ИГРОМАНЕР
СПЕЦ
03|64|МАРТ 2006
ЕЖЕМЕСЯЧНЫЙ
ТЕМАТИЧЕСКИЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ

GAME CODING

(game)land hi-fun media
9 771609 102006 03>

Создай свою реальность

с компьютером DEPO Ego на базе процессора Intel® Pentium® 4 с технологией HT

Включи DEPO Ego — и перед тобой откроется новая реальность твоих любимых компьютерных игр. Наслаждайся быстротой реакции и скоростью, исследуй распахнувшийся перед тобой мир высококачественной компьютерной графики и настоящего экшена. Теперь эта цифровая реальность может стать твоей благодаря компьютеру DEPO Ego на базе процессора Intel® Pentium® 4 с технологией HT.

DEPO Ego 525 DHR:

- процессор Intel® Pentium® 4 640 с технологией HT
- чипсет Intel® 925XE с улучшенной архитектурой
- сверхбыстрая память DDR2
- новые возможности графики PCI-Express
- реалистичный объемный 8-канальный звук

Компания DEPO Computers

Если нужен компьютер, покупай у производителя
на www.depo.ru или по тел. (495) 969-22-00

Товар сертифицирован

intro



Когда я был маленький, когда интернета и компьютеров еще не было, деревья были большими, солнце всходило с другой стороны, холмы были выше, из конопли делали только веревки и все знания приходилось черпать из бумажной литературы, — в те времена я любил полистать энциклопедический словарь. По данным этого авторитетного издания получалось, что игра является видом непродуктивной деятельности, мотив которой заключается не в результате, а в самом процессе. Во как!

Миллионы людей по всему миру, презрев результат, посвящают свое время непродуктивной деятельности, отдают за нее деньги тем, кто как раз устремлен к производству игр. В общем, целые миллионы прожигают свои жизни. Почему? Читаем дальше: «Имеет важное значение в воспитании, обучении и развитии детей как средство психологической подготовки к будущим жизненным ситуациям. Свойственна также высшим животным». Теперь все ясно! И правда, компьютерные игры — отличная школа жизни! Вот, допустим, «Цивилизация». Народ недоволен, бунтует? Надо построить храм, а потом и Колизей. Как вариант — сменить госстрой и ввести в город войска. Сколько же людей оптимизировали свою личную жизнь, усвоив эти простые истины? Я в их числе :). А FallOut? Правило трех «У» (Укради, Уговори, Убей) в реальной жизни работает не очень из-за ограничений, хотя и позволяет сделать выводы. А какую нечеловеческую реакцию мы развивали, когда играли deathmach в Quake 3? Сколько отрицательных эмоций было утилизировано на неживых монстров? Да, получается, что игры — это совсем не фигня, а геймдевелоперы несут в массы добро и процветание. Крепко подумав, мы решили посвятить читателей в этот нелегкий труд. К тому же, что самое интересное, геймдев-труд далеко не альтруистский!

Получай! Целый номер, посвященный разработке игр. Мне было очень приятно делать его, так как когда-то я наполовину портировал игру «Лестница» на Delphi, а потом проимел все наработки :). Кстати, не забывай посещать наш форум на **forum.xakep.ru** — нам очень интересно твое мнение о СПЕЦе.

Александр Лозовский,
высшее животное,
которому тоже
свойственно играть
alexander@real.xakep.ru

GAME CODING

СПЕЦ

ЕЖЕМЕСЯЧНЫЙ
ТЕМАТИЧЕСКИЙ
КОМПЬЮТЕРНЫЙ
ЖУРНАЛ
03(64) МАРТ 2006

www.xakep.ru

РЕДАКЦИЯ
Главный редактор
Николай «AvaLANche» Черепанов (avalanche@real.xakep.ru)
Выпускающие редакторы
Александр «Dr.Klouniz» Лозовский (alexander@real.xakep.ru)
Андрей Каролик (andrusha@real.xakep.ru)
CD/OFFTOPIC
Иван «SkyWriter» Касатенко (sky@real.xakep.ru)
Литературный редактор
Валентина Иванова (valy@real.xakep.ru)
Арт-директор
Иван Васин (vasin@real.xakep.ru)
Дизайнер
Наталья Жукова
Иллюстратор
Анна Журко
Цветокорректор
Александр Киселев

РЕКЛАМА
Директор по рекламе ИД (game)land
Игорь Пискунов (igor@gameland.ru)
Руководитель отдела рекламы цифровой группы
Ольга Басова (olga@gameland.ru)
Менеджеры отдела
Ольга Емельянцева (olgaeml@gameland.ru)
Евгения Горячева (goryacheva@gameland.ru)
Оксана Алехина (alekhina@gameland.ru)
Менеджер по работе с сетевыми РА, корпоративные продажи
Максим Григорьев (grigoriev@gameland.ru)
Трафик-менеджер
Марья Алексеева (alekseeva@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

РАСПРОСТРАНЕНИЕ
Директор отдела дистрибуции и маркетинга
Владимир Смирнов (vladimir@gameland.ru)
Оптовое распространение
Андрей Степанов (andrey@gameland.ru)
Подписка
Алексей Попов (popov@gameland.ru)
тел.: (495) 935.70.34
факс: (495) 780.88.24

PUBLISHING
Издатель
Сергей Покровский (pokrovsky@gameland.ru)
Учредитель
ООО «Гейм Лэнд»
Директор
Дмитрий Агарунов (dmitri@gameland.ru)
Финансовый директор
Борис Скворцов (boris@gameland.ru)

ГОРЯЧАЯ ЛИНИЯ ПО ПОДПИСКЕ
тел.: 8 (800) 200.3.999
Бесплатно для звонящих из России
ДЛЯ ПИСЕМ
101000, Москва,
Главпочтамт, а/я 652, Хакер Спец
spec@real.xakep.ru
http://www.xakep.ru

Отпечатано в типографии «ScanWeb», Финляндия
Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций
ПИ № 77-12014 от 4 марта 2002 г.
Тираж 42 000 экземпляров.
Цена договорная.

ИГРОВЫЕ ТЕХНОЛОГИИ

ТЕОРИЯ

ЭКСПЕРТ НОМЕРА

АЛЕКСАНДР ГЛАДЫШ

ВЕДУЩИЙ ПРОГРАММИСТ STEP CREATIVE GROUP (ПОСЛЕДНЯЯ ИЗДАННАЯ РАЗРАБОТКА ЭТОЙ КОМПАНИИ — КВЕСТ «ЗВЕЗДНОЕ НАСЛЕДИЕ»)

РАЗРАБОТКА

SPECIAL DELIVERY

offtopic

HARD

SOFT

CREW

STORY

КТО НЕ МЕЧТАЛ, ЕЩЕ В САМОМ ДЕТСТВЕ СЕВ ЗА КОМПЬЮТЕР, НЕ ПРОСТО ПОИГРАТЬ В ИГРЫ, А СТАТЬ КЕМ-ТО БОЛЬШИМ В ЭТОМ ВИРТУАЛЬНОМ МИРЕ — ТВОРЦОМ. НИКОГДА РАНЬШЕ ТВОЯ МЕЧТА НЕ БЫЛА ТАК БЛИЗКА К ОСУЩЕСТВЛЕНИЮ! НА ЭТОМ ДИСКЕ ТЫ НАЙДЕШЬ ВСЕ, ЧТО МОЖЕТ ПОТРЕБОВАТЬСЯ ТЕБЕ, ЧТОБЫ НАПИСАТЬ СОБСТВЕННУЮ ИГРУ

CD:

DIRECTX

DirectX 9 SDK (февраль 2006)
DirectX SDK Symbols (декабрь 2005)

ДВИЖКИ

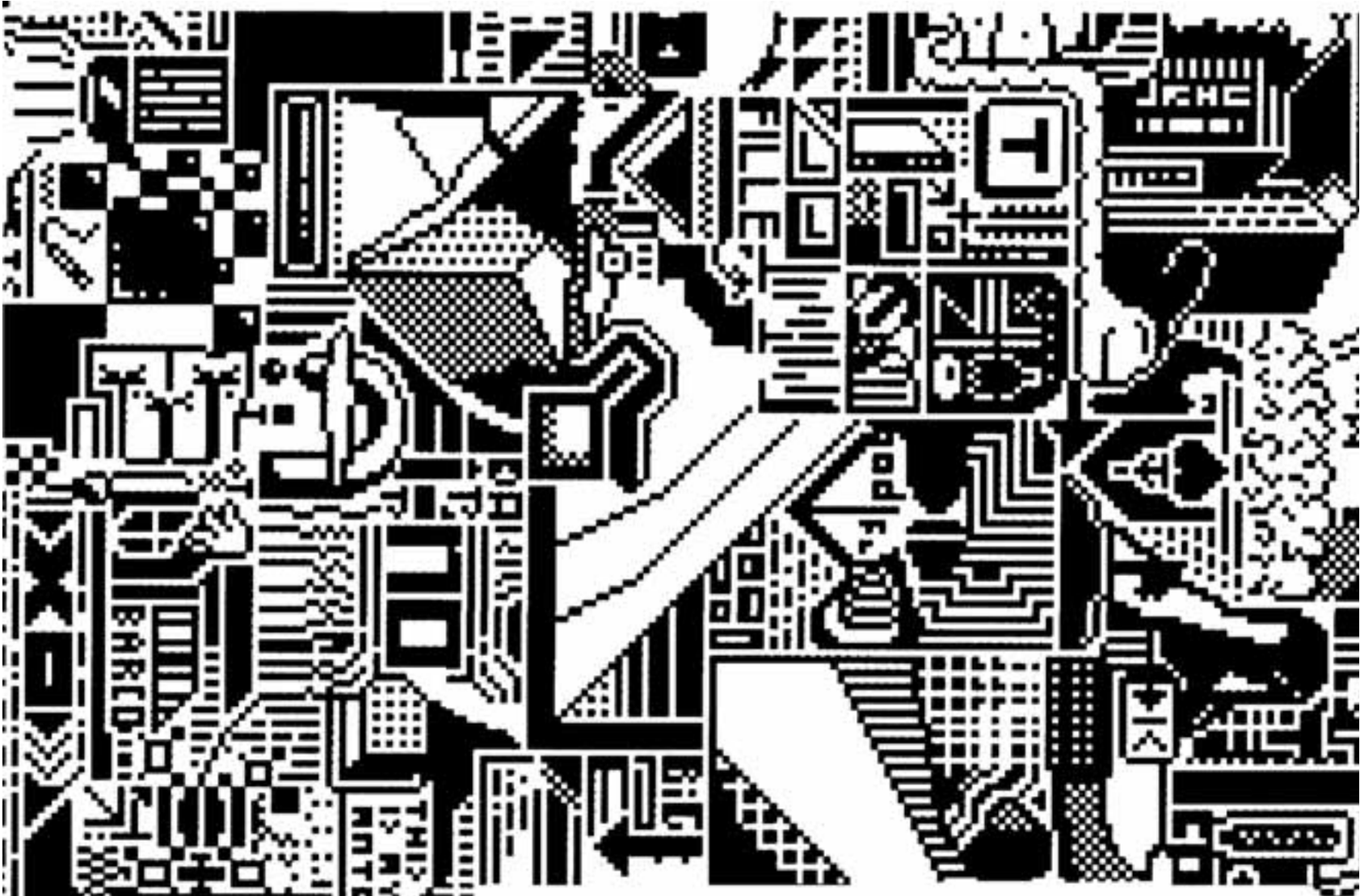
AGL 1.1
Irrlicht 0.14.0
Nebula2 SDK
OGRE SDK 1.0.7
Torque Game Engine 1.4

БИБЛИОТЕКИ

Boost 1.33.1
FMOD 4.03.00
Lua 5.0.2
ODE 0.5
OPAL 0.3.1
OpenAL
SDL 1.2.9

СОФТ ОТ NONAME

AimOne Screen Recorder v1.31
CUE Splitter v0.5
Crypto 2.0
DriveCrypt 4.20
GIF Movie Gear 4.1.0
Jet Audio 6.2.5 Plus VX
Light Alloy 3.5.5944
Miranda IM MDpack '07.02.06
New Weather v1.0
Opera 9 TP 2 (en)
Quiet Internet Pager (QIP) Build 7810 Alpha
Right Click Image Converter 2.2.2
Total Video Converter v2.4
Unlocker 1.7.9



ИГРОВЫЕ ТЕХНОЛОГИИ

По данным сети «Союз», десятка самых продаваемых PC-игр в России (на 7 февраля 2006 года) выглядит следующим образом (данные **3dgamers.ru**):

1. CALL OF DUTY 2
2. МАГИЯ КРОВИ
3. MAX PAYNE 2
4. ХРОНИКИ НАРНИИ. ЛЕВ, КОЛДУНЬЯ И ВОЛШЕБНЫЙ ШКАФ
5. SILENT HUNTER 3
6. КОРСАРЫ III
7. МАДАГАСКАР
8. КОРСАРЫ III
9. ТАКЕДА 2: ПУТЬ САМУРАЯ
10. COUNTER-STRIKE 1. ANTHOLOGY

По данным интернет-магазина Ozon, десятка самых продаваемых PC-игр в России (на февраль) выглядит следующим образом (данные **3dgamers.ru**):

1. ПРИНЦ ПЕРСИИ 3: ДВА ТРОНА (PRINCE OF PERSIA 3: TWO THRONES)
2. LADA RACING CLUB
3. HALF-LIFE 2. КОЛЛЕКЦИОННОЕ ИЗДАНИЕ
4. ПРОКЛЯТЫЕ ЗЕМЛИ: ЗАТЕРЯННЫЕ В АСТРАЛЕ
5. LADA RACING CLUB
6. WORLD OF WARCRAFT
7. CALL OF DUTY 2
8. NEED FOR SPEED: MOST WANTED
9. X3: ВОССОЕДИНЕНИЕ
10. БАБА-ЯГА УЧИТСЯ ЧИТАТЬ

По данным NPDTechWorld, десятка самых продаваемых PC-игр в США (за период с 22 по 28 января 2006 года) выглядит следующим образом (данные **3dgamers.ru**):

1. WORLD OF WARCRAFT
2. CIVILIZATION IV
3. THE SIMS 2
4. AGE OF EMPIRES III
5. BATTLEFIELD 2
6. ZOO TYCOON 2
7. CALL OF DUTY 2
8. THE SIMS 2 NIGHTLIFE
9. BATTLEFIELD 2 SPECIAL FORCES
10. GUILD WARS

жанровое жаркое

ВЗГЛЯД НА ИГРОВЫЕ ЖАНРЫ С ПОЗИЦИИ РАЗРАБОТЧИКА

НА ДВОРЕ 2006 ГОД, И ЕЩЕ КАКИХ-НИБУДЬ СЕМЬ ЛЕТ НАЗАД МЫ ТВЕРДО МОГЛИ БЫ СКАЗАТЬ: «ВРЕМЕНА РАЗРАБОТЧИКОВ-ОДИНОЧЕК УШЛИ, КАК УШЛИ ВРЕМЕНА ПРОСТЫХ И ГЕНИАЛЬНЫХ 2D-ИГР. НАСТАЛА ЭРА КРУПНЫХ КОМПАНИЙ, БОЛЬШИХ ДЕНЕГ И ГИГАНТСКИХ КОМАНД ГЕЙМ-ДЕВЕЛОПЕРОВ». ОДНАКО XXI ВЕК, ПРИНЕСШИЙ НАМ МОБИЛЬНЫЕ ПЛАТФОРМЫ, ОТКРЫЛ И НЕМАЛУЮ НИШУ ДЛЯ РАЗРАБОТЧИКОВ-ИНДИВИДУАЛОВ И НЕБОЛЬШИХ КОНТОР, КОТОРЫЕ ТОЖЕ МОГУТ РАБОТАТЬ И ПОЛУЧАТЬ ДЕНЬГИ (А МОГУТ ПОРАБОТАТЬ FOR FUN). НЕВАЖНО |

ЛОЗОВСКИЙ АЛЕКСАНДР (ALEXANDER@REAL.XAKEP.RU)

rpg

Сейчас уже трудно найти человека, который не знает, что RPG расшифровывается как «Role Playing Game», а не «Ручной Противотанковый Гранатомет». Эти игры требуют порядочных усилий разработчика: построения внятного сюжета (получается, что в XXI веке немногим нравятся задания в духе «Возьми свиток у Купца Ивана и отнеси его через всю карту в деревню Большие Бугры, перебив по дороге всех монстров») и интересных заданий. Эта задача оказывается первостепенной, превосходя даже графику и звук :). Кстати, проблема выбора графики для кодера под большие компьютеры тоже не беспроблемна. Что предпочесть? 3D или симпатичное 2D? Для RPG этот вопрос так и не решен. В области мобильных платформ ситуация проще: играть в них на КПК относительно удобно благодаря стилусу, а 3D RPG отсутствуют как класс и пока никому не нужны. Кстати, клон Fallout на КПК давно имеется — это Nuclear Time, Kings Bounti портирована (очень советую!). Очень скоро будет и Heroes 2.

ДОСТОИНСТВА:

— ПОЗВОЛЯЮТ УБИТЬ МНОГО ВРЕМЕНИ.
— ИНТЕРЕСНЫ.
— СОБИРАЮТ ВОКРУГ СЕБЯ ТОЛПЫ ПОЧИТАТЕЛЕЙ И ФАНАТОВ.

НЕДОСТАТКИ:

— СЛОЖНЫ В РАЗРАБОТКЕ СЮЖЕТА: НУЖНЫ ГРАМОТНЫЕ КРЕАТОРЫ И СВЕЖИЕ ИДЕИ.
—ОТНОСИТЕЛЬНО СЛОЖНА РЕАЛИЗАЦИЯ ГРАФИКИ, ОСОБЕННО ИНТЕРФЕЙСА.
— НЕ ПОДХОДЯТ ЛЮБИТЕЛЯМ ДИНАМИКИ И НЕМНОГО НАПРЯГАЮТ МОЗГ, ЧТО В НЕКОТОРЫХ СИТУАЦИЯХ НЕДОПУСТИМО :).

стратегии

Уменьшился ли интерес пользователей к стратегиям? Действительно ли народ режется в мультиплеерные RPG, оторвался от корней и не чтит заветы предков? Пожалуй, вряд ли. На самом деле время от времени в обществе возникала идея: «Ну все, от стратегий нам ждать больше нечего. Хедлайнеры уже реализовали все идеи, какие есть. И ничего нового уже не придумаешь». Такие мысли возникали и до выхода Dungeon Keeper, который перечеркнул их как раз своим выходом и последовавшей бешеной популярностью. Отсюда следует простой вывод: по-настоящему революционная идея выведет любой продукт на первые строчки хит-парадов. Правда, идея должна быть по-настоящему крутой. Еще одна «правда» в том, что некий интерес пред-ставляет и извращение/new life of/портирование проверенных продуктов. Но и тут есть одно «но». Продукт должен быть законченным. Если помнишь, однажды в конце прошлого века на лотках мы увидели Warcraft 2000 — революционный извращ над оригинальным War2, который был порожден на свет несколькими вольными каменщиками-программерами за полгода. В итоге они вытворили незаконченный извращ с двумя миссиями. Зато там была атомная бомба, и вообще играть было небезынтересно :). Какая же мораль следует? Взялся извращать классику — делай это до конца. Например, думаю, многие old_school'ные компьютерщики не откажутся погонять на покете в C&C или «Дюну 2». Правда, извращать такие продукты можно только с разрешения официальных лиц. Ты же не какой-нибудь пират, правда?

ДОСТОИНСТВА:

— УДОБНО ИГРАТЬ НА ПОКЕТЕ.
— РАЗВИВАЕТ МОЗГ.
— ВОЗМОЖНО, ДАЕТ НОВУЮ ЖИЗНЬ КЛАССИЧЕСКИМ ВЕЩАМ И ОЧЕНЬ ВЕРОЯТЕН ИСХОДНЫЙ КОД ТОЙ САМОЙ КЛАССИКИ.
— ОТНОСИТЕЛЬНО ПРОСТАЯ ГРАФИКА.

НЕДОСТАТКИ:

— СЛОЖНОСТИ С AI.
— СЛОЖНОСТИ СО СВЕЖИМИ ИДЕЯМИ, ЕСТЬ РИСК АБСОЛЮТНОГО ПРОВАЛА.
— ПРИ ПОРТИРОВАНИИ — РИСК ПОДДАТЬСЯ СОБЛАЗНУ ЧРЕЗМЕРНЫХ УЛУЧШЕНИЙ, УБИТЬ АТМОСФЕРУ ОРИГИНАЛЬНОЙ ИГРЫ, ПРОВАЛИТЬСЯ.

драки

Mortal Combat, Street Fighter, Tekken… Знакомые названия? Эти игры отнюдь не потеряли своей актуальности, особенно в наш век беспроводных коммуникаций, когда всем хочется в свободное (и рабочее) время набить морду соседу по офису, используя, конечно, смартфон. Так чего же мы ждем? :)

ДОСТОИНСТВА:

— АКТУАЛЬНО В ЭРУ БЕСПРОВОДНЫХ КОММУНИКАЦИЙ.
— НЕ НАПРЯГАЕТ МОЗГ.

НЕДОСТАТКИ:

— НЕУДОБНО ДЛЯ КПК.

азартные и интеллектуальные игры

С этими играми сложилась неоднозначная ситуация. С одной стороны, рынок шахмат пресыщен сверх всякой меры :), и, наверное, нет смысла писать очередной шахматный интеллект. С другой стороны, слабо развиты шашки (в том числе русские) и карточные игры. Не ошибся ли я насчет карточных игр? :) Нет, не ошибся. Действительно, у нас есть несколько преферансов и дураков для больших компьютеров, у нас есть Pocket Pref, у нас есть Open Source-преферанс для Linux, существует куча блэк-джеков и стрип-блэк-джеков, но… Преферанс для покета совсем неидеален, и, по-моему, есть куда развивать его AI. Нормального «дурака» для покета я не видел. Стрип-блэк-джеки можно писать тоннами :). Приступим? Кстати, самый сильный среди преферансов AI — у «Марьяжа», разработка которого прекращена. Вероятно, энтузиасты могут обсудить вопрос дальнейшего развития (на разных платформах) с Дмитрием Лесным и Львом Натансоном. Еще одним хитрым ходом может быть логическая игра собственной разработки или компьютерная реализация какой-нибудь игры шаманов древнего Тибета, забытой всеми. Как знать, может быть, именно это принесет тебе славу Пажитнова?

ДОСТОИНСТВА:

— НЕЗАМЕНИМЫ ДЛЯ ИНТЕЛЛЕКТУАЛОВ.
— ОТНОСИТЕЛЬНО ПРОСТОЙ КОДИНГ ИНТЕРФЕЙСА.
— ЛОГИЧЕСКИХ ИГР МНОГО, В ТОМ ЧИСЛЕ ОФЛАЙНОВЫХ (МОЖЕТ БЫТЬ, ЭТО ТВОЙ СЧАСТЛИВЫЙ БИЛЕТ).
— АЗАРТНЫЕ ИГРЫ МОГУТ ПРИНОСИТЬ ДЕНЬГИ.

НЕДОСТАТКИ:

— ВЕСЬМА НЕПРОСТОЙ КОДИНГ AI В НЕКОТОРЫХ ИГРАХ.
— ОПРЕДЕЛЕННАЯ НАСЫЩЕННОСТЬ РЫНКА.
— СЕТЕВЫЕ РЕАЛИЗАЦИИ ПОСТОЯННО НАГИБАЮТСЯ ХАКЕРАМИ.

fps

Наверное, абсолютно бесперспективный жанр, если судить с позиции «домашней» команды разработчиков. Эти игры диктуют моду в графике, ими занимаются огромные корпорации, в них вваливаются кучи долларов. Старые и добрые шутеры уже портированы под мобильные платформы, и в этом направлении нам ловить нечего. Наверное, не портированы только Catacombs (если помнишь, был такой шутер то ли до Wolf 3D, то ли чуть после, в нем нужно было глушить врагов магией из руки и собирать какие-то ключи) и Blood. Однако то, нужен ли кому-то такой порт, — большой вопрос. Pocket Hexen, Doom и Quake у нас уже есть, и мне, к примеру, не понравилось играть в них на КПК :).

ДОСТОИНСТВА:

— КРУТО И КРАСИВО УБИВАЕШЬ ВРАГОВ.
— ДИНАМИЧНО.
— НЕ ПРОИЗВОДИШЬ ДВИЖЕНИЙ ИЗВИЛИНАМИ («ВЫНЬ ЭНЕРГОКУБЫ, ВСТАВЬ ЭНЕРГОКУБЫ»).

НЕДОСТАТКИ:

— РАЗРАБОТКА НОРМАЛЬНОГО ПРОЕКТА ПОЧТИ НЕРЕАЛЬНА — НАМ ОСТАЕТСЯ ТОЛЬКО РАЗРАБОТКА МОДОВ.

аркада

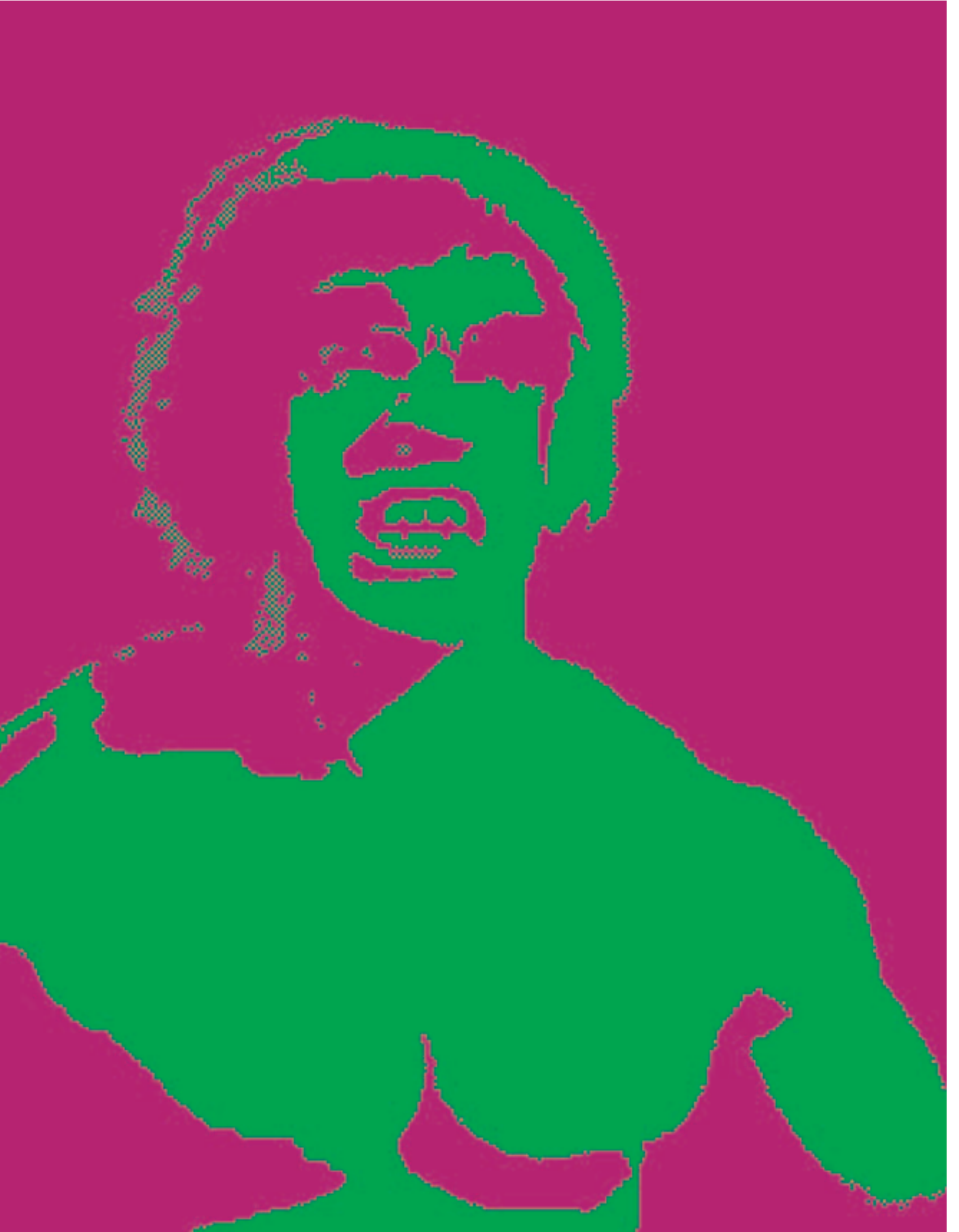
Самый древний и лучший игровой жанр для разработчика. На больших компьютерах этот жанр уже умер и разложился, а на телефонах — испытывает творческий подъем по одной простой причине: управление большинства телефонов поддерживает судорожные нажатия на вверх-вниз-вправо-влево, а маленький экран и скудные звуковые возможности никогда не были ограничением для этого жанра. Правда, для КПК он является приемлемым далеко не всегда по причине управления — совсем неудобно гонять стилусом человечков по лабиринтам, самолетики над полями с танками, а клавишное управление на большинстве КПК оставляет желать лучшего

ДОСТОИНСТВА:

— ЛЕГКОСТЬ РЕАЛИЗАЦИИ;
— ПРОСТОТА И ЛЕГКОСТЬ ИГРЫ;
— ИНТЕРЕСНОСТЬ;
— ДИНАМИКА;

НЕДОСТАТКИ:

— ЕСТЬ ПРОБЛЕМЫ С УПРАВЛЕНИЕМ НА КПК.
— НЕ ПОДХОДЯТ ИНТЕЛЛЕКТУАЛАМ И ЛЮБИТЕЛЯМ АЗАРТНЫХ ИГР.



неуправляемый directx

ЧТО .NET ГЕЙМДЕВЕЛОПЕРАМ ГОТОВИТ?

НЕ СЕКРЕТ, ЧТО ТЕХНОЛОГИЯ .NET ПОЯВИЛАСЬ ИЗ-ЗА ЗАВИСТИ MS УСПЕХУ JAVA: «КАК ЖЕ ТАК? В МОБИЛЬНЫХ ТЕЛЕФОНАХ НЕ СТОЯТ ОКНА И РАБОТАЮТ ПРОГРАММЫ НА КАКОЙ-ТО JAVA?!» И ВОТ, ВЫПУСКАЯ ИЗ БУТЫЛКИ ДЖИНА ПО ИМЕНИ .NET, MS СОЗДАЛА СЕРЬЕЗНУЮ ПРОБЛЕМУ ДЛЯ КОНКУРЕНТОВ В ВИДЕ БИБЛИОТЕКИ MANAGED DIRECTX (УПРАВЛЯЕМЫЙ DIRECTX) | **ФЛЕНОВ МИХАИЛ АКА HORRIFIC**

«БУДУЩЕЕ MANAGED DIRECTX ПРИЗРАЧНО, НО ЕСЛИ ЕГО СУМЕЮТ ПОРТИРОВАТЬ НА РАЗНЫЕ ПЛАТФОРМЫ, ТО МЫ СМОЖЕМ СОЗДАТЬ QUAKE, КОТОРЫЙ БУДЕТ РАБОТАТЬ КАК В LINUX, ТАК И В WINDOWS»

кто таков? Как известно, Managed DirectX — это поддержка DirectX из управляемого кода, то есть из программ, написанных на платформе .NET и для нее. Изначально эта технология даже называлась DirectX .NET, но позже была переименована в Managed DirectX.

Первый раз я услышал о новом DirectX примерно в 2003 году. Информация, которая просачивалась в интернет, была очень обрывочной, а на официальном сайте появилась только бета-версия управляемого DirectX. По всей видимости, эта библиотека была написана на C# на базе устаревшего к тому времени DirectX8. Я не видел эту библиотеку в действии, потому что в те времена пользовался телефонным подключением к интернету и был не в состоянии качать большой файл. Судя по отзывам «счастливчиков», библиотека была провальной и тормозила как ржавый «Запорожец» по сравнению с «Мерседесом» (классическим DirectX).

Затем наступило затишье. Информации о новой технологии стало минимум, в основном в виде слухов, судя по которым библиотеку переписывали дважды, причем полностью. Как всегда, официальные лица либо отмалчивались, либо несли какую-то чушь, поэтому было слишком сложно определить, где правда.

первый релиз И вот перед самым появлением DirectX 9 мы узнаем, что MS рассылает бета-версию обновленной библиотеки Managed DirectX, а в девятую версию DX SDK должен попасть полный вариант. Бета-версию мне таки не удалось увидеть, но когда на жестком диске появился установочный файл DX SDK 9.0, то в мои глаза бросилось прежде всего наличие в папке Help двух файлов помощи directx9_c.chm и directx9_m.chm. Первый файл описывает классический DirectX для C++, а второй — DirectX 9.0 for Managed Code или просто Managed DirectX.

После установки DX SDK сразу заглядываем в директорию Samples и видим директории C# и VB.NET с примерами использования DirectX для соответствующих языков. Да, если раньше создавать игры на VB было проблематично, то с помощью VB.NET — милости просим, легко и непринужденно создавай любое приложение DirectX.

поддержка Библиотека Managed DirectX разделена на следующие пространства имен:

- **MICROSOFT.DIRECTX.DIRECT3D** — ИНТЕРФЕЙСЫ ДЛЯ РЕАЛИЗАЦИИ 3D-ГРАФИКИ;
- **MICROSOFT.DIRECTX.DIRECTDRAW** — СТАРЫЕ, НО ОЧЕНЬ ДОБРЫЕ ФУНКЦИИ ДЛЯ РАБОТЫ С 2D-ПОВЕРХНОСТЯМИ И СООТВЕТСТВУЮЩЕЙ ГРАФИКОЙ;
- **MICROSOFT.DIRECTX.DIRECTSOUND** — ИНТЕРФЕЙСЫ ДЛЯ РАБОТЫ СО ЗВУКОМ;
- **MICROSOFT.DIRECTX.DIRECTINPUT** — ИНТЕРФЕЙСЫ ДЛЯ РАБОТЫ С УСТРОЙСТВАМИ ВВОДА.

Это основные пространства, и уже становится ясно, что у нас есть все необходимое для создания игр. Кроме того, в managed DirectX есть интерфейсы для реализации средств безопасности и поддержка сети.

Все эти интерфейсы можно использовать в следующих языках программирования:

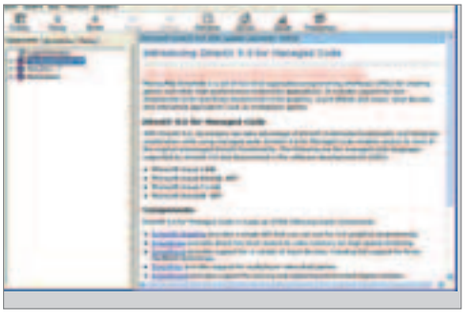
- MICROSOFT VISUAL C#
- MICROSOFT VISUAL BASIC .NET
- MICROSOFT VISUAL C++
- MICROSOFT JSCRIPT .NET

Самое интересное — последний язык. Судя по всему, Managed DirectX планируют использовать в интернете для создания сервисов с поддержкой 3D-графики!

тест производительности Попробуем провести достаточно простой тест производительности. Так как примеры, которые идут в поставке с DX SDK, схожи по выполняемым функциям и притом выводится FPS, нам будет достаточно сначала запустить на одном компьютере пример на классическом DirectX, а потом — с использованием Managed DirectX. В качестве примера возьмем плавающего дельфина DXSDK\Samples\C#\Direct3D\DolphinVS.

На ноутбуке Pentium M 1.7 (видео от ATI Mobility Radeon 9700) получились следующие результаты FPS для плавающего дельфина: 1) программа на C# с использованием Managed DirectX в среднем показала 540 FPS; 2) программа на C с использованием классического DirectX в среднем показала 620 FPS.

Разница в 80 FPS (в данном случае — примерно 15%) оставляет двусмысленное впечатление. С одной стороны, разработчики хорошо постарались и для управляемого кода получилась достаточно высокая производительность. Не забываем, что код C# выпол-



Файл помощи directx9_m.chm по Managed DirectX

Блог Тома Миллера (один из разработчиков MS). Лучший источник информации о Managed DirectX

няется как бы в виртуальной машине. С другой стороны, падение производительности на 15% неоправданно, я бы просто зажал их. Лучше потратить драгоценные потерянные такты процессора на что-то пополезнее.

И все же для Managed DirectX найдется применение в простых играх, а разработчики на таких языках, как C# и VB.NET, будут очень довольны. Раньше создание игр на VB было сущей каторгой!

источники информации Как все уже заметили, пока распространилось очень мало информации о Managed DirectX, потому что технология новая и многие относятся к ней с опаской. Самый лучший источник — это блог Тома Миллера. Том является одним из разработчиков API для Managed DirectX. Кто как не он знает все тонкости и последние новости?

Кроме того, Том написал даже целую книгу, в которой подробно описал Managed DirectX девятой версии. Книга посвящена разработке графики и игр и называется Managed DirectX 9 Graphics and Game Programming (Sams Publishing, 2004). В России эта книга была переведена в издательстве «КомБук», причем название книги и ее обложка не изменились.

субъективные размышления Технология Managed DirectX — мощное средство создания игр. Представь себе игру, написанную на C# в сочетании с Managed DirectX. По заявлениям MS, программы .NET могут выполняться на любой платформе при наличии соответствующего .NET Framework, следовательно, мы получаем межплатформенную игру. Однако с последним есть серьезные проблемы. Во-первых, не существует .NET, нормально портированной на системы, отличные от Windows. Есть лишь попытки сделать что-то под Linux, но они неработоспособны на все 100%.

Как мне кажется, нормальные люди вообще не будут портировать DirectX, потому что не хотят нарываться на серьезные проблемы. Managed DirectX использует библиотеки неуправляемого DirectX, портировать который мегасложно из-за применения технологии COM. Одну эту технологию много раз пытались перенести на Linux, половина попыток закончилась неудачно, а остальные накрылись медным тазом. То же самое может ожидать Managed DirectX. Если вспомнить, что есть еще такие ОС, как Sun, FreeBSD, MacOS, в которые не ступала нога .NET, то все поймут, что межплатформенность — мечта, которая может и не сбыться.

На первый взгляд, вырисовывается страшная картина. Это мое, субъективное мнение, но оно основано на подтвержденных фактах. С другой стороны, MS уже вложила сумасшедшие соответствующие деньги и вероятность провала всей технологии .NET стремится к нулю.

итого Что же ожидает технологию Managed DirectX? Я думаю, она будет развиваться, но только в сегменте стационарных компьютеров и мобильных устройств с ОС от Microsoft. Так как наладонники и вкарманники пока не обладают графическими возможностями ATI и GeForce, развитие будет протекать медленно и печально. Я отслеживаю это развитие и одним глазком подглядываю в планы, чтобы в случае чего вскочить на коня и начать разработку игр на этой технологии. Сейчас же…наверное, нет смысла тратить время на подобное, так как Managed DirectX работает только на стационарах, на которых есть быстрый классический DirectX. Как только появятся мобильники с полноценной поддержкой, сразу можно будет броситься изучать Managed, благо он не очень сложен и не сильно отличается от классики

Плавающий дельфин в программе, написанной на C#

ЭНЦИКЛОПЕДИЯ GamePost

Незаменимый помощник при выборе игры

Описание:

Fahrenheit (также известный как Indigo Prophecy) – один из главных хитов 2005 года. Это интерактивный триллер, где вы играете и за детективов, и за подозреваемого – и каждое ваше действие, каждый выбор имеет значение. Интуитивное управление, и интерфейс, доведенный до минимализма, помогают погрузиться в игру с головой, а повороты сюжета продержат вас в напряжении до самой развязки.

Fahrenheit (Indigo Prophecy)	Жанр:
$69.99	**Adventure**

Описание:

Разработчики Guild Wars взяли все лучшие черты из других MMORPG и смешали их таким образом, что вы забудете обо всем том, что до сих пор раздражало вас в многопользовательских играх. Вы можете встретить новых друзей в городах и крепостях, сформировать партию и тут же отправиться выполнять задания вместе. В вашей партии всегда будет копии карты квеста.

Guild Wars Special Edition (EURO)	Жанр:
$79.99	**RPG**

Описание:

Age of Empires III погрузит вас в атмосферу XVI-XIX веков. Вам предстоит строить собственную империю, колонизировать и завоевывать Северную и Южную Америку и участвовать в эпических войнах. Невиданный уровень реализма и великолепно отображенное культурное разнообразие порадуют даже самых утонченных эстетов.

Age of Empires III	Жанр:
$79.99	**Strategy**

САМАЯ ПОЛНАЯ ИНФОРМАЦИЯ ОБ ИГРАХ

* Огромное количество скриншотов
* Исчерпывающие описания
* Возможность посмотреть внутренности коробок

Играй просто!
GamePost

Тел.: (495) 780-8825
Факс.: (495) 780-8824
www.gamepost.ru
GAMEPOST

тяга к революциям

НАСТОЯЩЕЕ И НЕДАЛЕКОЕ БУДУЩЕЕ ТРЕХМЕРНЫХ ТЕХНОЛОГИЙ

«ШЕЙДЕРЫ» — ОДИН ИЗ САМЫХ НЕУДАЧНЫХ ТЕРМИНОВ ВО ВСЕЙ ИСТОРИИ IT, ТЕМ НЕ МЕНЕЕ ЭТА ТЕХНОЛОГИЯ ПРИШЛА К НАМ ДОВОЛЬНО ДАВНО И НАДОЛГО ЗАНЯЛА КЛЮЧЕВОЕ МЕСТО В НАШЕМ БЮДЖЕТЕ. ОДНАКО ШЕЙДЕРЫ — ЭТО НЕ ЕДИНСТВЕННЫЙ ВИД ГРЫЗУНОВ, КОТОРЫЕ ГРОЗЯТ НАШЕМУ БЮДЖЕТУ ДЫРКАМИ. БУДЕМ ГОТОВИТЬСЯ К КУПЮРНОМУ ЛИСТОПАДУ |TONY (TONY@EYKONTECH.COM)

история На платформе PC точкой отсчета для шейдеров является видеокарта NVidia GeForce 2 GTS. Именно в этой карте впервые появились шейдеры, но они не произвели прорыва. В связи с тем, что в то время шейдеры не были специфицированы и не поддерживались производителями игр, эта видеокарта так и осталась неким пробным камнем, запущенным в направлении наших бумажников.

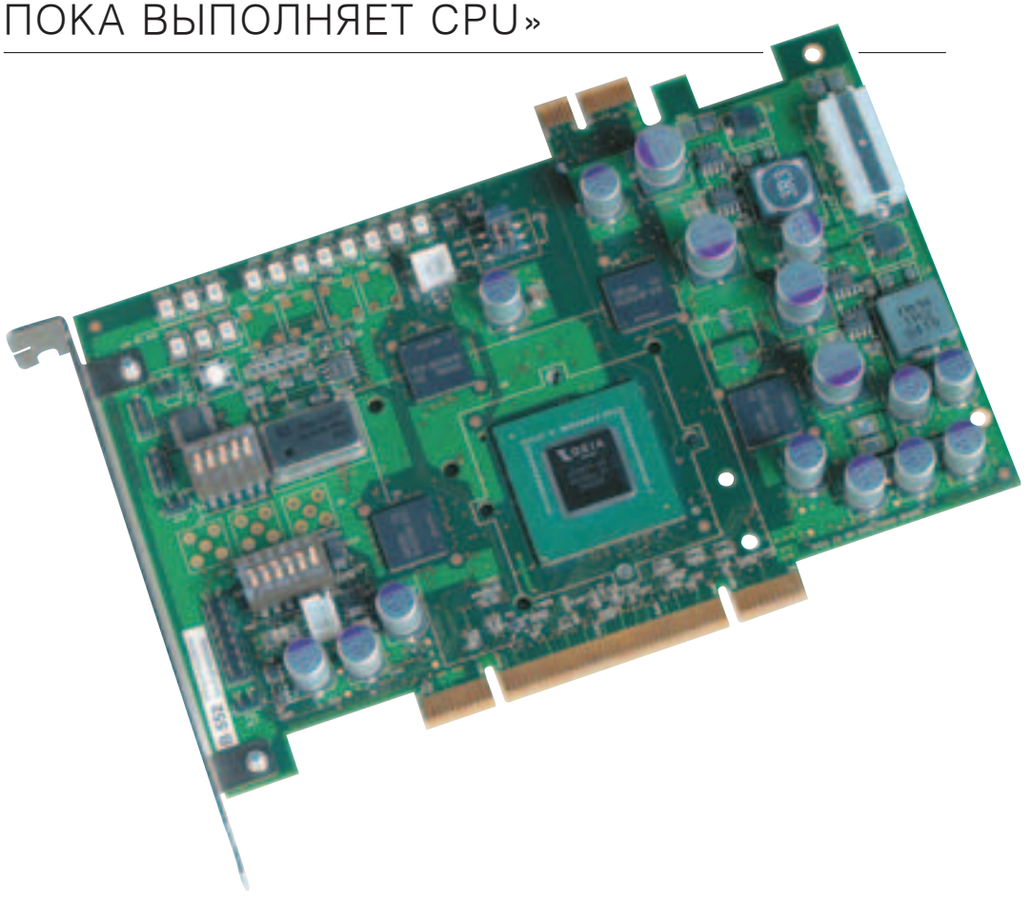
Официальная история шейдеров начинается с серии GeForce 3 — в то время появились шейдеры версии 1.X и спецификации DirectX 8.1. Для того чтобы не отягощать тебя лишними индексами и моделями карт, далее я буду рассказывать только о поколениях шейдеров (с точки зрения программистов). Итак, сейчас мы имеем три поколения: 1.X, 2.X и 3.X. На сегодняшний день (до релиза Windows Longhorn) последняя программная спецификация от Microsoft называется DirectX 9.0c, она включает в себя поддержку всех существующих версий шейдеров.

терминология С точки зрения разработчика графической программы, шейдер — это набор состояний графического конвейера плюс программный код, который выполняется графическим процессором (GPU). Например, в первом Quake тоже были шейдеры… Удивлен? Все очень просто, таким термином обзывали программную абстракцию, которая управляла различными состояниями видеокарты. Бессмысленно рассматривать код, исполняемый GPU, отдельно от состояний графического конвейера — это единое и неделимое целое.

графический конвейер Прежде чем начинать изучение шейдеров, необходимо представлять себе, как работает современный графический конвейер, какие данные приходят ему на вход и как он обрабатывает их. Представь себе, что ты хочешь нарисовать на экране трехмерный куб. В памяти твой куб будет представляться в виде набора вершин, которые описывают составляющие его треугольники (примитивы). Каждая вершина характеризуется своими координатами в 3D-пространстве: x, y, z — минимум, без которого не обойтись. Обычно вершина содержит дополнительные параметры: нормаль, диффузный и спекулярный цвета, несколько наборов текстурных координат и т.д. Для оптимизации и чтобы задействовать вершинный кеш более эффективно, примитивы используют общие вершины (например, семь вершин, образующие пять треугольников).

Формирование изображения на экране происходит следующим образом. В каждый момент времени главный поток программы генерирует кадры изображения. Например, для комфортной игры в шутер необходимо генерировать минимум 25 кадров в секунду. Посмотри на диаграмму «Графический конвейер». Создание кадра начинается с того, что карте сообщается, какую именно гео-

«ТЕХНОЛОГИЯ PHYSX ВОЗЬМЕТ НА СЕБЯ ОБСЧЕТ ФИЗИКИ СЦЕНЫ, КОТОРЫЙ ПОКА ВЫПОЛНЯЕТ CPU»



Опытный образец платы PhysX

метрию ей необходимо нарисовать для этого кадра (шаг «буфер вершин»). Каждая вершина обрабатывается вершинным процессором, в который заранее загружена вершинная программа (вершинный шейдер). Обычно эта программа трансформирует вершины исходя из заданных мировой и видовой матриц и для каждой вершины вычисляет специфические данные (нормаль, цвет). Как ты помнишь, вся геометрия разбивается на треугольники. Видимые на экране фрагменты треугольников (после отброса невидимых фрагментов) попадают в пиксельный процессор, где они обрабатываются фрагментным процессором (пиксельный шейдер), который рассчитывает цвет каждого фрагмента. Откуда берутся компоненты цвета фрагмента? Из текстур, покрывающих треугольник, из освещенности и затененности фрагмента и прочих эффектов. Параметры для каждого фрагмента передаются в пиксельный шейдер интерполированными по всему обрабатываемому треугольнику, то есть они находятся в прямой зависимости от параметров вершин треугольника. После расчета цвета фрагмента он смешивается с текущим кадром в зависимости от настроек конвейера. Буферы между различными частями конвейера позволяют задействовать конвейер наиболее эффективно — с целью минимизации его простоев.

поколения процессоров Сравним возможности вершинных и фрагментных процессоров трех поколений. Поскольку первое поколение шейдеров было самым многочисленным (в плане версий), я возьму для сравнения только версию 1.4, остальные практически не отличаются от нее. Для версии шейдеров 2.0 буду рассматривать спецификацию расширенных шейдеров 2.X.

Как видно по сравнительной таблице, с каждым новым поколением возрастает длина шейдеров и количество регистров, что позволяет усложнять и усложнять используемые эффекты. Введение инструкций для управления кодом (циклы, условия) позволяет проверять передаваемые из базового кода, выполняемого обычным CPU, условия и параметры (например позиции источников света). Благодаря этому становится возможным создавать реалистичное попиксельное освещение, зависящее от нескольких различных лампочек, создавать реальные (мягкие) тени, эффекты глубины поверхностей и т.д.

вершинный процессор Теперь покопаемся в кишочках каждого процессора, первый в очереди — вершинный. Для каждой вершины выполняется вершинный шейдер, заранее загруженный в видеокарту. Посмотри на рисунок «Пример шейдера»: в XML-узле VertexShader содержится код вершинной программы (функция main). Эта программа довольно простая, к каждой вершине она добавляет диффузный цвет и вычисляет значение текстурных координат для специальной текстуры, которые изменяются в зависимости от того, под каким углом мы смотрим на сцену. Однако для того

Регистры фрагментных процессоров

В версии шейдеров	1.4	2.X	3.0
Константных регистров (с плавающей точкой)	8	32	224
Константных регистров (целочисленных)	–	16	16
Константных регистров (логических)	–	16	16
Временных регистров	2	12...32	32
Текстурные регистры/семплеры	4	16	16
Регистров цвета/текстурных координат	2	8	10
Предикаты	–	1	1
Управление выполнением (циклы, условия)	–	Есть	Есть
Выбор прямой/обратной стороны примитива	–	–	Есть

чтобы начать расчет нужных данных, мы должны перевести координаты вершины из базового пространства (единичной системы координат) в мировое пространство — эти вычисления обязательны, если мы хотим, чтобы сцена отображалась на экране с учетом камеры и координат в мировом (сценическом) пространстве. Это преобразование выполняется с помощью строчки: o.pos = mul(i.pos, WorldViewProjection). Данные о вершине попадают в функцию main() из потока входных данных — это так называемый однородный (uniform) ввод данных, которые попадают в процессор из локальной памяти видеокарты в регистры входных данных (Input registers). Данные, которыми ты влияешь на шейдер из своей управляющей программы, работающей на CPU, попадают в программу через константные регистры (варьируемый вход). Эти данные не могут быть изменены в коде шейдера, поэтому и называются константными. В нашем примере константные данные — различные матрицы (видовая и мировая), а также диффузный цвет вершины. Временные регистры используются для сохранения промежуточных результатов вычислений (в нашем случае — переменные nrm и e2v). Регистр предикатов содержит в себе флаг — некоторое условие, которое может быть задано заранее (извне шейдера) или рассчитано в программе. С помощью этого условия можно повлиять на ход выполнения программы (ветвления). В шейдерах второго поколения (2.0 и 2.X) возможны ветвления кода, причем в шейдерах версии 2.0 возможны только статические ветвления (циклы и подпрограммы), то есть проверка условий, заданных извне шейдера. В расширенных шейдерах 2.X возможно и динамическое управление ходом выполнения шейдера. Например, если рассчитанный диффузный цвет вершины красный, то следует делать одно, если красной компоненты не содержится — делать другое.

После того как ты выполнишь все необходимые вычисления с входными данными, настанет время заполнить структуру выходных данных. Она ничем не отличается от структуры входных данных, за исключением текстурных координат. Выходные данные передаются через выходные регистры. Подведем итоги: вершинный процессор получает вершину, переводит ее из базового пространства в пространство мира и камеры, подготавливает данные вершины (текстурные координаты) для последующей обработки фрагментным процессором.

фрагментный процессор После обработки вершинным процессором данные геометрии сцены складируются в промежуточном буфере готовых вершин. Из вершин формируются треугольники, а из треугольников — их фрагменты. С точки зрения программ, выполняемых GPU, этот процесс незаметен, поскольку на вход пиксельному шейдеру, завершающему обработку, попадают видимые фрагменты треугольников, но ты должен понимать все происходящее внутри видеокарты.

Три готовые вершины образуют треугольник. Треугольник разбивается на фрагменты. Эти фрагменты проверяются на видимость на экране с помощью Z-теста (проверка на видимость), и уже видимые фрагменты попадают на вход фрагментной программе. Почему я заостряю твое внимание на этом процессе? Здесь есть одна тонкость. Ранее ты рассчитывал текстурные координаты для вершин, а фрагментному процессору достаются текстурные координаты для фрагментов. Откуда они берутся? Интерполированием по всему треугольнику, между текстурными координатами трех его вершин.

Посмотри еще раз на код примера (XML узел PixelShader). Входные данные для фрагментного процессора аналогичны выходным данным вершинного процессора с учетом того, что они интерполированы для конкретного фрагмента внутри треугольника. На выходе из фрагментной программы мы должны получить результирующий цвет фрагмента с учетом текстур, освещения, затенения и т.д. (в зависимости от решаемой задачи). В константных регистрах процессора задаются константы из главной программы. Временные регистры служат для сохранения промежуточных результатов вычислений. Семплеры — это то же самое, что и текстуры. Регистры текстурных координат содержат интерполированные текстурные координаты для фрагмента. Регистр предикатов и, соответственно, статические и динамические ветвления появляются только в 2.X-версии пиксельных шейдеров. В последнем (третьем) поколении шейдеров появляется регистр стороны треугольника, который содержит положительное значение для передней стороны и отрицательное — для задней. Эта возможность используется для выбора различных схем расчета результирующего цвета (выбор освещения) для разных сторон треугольников.

Слева направо: Без шейдеров, шейдеры 1.X, шейдеры 3.0

Самый сильный физик планеты

будущее Гадание на кофейной гуще — чрезвычайно увлекательный процесс, именно им и занимаются многие аналитики IT-рынка. Их фантазии нельзя назвать никак по-другому. В пришествии технологии шейдеров на платформу PC нет ничего удивительного. Если в прошлом видеокарты выполняли фиксированный набор действий, то теперь этих действий стало на порядок больше за счет того, что появилась возможность писать управляющие программы (шейдеры) для графических процессоров. После резкого перехода от технологий фиксированного к технологиям программируемого графического конвейера индустрия продолжила свою эволюцию. Шейдеры обрастали мясом функциональности, новыми возможностями и регистрами, высокой частотой и возможностями распараллеливания вычислений. На сегодня мегамонстр PC-графики держит на своем борту 512 Мб памяти, работает на частоте 600 МГц, данные обрабатываются 24-мя пиксельными конвейерами. Также имеются решения на базе технологии SLI из двух видеокарт. Налицо тенденции к распараллеливанию вычислений. Почему существуют такие тенденции? Все очень просто, инженеры подходят к границам миниатюризации технологического процесса, преодолеть которые невозможно физически. Так что остается двигаться в сторону параллельных вычислений: многоядерные процессоры, многоканальные контроллеры памяти, мультиконвейерные графические решения. В ближайшем будущем нас ожидает новая операционная система от Microsoft, вместе с ней к нам придет новая архитектура графических драйверов и новый DirectX. Прыгнут ли графические технологии еще раз вперед? Вряд ли. Получим ли мы ощутимую прибавку к производительности и качеству? Наверняка. Спрашиваю, нужно ли нам это как пользователям? Вот здесь я сильно сомневаюсь, мне, к примеру, надоело менять видеокарту каждые два года, отваливая за нее от ста до двухсот зеленых бакинских комиссаров? А тебе? Если не надоело, то стоит прочитать вот это: **http://thg.ru/graphic/20060114/onepage.html**.

Регистры вершинных процессоров

В версии шейдеров	1.1	2.X	3.0
Инструкций	128	256	512*
Константных регистров (с плавающей точкой)	96**	256**	256**
Константных регистров (целочисленных)	–	16	16
Константных регистров (логических)	–	16	16
Регистров входных данных (атрибуты вершины)	16	16	16
Временных регистров	12	12***	32
Предикаты	–	1	1
Управление выполнением (циклы, условия)	–	Есть	Есть
Самплеры	–	–	4
Выходные регистры (текстурные координаты)	8	8	12

* – минимум инструкций, максимум для каждой карты определен в параметре D3DCAPS9.MaxVertexShader30InstructionSlots
** – минимум регистров, максимум задается для каждого устройства в D3DCAPS9.MaxVertexShaderConst
*** – минимум регистров, максимум задается для каждого устройства в D3DCAPS9.VS20Caps.NumTemps

в гостях у ньютона Графика — это не единственное направление, в котором наблюдается устойчивое появление все новых технологий на один геймерский доллар. Кроме шейдеров, в твой карман стремится залезть еще один ускоритель, на этот раз — ускоритель физики PPU (Physic Processing Unit). Итак, в начале марта 2005 года, когда все российские женщины готовились принимать очередную дозу дежурных комплиментов, молодая калифорнийская компания AGEIA объявила о разработке устройства, способного совершить революцию, аналогичную некогда триумфальному появлению Voodoo от компании 3DFx. По заверениям разработчиков, технология PhysX должна была взять на себя задачи по обсчету физики сцены, которые на тот момент решались CPU. Динамика твердых, гибких и жидких тел, столкновения, моделирование волос, меха и т.д. отныне будут осуществляться в процессоре PhysX. В отличие от современных игр, те игры, которые поддерживают технологии AGEIA, смогут отображать на экране не сотни объектов, подчиняющихся упрощенным законам физики, а десятки тысяч реальных объектов, полностью подчиняющихся всем законам физики. В новой эре будут сняты, казалось бы, монументальные ограничения на целостность (не разрушаемость) мира и ограниченность модели освещения и построения теней — PhysX может взять на себя и эти вычисления. Чип, состоящий из 125 млн. транзисторов, будет выпускаться по 130-нм техпроцессу в виде платы расширения с интерфейсом PCI или PCI-Express. Потребление энергии составит около 25 Вт. Кроме самого чипа, на борту платы будет располагаться 128 Мб 2.0 нс GDDR3-памяти, работающей на гигагерцовой частоте. В будущем планируется использование XDR-памяти.

AGEIA представляет собой небольшое конструкторское бюро в Калифорнии с офисами в Сент-Луисе и отделами разработчиков в Швеции, Швейцарии и Китае. Чипы будут производиться, по всей видимости, на производственных мощностях TSMC. Стоимость чипа составит от $100 до 400. В качестве API для железа будет выступать PhysX SDK, включающий в себя технологии Megon и NovodeX, купленных в 2005 году. Обещанный на Рождество 2005 года, выпуск плат был отложен на второй квартал 2006 года. По заявлениям официальных лиц AGEIA, такое решение было принято для того, чтобы плата не затерялась в толпе рождественских новинок. Однако не все так просто, как хотелось бы AGEIA. Сдвиг даты выпуска PhysX нужен для доработки платы в плане ее удешевления. Сейчас в Сети циркулируют слухи о том, что железка будет стоить $199. Однако главная причина кроется, по всей видимости, в немногочисленности игр, поддерживающих технологию AGEIA. Напомню, что PhysX SDK используется в Unreal Engine 3. 17 января 2006 года о поддержке этой технологии объявила компания Emergent Game Technologies (производители движка Gamebryo 3D). Таким образом, следующие тайтлы наверняка будут поддерживать PhysX: The Elder Scrolls IV: Oblivion, Loki, Warhammer MMORPG, City of Villains, Tom Clancy's Ghost Recon Advanced Warfighter. Эти и наверняка многие другие игры создадут солидную критическую массу игр, ускоряющихся железкой, и будут являться солидными аргументами для игроков в пользу приобретения PhysX.

сомнения Уже мало кто сомневается в триумфе AGEIA. Однако их конкурент, компания Havok, анонсировала свою технологию — Havok FX, которая использует для физических расчетов и ресурсы CPU, и ресурсы GPU третьей шейдерной модели. Производитель чипов Radeon, канадская компания ATI, также предлагает использовать пиксельные шейдеры 3.0 в видеокартах серии X1000 для произвольных вычислений, при этом текстуры используются для хранения данных, а фрагментный процессор — для их обработки. Для сравнения: процессор Pentium 4 3 ГГц имеет производительность порядка 12 GFLOP, а его системная шина — полосу пропускания порядка 6 Гб в секунду; фрагментный процессор ATI Radeon X1800XT — 120 GFLOP и, соответственно, 42 Гб в секунду. Тесты показали то же, что и предполагала теория. Производительность вычислений возросла в два-три раза при их переносе с CPU на GPU. Однако меня, например, одолевают все новые и новые сомнения, когда ATI надувает щеки и поигрывает бицепсами.

Все дело в стоимости решений от AGEIA и ATI. AGEIA предлагает железку не дороже чем за $400, а ATI — за $600. Кроме того, пользователям придется приобретать вторую такую же видеокарту. Я говорю о технологии ATI Crossfire — аналоге SLI решения от NVidia: в паре работают две карты, одна занимается только физическими расчетами, вторая — только графикой. Так что стоимость решения поднимается до $1200, что, согласись, не идет ни в какое сравнение с максимальными 400 у.е. от AGEIA. Однако вновь мои сомнения возрастают, когда в голову приходит мысль о том, нужны ли вообще современному рынку игр физические ускорители. Единственным удачным примером использования физики в игровом процессе является Half-Life 2. Причем с этой физикой (Havok 2) легко справляются современные процессоры от Intel и AMD. Обе компании, между прочим, также не дремлют и объявляют от том, что многоядерные процессоры будут лучше справляться с играми… Что же станут делать дизайнеры игр, если пользователь будет ждать от них полной разрушаемости игрового уровня? Куда денутся классические сюжетные преграды, если можно отыскать где-нибудь динамита и взорвать к чертовой матери дверь, не пропускающую по сюжету? Впрочем, ответов на многие вопросы, которыми будут развеяны мои сомнения, осталось ждать недолго — не позднее лета 2006. На это время запланировано с десяток разных «революций»: Longhorn, AGEIA PhysX, Sony PlayStation 3, Nintendo Revolution и, конечно, куча мегатайтлов, вроде Oblivion.

Странное совпадение, не правда ли?..

Без шейдеров картинка выглядит ужасно

Работа шейдеров 1.X в F.E.A.R

Работа шейдеров 2.X в F.E.A.R

Работа шейдеров 3.0, ParalaxMapping дает картинку с неровностями стен

мнение профессионалов

УСПЕШНОЙ МОЖЕТ СТАТЬ ЛЮБАЯ ИГРА

СПЕЦ: КАКИЕ ИГРЫ БОЛЕЕ УСПЕШНЫ: КВЕСТЫ, АРКАДЫ, СТРАТЕГИИ?.. ИЛИ ЖЕ ЖАНР ИГРЫ — ВТОРОСТЕПЕННОЕ? КАК СОЗДАТЬ ХИТ?

МИХАИЛ РАЗУМКИН: Самые успешные игры — те, при разработке которых четко учитывались нужды предполагаемой аудитории и в которых отсутствуют явные недоработки. Так из года в год работают Blazzads, шлифуя до блеска когда-то изобретенную идею Warcraft. Их игры очень дружественны к новичку, позволяют уже с первых минут почувствовать себя хозяином в виртуальном мире. Однако при этом они имеют и достаточную глубину, позволяющую бывалому геймеру проводить часы за компьютером, оттачивая мастерство. И серьезные графические навороты здесь не играют решающей роли. Красивым, но неиграбельным проектом повосторгаются, и потом его забудут.

Вообще во многом хитовость игры решает еще и такой фактор, как replayability, то есть то, насколько геймеру захочется поиграть еще раз. Игрой, точно попавшей в своего геймера, можно назвать The Sims, которая сделала миллионные продажи за счет незадействованной ранее женской аудитории. Конечно, есть и востребованные жанры. FPS, стратегии всегда популярнее, чем остальные виды игр. Зато квесты проще сделать.

Немалую роль играет и известность марки. Так, например, совершенно непотребная игра «Бой с тенью» разошлась у нас очень достойным тиражом. Похожая ситуация была и на Западе с Enter the Matrix — гораздо более качественной, но далеко не такой хорошей, как хотелось бы.

АЛЕКСАНДР ФЕДОРА: Успешнее то, что качественно сделано и вышло в свое время.

ЮРИЙ МАТВЕЕВ: Успешной может стать любая игра. В любом жанре. Но стать успешной игра может только если обладает какой-то своей, определенной изюминкой. Рецептов создания хитов не существует так же, как не существует рецептов создания хитовых книг, фильмов... Впрочем, в музыке уже существуют способы «сгенерировать» потенциальный хит, просчитав его заранее. Подобные попытки «просчитать успешность» того или иного произведения предпринимают и в других областях. Надо только сказать, что настоящими хитами становятся не те, что были просчитаны, как ни странно, а те, что были созданы с душой, под вдохновением... Под озарением свыше, необъяснимым даже для автора :).

СЕРГЕЙ АЗАРОВ: Очень хочется ответить, что жанр игры — это все второстепенное. Но реалии таковы, что спрос рождает предложение, а сейчас очень большой спрос на игры динамичных жанров с высокой степенью погружения в мир: шутеры и экшен-игры, причем зачастую с видом от первого лица. Конечно, это не означает, что если кто-то выпустит, скажем, высококачественный продукт в жанре RPG, то он провалится. Да, он продастся и, возможно, даже очень хорошо, однако это не будет означать, что начни разработчики выпускать пачками RPG-игры, все они тоже станут продаваться успешно. Сложно ответить на вопрос о том, как создать хит. Я, например, еще не создал хит :), поэтому и ответить четко не смогу. Могу только предположить, что для этого, как минимум, нужна слаженная команда профессионалов и четко поставленное руководство процессом разработки.

АНДРЕЙ БЕЛКИН: Наиболее успешны игры для домашних консолей — PlayStation2, Xbox, Xbox360, GameCube. Соответственно, жанры, характерные исключительно для PC, выражаясь интеллигентно, явно уступают. Этим занимаются, главным образом, квесты и стратегии, которых фактически нет на консолях.

Успешны экшены, шутеры от первого и третьего лица, action/adventure, платформеры, RPG и т.д. Создать хит не так уж сложно. В первую очередь нужна работоспособная студия: менеджер проекта, два-три геймдизайнера, пара концепт-художников, десяток программистов, десяток моделеров, три-четыре аниматора и звукорежиссер. Нужны деньги, чтобы в течение года-двух платить всем этим людям зарплату. Нужен издатель, который выпустит готовый продукт. Нужно знать, что будет популярно на игровом рынке через год-два (то есть тогда, когда игра будет готова). Ну и, конечно, нужно четкое понимание того, что ты хочешь сделать, и того, каким именно образом ты это будешь делать.

МИХАИЛ ПИСКУНОВ: Жанр игры, конечно, имеет значение. По крайней мере, есть «нехитовые» жанры — те же квесты, например. Но в хитовых жанрах конкуренция выше. Если у геймдизайнера есть что-то интересное, что он хочет сказать, то на его игру обязательно обратят внимание вне зависимости от жанра. Рецепт хита известен. Нужно сделать то, что хотят люди. Лучше, если желание людей угадано до того, как оно сформировалось в конкретный запрос, как было в случае с игрой The Sims.

АНДРЕЙ БЕЛКИН

РУКОВОДИТЕЛЬ ПРОЕКТА SWASHBUCKLERS: BLUE AND GREY. НА ДАННЫЙ МОМЕНТ ЗАНИМАЕТСЯ ОСВОЕНИЕМ КОНСОЛЬНОЙ ПЛАТФОРМЫ, НА КОТОРОЙ ОСНОВЫВАЕТСЯ ПРОЕКТ «ГОЛОВОРЕЗЫ». В ПРОШЛОМ СОЗДАТЕЛЬ СКАНДАЛЬНОЙ ИГРЫ «БОЙ С ТЕНЬЮ»

ОЛЬГА PACHELLA ПАК

СПЕЦИАЛИСТ ПО РАБОТЕ С ОБЩЕСТВЕННОСТЬЮ В СФЕРЕ КОМПЬЮТЕРНЫХ ИГР. НА ДАННЫЙ МОМЕНТ — PR-МЕНЕДЖЕР КОМПАНИИ «АКЕЛЛА». ПО СОВМЕСТИТЕЛЬСТВУ — КАПИТАН ЖЕНСКОЙ КИБЕРСПОРТИВНОЙ КОМАНДЫ ДЕЙСТВУЮЩИХ ЧЕМПИОНОК РОССИИ

МИХАИЛ CHSNARK ПИСКУНОВ

ВЕДУЩИЙ СЦЕНАРИСТ KDV GAMES. ЗА 15 ЛЕТ ИГРОСТРОИТЕЛЬНОГО (И ИГРОВОГО) СТАЖА УСПЕЛ ПОБЫВАТЬ В РОЛЯХ ПРОГРАММИСТА, 3D-МОДЕЛЕРА, ГЕЙМДИЗАЙНЕРА И ПРИНЯЛ УЧАСТИЕ В ПРОЕКТАХ «ВАНГЕРЫ», «ПЕРИМЕТР», «БРАТЬЯ ПИЛОТЫ 3D» И ДР.

МИХАИЛ РАЗУМКИН

ГЛАВНЫЙ РЕДАКТОР МУЛЬТИПЛАТФОРМЕННОГО ИГРОВОГО ЖУРНАЛА «СТРАНА ИГР». РАНЕЕ ВОЗГЛАВЛЯЛ НЕ МЕНЕЕ ИГРОВОЙ ЖУРНАЛ — «OFFICIAL PLAYSTATION РОССИЯ». РАЗРАБОТКОЙ ИГР НИКОГДА НЕ ЗАНИМАЛСЯ, НО В СИЛУ ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ КОЕ-ЧТО В НИХ ПОНИМАЕТ, ПОСКОЛЬКУ УВЛЕКАЕТСЯ ЭТИМ ДЕЛОМ УЖЕ ОКОЛО 20-ТИ ЛЕТ

СЕРГЕЙ АЗАРОВ

ВЕДУЩИЙ ГЕЙМДИЗАЙНЕР И И.О. РУКОВОДИТЕЛЯ НЕАНОНСИРОВАННОГО ПРОЕКТА В КОМПАНИИ STEP CREATIVE GROUP (**www.stepgames.ru**). РАНЕЕ РУКОВОДИЛ СТУДИЕЙ DESTINY GAMES, ГДЕ ТАКЖЕ ИСПОЛНЯЛ РОЛЬ РУКОВОДИТЕЛЯ ПРОЕКТА И ВЕДУЩЕГО ГЕЙМДИЗАЙНЕРА

ЮРИЙ МАТВЕЕВ

ОСНОВАТЕЛЬ И ГЕНЕРАЛЬНЫЙ ДИРЕКТОР КОМПАНИИ STEP CREATIVE GROUP (**www.stepgames.ru**). АВТОР ИГРЫ «ЗВЕЗДНОЕ НА-СЛЕДИЕ» И УЧАСТНИК РАЗРАБОТКИ РЯДА ДРУГИХ ИГР КОМПАНИИ В КАЧЕСТВЕ СЦЕНАРИСТА И ГЕЙМДИЗАЙНЕРА

ВОЛОДЯ КОРОТКОВ

РУКОВОДИТЕЛЬ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ В КОМПАНИИ «АКЕЛЛА»

РОМАН ЦОЙ

РУКОВОДИТЕЛЬ ПРОЕКТА «МОРСКОЙ ОХОТНИК»

ВАДИМ ГАЙДУКЕВИЧ

ВЕДУЩИЙ ПРОГРАММИСТ ПРОЕКТА «MASSIVE ASSAULT: РАСЦВЕТ ЛИГИ» (BY WARGAMING.NET)

АНДРЕЙ TERAN ТЕРТИЧНИКОВ

ПРОГРАММИСТ КОМПАНИИ MEDIAMOBILE В СОСТАВЕ GFI. ЗАНИМАЕТСЯ РАЗРАБОТКОЙ МОБИЛЬНЫХ ИГР. ХОББИ — ТРЕХМЕРНАЯ ГРАФИКА

ДМИТРИЙ ЖУКОВ: С тем же успехом можно спросить, какие фильмы самые успешные. У каждой игры есть своя аудитория, хотя, наверное, самые успешные игры — это простые незатейливые «Солитер» и «Сапер», которые стоят на каждом компьютере. В каждой области есть свои хиты — игры, оказавшие сильное влияние и вдохновившие кого-то на создание других игр. Как создать такой хит — не скажу, но, однозначно, надо предложить что-то новое.

АНДРЕЙ ТЕРТИЧНИКОВ: Основное отличие мобильных игр от «немобильных» в том, что человек играет в них в основном в те моменты, когда находится вне дома и ему нечем заняться. То есть он либо едет куда-то с места на место, либо сидит на каком-нибудь скучном совещании/лекции и т.д. Играют обычно урывками, поэтому игры аркадного типа пользуются большой популярностью. Когда играешь в них, не надо запоминать, что было раньше. Также определенные рамки накладывает управление: клавиатура компьютера или приставочный джойстик намного более удобны, чем телефонные клавиши, поэтому игры типа квестов мало распространены — так же, как и стратегии, хотя некоторые урезанные версии в плане управления все-таки есть. А вот создание хитов — процесс непредсказуемый, мало кто знает «секретную формулу» :).

СПЕЦ: ЧТО ВЕРНО: ЖЕЛЕЗО ЗАДАЕТ РАМКИ ИГРАМ ИЛИ ИГРЫ — ЖЕЛЕЗУ?

МИХАИЛ РАЗУМКИН: Естественно, железо. Именно с ним зачастую связан столь длительный срок разработки. Особенно это заметно в стане компьютерных игр, так как разработчикам нужно не только создать проект, удовлетворяющий самым современным требованиям, но и удовлетворить большинство геймеров, до сих пор играющих на устаревшем железе. А поскольку процесс разработки зачастую длится несколько лет, то возникают ситуации, когда графический движок приходится дотягивать до новых стандартов.

АЛЕКСАНДР ФЕДОРА: Железо развивается благодаря играм, но текущее поколение игр зависит от того железа, которое есть у большинства игроков. Без игр не развивались бы так уверено графические акселераторы, а про ускоритель физики вообще никто не услышал бы. Для повседневного использования вполне годится и компьютер трех-/пятилетней давности, но люди покупают новое железо. И 90% покупателей делают это исключительно для того, чтобы «шла новая игра».

ЮРИЙ МАТВЕЕВ: Первое. Это и доказывает наш многолетний опыт работы над различными проектами под разные платформы. Если посмотреть на наш недавний римейк — «Звездное наследие» (которое было перенесено с платформы ZX-Spectrum) — можно оценить разницу того, как влияет «железо» и, соответственно, его возможности на конечный вид игры, по сути, с одной и той же идеей...

СЕРГЕЙ АЗАРОВ: Оба утверждения имеют право на жизнь, поскольку одно вытекает из другого, и наоборот. Фактически, данный вопрос из разряда «Что было первым: курица или яйцо?». Хотя можно сме-

АЛЕКСАНДР ФЕДОРА

ПРОГРАММИСТ КОМПАНИИ STEP CREATIVE GROUP (**www.stepgames.ru**). ЗАНИМАЕТСЯ РАЗРАБОТКОЙ КОМПЬЮТЕРНЫХ ИГР. ПОСЛЕДНИЙ ЗАВЕРШЕННЫЙ ПРОЕКТ — «ЗВЕЗДНОЕ НАСЛЕДИЕ 1: ЧЕРНАЯ КОБРА»

ДМИТРИЙ ЖУКОВ

C++ DEVELOPER. СПЕЦИАЛИЗАЦИЯ — РАЗРАБОТКА САПР. В НАСТОЯЩЕЕ ВРЕМЯ РАБОТАЕТ В КОМПАНИИ PATHTRACE, ВЕЛИКОБРИТАНИЯ

СЕРГЕЙ ЗАГУРСКИЙ

ТЕХНИЧЕСКИЙ ДИРЕКТОР КОМПАНИИ MIST LAND — SOUTH В СОСТАВЕ GFI (**www.mistgames.ru**). В ГЕЙМДЕВЕ БОЛЕЕ ПЯТИ ЛЕТ. ПРИНИМАЛ УЧАСТИЕ В ПРОЕКТАХ: «КОД ДОСТУПА: РАЙ», «ВЛАСТЬ ЗАКОНА» И «АЛЬФА: АНТИТЕР-РОР»

ло сказать, что на данный момент железо, как правило, опережает игры по технологическому развитию, ибо появление новых возможностей в железе находит широкое применение через некоторый промежуток времени. Но и здесь есть свое исключение из правил: например, компания Crytek, которая уже анонсировала свой новый движок, требующий еще не вышедшего DirectX 10 и еще не реализованных в железе Shader Model 4.0.

ОЛЬГА ПАК: Все сильно взаимосвязано. В первую очередь, новые разработки в области компьютерных инноваций дают стимул и возможность создавать новые и все более совершенные игры. С другой стороны, спрос на новые игры и на их создание порождает спрос на развитие новых технологий, откуда и возникает предложение. Законы рыночной экономики действуют и здесь.

МИХАИЛ ПИСКУНОВ: Не зря же Windows называет игры и вообще программы приложениями, то есть не основным. Так что, в общем и целом, конечно, железо диктует то, какими должны быть игры. Для специальных платформ вроде сотовых телефонов или игровых приставок рамки совсем жесткие. А в случае PC возможен вариант, когда требования для игры объявляются после того, как она сделана, и в зависимости от того, что получилось.

ДМИТРИЙ ЖУКОВ: Однозначно, железо ограничивает возможности разработчиков — приходится идти на всяческие ухищрения и придумывать гениальные алгоритмы, о которых потом пишут в книгах. Ограниченные возможности железа — второй стимул к развитию новых алгоритмов после первого — лени.

СЕРГЕЙ ЗАГУРСКИЙ: С одной стороны, передовая (с технологической точки зрения) игра не может быть выпущена без соответствующей поддержки в железе. С другой — производители железа всегда рассчитывают на разработчиков игр, чтобы те поддержали определенные нововведения в аппаратной части. Безусловно, высокотехнологичные игры ограничиваются железом. Но в то же время железки, как правило, не выходят до тех пор, пока какие-либо игры не будут использовать их аппаратные возможности. Так что этот вопрос из разряда «Что появилось раньше: софт или железо?».

АНДРЕЙ ТЕРТИЧНИКОВ: Мобильные игры существенно ограничены аппаратными возможностями телефонов, поэтому приходится очень сильно ужиматься как в плане графики, так и в плане сложности уровней и самой игры. Да и то, что делать игры приходится с оглядкой на самые слабые модели телефонов (ведь для них тоже надо делать специальные версии игры, пускай и урезанные, но они должны сохранить геймплей оригинала), тоже накладывает свои ограничения.

СПЕЦ: ПОЧЕМУ ПРАКТИЧЕСКИ ВСЕГДА ПРИ ПОКУПКЕ НОВОЙ ИГРЫ МЫ ВЫНУЖДЕНЫ ДЕЛАТЬ КОНФИГУРАЦИЮ КОМПЬЮТЕРА ВСЕ БОЛЕЕ НАВОРОЧЕННОЙ? ИНОГДА, ЧТОБЫ ПОИГРАТЬ В ИГРУ, НУЖНО НЕ ТОЛЬКО КУПИТЬ ДИСК, НО И СДЕЛАТЬ АПГРЕЙД КОМПЬЮТЕРА. ПОЛУЧАЕТСЯ ЗАТРАТНО...

МИХАИЛ РАЗУМКИН: Это не совсем так. Точнее, это бывает верно только в случае самых топовых проектов, а их, как правило, единицы. Например, тот же Warcraft III имел весьма умеренные аппетиты. А вот про монстров вроде Battlefield 2 или F.E.A.R. такого не скажешь. Чем более грандиозные планы ставит перед собой разработчик, тем более мощное железо необходимо для их реализации. Всем так нравится естественная физика последних шутеров и автосимуляторов, но для ее просчета нужен весьма шустрый процессор. То же касается и графики. Новые технологии позволяют добиваться все более реалистичной картинки, однако, чтобы она еще и двигалась :), понадобится последнее слово техники в области видеокарт. Но рука производителей железа в этом тоже чувствуется. Вряд ли кто признается, но ведь ни для кого не секрет, что та же Half-Life быстрее бегает на radeon'ax, а Doom — на geforce'ax...

АЛЕКСАНДР ФЕДОРА: Потому что игры — это ПО, которое выжимает из компьютера 110% его возможностей. Всегда хочется сделать игру такой, какой еще нет. В то же время некоторые разработчики улучшают графику и физику, пренебрегая при этом геймплеем, — и это очень плохая тенденция. В какой-то степени этому способствуют и сами игроки, многие из которых встречают игры, что называется, «по одежке», отдавая в первую очередь предпочтение наиболее навороченной графике, но не геймплею, который сложно передается статичными скриншотами на обложке диска. По этой причине большинство игр разрабатывается для продвинутой конфигурации, а на обычном железе от полноценной игры остается только обрубок, лишь отдаленно соответствующий задумке авторов проекта. В худшем случае — теряется атмосфера игры.

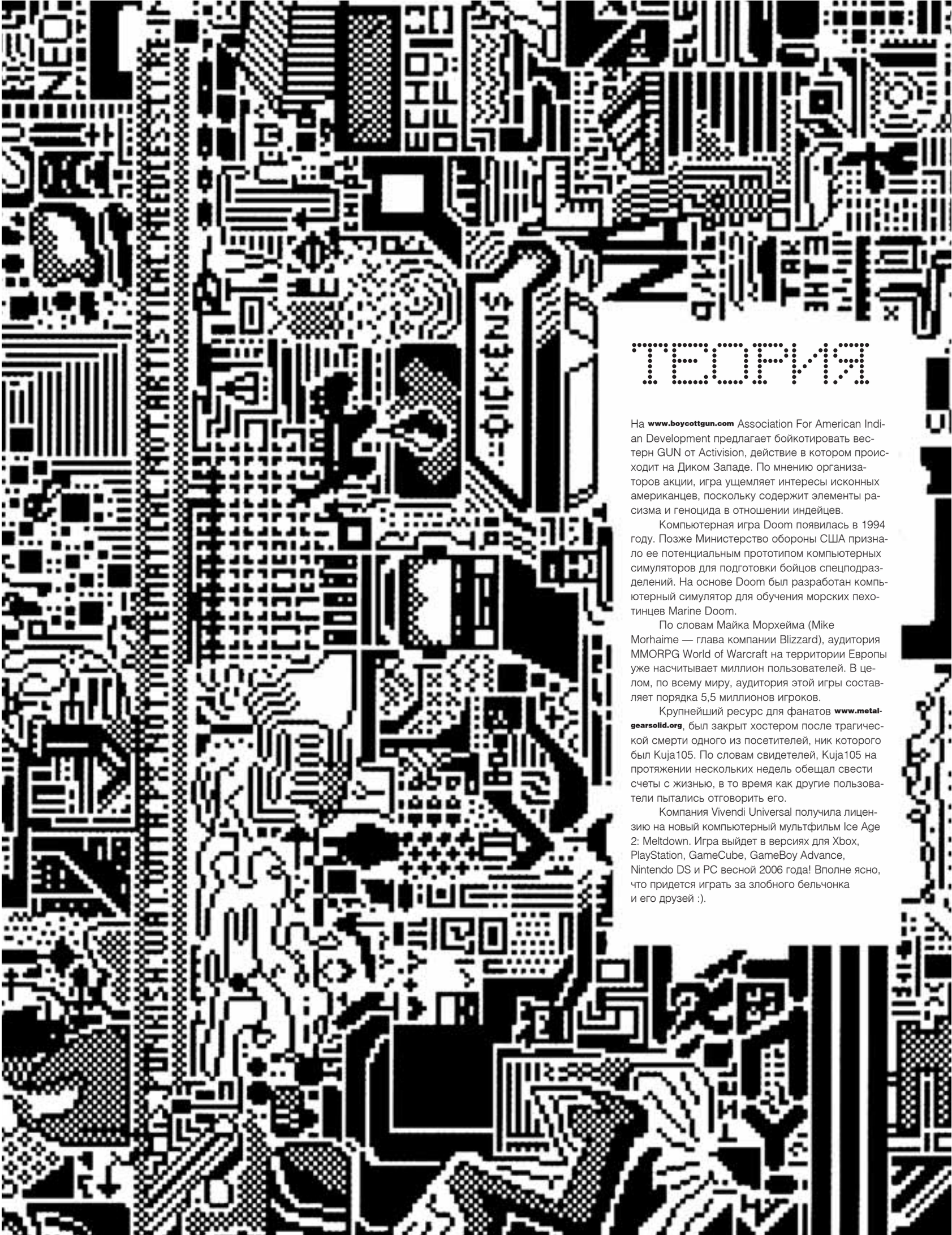
ЮРИЙ МАТВЕЕВ: Прогресс не остановить (это если кратко). В остальном — придется прочесть целую лекцию по истории развития интерактивных развлечений :).

СЕРГЕЙ АЗАРОВ: «Спрос рождает предложение» — известная экономическая истина. Если игрок хочет увидеть все прелести технологического прорыва в области игростроения, то ему неизбежно придется выложить кругленькую сумму, как минимум, на видеокарту. Но есть и другая сторона медали: нужно уметь собирать компьютеры, чтобы их хватило хотя бы на один-два года. Немудрено, если вдруг выяснится, что разработчики железа находятся в тайном заговоре с ключевыми фигурами игроиндустрии.

ОЛЬГА ПАК: Мир не стоит на месте, и постоянное обновление никому не помешает. Учитывая запросы наших потребителей (сейчас уже никого не удивишь появлением на экране 3D-модели), стараемся создавать все более «навороченные» игры, ведь народ именно этого хочет. Но нельзя сделать прорыв в игровой индустрии без сопутствующего прорыва в технической сфере. И если шагать в ногу со временем, то стоит почаще совершенствовать свой домашний компьютер.

ДМИТРИЙ ЖУКОВ: Компьютерная индустрия вообще быстро развивается. Любые программы со временем требуют больше и больше — просто игровая индустрия в данном случае самая жадная. Конкуренция высокая, поэтому надо предлагать что-то новое, интересное и увлекательное. Платит за это пользователь — разумеется, делая постоянные апгрейды. Наверное, это можно сравнить с производством фильмов: эффекты становятся все интересней, а бюджет картин постоянно растет.

СЕРГЕЙ ЗАГУРСКИЙ: Дело в том, что практически каждая новая игра в какой-то мере раздвигает технологические горизонты, охваченные уже существующими играми, вводя новые масштабы игровых локаций, высокую детализацию персонажей, качественный искусственный интеллект, массовость игрового действия



ТЕОРИЯ

На **www.boycottgun.com** Association For American Indian Development предлагает бойкотировать вестерн GUN от Activision, действие в котором происходит на Диком Западе. По мнению организаторов акции, игра ущемляет интересы исконных американцев, поскольку содержит элементы расизма и геноцида в отношении индейцев.

Компьютерная игра Doom появилась в 1994 году. Позже Министерство обороны США признало ее потенциальным прототипом компьютерных симуляторов для подготовки бойцов спецподразделений. На основе Doom был разработан компьютерный симулятор для обучения морских пехотинцев Marine Doom.

По словам Майка Морхейма (Mike Morhaime — глава компании Blizzard), аудитория MMORPG World of Warcraft на территории Европы уже насчитывает миллион пользователей. В целом, по всему миру, аудитория этой игры составляет порядка 5,5 миллионов игроков.

Крупнейший ресурс для фанатов **www.metalgearsolid.org**, был закрыт хостером после трагической смерти одного из посетителей, ник которого был Kuja105. По словам свидетелей, Kuja105 на протяжении нескольких недель обещал свести счеты с жизнью, в то время как другие пользователи пытались отговорить его.

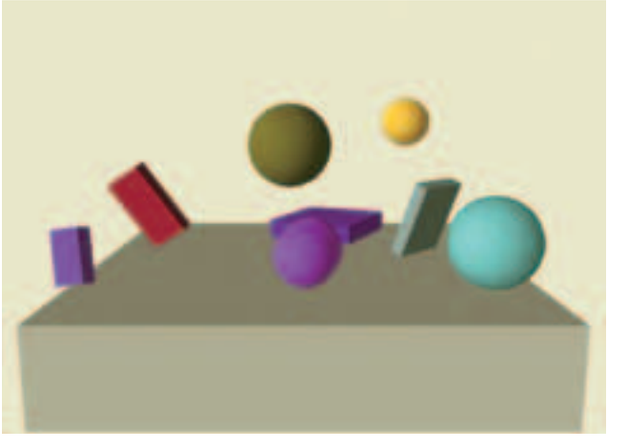
Компания Vivendi Universal получила лицензию на новый компьютерный мультфильм Ice Age 2: Meltdown. Игра выйдет в версиях для Xbox, PlayStation, GameCube, GameBoy Advance, Nintendo DS и PC весной 2006 года! Вполне ясно, что придется играть за злобного бельчонка и его друзей :).

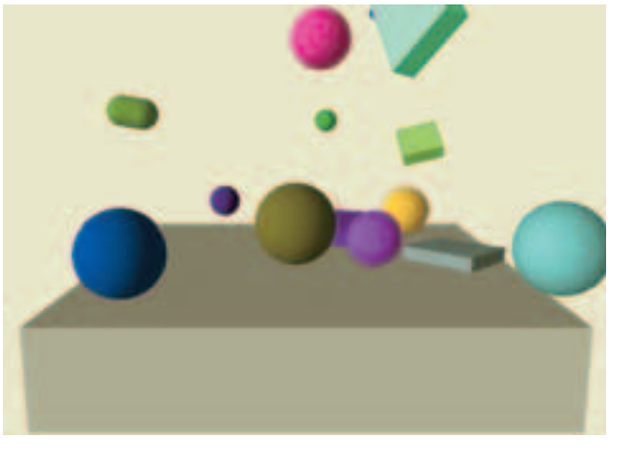
физика в играх

ВОЗМОЖНОСТИ СУЩЕСТВУЮЩИХ ДВИЖКОВ

НЕ ИМЕЕТ СМЫСЛА МНОГО ГОВОРИТЬ О ТОМ, ЧТО В СОВРЕМЕННЫХ ИГРАХ ОЧЕНЬ ЧАСТО ИСПОЛЬЗУЮТСЯ ФИЗИЧЕСКИЕ СИМУЛЯЦИИ РАЗЛИЧНОГО РОДА. ФИЗИКА УЖЕ ДАВНО (И НАДОЛГО) ПРОТОПТАЛА ДОРОЖКУ В СИМУЛЯТОРЫ, В ОСНОВНОМ АВТОМОБИЛЬНЫЕ |

ЕВГЕНИЙ КОРНЮШЕНКО WWW.STEPGAMES.RU





немного терминологии Физика используется и как основной элемент геймплея (хардкорные симуляторы), и как средства к получению дополнительных эмоций игроком (различного рода разрушения). Также есть ряд жанров, в которых массовое применение физики началось сравнительно недавно — это прежде всего шутеры и игры с видом от третьего лица. Спектр использования физических эффектов в них очень широк: от достаточно простых rag-dolls и пинания ящиков до попыток завязать часть геймплея на физику (Half-life 2, Psi-ops и т.д.). Кроме всего перечисленного, существует ряд игровых жанров, в которых физика не нужна в принципе, к примеру в пошаговых глобальных стратегиях. В целом на сегодня физика в играх является важнейшим элементом, но несмотря на это раскручена не так, как графика.

Физический движок (фд)

ФД — библиотека, которая рассчитывает физические взаимодействия между объектами игрового мира (симулируется физика, описываемая законами Ньютона). Физические движки, используемые при разработке игр, как правило, не симулируют физические процессы игрового мира со 100% точностью, а лишь производят достаточно точную аппроксимацию физических законов. Современные игровые физические движки состоят из двух частей: подсистемы определения столкновений и подсистемы расчета физических взаимодействий.

Подсистема определения столкновений

Основные два параметра подсистемы определения столкновений: скорость работы и точность определения столкновений. Недостаточная точность приводит к появлению разных артефактов, таких как перекрытие объектов, неопределение столкновений при существенно разных размерах и скоростях объектов и т.д. Для ускорения работы подсистемы столкновений используют различного рода разбиения пространства на подпространства, такие как quadtree (рекурсивное деление пространства на четыре подпространства) или octree (деление на восемь подпространств), для снижения количества проверок столкновений. Системы столкновений работают дискретно — столкновения рассчитываются через определенные промежутки времени. Итак, такого рода системы могут приводить к тому, что столкновения с участием быстро движущихся объектов не фиксируются — для борьбы с подобного рода артефактами некоторые системы столкновений поддерживают так называемый continuous collision detection (CCD)

Таблица рентабельности

преимущества	PhysX	Havok	TrueAxis	ODE	Tokamak	Newton
Стоит ли денег	да	да	да	нет	нет	нет
Есть ли CCD	да	да	да	нет	нет	да
Встроенная поддержка rag-dolls	нет	да	нет	нет	нет	да
Встроенная поддержка автомобилей	да	да	да	нет	нет	да
Дополнительная поддержка персонажей	да	да	нет	нет	нет	нет
Поддержка многопоточности	да	да	нет	нет	нет	нет
Симуляция жидкостей	да	да	нет	нет	нет	нет

(системы непрерывного отслеживания столкновений). Суть метода continuous collision detection заключается в том, что проверка столкновений между двумя объектами производится не между ними самими в дискретные моменты времени, а между вытянутыми объемами, которые представляют движение объектов в течение всего временного шага.

Подсистема симуляции (пс)

ПС, помимо скорости работы, характеризуется таким параметром, как стабильность симуляции. Этот параметр влияет непосредственно на достоверность самой физической симуляции: если симуляция нестабильна, видны разные артефакты, например подергивающиеся объекты. Симуляция дискретна, то есть программист извне задает шаг времени, который необходимо рассчитать. От размера этого шага и зависит стабильность симуляции. Соответственно, чем больший размер шага позволяет устанавливать движок, тем лучше.

Подавляющее большинство физических движков может (с различным успехом) симулировать физику твердого тела (Rigid body simulation). Твердое тело — это тело, которое не меняет свою форму (например кирпич, стол, стена и т.д.). На данный момент подавляющее большинство игр использует именно физику твердого тела, главным образом потому, что с приемлемой производительностью может обсчитываться лишь физика твердого тела. Для представления объема твердых тел, в зависимости от движка, могут использоваться как различные примитивные тела (прямоугольники, сферы, цилиндры, конусы и т.д.), так и более сложные (карты высот, выпуклые многогранники или невыпуклые многогранники). Если используются только примитивные тела, более сложные тела описываются с помощью аппроксимации примитивами. Для описания свойств твердых тел используется понятие материала, который описывается параметрами: коэффициент трения (может быть два: коэффициент трения покоя, который показывает, как тяжело сдвинуть тело, и коэффициент трения движения, который показывает, как тяжело удерживать тело в движении), упругость (сколько энергии остается после столкновения с другим телом). Помимо этих параметров, могут быть и другие. Твердое тело также имеет массу. Движение твердых тел описывается при помощи линейной, угловой скорости и ускорения. Хотя движки позволяют устанавливать эти параметры непосредственно, воздействия на тела, как правило, осуществляют при помощи приложения либо физических сил (влияют на ускорения тел), либо импульсов (влияют на скорости).

На базе физики твердого тела реализуется также физическое поведение персонажей и широко известный эффект тряпичной куклы (rag-doll), который заключается в том, что персонаж (чаще всего мертвый) падает вниз, как тряпичная кукла. Для реализации подобного рода поведения только твердых тел недостаточно. Используются так называемые сочленения (joint) или ограничения (constraint).

Джоинт — это точка, которая соединяет два твердых тела (соединение обычно задается относительно точки джоинта) и накладывает ограничения на положение тел в пространстве друг относительно друга или на скорости тел относительно друг друга. Типов джоинтов много (некоторые движки позволяют писать собственные). Для реализации рэгдолов используют ball-joint и hinge-joint. Ball-joint — это шарнирное соединение двух тел, оно ограничивает перемещение тел друг относительно друга и налагает ограничение на то, как повернуты тела относительно друг друга по трем осям (при помощи джоинтов такого типа в регдолле крепится плечо или голова к телу). Hinge-joint — это петельное соединение двух тел, которое ограничивает повороты тел относительно друг друга одной осью. В целом же спектр применения джоинтов очень широк и не ограничивается только созданием rag-dolls.

Помимо физики твердого тела, различные физические движки могут реализовывать дополнительные возможности: специальную поддержку симуляции движения автомобилей, симуляцию воды и прочих жидкостей, симуляцию тканей и одежды, симуляцию частиц, дополнительную поддержку для симуляции персонажей — высокоуровневые контроллеры персонажей, встроенную поддержку rag-dolls, поддержку анимации и т.д.

физика и разработка

Наличие физики в игре накладывает отпечаток на весь процесс разработки. В зависимости от того, каков объем и характер физических фич, могут быть затронуты практически все аспекты производственного процесса — от геймплея до контента.

Как делать физику — этот вопрос требует решения. На одном полюсе — решение о том, чтобы писать физику самостоятельно. На другом полюсе — покупка готового решения, которое полностью покроет спектр поставленных задач. Где-то посредине находится такой вариант: модифицировать существующий, но далеко не на 100% подходящий движок. Как и в остальных областях разработки ПО, данное решение зависит от множества факторов: наличия средств, наличия персонала, наличия подходящих решений и т.д.

В зависимости от выбора и требований к физике сложность разработки может значительно колебаться. Если же рассматривать одну и ту же игру с физикой и без нее, при прочих равных, то вариант с физикой (достаточно объемной) затронет всех участников производственного процесса. Наличие физики прежде всего влияет на архитектуру кода игры: введение такой сложной подсистемы влияет и на графику, и на искусственный интеллект, и даже на ввод пользователя.

Помимо архитектурных моментов, в команде разработчиков появляется новая роль — программист физики, который будет вынужден заниматься вопросами реализации воплощения физических фич (эта доля тяжелая и неблагодарная). Наличие физики повлияет и на геймдизайн. С одной стороны, можно будет добавлять различные физические геймлей-фичи, с другой стороны, появится дополнительная забота о том, как физика скажется на остальном геймплее. Также физика затрагивает и производство контента для игры: придется создавать физические модели объектов, настраивать материалы, rag-dolls и джоинты.

обзор существующих решений

До недавнего времени ситуация на рынке физических движков была довольно стабильна. Существовало несколько бесплатных физических движков (разного качества), а также ряд коммерческих физических движков. Однако в ушедшем 2005 году компания AGEIA анонсировала первый в истории игровой индустрии ускоритель физики, который снимает с центрального процессора задачу расчета физики. Ускоритель пока еще не появился в продаже, но его создатели утверждают, что их продукт будет способен обсчитывать порядка 40 000 объектов (на данный момент количество физических объектов, обсчитываемых на ПК, составляет десятки).

Ниже рассмотрим основные физические движки, которые существуют на сегодня. Помимо чисто физических движков, существует масса игровых движков, которые содержат физические подсистемы, написанные поверх одной из библиотек (на **www.devmaster.net/engines** можно ознакомиться с игровыми движками).

PhysX (www. ageia.com, коммерческий)

Эта физическая технология предоставляется компанией AGEIA и как физический движок, и как АПИ для работы с их физическим ускорителем. На данный момент ряд игровых компаний заявили о поддержке этой технологии в своих продуктах (в том числе Unreal Engine).

возможности:

— СИМУЛЯЦИЯ ТВЕРДЫХ ТЕЛ — В КАЧЕСТВЕ ФИЗИЧЕСКОГО ПРЕДСТАВЛЕНИЯ МОГУТ ИСПОЛЬЗОВАТЬСЯ ПРИМИТИВЫ (ПЛОСКОСТИ, БОКСЫ, КАПСУЛЫ) И ВЫПУКЛЫЕ МНОГОУГОЛЬНИКИ, А ТАКЖЕ ИХ КОМБИНАЦИИ;

> «ДОБАВЛЕНИЕ ФИЗИКИ В ИГРУ УСЛОЖНЯЕТ ПРОЦЕСС РАЗРАБОТКИ ПРОДУКТА И ТРЕБУЕТ ДОПОЛНИТЕЛЬНЫХ РЕСУРСОВ»

- ПОЛНОСТЬЮ НАСТРАИВАЕМЫЕ ДЖОИНТЫ С ШЕСТЬЮ СТЕПЕНЯМИ СВОБОДЫ;
- СИМУЛЯЦИЯ ЖИДКОСТЕЙ;
- ФИЗИЧЕСКОЕ ВЗАИМОДЕЙСТВИЕ МЕЖДУ ЖИДКОСТЯМИ И ТВЕРДЫМИ ТЕЛАМИ;
- КОНТРОЛЛЕРЫ ПЕРСОНАЖЕЙ (ПЕРСОНАЖ МОЖЕТ СОСТОЯТЬ ИЗ БОКСОВ И КАПСУЛ), ПОЗВОЛЯЮЩИЕ АВТОМАТИЧЕСКИ ПЕРЕМЕЩАТЬСЯ ПО ЛЕСТНИЦАМ;
- АВТОМОБИЛИ НА ОСНОВЕ ТРЕЙСИНГА ЛУЧА;
- ПОДДЕРЖКА НЕСКОЛЬКИХ СЦЕН;
- СИСТЕМА СТОЛКНОВЕНИЯ ПОДДЕРЖИВАЕТ НЕПРЕРЫВНОЕ ОТСЛЕЖИВАНИЕ СТОЛКНОВЕНИЙ;
- МУЛЬТИПЛАТФОРМЕННОСТЬ;
- МНОГОПОТОЧНОСТЬ.

Havok (www.havok.com, коммерческий)

Один из старейших физических движков. На нем сделаны десятки игр (посмотреть их список можно по адресу **www.havok.com/content/blogcategory/29/73**). Недавно Havok анонсировал новую технологию расчета физики Havok FX, которая производит расчеты физики на видеокартах используя шейдеры 3.0.

возможности:

- ФИЗИКА ТВЕРДОГО ТЕЛА;
- ДЖОИНТЫ;
- СИСТЕМА СТОЛКНОВЕНИЯ ПОДДЕРЖИВАЕТ НЕПРЕРЫВНОЕ ОТСЛЕЖИВАНИЕ СТОЛКНОВЕНИЙ;
- ПОДДЕРЖКА СИМУЛЯЦИИ АВТОМОБИЛЕЙ;
- RAG-DOLLS;
- КОНТРОЛЛЕРЫ ПЕРСОНАЖЕЙ (ПОЗВОЛЯЮТ ПЕРСОНАЖАМ ПЕРЕМЕЩАТЬСЯ ПО ЛЕСТНИЦАМ);
- МНОГОПОТОЧНОСТЬ.

Trueaxis (www.trueaxis.com)

Бесплатный для некоммерческого использования. Исходники закрыты.

возможности:

- ФИЗИКА ТВЕРДОГО ТЕЛА;
- СИСТЕМА СТОЛКНОВЕНИЯ ПОДДЕРЖИВАЕТ НЕПРЕРЫВНОЕ ОТСЛЕЖИВАНИЕ СТОЛКНОВЕНИЙ;
- В КАЧЕСТВЕ ПРЕДСТАВЛЕНИЯ МОГУТ ИСПОЛЬЗОВАТЬСЯ ВЫПУКЛЫЕ МНОГОУГОЛЬНИКИ, КАПСУЛЫ, ЦИЛИНДРЫ И СФЕРЫ;
- ПОДДЕРЖКА СИМУЛЯЦИИ АВТОМОБИЛЕЙ;
- ДЖОИНТЫ.

ODE (www.ode.org, BSD, исходники открыты)

Единственный в списке движок с открытыми исходниками, что позволяет ему служить в качестве базы для построения собственного физического движка (небезызвестный проект S.T.A.L.K.E.R. использует модифицированный ODE), а также просто для изучения того, как «оно все внутри устроено». ODE очень часто используется различными игровыми движками в качестве физической подсистемы.

возможности:

- ФИЗИКА ТВЕРДОГО ТЕЛА;
- ДЖОИНТЫ;
- СИСТЕМА СТОЛКНОВЕНИЙ ОТДЕЛЕНА ОТ ФИЗИЧЕСКОЙ СИМУЛЯЦИИ, ЧТО ПОЗВОЛЯЕТ ИНТЕГРИРОВАТЬ ODE С РАЗНЫМИ СИСТЕМАМИ СТОЛКНОВЕНИЙ.

Tokamak (www.tokamakphysics.com, бесплатный, исходники закрыты)

возможности:

- ФИЗИКА ТВЕРДОГО ТЕЛА;
- ДЖОИНТЫ (ВСЕГО ДВА ВИДА: BALL И HINGE — ДОСТАТОЧНО ДЛЯ ПОСТРОЕНИЯ RAG-DOLLS).

Newton (www.physicsengine.com, бесплатный, исходники закрыты)

возможности:

- СИСТЕМА СТОЛКНОВЕНИЯ ПОДДЕРЖИВАЕТ НЕПРЕРЫВНОЕ ОТСЛЕЖИВАНИЕ СТОЛКНОВЕНИЙ;
- ФИЗИКА ТВЕРДОГО ТЕЛА;
- RAG-DOLLS;
- ПОДДЕРЖКА СИМУЛЯЦИИ АВТОМОБИЛЕЙ.

физика на примере

Для практического примера возьмем демонстрационную программу из OPAL (open physics abstraction layer: **http://ox.slug.louisville.edu/opal/wik**i) — это открытый физический движок, предоставляющий единый высокоуровневый интерфейс для работы с другими физическими движками. На данный момент существует единственная реализация поверх ODE, в разработке находится оболочка над TrueAxis. Для работы необходимо скачать сам движок — **http://prdownloads.sourceforge.net/opal/opal-0.3.1-src.zip?download.**

Нас интересует дема, которая находится в папке opal-0.3.1-src\samples\simple\ и показывает базовые аспекты работы с физическим движком OPAL, демонстрируя работу с твердыми телами. В качестве визуализатора используется библиотека SDL (**www.libsdl.org/index.php**). Приложение состоит из единственного файла main.cpp и нескольких h-ков. При старте появляется статический бокс. При нажатии клавиши «пробел» на этот бокс падают твердые тела, представляемые при помощи различных примитивов. Рассмотрю файл main.cpp и прокомментирую те его части, которые относятся к физической симуляции.

набор глобальных переменных

```
std::vector<opalSamples::Entity*> gEntities;
opal::Simulator* gSimulator = NULL;
```

Переменная gSimulator типа opal::Simulator указывает на экземпляр физического движка OPAL. Тип opal::Simulator — собственно, и есть сам физический движок. Он инкапсулирует внутри систему расчета столкновений и подсистему расчета физики. Как правило, в приложении достаточно одного экземпляра физического движка. Подобного рода архитектура, при которой существует некий объект, содержащий информацию обо всех физических объектах игрового мира, и производит обсчет физики и используется практически во всех высокоуровневых физических движках.

Переменная gEntities содержит список сущностей типа opalSamples::Entity, которые имеют физическое представление. Сам класс и его наследники мало интересны, так как они в данной программе служат для отрисовки симулируемых объектов. В самом классе opalSamples::Entity нас интересует одно поле — opal::Solid* mSolid. Тип opal::Solid представляет твердое тело в физическом движке OPAL. Твердое тело обладает позицией, скоростью, ускорением и является базовой единицей симуляции — такого рода сущность есть в любом физическом движке. В данной программе для рисования сущности необходимо знать позицию твердого тела — эта информация получается путем вызова метода getTransform у mSolid.

содержимое функции main

```
gSimulator = opal::createSimulator();
gSimulator->setGravity(opal::Vec3r(0, (opal::real)-9.81, 0));
```

В первой строчке создаем экземпляр симулятора при помощи функции opal::createSimulator(). Во второй строчке устанавливаем силу гравитации для этого симулятора при помощи метода setGravity и передачи в качестве параметра вектора силы тяжести. Заметь, что можно задать гравитацию, отличную от земной, или установить ее равной нулю. С точки зрения физической симуляции, гравитация — это всего лишь одна из сил, действующих на тело, подобного рода метод применяют из соображений удобства и эффективности, так как, как правило, в симулируемом мире гравитация присутствует.

```
1. opal::Solid* platformSolid = gSimulator->createSolid();
2. platformSolid->setStatic(true);
3. opal::BoxShapeData boxShape;
4. boxShape.dimensions = gGroundDimensions;
5. platformSolid->addShape(boxShape);
```

В приведенном коде создаем объект в виде прямоугольной коробки, он служит в землей. В первой строчке создаем твердое тело в нашем симуляторе (это твердое тело будет обрабатываться именно данным симулятором и никаким другим).

Во второй строчке делаем тело статическим, то есть оно не может менять свою позицию (обладает бесконечной массой), но может взаимодействовать с другими физическими телами. В играх такие статические тела используются для симуляции неподвижных объектов локаций, таких как земля, стены и т.д.

В третьей строчке создаем экземпляр структуры boxShape, которая описывает трехмерный прямоугольник (бокс).

В четвертой строчке устанавливаем его размеры по трем осям, а в пятой — устанавливаем этот прямоугольник в качестве объема для нашей земли, вызвав метод addShape. Можно добавлять сколько угодно разных фигур (Shapes), описывая таким образом сложный объем. В OPAL, помимо боксов, в качестве объема можно использовать сферы, цилиндры, плоскости и многоугольники. Все фигуры имеют, помимо специфичных для каждой фигуры параметров (как dimensions в нашем примере), ряд общих полей:

```
Matrix44r offset;
Material material;
```

Поле offset задает смещение фигуры относительно центра твердого тела, поле material задает свойства материала фигуры. В OPAL'е материал имеет следующие свойства:

hardness — от 0 до 1. Показывает, насколько допустимо перекрытие объемов тел.
friction — от 0 до 1, трение движения. Чем больше это значение, тем быстрее будет останавливаться движущееся тело.
bounciness — от 0 до 1, упругость. Чем больше значение, тем больше энергии будет поглощаться при соударении.
density — плотность материала. На основании этого параметра и объема фигуры вычисляется масса фигуры.

основной цикл программы (вернее, те его части, которые представляют интерес в контексте рассматриваемого вопроса)

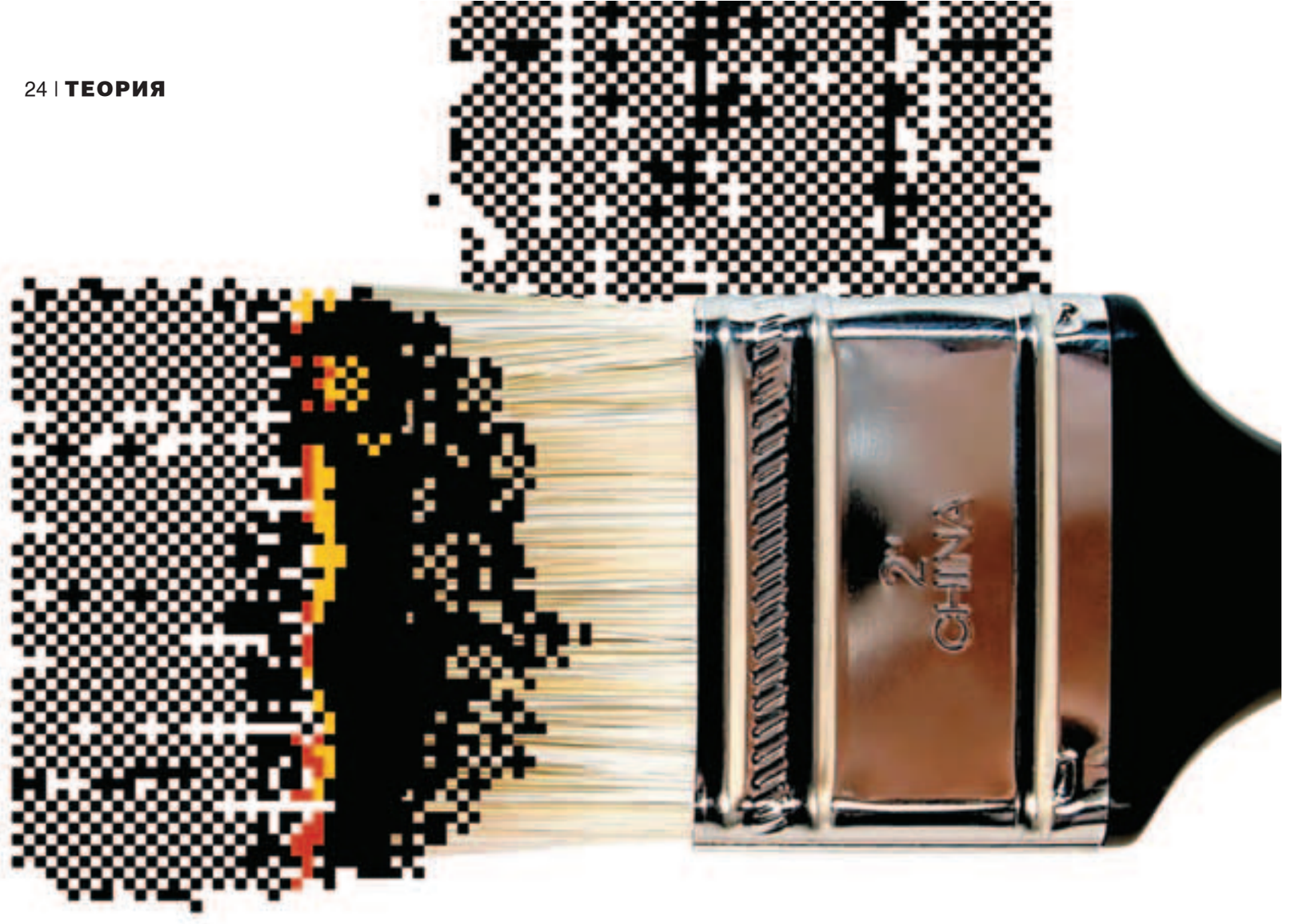
```
while (!gQuit)
{
  opal::real dt = (opal::real)timer.getElapsedSeconds();
  gQuit = processInput();
  if (!gPaused)
  {
    gSimulator->simulate(dt);
  }
}
```

Нам интересна только последняя строчка. В метод simulate передаем единственный параметр — время, которое прошло в «физическом мире». Именно внутри этого вызова происходит расчет столкновений и расчет взаимодействия физических объектов, происходит обновление позиций, скоростей и т.д.

```
gSimulator->destroy();
```

Заключительная строчка программы вызывает метод destroy объекта-симулятора, тем самым освобождая все ресурсы, занятые им, и уничтожая все физические объекты, созданные через методы симулятора, подобные createSolid.

В примере я рассмотрел весьма простое приложение. В реальной игре больше задач, эти задачи сложнее. Тем не менее, даже если судить по этому приложению, можно сказать, что на сегодня не нужно быть специалистом по высшей математике и физике только для того, чтобы писать приложения, включающие в себя физические симуляции



графоманские улучшения

ОПТИМИЗАЦИЯ ИГРОВОЙ ГРАФИКИ

ОБОЖАЮ ОПТИМИЗАЦИЮ: ВО ВРЕМЯ ПРОФИЛИРОВАНИЯ КОДА ПРИХОДИТСЯ СЕРЬЕЗНО НАПРЯГАТЬ МОЗГИ И ИСКАТЬ ИНТЕРЕСНЫЕ РЕШЕНИЯ. ПРИ ПРОГРАММИРОВАНИИ ИГР МНЕ ДОСТАВЛЯЕТ БОЛЬШЕЕ УДОВОЛЬСТВИЕ СОЗДАНИЕ ЭФФЕКТОВ И ПРОЦЕСС ОПТИМИЗАЦИИ. МОЖЕТ БЫТЬ, ПОЭТОМУ Я ЕЩЕ НЕ СОЗДАЛ НИ ОДНОЙ ИГРЫ ;) | **ФЛЕНОВ МИХАИЛ АКА HORRIFIC (WWW.VR-ONLINE.RU)**

Что будем рассматривать в этой статье? Нас интересует графика, а значит, будем говорить о тех вопросах, которые связаны с графикой и играми. Оптимизировать отображение стандартного окна с настройками игры мы не будем. Опускаться до такой пошлости :), как ассемблер, тоже не будем, потому что если опуститься туда, то можно будет писать целую книгу. Здесь лучше меня расскажет Касперски или другие уважаемые спецы по этому языку. Я знаю ассемблер, но не в такой степени, чтобы заниматься оптимизацией, хотя, даже если просто переписать какую-то функцию C++ на ассемблере, то код может работать быстрее. «Может», но не обязательно, поэтому не будем трогать эту тему.

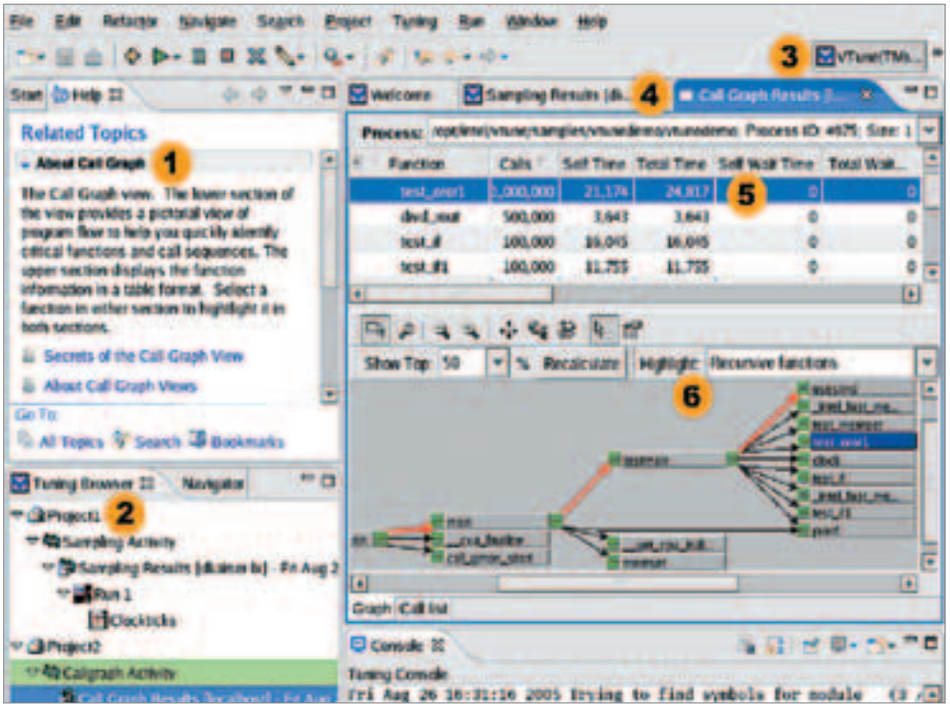
компилятор Не секрет, что львиная доля игр создается на C++, и чаще всего для компиляции используется компилятор от Microsoft. Этот стандарт признанный, и кто оптимизирует программу для работы в Windows лучше, чем сам производитель ОС? Однако нужна ли эта оптимизация в отношении Windows настолько сильно? С полной уверенностью говорю, что не нужна!!! Почему? Об этом читай буквально в следующем абзаце.

Всегда удивляюсь тем паразитам, которые разрабатывают программы (в том числе игры) с минимальными требованиями в Pentium 4, причем компилируют с помощью Visual C++ 6.0 без патчей и обновлений. Что тут удивительного? То, что Visual Studio 6.0 без патчей не знает о существовании Pentium 4, он даже MMX-команды по умолчанию не использует. Почему бы не использовать то, что есть в минимальных требованиях? Одни только команды

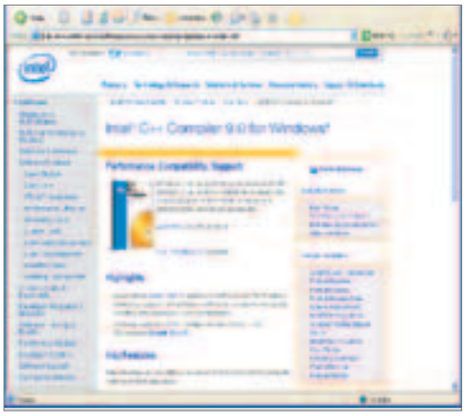
MMX могут повысить производительность, а если включить и SIMD, то выиграешь в производительности до 20%, не внося ни капли изменений в код.

Так почему же не нужна оптимизация под ОС Windows? Игры используют возможности ОС минимально, то есть больше DirectX и процессора, а кто знает архитектуру процессоров лучше, чем сама Intel? По некоторым тестам, одна смена компилятора может повысить производительность программы на 10-15%, особенно в программах, интенсивно рассчитывающих графику. Действительно, такие показатели видишь не очень часто, а может приключиться и замедление работы, но 10% — вполне реальная цифра, причем абсолютно без каких-либо вмешательств в код. Главное — использовать процессор по максимуму и задействовать все возможности.

профилятор-турбулятор Когда игра готова, запускаем ее и ищем самое слабое место. Я всегда искал его на глаз (в принципе, в игровом движке не так уж сложно определить слабое место), но недавно познакомился с программой Intel VTune Performance Analyzer. Эта утилита предназначена для анализа скорости выполнения программы. Настраиваем VTune, запускаем программу и через какое-то время смотрим, какие замеры сделал VTune. Главное, что нас интересует, — какая функция работает дольше всех. Оптимизацией именно такой функции нужно заниматься в дальнейшем.



Intel VTune Performance Analyzer в действии



Компилятор от Intel для Windows

Только не торопись. Одного времени выполнения мало. Дольше всех может выполняться функция инициализации игрового уровня, где происходит загрузка текстур и т.д., но ее оптимизацией заниматься необязательно. Необходимо учитывать прежде всего те функции, которые участвуют непосредственно в рендеринге.

Кроме того, нужно учитывать количество вызовов. Например, одна функция может выполняться достаточно быстро, но вызываться очень часто. Проанализируй: если вызов происходит из одного или двух мест, то, может, лучше ты сделаешь ее inline, то есть избавишься от нее вовсе и сэкономишь лишний переход, возврат, передачу параметров и т.д. В целом экономия составит примерно три и более команд (в зависимости от количества параметров), но если умножить это значение на количество вызовов функции, то экономия становится существенной. Например, если за секунду произошло 1000 вызовов функции, то избавление от нее даст экономию в 3000 команд в секунду — не так плохо!

Кстати, в играх вообще не должно быть маленьких функций, выполняющих от одного до трех простых действий. Именно они чаще всего вызываются по сотни раз и съедают драгоценные такты.

секреты от intel Корпорация Intel заботится о разработчиках, в том числе о разработчиках игр. На **intel.com** есть много полезной информации о различных методах оптимизации и эффективных способах по использованию возможностей современных процессоров. Сейчас можно найти множество документов по использованию Hyper-Threading в играх, а двухъядерные процессоры действительно могут повысить производительность на порядок. Если для игры два ядра являются минимумом, то почему не воспользоваться ими для своих нужд?

мнение эксперта

Михаил ChSnark Пискунов — ведущий сценарист KDV GAMES

РОСТ ТРЕБОВАНИЙ ДЛЯ ИГР СВЯЗАН В ОСНОВНОМ С ПОСТОЯННЫМ РОСТОМ КАЧЕСТВА ГРАФИКИ. УЛУЧШИТЬ ГРАФИКУ В ИГРЕ — САМЫЙ ОЧЕВИДНЫЙ И ПРОСТОЙ ПУТЬ СДЕЛАТЬ ЕЕ ЛУЧШЕ. ПО КРАЙНЕЙ МЕРЕ, В ВОСПРИЯТИИ ПОКУПАТЕЛЕЙ. ТУТ МОЖНО ДАЖЕ ПРОВЕСТИ АНАЛОГИЮ С ДРУГИМИ ПРОДУКТАМИ ВЫСОКИХ ТЕХНОЛОГИЙ. НАПРИМЕР, ПОКУПАТЕЛИ ТВЕРДО ЗНАЮТ, ЧТО ЧЕМ БОЛЬШЕ «МЕГАГЕРЦ В ПРОЦЕССОРЕ» И «МЕГАПИКСЕЛЕЙ В КАМЕРЕ», ТЕМ ЛУЧШЕ, ПОЭТОМУ ПРОИЗВОДИТЕЛИ И НАРАЩИВАЮТ ИХ ЧИСЛО. ТАК ЖЕ И В ИГРАХ: С КАЖДЫМ ПРОЕКТОМ БОЛЬШЕ ПОЛИГОНОВ НА МОДЕЛЬ И СПЕЦЭФФЕКТОВ, СООТВЕТСТВЕННО, БОЛЬШЕ ТРЕБОВАНИЯ. НО НИ ОДНА ИГРА НЕ БЕЖИТ ВПЕРЕДИ ЖЕЛЕЗНОГО ПАРОВОЗА, КОТОРЫЙ ПОДЧИНЯЕТСЯ СВОЕЙ ЛОГИКЕ ДВИЖЕНИЯ.

Классика оптимизации — игра .kkrieger. Размер игры — 97 Кб (включая код, звук, текстуры)

На сайте есть информация и на русском языке. Да, ее намного меньше, чем на английском, но основные и последние документы переводятся. Советую заглянуть на следующую страничку: **www.intel.com/cd/ids/developer/emea/rus/dc/games/index.htm.**

Запомнить этот адрес трудновато, поэтому выжги его в «Избранном».

ati и nvidia Компании ATI и NVidia являются законодателями моды на рынке домашних видеоускорителей и так же, как Intel, заботятся о разработчиках. На их сайтах можно найти тонны информации о разработке графических эффектов, игр и использовании максимальных возможностей последних видеочипов.

Чем сильнее ты нагрузишь видеочип, тем быстрее будет работать программа. Почему? Чаще всего видеоускорители выполняют задачи быстрее, чем центральный процессор, потому что они специально заточены для графических вычислений, матриц, векторов и т.д. Некоторые задачи могут выполняться параллельно с вычислениями центральным процессором, что также повысит производительность.

Если ты решил использовать какую-то информацию с сайтов **www.ati.com** или **www.nvidia.com**, то проявляй осторожность. Некоторые функции работают на видеочипе одного производителя, но вызывают серьезные проблемы на чипе другого производителя. В этом случае можно потерять большую часть потенциальных пользователей, а если проект коммерческий, то и покупателей. Чтобы решить проблему, попробуй создать две сборки исполняемых файлов: одна для чипа ATI, вторая — для чипа NVidia. Осуществляется не так сложно. Создаешь две функции для разных SDK и подключаешь их в зависимости от сборки, выполняемой в данный момент. Да, усложняется отладка, но прямой вызов функций драйвера видеокарты лучше.

«ИСПОЛЬЗУЯ DIRECTDRAW, НИКОГДА НЕ ПРИМЕНЯЙ GDI: ОНИ ТОРМОЗЯТ ПРОГРАММУ. ПОЛЬЗУЙСЯ ПРЯМЫМ ДОСТУПОМ К ПОВЕРХНОСТИ И КОПИРОВАНИЕМ ПОВЕРХНОСТЕЙ»

качество — производительность Оптимизация графики — это достаточно интересный процесс, потому что здесь, помимо общих принципов оптимизации программ, применяется множество других приемов. Например, чтобы оптимизировать размер картинки, ее сжимают классическим архиватором. Размер файла уменьшится, но коэффициент сжатия зависит от самой картинки. Например, фотографии с большим количеством цветов сжимаются очень плохо.

Человеческий глаз не способен различить два оттенка цвета, если в них отличается на единицу только одна составляющая. Получается, что если объединить близлежащие точки с небольшим отличием в оттенке и заменить их одним цветом, то файл сожмется больше, а пользователь ничего и не заметит. Такое сжатие называется компрессией с потерей качества. Тут нужно добиваться компромисса качества изображения и качества сжатия, что достигается выбором максимальной разницы между оттенками, которые можно объединить. Если будут объединены два цвета, разница которых незаметна человеческому глазу, то качество пострадает несильно. Однако если отличие заметно, то качество теряется.

Где это используется? Конечно же, в файлах изображений, которые способна загружать программа. Чем меньше файл, тем быстрее он загружается в память с жесткого диска, который является самым слабым звеном компьютера. Далее процессор восстановит изображение, и если алгоритм сжатия несложный, то, может быть, на это потребуется меньше времени, чем на загрузку с диска несжатых данных.

Количество используемых цветов влияет и на потребности в ресурсах. Чем больше цветов мы используем, тем больше памяти требуется для хранения изображений, поверхности и тем больше потребности в процессорном времени, чтобы копировать всю эту память между поверхностями.

Предположим, у нас есть картинка размером 100x100 пикселей. Если используется глубина цвета в один байт, для хранения изображения понадобится 10 000 байт памяти. Такая картинка может содержать максимум 256 цветов (что очень мало), и поэтому желательно использовать минимум два байта, когда количество цветов будет равно 65 535. Получается достаточно для хранения более качественного изображения, но понадобится 20 000 байт памяти, что в два раза больше, соответственно, и копирование данных будет требовать от процессора в два раза больше ресурсов.

Если же выбрать глубину цвета в 24 бита, то тут уже понадобится в три раза больше ресурсов, хотя количество цветов будет исчисляться 16 777 216. Вот и думай после такого, что выбрать: качество или скорость. Наша задача — найти оптимальный вариант, который позволит получить лучшее соотношение скорости и качества.

Размер изображений также играет немаловажную роль. Если мы выбрали разрешение экрана в 800x600 пикселей при 16-битном цвете, то для хранения поверхности понадобится 800*600*2 = 960 000 байт памяти. Почти 1 Мб! А если изображение будет 1024x768, то объем необходимой поверхности памяти составит 1 572 864 байт. Снова увеличение ресурсов в два раза, следовательно, мы получаем падение производительности во время копирования содержимого поверхностей.

Оптимизация графики на первоначальном этапе — это борьба качества и скорости, выбор между ними. Современным стандартом стали ЖК-мониторы с диагональю 15 дюймов. Чтобы картинка

на них выглядела приемлемо, необходимо использовать разрешение 800x600, а лучше 1024x768.

Однако неплохо было бы предусмотреть возможность выбирать необходимое разрешение. Например, у меня широкоформатный ноутбук, и для него идеально разрешение 1280x800, а разрешения 800x600 и 1024x768 выглядят растянуто, что очень неудобно. Вот почему в данном случае можно предложить пользователю самостоятельно выбирать необходимое разрешение в зависимости от имеющихся ресурсов.

В качестве глубины цвета чаще всего используется 16 бит, потому что так мы добиваемся приемлемого качества при минимуме затрат. Меньше уже нельзя, а больше в основном избыточно. Можно предложить пользователю выбирать и глубину цвета, но тогда придется писать слишком универсальный код, который понизит производительность. Чтобы не терять скорость, лучше все-таки привязаться к определенной глубине экрана.

не дай себе засохнуть Двумерная графика может формироваться двумя способами: математическим и спрайтовым. В случае с математическим способом поверхность заполняется через прямой доступ по определенному алгоритму. Некоторые из алгоритмов создания эффектов я рассмотрю, только не сейчас. При использовании «математики» скорость формирования сцены, помимо глубины цвета и разрешения, очень сильно зависит от используемого алгоритма.

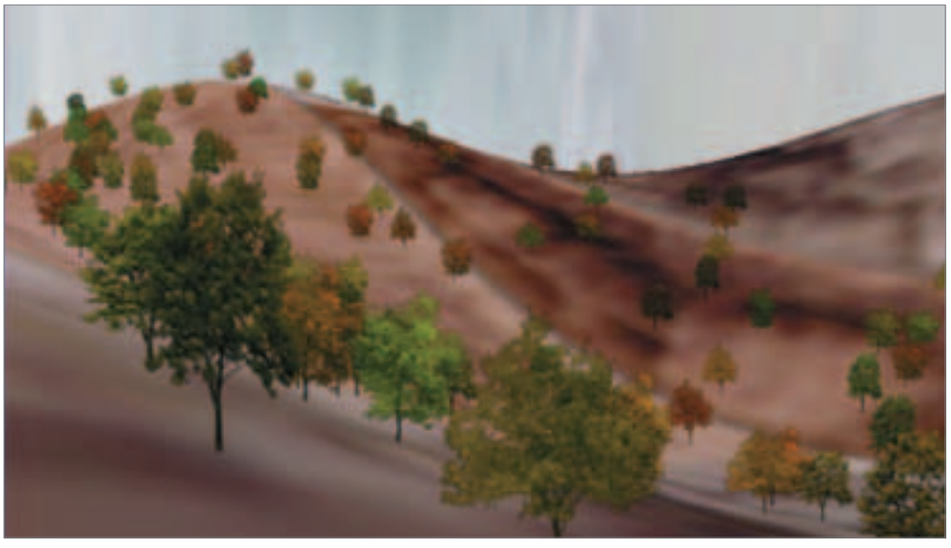
При спрайтовом формировании сцены дополнительным фактором торможения является количество спрайтов, поскольку для вывода каждого из них необходимо выполнить операцию Blt или BltFast (я имею в виду функции DirectDraw). Чем больше обращений, тем больше потеря, потому что тут проявляется эффект цикла: приходится много раз вызывать одну и ту же функцию, во время вызова несколько раз параметры поднимаются в стеке (у Blt их немало) и происходит вызов удаленной функции, а также куча лишних проверок со всеми вытекающими последствиями. Иногда функцию Blt даже лучше заменить копированием данных через прямой доступ к памяти и WinAPI-функцию memcpy.

Потери данных при спрайтовом выводе могут быть и неоправданными. Допустим, у нас в распоряжении находится экран размером в 800x600 пикселей, как показано на рисунке. Все, что закрашено черным цветом, — это оборка, которая статична и не изменяется, а в белой области данные формируются динамически, причем полностью. Эта задача решается вот так: перед формированием кадра вывести на экран изображение 800x600 с оборкой, затем сформировать среднюю часть. Стоп. Зачем делать это, если оборка статична? Не лучше ли один раз вывести оборку, а потом перекрашивать только центральную часть, которая изменяется? Конечно же, лучше. Таким образом мы сэкономим целую операцию Blt, которой приходится копировать большой объем информации.

Если экран программы каждый раз формируется полностью, то и не стоит каждый раз перед формированием сцены очищать его. Все равно каждый пиксель будет закрашен, и от нашей заливки ничего не останется. Другое дело, когда рисуется движение звезд на черном небе. Тут действительно удобнее закрасить экран черным, а потом поверх неба нарисовать звезды. Но если на экране присутствует поверхность земли (например, из 600 пикселей в высоту: нижние 200 — это земля, а верхние 400 — черное небо), то нужно закрашивать только ту часть, где есть небо. Нет смысла очищать весь экран!

Схема экрана программы

Пример использования сортировки при отображении ландшафта



алгоритм Самое главное, что определяет скорость работы игры, — это алгоритм. Хороший алгоритм не требует дополнительной оптимизации. Не могу предложить однозначное решение, но некоторые рекомендации по оптимизации есть.

Игровые сцены чаще всего характеризуются значительным множеством объектов. Давай вспомним знаменитую игру Command&Conquer (или, проще, C&C). На поле боя одновременно может находиться тысяча объектов. Одних только солдатиков можно наделать пару тысяч, плюс здания, деревья, мосты и т.д. Теперь представим, что произойдет, если цикл отображения сцены будет выглядеть следующим образом:

```
for (int i=0; i<OBJECT_COUNT; i++)
 {
 if (объект видим)
 Отобразить объект
 }
```

Если значение OBJECT_COUNT равно 2 000, то для отображения сцены придется выполнить 2 000 шагов цикла и на каждом из них произвести проверку видимости, которая может состоять из четырех операций if. Сумасшедшие расходы! Не по силам даже современному компьютеру, если не оптимизировать код. Что же можно сделать здесь? Как сократить количество циклов?

Карта игры C&C большая, и в определенный момент времени видно не более 10% ее. Можно разбить карту на 10-20 квадратов и привязывать объекты именно к определенному квадрату на карте. В результате цикл отображения будет таким:

```
for (int i=0; i<QUAD_COUNT; i++)
 {
 if (квадрат видим)
 {
 Запустить цикл проверки видимости
 и отображения объектов
 данного квадрата на карте.
 }
 }
```

Теперь будут проверяться не все объекты, а только те, которые находятся на квадрате, видимом в данный момент. Единственное, что нужно сделать дополнительно, — после перемещения объекта проверить, вышел ли он за пределы своего квадрата, и если вышел, то присоединить объект к новому квадрату, на плоскость которого попал объект. Такие объекты, как деревья и здания, не перемещаются, поэтому для них проводить проверку не нужно.

сортировка Следующий способ ускорения отображения — сортировка. Желательно отсортировать все объекты по определенной оси и отображать их в этом же порядке. Замечу, что если очередной объект вышел из зоны видимости по отсортированной оси, то остальные с еще большей вероятностью вышли и дальнейшая проверка объектов бессмысленна — цикл можно прерывать. Соответственно, мы снова экономим цикл, а именно циклы являются самым слабым местом, особенно в играх, где очень много объектов.

В составе DirectX SDK есть очень хороший пример рисования деревьев на поверхности ландшафта (DXSDK\Samples\C++\Direct3D\Billboard). Деревьев очень много, и чтобы не перебирать их все, во время отображения используется сортировка. Попробуй убрать сортировку и производить проверку всех деревьев при каждом формировании сцены — производительность примера заметно упадет.

оптимизация 3d При создании фигуры с помощью Direct3D используй минимально необходимые размеры. Например, при создании буфера индексов IDirect3DIndexBuffer9 каждый элемент массива может быть 16- или 32-битным. Если количество индексов не превышает 65 535, то следует использовать 16-битный массив. В данном случае переход на 32 бита будет неоправданным и принесет лишние расходы памяти, которой никогда не бывает много.

В Direct3D мы формируем сцену с помощью вершин и треугольников. Чем их больше, тем больше времени движок и видеокарта тратят на формирование сцены. И вновь мы выбираем между количеством и качеством! Посмотри на рисунок, где показана сфера, созданная из 32-х сегментов. Сфера получается гладкой, но количество необходимых треугольников слишком велико.

Можно сократить количество сегментов до 16-ти, то есть уменьшить его в два раза. В результате сокращается количество ресурсов, необходимых для отработки фигуры, но сфера получается угловатой. Я думаю, такое положение дел устроит далеко не многих.

Если же нам нужны сглаженные поверхности, то приходится выбирать между гладкостью и скоростью. В большинстве игр разработчики стараются использовать прямоугольные поверхности — для создания четырехугольной плоскости достаточно всего лишь двух треугольников.

Получается, что скорость создания сцены зависит не только от количества объектов на экране (объектов интерьера, освещений, существования теней). На сцену влияют и количество вершин, из которых состоит объект. Здесь необходимо подходить к решению задачи с деликатностью и осторожностью.

С другой стороны, если создать сферу из ста сегментов, то мы получим явную избыточность, особенно если эта сфера в сцене будет находиться очень далеко от зрителя и выглядеть мелкой. Пользователь просто не сможет увидеть мелкие детали, которые ты захочешь передать. Если у тебя есть 3D-редактор, позволяющий контролировать количество сегментов (например 3D Studio Max), то попробуй сейчас создать сферу размером во весь экран. Для нормального отображения и получения гладкой поверхности необходимо не менее 30-ти сегментов. Если же отдалить сферу как можно дальше (чтобы в диаметре она казалась не больше десятикопеечной монеты), то будет достаточно и 12-ти сегментов.

Получается, что при формировании сцены мы можем использовать разное количество сегментов для объектов с разной удаленностью и значительно сэкономить жизнь процессору и видеокарте, а следовательно, увеличить скорость создания сцены.

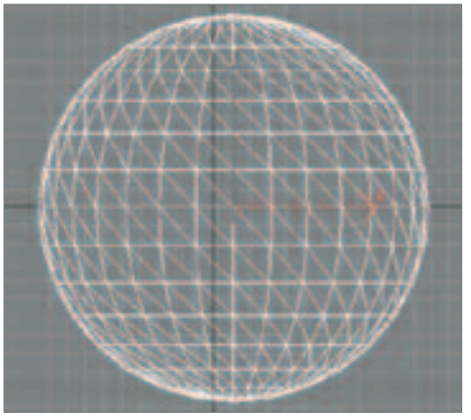
Посмотри на рисунок, где показаны сферы различного размера. Самая большая сфера создана из 32-х сегментов, а остальные — из 12-ти. Обрати внимание на то, что самая маленькая сфера выглядит вполне гладкой, хотя на сферах побольше хорошо видны угловатости. Использовать множество сегментов на маленьких объектах — лишняя растрата ресурсов.

То же касается и текстур. Чем ближе объект, тем лучше видна каждая деталь на текстуре. На самой большой сфере можно увидеть, что текстура есть, но на самой дальней сфере обычно не разглядишь ничего. Она настолько мала, что кажется окрашенной в один цвет. Тогда зачем использовать текстуру? Не лучше ли просто окрасить сферу в нужный цвет? Пользователь ничего не заметит, особенно если сфера будет находиться в движении.

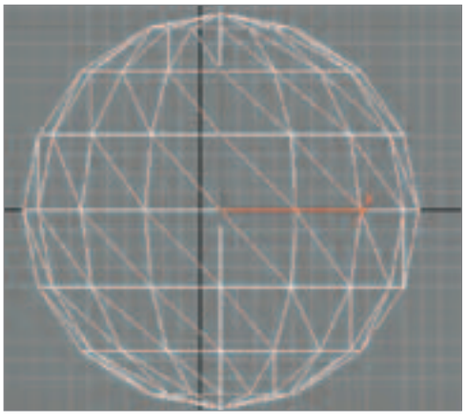
Чтобы повысить скорость работы с текстурами, предлагаю действовать следующим образом:

- СОВСЕМ НЕ ИСПОЛЬЗОВАТЬ ТЕКСТУРЫ ДЛЯ САМЫХ ДАЛЬНИХ (ОТ ЗРИТЕЛЯ) ОБЪЕКТОВ, ТОЛЬКО ОКРАШИВАТЬ ИХ ЦВЕТОМ, МАКСИМАЛЬНО СООТВЕТСТВУЮЩИМ ЦВЕТУ ТЕКСТУРЫ.
- ПО МЕРЕ ПРИБЛИЖЕНИЯ ОБЪЕКТА НАТЯНУТЬ НА НЕГО ТЕКСТУРУ НЕБОЛЬШОГО РАЗМЕРА, НАПРИМЕР 16X16. ВОТ ТАК МЫ ЗАРАНЕЕ МАСШТАБИРУЕМ БИТОВЫЙ ФАЙЛ, ОБЛЕГЧАЯ ЖИЗНЬ ВИДЕОКАРТЕ.
- ДЛЯ БОЛЬШИХ ОБЪЕКТОВ, НАХОДЯЩИХСЯ ПОБЛИЗОСТИ, ИСПОЛЬЗОВАТЬ БОЛЕЕ КАЧЕСТВЕННЫЕ ТЕКСТУРЫ.

Если масштабировать текстуры заранее, то качество изображения будет лучше, чем если бы оно было сделано DirectX, так как профессиональный художник сможет сделать необходимые сглаживания, чтобы вид уменьшенного изображения был приемлемым. Конечно, невозможно подготовить все возможные варианты текстур, но хотя бы несколько вариантов должны быть сделаны. Меньшие по размеру текстуры, которые будут использоваться для дальних объектов, обработаются быстрее благодаря меньшему размеру и меньшему коэффициенту программного масштабирования. Текстуры, масштабированные заранее, — это один из немногих трюков, благодаря которому мы повышаем и скорость, и качество сцены.



Сфера из 32-х сегментов



Сфера из 16-ти сегментов

итого Я мог бы еще долго рассказывать об оптимизации в играх, но эта тема слишком широкая и требует отдельной книги, над чем я сейчас и работаю. Прошу ко мне на сайт **www.vr-online.ru** — уже к моменту выхода этого номера я постараюсь выложить что-нибудь новое по оптимизации и графике.

Удачи! И не бойся экспериментировать и обманывать пользователя (они это любят)

НЕ ХВАТАЕТ ЧЕГО-ТО
ОСОБЕННОГО?

Играй просто!
GamePost

Age of Empires III Collector's Edition

$125.99

Call of Duty 2 Collector's Edition

$99.99

Command & Conquer: Collection

$49.99

* Diablo Action Figure: **Necromancer** **$42.99**

У НАС ПОЛНО
ЭКСКЛЮЗИВА

* Эксклюзивные игры

* Коллекции фигурок из игр

* Коллекционные наборы

Тел.: (495) 780-8825
Факс.: (495) 780-8824
www.gamepost.ru

GAMEPOST

звуковое насилие

УЧИМСЯ ПРАВИЛЬНО БУДИТЬ СОСЕДЕЙ ПО НОЧАМ

ЗВУКОВОЕ СОПРОВОЖДЕНИЕ ПОКА ИГРАЕТ В КОМПЬЮТЕРНЫХ ИГРАХ ТОЛЬКО ВТОРОСТЕПЕННЫЕ РОЛИ. СИЛЬНЫЙ РАЗБРОС ПО КАЧЕСТВУ АУДИОАППАРАТУРЫ, КОТОРОЙ ПОЛЬЗУЮТСЯ ИГРОКИ, ПОКА НЕ ПОЗВОЛЯЕТ ЗВУКУ ВСТАТЬ НА ОДНУ СТУПЕНЬКУ С ГРАФИКОЙ ТАК, КАК ЭТО ПРОИЗОШЛО В КИНОИНДУСТРИИ |

ПАЛАГИН АНТОН АКА TONY (TONY@EYKONTECH.COM)

synopsis Владельцев интегрированных домашних комплексов развлечений можно пересчитать по пальцам, а обладателей качественной аппаратуры — не сыскать и днем с огнем. Вот почему не во всех играх можно встретить качественный звук, который гармонично дополняет игровой процесс. Обычно хороший звук присутствует в симуляторах техники и экшенах, то есть в играх, где есть честное трехмерное пространство, по которому игрок свободно перемещается. В подобных играх применяется трехмерное позиционирование звука и его обработка EAX-подобными эффектами, которые подчеркивают звуковые свойства помещения, в котором находится игрок. Чтобы ты воспринимал сцены поглубже и поэмоциональнее, талантливые дизайнеры игр динамически оперируют музыкальным сопровождением и звуковыми эффектами: ломание дверей, взрывы, злобные крики и рыки и т.д. Помню, как после недели игры в System Shock 2 я некоторое время пугался, если слышал какие-нибудь визжащие зву-ки — настолько хорошо мой мозг запомнил ситуации, в которые попадал герой игры после того, как раздавались вскрики-визги.

Для качественного воспроизведения звуков обычно не хватает возможностей API операционной системы, и разработчики используют специализированные звуковые библиотеки. В этой статье я расскажу о том, как работать с кроссплатформенной библиотекой FMOD. Принципы

работы с прочими библиотеками похожи на описанные как две капли воды. Этот инструмент довольно часто используется в играх, поддерживает и PC, и Mac, и все современные игровые консоли, в том числе консоли следующего поколения PlayStation 3 и Xbox 360. FMOD доступен для скачивания и бесплатен для некоммерческого использования. Не стоит забывать, что некоторые звуковые форматы, такие как MP3, также требуют лицензионных отчислений, поэтому в коммерческих продуктах стоит задуматься о бесплатных альтернативах, например OGG.

инициализация Методы библиотеки возвращают результат своего выполнения FMOD_RESULT. Если это значение не равно FMOD_OK, то произошла ошибка. Диагностическое сообщение об ошибке можно получить с помощью функции FMOD_ErrorString(). FMOD поддерживает два интерфейса C и C++, названия функций и методов классов, а также их сигнатуры в этих интерфейсах идентичны. Функциональность FMOD расширяется с помощью плагинов, из них поддерживаются следующие: DSP (digital signal processing), форматы данных, вывод звука.

```
//Создаем системный объект
result = FMOD::System_Create(&system);
//Проверяем версию библиотеки
result = system->getVersion(&version);
if (version < FMOD_VERSION)
{
printf("Error! You are using an old version
of FMOD %08x. This program requires %08x\n",
version, FMOD_VERSION);
exit(-1);
}
//Получаем параметры конфигурации
акустики пользователя, установленных
в панели управления.
result = system->getDriverCaps
(0, 0, 0, 0, &speakermode);
//Устанавливаем для программного
микшера FMOD параметры акустики
result = system->setSpeakerMode(speakermode);
//Инициализируем FMOD
result = system->init(32, FMOD_INIT_NORMAL, 0);
```

Чтобы начать работу с FMOD, необходимо создать системный объект, проверить версию используемой бинарной библиотеки FMOD, дальше настроить программный микшер и инициализировать FMOD. Первый параметр в методе init() — это количество используемых каналов, то есть число проигрываемых одновременно звуков, и чем больше это число, тем больше нагружается процессор и память. Два других параметра характерны для используемой аппаратной платформы.

```
//Освобождаем ресурсы звуков
result = sound1->release();
result = sound2->release();
//Завершаем работу системного объекта
result = system->close();
//Освобождаем системные ресурсы
result = system->release();
```

При завершении работы библиотеки для каждого созданного объекта библиотеки вызываем метод release(). Перед высвобождением системных ресурсов вызываем метод close().

```
//Загружаем стереозвук, звук будет
обрабатываться программно
result = system->createSound("../media/drum-
loop.wav", FMOD_SOFTWARE | FMOD_2D, 0,
&sound1);
//Отключаем циклическое воспроизведение звука
result = sound1->setMode(FMOD_LOOP_OFF);
//Загружаем стереозвук, обрабатываться
он будет аппаратно, и, кроме того, будем
его позиционировать в 3D-пространстве
result = system->createSound("../media/ste-
reo.ogg", FMOD_HARDWARE | FMOD_3D, 0,
&sound2);
```

Загрузка звуков осуществляется с помощью метода System::createSound(), который загружает и распаковывает звук в память и подходит только для оперативных коротких звуков (выстрелы, шаги и т.д.). Музыку лучше загружать в режиме потока с помощью функции System::createStream(), в этом случае можно указать размеры буфера для обработки этого звукового потока, что делается с помощью метода System::setStreamBufferSize(). Размер памяти, занятой FMOD и его ресурсами, можно выяснить с помощью метода FMOD::Memory_GetStats(). Добавить собственный механизм менеджмента памяти — FMOD::Memory_Initialize(). Кроме имени файла со звуком, эти функции принимают также флаги создания. Например, в листинге загруженный звук обрабатывается без использования аппаратного ускорения и только в двумерном режиме. Для позиционирования звука в трехмерном пространстве необходимо указывать флаги семейства FMOD_3D. Если режим использования звука необходимо динамически изменить, то после загрузки можно установить нужный режим использования с помощью метода Sound::setMode().

работа со звуком После инициализации FMOD и загрузки звуков в память уже можно прослушать их. Методы, проигрывающие звуки, работают в асинхронном режиме, то есть управление возвращается вызвавшей программе сразу после окончания подготовки звука к проигрыванию. Работу по выводу звука на звуковую карту и микширование FMOD выполняет в фоновых потоках. Периодически (раз в кадр) необходимо вызывать метод системного объекта update() для обновления состояния библиотеки.

```
//Проигрываем звук на свободном канале,
номер канала — возвращаемый параметр
result = system->playSound(system, FMOD_CHAN-
NEL_FREE, sound1, TRUE, &channel);
//Эта конфигурация микшера проиграет звук на
всех семи колонках акустической системы 7.1
result = channel->setSpeakerMix(channel,
1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f, 1.0f);
//Снимаем звук с паузы
result = channel->setPaused(channel, FALSE);
//Обновляем состояние FMOD
system->update();
```

Если по твоей задумке звуки располагаются в трехмерном мире и, кроме того, они могут перемещаться, а герой (слушатель звуков, то есть ты) также перемещается по пространству, то для правильного смешивания в микшере для каждого звука должны быть установлены его позиция и скорость. Эти же параметры плюс ориентация в пространстве должны быть установлены для слушателя. Плюс должны быть установлены значения среды, в которой распространяется звук: доплеровское смещение и ослабление из-за расстояния до источника звука.

> «FMOD ДОСТУПЕН ДЛЯ СКАЧИВАНИЯ И БЕСПЛАТЕН ДЛЯ НЕКОММЕРЧЕСКОГО ИСПОЛЬЗОВАНИЯ»

```
//Устанавливаем параметры среды
распространения звука
result = system->set3DSettings( 0.9, 0.9, 1.0 );
//Устанавливаем позицию и скорость
источника звука
FMOD_VECTOR vel, pos;
...
result = channel->set3DAttributes(&pos, &vel);
//Устанавливаем позицию и скорость
слушателя звука
FMOD_VECTOR vel, pos, forward, up;
...
result = channel->set3DListenerAttributes
( 0, &pos, &vel, &forward, &up);
```

Как видишь, ничего сложного нет. Главное — помни о соседях, о магической цифре 22.00 и о том, что судебные приставы могут пожаловать к тебе в любую минуту для конфискации дорогостоящей акустики ;)

мнение профессионалов

СТОПРОЦЕНТНЫХ СПОСОБОВ ЗАЩИТЫ НЕ СУЩЕСТВУЕТ

СПЕЦ: КАКИЕ СРЕДСТВА РАЗРАБОТКИ ИСПОЛЬЗУЮТ ПРИ СОЗДАНИИ ИГР? АЛЬТЕРНАТИВЫ, ПЛЮСЫ/МИНУСЫ, ОПТИМАЛЬНЫЙ ВЫБОР…

АЛЕКСАНДР ФЕДОРА: Для PC — Microsoft Visual C++ .NET 2003. Плюс обязательно утилиты для контроля версий (CVS, SVN). Для менеджмента багов — Test Track Pro от Seapine Software.

СЕРГЕЙ АЗАРОВ: Средства разработки — понятие очень растяжимое, ибо, как правило, эти средства пишутся специально для конкретной игры или линейки игр. Уместнее такой вопрос: «Какие средства разработки используют для разработки средств разработки игры?» Запутанно, но похоже на правду больше. Если говорить о более низком уровне, то используется достаточно стандартный набор: Visual Studio, 3DS Max/Maya, Photoshop, Microsoft Office. Если говорить о более высоком уровне, то здесь уже средства разработки создаются специально под проект: редакторы уровней (хотя вполне можно использовать тот же 3DS Max), редактор диалогов, редакторы систем частиц и т.п. Конечно, помимо описанного выше, существует масса довольно полезных программ, не бесплатных. Например, Facegen Modeller — программа, позволяющая генерировать различные формы лиц, с возможностью экспортирования во все известные графические пакеты как самого лица, так и состояний его морфинга для соответствующей лицевой анимации.

РОМАН ЦОЙ: Программирование: C++ — отраслевой стандарт. Арт: Maya + Photoshop = оптимальный выбор (достоинства: логичный интерфейс и неограниченные возможности для творчества и производства). А вообще в производстве используется целый ряд инструментов. Секрет же успеха — творческое начало и правильное сочетание средств разработки.

ВОЛОДЯ КОРОТКОВ: В основном это Visual C++, 3D Max, Maya, рукописные плагины для экспорта, другие рукописные утилиты. Альтернатив просто нет, этот выбор оптимальный.

АНДРЕЙ БЕЛКИН: Тема очень широкая, если раскрывать ее «в полном объеме». Само понятие «средства разработки» можно истолковать очень неоднозначно. Скажем так, типовой набор средств, используемых в большинстве известных мне проектов, — это: MS Visual Studio (например, версии .net) в качестве среды для написания кода плюс какой-нибудь модный компилятор, например GNU или CodeWarrior; Maya или 3DS Max и их многочисленные плагины — в качестве среды для работы с 3D-моделями, спецэффектами и т.п. Photoshop и Painter и их плагины — для работы с текстурами и концепт-артом. Sound Forge — для работы со звуком. MS Office во всех его проявлениях — для работы с документами всех мастей.

ВАДИМ ГАЙДУКЕВИЧ: Visual Studio, Maya 3D, Photoshop и золотые руки — вот истинные инструменты создания современных компьютерных игр. Правда, отдельные разработчики пытаются использовать различные готовые middleware-решения, но истинный игродел никогда не опустится до того, чтобы взять хотя бы одну строчку готового кода со стороны. Он все напишет сам.

МИХАИЛ ПИСКУНОВ: Мне, как сценаристу, для работы с литературной частью вполне хватает Word'а, Excel'а. Иногда — Visio.

ДМИТРИЙ ЖУКОВ: Как правило, сначала создают некое базовое ядро, собственную среду разработки — «движок». Обычно пишут на языке C++, с использованием ассемблера для самых ответственных операций. Поскольку большинство современных игр так или иначе трехмерны, то используют библиотеку OpenGL или DirectX.

Отдельная задача — это программирование AI, то есть искусственного интеллекта. Здесь начинается чистая наука: технологии сетей, теория графов и т.д. Очень многие компании предпочитают свести кодирование к минимуму, купив готовый движок сторонней фирмы, особенно это касается трехмерных стрелялок (First Person Shooter). Один из самых популярных движков — от игры Quake 3, на котором создано около десятка игр. Затем за дело берутся художники, композиторы и т.д. — те, кто создает содержание в игре. Вообще игры перестали быть готовым к употреблению продуктом, и зачастую пользователь сам привлекается к процессу разработки контента, а многие уровни к играм создаются энтузиастами.

АНДРЕЙ ТЕРТИЧНИКОВ: Для программирования под J2ME необходим только соответствующий SDK, сам код можно писать хоть в «Блокноте». Однако для удобства применяются различные IDE (интегрированные среды разработки), заточенные именно под создание приложений для J2ME. Среди самых популярных можно выделить Borland JBuilder, IDEA и NetBeans. По большому счету, между ними нет каких-либо серьезных различий, поэтому каждый выбирает то, что нравится.

СПЕЦ: НИ ОДИН ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ НЕ МОЖЕТ ПРОТИВОСТОЯТЬ ГЕЙМЕРУ. ВОПРОС, СКОРЕЕ, ВО ВРЕМЕНИ, КОТОРОЕ ИГРОК ПОТРАТИТ НА ПОИСК СЛАБЫХ МЕСТ ИИ. РАЗРАБОТЧИКИ ИГР СОЗНАТЕЛЬНО ДОБИВАЮТСЯ ЭТОГО ИЛИ СОЗДАНИЕ ДОСТАТОЧНО РАЗУМНОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА НЕВОЗМОЖНО?

МИХАИЛ РАЗУМКИН: Неужели было бы лучше, если бы геймер не смог справиться с компьютером? Такую игру точно никто не купил бы. Это же в первую очередь отдых, а не пионерское создание трудностей, чтобы потом героически их преодолевать. Но если судить по моему опыту, создать достаточно правдоподобного (живого) противника чрезвычайно сложно. Вспомни ботов в Counter-Strike: им не нужно целиться. Один выстрел — и у тебя дырка в голове. Но вот поступить нестандартно они не могут, так как бегают по маршрутам. Похожая ситуация и в гонках, стратегиях, драках. Вообще вопрос AI очень сложен. Обычно это многоуровневый набор скриптов, создающих подобие реальности. Агрессивный гонщик будет постоянно бортовать вас, лидер — идеально вписываться во все повороты, но не на максимальной скорости (должен же ты его все-таки догнать), «лох» же будет собирать все кочки на дороге. Но это только подобие жизни, а не настоящий AI. Искусственный интеллект должен обучаться, то есть серьезно менять свое поведение в зависимости от твоих действий и изменения окружающей среды. Играть с таким противником гораздо интереснее, но не всегда оправданно с точки зрения трудозатрат со стороны разработчика. Хороший набор скриптов проще и надежнее. Вот они обычно и стараются держать баланс

между всесильным компьютерным противником, который точно знает все бреши в твоей обороне и вдруг досадно ошибается, вообще прекращает сопротивление, начинает тупить. А смещение баланса в ту или иную сторону называют «уровнем сложности».

АЛЕКСАНДР ФЕДОРА: Во-первых, ограничения по производительности. ИИ забирает много процессорного времени, причем чем больше NPC, тем больше требуется ресурсов. И во-вторых, ограничения бюджета. Создание «умного» противника в игре — достаточно тяжелая задача, требующая больших трудозатрат на разработку, тестирование и балансировку.

ЮРИЙ МАТВЕЕВ: Боюсь, что когда придумают «разумный» ИИ, это будет начало конца человечества в его нынешней форме существования...

СЕРГЕЙ АЗАРОВ: Есть такое понятие, как геймплей, определяющее саму суть игры. Если в игре нет игрового процесса, то в ней нет никакого смысла. Искусственный интеллект является неотъемлемой частью геймплея, его назначение состоит в том, чтобы обеспечить естественные препятствия для игрока на пути к его цели — прохождению игры. Так вот, если искусственный интеллект начинает мешать достижению этой цели (а любой очень умный и сложный ИИ будет этому способствовать), то наличие подобного ИИ является абсолютно нецелесообразным. Поэтому ответ на вопрос — «Да, так и задумывается».

АНДРЕЙ БЕЛКИН: Достаточно разумный искусственный интеллект придумать возможно. Весь вопрос в степени достаточности. Более того, идеальный искусственный интеллект даже можно реализовать. Дело в том, что в игру с таким вот «достаточно разумным» искусственным интеллектом никто не будет играть, поскольку игры делаются чтобы развлекать потребителя, а не чтобы поражать его нечеловеческим искусственным интеллектом.

МИХАИЛ ПИСКУНОВ: Разработчики игр не придумывают «достаточно разумный» искусственный интеллект — это, скорее, научная задача. Разработчики просто стараются придумать набор правил и алгоритмов для компьютера, чтобы с ним можно было более или менее интересно играть, то есть которого обязательно можно побить, но не сразу :). Этого вполне достаточно.

ДМИТРИЙ ЖУКОВ: Противостоять-то, возможно, не сможет. Вопрос времени и только времени. А вот в быстродействии AI всегда берет верх — поэтому его и не делают шибко продвинутым.

СЕРГЕЙ ЗАГУРСКИЙ: При разработке игры, как правило, не составляет труда создать ИИ, который бы «уделывал» практически любого игрока. Сложно как раз наоборот, сделать его более «человечным», чтобы он допускал ошибки такие, какие допускает и обычный игрок. Чтобы реагировал не моментально, а изображал испуг, панику и т.п.

АНДРЕЙ ТЕРТИЧНИКОВ: Проблема в том, что современный уровень развития технологий (я имею в виду не только и не столько программирование, а вообще научную базу, на которой строится программирование AI) не позволяет создать программу, которая могла бы действительно ДУМАТЬ. Все противники даже в самых современных играх — это всего лишь набор алгоритмов, жестко заданных программистом. Чем эти алгоритмы адекватнее самой игре, тем более «живым» кажется противник. Однако сложность и проработанность современных игр таковы, что вложить в искусственный интеллект все возможные реакции на действия игрока просто невозможно. Поэтому ограничиваются неким набором шаблонов и набором условий, при которых эти шаблоны срабатывают. Это существенно ограничивает «разумность» AI (если к нему можно вообще применить это слово).

СПЕЦ: КАКИЕ СПОСОБЫ ЧАЩЕ ВСЕГО ИСПОЛЬЗУЮТ ДЛЯ ЗАЩИТЫ ИГР? АЛЬТЕРНАТИВЫ, ПЛЮСЫ/МИНУСЫ, ОПТИМАЛЬНЫЙ ВЫБОР...

АЛЕКСАНДР ФЕДОРА: В основном для защиты используются уже готовые технологии типа StarForce, Securom и т.д. либо самописные методы. Причем некоторые пользователи вообще отказываются брать диски с мощной защитой типа StarForce, так как распространено мнение, что такие методы защиты вносят баги в игры и со временем могут перестать опознавать даже лицензионные диски. Самописные методы защиты в основном делают несложными, а-ля «от дурака». Это вызвано тем, что написать полноценную систему занимает много времени и денег. Но простые решения могут позволить защититься от копирования большинством пользователей. Вообще ломается любая система защиты. Вопрос в том, как затратить достаточное количество ресурсов на защиту, чтобы большинство игроков не смогли легко взломать или найти крэк и купили лицензионный диск.

ВОЛОДЯ КОРОТКОВ: Чаще всего для защиты игр используют программные комплексы сторонних производителей: StarForce, SecuRom, SafenSec... У каждого из них есть свои плюсы и минусы. Оптимальный выбор для нас — отечественный производитель StarForce.

СПЕЦ: ЧТО ЯВЛЯЕТСЯ НАИБОЛЕЕ РЕСУРСОЕМКИМ В ИГРЕ? КАКИЕ СПОСОБЫ ОПТИМИЗАЦИИ МОЖНО ПРИМЕНЯТЬ К ГОТОВОЙ ИГРЕ И ИГРЕ, НАХОДЯЩЕЙСЯ НА ЭТАПЕ РАЗРАБОТКИ?

СЕРГЕЙ АЗАРОВ: Самым ресурсоемким в игре, естественно, является контент: модели, текстуры, анимации, системы частиц и т.д. Причем для разных игр процентное соотношение этих элементов может сильно различаться. Плюс в условиях современного развития игровых технологий добавляются еще и такие немаловажные вещи, как динамическое освещение и тени. Если говорить о способах оптимизации, то оптимальным является четкая выработка требований и условий к контенту и технологиям на этапе проектирования игры и следование им на протяжении процесса разработки.

АНДРЕЙ БЕЛКИН: Самое ресурсоемкое в игре — это фонд заработной платы сотрудников. Оптимизировать можно по-разному: платить зарплаты меньших размеров или же ограничивать штат студии. Как-то оптимизировать уже готовую игру вряд ли получится и с точки зрения финансов, и с точки зрения системных требований.

МИХАИЛ ПИСКУНОВ: Это графика и логика (то есть AI).

АНДРЕЙ ТЕРТИЧНИКОВ: Самое ресурсоемкое — создание графики, потому что она занимает много места как в самом приложении, так и в памяти телефона. Так что ее оптимизация — главная задача. В принципе, оптимизацию изображений можно проводить и после отрисовки всей игры, но лучше подумать об этом заранее. На втором месте стоит оптимизация ресурсоемких участков кода: аппаратные возможности современных телефонов хоть и находятся в постоянном развитии, на данный момент времени не позволяют забывать о своем существовании.

СПЕЦ: КАКИЕ МОМЕНТЫ ВАЖНО УЧИТЫВАТЬ ПРИ РАЗРАБОТКЕ СЕТЕВЫХ ИГР?

АНДРЕЙ ТЕРТИЧНИКОВ: Учитывать надо то, что несмотря на активное развитие сетевых технологий в мобильной сфере доступ телефонов в интернет все еще сильно ограничен и скоростью, и объемами передаваемых данных. Да и стоимость трафика при активной передаче данных все-таки высока для конечного пользователя. Поэтому создавать полноценные динамичные сетевые игры (на базе интернета) пока не представляется возможным. Конечно, у современных телефонов есть еще возможность обмениваться данными с другими телефонами посредством таких технологий, как, например, Bluetooth, однако это резко сужает возможности сетевого режима


ЖУКИ@MAIL.RU
http://zhuki.mail.ru
Самая ожидаемая игра 2005 года уже на Mail.ru!
Заведи своих жуков. Тренируй их. Вырасти чемпионов тараканьих забегов!
Все подробности на http://zhuki.mail.ru
@mail.ru

комфортное программирование игр

СКРИПТОВЫЕ ЯЗЫКИ

ЧТО ХОЧЕТ ОТДЕЛ ГЕЙМДИЗАЙНА ОТ ПРОГРАММИСТА? ЧТОБЫ ПРОГРАММИСТ РЕАЛИЗОВАЛ ЛОГИКУ ИГРЫ КАК МОЖНО БЛИЖЕ К ОРИГИНАЛЬНОЙ ИДЕЕ. РАЗРАБОТКА ИГРЫ ВО МНОГОМ ЯВЛЯЕТСЯ ИТЕРАТИВНЫМ ПРОЦЕССОМ — НЕДОЧЕТЫ ОБНАРУЖИВАЮТСЯ НЕИЗБЕЖНО И В ЗАДУМКЕ ГЕЙМДИЗАЙНЕРА, И В ЕЕ РЕАЛИЗАЦИИ ПРОГРАММИСТОМ | АЛЕКСАНДР ГЛАДЫШ (WWW.STEPGAMES.RU)

Некоторые стадии процесса, такие как балансировка игры, сложно проводить до того, как появится возможность «пощупать» реализацию. Неизбежное следствие — необходимость малоэффективного итеративного взаимодействия: геймдизайнер объясняет программисту, что нужно изменить в игровом процессе, программист меняет, дизайнер обнаруживает, что хотел нечто совершенно другое, и идет объяснять заново. Чтобы повысить эффективность работы, геймдизайнеру нужен инструмент, при помощи которого он самостоятельно сможет влиять на то, что происходит в игровом мире.

какой язык выбрать

Если бы программист сам занимался геймдизайном или геймдизайнер был квалифицированным программистом, проблем бы не было. Казалось бы, самый мощный инструмент для реализации игровой логики — C++. Однако, к сожалению (скорее, к счастью), в современной игровой индустрии существует разделение труда. Следовательно, здесь работают в основном специалисты узкой квалификации. Редко встретишь хорошего геймдизайнера и квалифицированного программиста в одном лице.

Итак, нужен такой инструмент, который (при гибкости, достаточной для эффективной реализации игровой логики) был бы достаточно простым и не требовал для своего использования развитых навыков в программировании. Подобным инструментом может стать скриптовый язык программирования, подключенный к движку игры. Даже если ты и программист, и геймдизайнер в одном лице, имеет смысл вынести высокоуровневую логику в скрипты — так ты значительно облегчишь свою жизнь.

Скриптовый язык — язык расширения функциональности движка игры. «Скриптовыми» языками в подобном контексте обычно называют интерпретируемые языки программирования, обладающие динамической типизацией. В то же время скриптовым языком может считаться и, например, язык описания конфигурационных файлов игры.

Интерпретируемость языка, при наличии достаточно удобного API к интерпретатору, значительно облегчает его интеграцию в движок и ускоряет разработку, позволяя изменять код, написанный на скрипте, «на лету» без длительной перекомпиляции и, возможно, даже непосредственно во время выполнения программы.

Однако одним из важнейших преимуществ должна быть сравнительная простота языка, чтобы была обеспечена доступность для понимания и непрограммисту, и геймдизайнеру, которые занимаются высокоуровневым программированием и наладкой/балансировкой логики. В идеале, код, написанный на скриптовом языке, должен быть как можно ближе к тексту, написанному пусть и формальным, но человеческим языком. Использовать хитроумные синтаксические конструкции, которые повышают эффективность работы профессионального программиста, в данном случае вредно — понимание кода затруднено даже для обычного человека.

написать язык самостоятельно?

Часто естественное желание программиста — написать решение любой проблемы с нуля самостоятельно. Скриптовые системы в играх — не исключение. Если ты не занимаешься самообразованием, конечно, с таким желанием нужно бороться всеми силами :).

Выбирая готовую скриптовую систему, ты сможешь использовать опыт множества пользователей, которые уже наступили на основные грабли до тебя. Даже если твои требования уникальны, придется обходить только те подводные камни, которые относятся к уникальным местам твоего проекта, а не к скриптовой системе в целом.

Для готовых скриптовых языков обычно уже реализованы удобные среды разработки, отладчики, профайлеры и прочий инструментарий, который пришлось бы писать с нуля. Языки программирования собственного изготовления обычно грешат отсутствием подобных удобств, вплоть до того, что не имеют вменяемых сообщений об ошибках — на такие вещи просто не хватает времени.

Кроме того, если геймдизайнер не знаком с выбранным готовым скриптовым языком, его изучение повысит квалификацию в общем, а не станет дополнительным грузом бесполезных сведений, пригодных для использования в одном-единственном проекте. Если же геймдизайнер уже знаком с языком — еще лучше, время на изучение нового инструмента будет сэкономлено.

не обязательно писать вручную

Геймдизайнер не обязан обладать квалификацией программиста для того, чтобы иметь возможность задавать логику игры (хотя так он создал бы хорошее подспорье для себя). Часто целесообразнее написать инструментарий для «визуального программирования» логики игры, особенно тогда, когда часть логики представляется в графическом виде более наглядно. Например, удобно делать визуальным редактирование диалогов, карт переходов между локациями и т.п.

Следи за тем, чтобы твой визуальный редактор логики не превратился в неудобную замену «Блокноту». Если нужна гибкость, сравнимая с непосредственным написанием текста программы, позволь дизайнеру писать код в виде текста. Если нужно, просто введи жесткую валидацию вводимого кода — необязательно поддерживать все возможности выбранного языка, достаточно того, что требуется для комфортной реализации нужной части функциональности игры.

Оптимальный вариант — тот, при котором визуальный редактор служит ширмой для генератора кода на готовом скриптовом языке. Реализация генератора кода обычно оказывается проще, чем реализация собственного интерпретатора логики. Это также пригодится, если функциональность визуального редактора окажется недостаточной — под рукой будешь иметь всю мощь полноценного скриптового языка.

Если дизайнеру необходима возможность управлять логикой игры посредством редактирования некоторого набора данных, особенно представимых в табличном виде, рассмотри возможность использовать Microsoft Excel в качестве визуального редактора. Этот достаточно мощный инструмент предоставляет богатые возможности по расширению функциональности благодаря диалекту Visual Basic. Обычно геймдизайнеры знакомы с Excel и нередко считают его такой же родной средой для выполнения собственной работы, как программисты — Visual Studio. Excel поддерживает экспорт данных в простой текстовый формат таблиц CSV. К тому же на Visual Basic'е достаточно легко можно написать скрипт для генерации кода на скриптовом языке твоей игры.

выводы:

- РАЗРАБОТКА ИГРЫ НЕЭФФЕКТИВНА, КОГДА ГЕЙМДИЗАЙНЕРУ НУЖНЫ УСЛУГИ ПРОГРАММИСТА, ЧТОБЫ ВНЕСТИ ИЗМЕНЕНИЯ В ИГРОВОЙ ПРОЦЕСС.
- ГЕЙМДИЗАЙНЕРУ НЕОБХОДИМ ИНСТРУМЕНТАРИЙ ДЛЯ РЕАЛИЗАЦИИ ВЫСОКОУРОВНЕВОЙ ЛОГИКИ ИГРЫ И БАЛАНСИРОВКИ ГЕЙМПЛЕЯ.
- РЕШИТЬ ПРОБЛЕМУ МОЖЕТ НЕСЛОЖНЫЙ СКРИПТОВЫЙ ЯЗЫК, ПОДКЛЮЧЕННЫЙ К ДВИЖКУ ИГРЫ.
- КАК ПРАВИЛО, ВЫГОДНЕЕ ИСПОЛЬЗОВАТЬ ГОТОВЫЙ И ОТЛАЖЕННЫЙ СКРИПТОВЫЙ ЯЗЫК, А НЕ ПИСАТЬ СОБСТВЕННЫЙ С НУЛЯ.
- В НЕКОТОРЫХ СЛУЧАЯХ УДОБНЕЕ ПРОГРАММИРОВАТЬ СПЕЦИФИЧЕСКИЕ ЧАСТИ ИГРОВОЙ ЛОГИКИ В ГРАФИЧЕСКОМ, А НЕ ТЕКСТОВОМ ПРЕДСТАВЛЕНИИ. В ТАКИХ СЛУЧАЯХ МОЖЕТ ПОМОЧЬ ВИЗУАЛЬНЫЙ РЕДАКТОР, РЕАЛИЗОВАННЫЙ КАК ШИРМА ДЛЯ ГЕНЕРАТОРА КОДА НА ВЫБРАННОМ СКРИПТОВОМ ЯЗЫКЕ.

популярные открытые скриптовые системы

Готовых скриптовых систем множество, но в играх прижились единицы. Причина? В основном из-за специфических требований, предъявляемых играми к скриптам. К примеру, необходимость в высокой производительности интерпретатора.

Python (www.python.org) Open Source

Автор: Гвидо ван Россум (Guido van Rossum)
www.python.org/~guido
Год создания: 1990
Текущая версия: 2.4.2, (28 сентября 2005 г.)

достоинства:

- БОГАТЕЙШАЯ ПО ФУНКЦИОНАЛЬНОСТИ БИБЛИОТЕКА;
- БОЛЬШОЙ ОБЪЕМ ДОСТУПНОЙ ДОКУМЕНТАЦИИ И ОБШИРНОЕ КОММЬЮНИТИ;
- ПРОСТОЙ СИНТАКСИС;
- УДОБСТВО ДЛЯ НАПИСАНИЯ МЕЛКИХ УТИЛИТ;
- ПОДДЕРЖКА ЮНИКОДА.

недостатки:

- ДОСТАТОЧНО ТРЕБОВАТЕЛЕН К ОБЪЕМУ ПАМЯТИ.
- ПРОИЗВОДИТЕЛЬНОСТЬ НЕВЫСОКАЯ.
- ОРИЕНТИРОВАН, СКОРЕЕ, НЕ НА РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ПРОГРАММ, А НА РАСШИРЕНИЕ СОБСТВЕННОЙ ФУНКЦИОНАЛЬНОСТИ ЗА СЧЕТ ВНЕШНИХ МОДУЛЕЙ.
- ОФИЦИАЛЬНАЯ ИДЕОЛОГИЯ ДИКТУЕТ ЕДИНСТВЕННЫЙ МЕТОД РЕШЕНИЯ КАЖДОЙ ПРОБЛЕМЫ.
- НЕТ ВОЗМОЖНОСТИ ЗАПУСКАТЬ СКРИПТЫ «В ПЕСОЧНИЦЕ», БЕЗ ДОСТУПА К ОПЕРАЦИОННОЙ СИСТЕМЕ.
- ЧУВСТВИТЕЛЕН К КОЛИЧЕСТВУ ПРОБЕЛОВ В НАЧАЛЕ СТРОКИ, КОТОРОЕ ОПРЕДЕЛЯЕТ УРОВЕНЬ ВЛОЖЕННОСТИ КОНСТРУКЦИИ (ВМЕСТО, НАПРИМЕР, ФИГУРНЫХ СКОБОК В C/C++). ВПРОЧЕМ, К ЭТОЙ МЕЛОЧИ НЕ СЛОЖНО ПРИВЫКНУТЬ.
- В БУДУЩЕЙ ВЕРСИИ, PYTHON 3000, ПЛАНИРУЕТСЯ СЕРЬЕЗНО НАРУШИТЬ ОБРАТНУЮ СОВМЕСТИМОСТЬ СО СТАРЫМ КОДОМ НА PYTHON 2.X.

Ruby (www.ruby-lang.org) Open Source

Автор: Юкихиро Мацумото (Yukihiro Matsumoto)
en.wikipedia.org/wiki/Yukihiro_Matsumoto
Год создания: 1995
Текущая версия: 1.8.4, (12 декабря 2005 г.)

достоинства:

- СИНТАКСИС УДОБНЫЙ.
- ДОКУМЕНТАЦИЯ БОЛЕЕ-МЕНЕЕ ВМЕНЯЕМАЯ, КОММЬЮНИТИ ДОСТАТОЧНО РАЗВИТОЕ.
- ОЧЕНЬ УДОБЕН В СОЗДАНИИ СПЕЦИАЛИЗИРОВАННЫХ ЯЗЫКОВ ДЕКЛАРАТИВНОГО ХАРАКТЕРА.
- НА БУДУЩУЮ ВЕРСИЮ 2.0 ЗАПЛАНИРОВАНЫ РАБОТЫ ПО УСТРАНЕНИЮ ОСНОВНЫХ НЕДОСТАТКОВ ЯЗЫКА.

недостатки:

- СЛАБОЕ API ДЛЯ ВСТРАИВАНИЯ. НЕТ ПРОДВИНУТЫХ ОБЕРТОК, ПОДОБНЫХ BOOST.PYTHON И LUABIND.
- НЕВЫСОКАЯ ПРОИЗВОДИТЕЛЬНОСТЬ.
- ТЕКУЩАЯ ВЕРСИЯ НЕ ПОДДЕРЖИВАЕТ ЮНИКОД.
- ОРИЕНТИРОВАН, СКОРЕЕ, НЕ НА РАСШИРЕНИЕ ФУНКЦИОНАЛЬНОСТИ ПРОГРАММ, А НА РАСШИРЕНИЕ СОБСТВЕННОЙ ФУНКЦИОНАЛЬНОСТИ ЗА СЧЕТ ИСПОЛЬЗОВАНИЯ ВНЕШНИХ МОДУЛЕЙ.

Lua (www.lua.org) MIT (Open Source)

Авторы: Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes
Год создания: 1993
Текущая версия: 5.0.2, (17 марта 2004 г.)

достоинства:

- ПРОСТОЙ И ДОСТАТОЧНО УДОБНЫЙ СИНТАКСИС, БЛИЗКИЙ К КЛАССИЧЕСКОМУ.
- ХОРОШИЙ API ДЛЯ ПОДКЛЮЧЕНИЯ.
- ДОКУМЕНТАЦИЯ ХОРОШАЯ, КОММЬЮНИТИ ДОСТАТОЧНО РАЗВИТОЕ.
- СПЕЦИАЛЬНО РАЗРАБАТЫВАЛСЯ КАК ЯЗЫК ДЛЯ РАСШИРЕНИЯ ФУНКЦИОНАЛЬНОСТИ (EXTENSIBILITY LANGUAGE).
- МАЛЕНЬКИЙ И ЛЕГКИЙ (ВСЕГО 150 КБ) ИНТЕРПРЕТАТОР, ОСНОВАННЫЙ НА РЕГИСТРОВОЙ ВИРТУАЛЬНОЙ МАШИНЕ (REGISTER-BASED VIRTUAL MACHINE). ИНТЕРПРЕТАТОР LUA — ОДИН ИЗ САМЫХ БЫСТРЫХ СРЕДИ ИНТЕРПРЕТАТОРОВ СКРИПТОВЫХ ЯЗЫКОВ.
- ПОДДЕРЖКА ГИБКОГО, МУЛЬТИПАРАДИГМЕННОГО СТИЛЯ ПРОГРАММИРОВАНИЯ.
- УДОБНО ИСПОЛЬЗОВАТЬ КАК ОСНОВУ ДЛЯ СОЗДАНИЯ СПЕЦИАЛИЗИРОВАННЫХ ЯЗЫКОВ (DOMAIN SPECIFIC LANGUAGES).
- УДОБНО ОПИСЫВАТЬ ДАННЫЕ, ИНТЕРПРЕТАТОР ЭФФЕКТИВНО РАБОТАЕТ С БОЛЬШИМИ ОБЪЕМАМИ КОДА.
- ЭТОТ СКРИПТОВЫЙ ЯЗЫК ИСПОЛЬЗУЕТСЯ В ИГРОВОЙ ИНДУСТРИИ ЧАЩЕ ВСЕГО.

Недостатки:

- НЕ ПОДДЕРЖИВАЕТ ЮНИКОД ЯВНЫМ ОБРАЗОМ, ОДНАКО ПОДДЕРЖКА НУЛЕВЫХ СИМВОЛОВ В СЕРЕДИНЕ СТРОК ПОЗВОЛЯЕТ ПРОЗРАЧНО ИСПОЛЬЗОВАТЬ UTF-8.
- ОТСУТСТВИЕ НЕПОСРЕДСТВЕННОЙ ПОДДЕРЖКИ ОБЪЕКТНО ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ. КОМПЕНСИРУЕТСЯ БОЛЬШОЙ ГИБКОСТЬЮ ЯЗЫКА И НАЛИЧИЕМ СПЕЦИАЛЬНОГО «СИНТАКСИЧЕСКОГО САХАРА». ТАКЖЕ ОБЪЕКТНАЯ СИСТЕМА ПРЕДОСТАВЛЯЕТСЯ НЕКОТОРЫМИ БИБЛИОТЕКАМИ-ОБЕРТКАМИ, НАПРИМЕР LU-ABIND (**luabind.sf.net**).
- ГЛОБАЛЬНАЯ ОБЛАСТЬ ВИДИМОСТИ ПЕРЕМЕННЫХ, ПРИНЯТАЯ ПО УМОЛЧАНИЮ, ЧРЕВАТА ОШИБКАМИ.

из опыта Опыт подсказывает, что Lua — самое удобное средство, которое можно использовать в качестве скриптового языка в игре. Python, благодаря развитым библиотекам, очень удобен как язык для написания утилит. Например, именно на нем реализована довольно удачная система управления сборкой проектов Scons (**www.scons.org**) Также стоит время от времени посматривать на Ruby — этот язык имеет хороший потенциал, который, может быть, раскроется после выхода версии 2.0

[не]интересные идиоты

СКАЗ О ТОМ, КАК ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ИГРАХ ДЕЛАЮТ

ИНТЕРЕСНЫЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ ЯВЛЯЕТСЯ ОДНИМ ИЗ ВАЖНЕЙШИХ КОМПОНЕНТОВ ЛЮБОЙ УСПЕШНОЙ ИГРЫ. ОН ПРИВНОСИТ ДУХ СОРЕВНОВАНИЯ, ПРИДАЕТ УВЛЕКАТЕЛЬНОСТЬ. В ЭТОЙ СТАТЬЕ ПОГОВОРИМ О ТОМ, ЧТО ТАКОЕ ИИ В ИГРАХ, ЗА ЧТО ОН ОТВЕЧАЕТ И КАК РАБОТАЕТ |YPP (YPP_PUBLIC@MAIL.RU)

За свою более чем 50-летнюю историю развития искусственный интеллект как отрасль компьютерной науки разработан достаточно глубоко. Создано множество алгоритмов решения задач широкого спектра. Многие алгоритмы пригодны для использования и в играх, правда, здесь акценты несколько смещены. Если основная цель классического ИИ — найти правильное решение поставленной задачи, то от игрового ИИ требуют высокой производительности и «интересности». Сложно представить себе бота, которому нужно полчаса, чтобы подумать и додуматься, по какому из двух коридоров бежать или из какого оружия стрелять.

кто такие? В общем случае под искусственным интеллектом (artificial intelligence) понимают «моделирование разумного поведения с помощью компьютера». Замечу, что применительно к игровому ИИ под словом «разумный» подразумевают поведение, характерное для человека. Кому нужны такие расчетливые противники, которые знают,

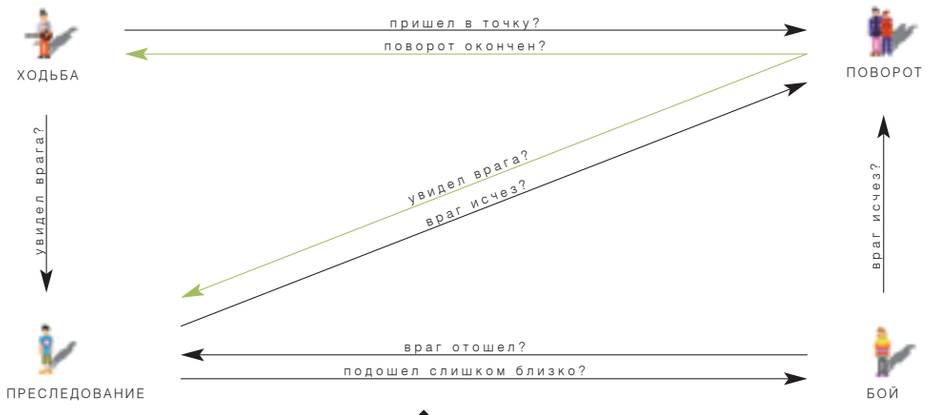


Схема состояний и переходов

куда им идти, что делать в любой ситуации, знают место расположения игрока и то, из какого оружия его легче будет подстрелить? Наверное, интереснее противники, которые иногда могут ошибаться, делать что-то не так, в общем, вести себя как обычные люди. Именно такого мы ждем от искусственного соперника, который призван заменить нам живого человека. «Агентом» будем называть носителя ИИ, то есть некую игровую сущность, которая принимает все свои решения с помощью ИИ.

Системы игрового ИИ можно условно разделить на два типа. Первый — это локальный, то есть ИИ отдельной единицы: солдат, танк, пробегающий мимо заяц или автоматическая дверь на уровне. Такие системы состоят из основных четырех элементов: система ввода информации (сенсор); память; модуль (ядро) принятия решений; система вывода (реагирования) системы. Первый элемент отвечает за то, чтобы агент воспринял все факторы окружающего мира, относящиеся к делу: положение противника, наличие препятствий, посторонние звуки и т.д. Принимая во внимание внешние факторы (ввод системы) и происходившее ранее (память), агент решает, что ему делать в следующий момент времени. Например, солдат, патрулирующий здание, должен уметь перемещаться по коридорам этого здания, видеть и (встречается нечасто) слышать. Он также должен правильно оценивать обстановку и принимать решение об отступлении/атаке/вызове подкрепления (если успеет, конечно ;)), когда встречается с врагом.

Второй тип систем ИИ отвечает за глобальные действия/события в игре. Например, в стратегических играх он управляет всей армией врага, определяет баланс сил на основе некоторых показателей (обычно: количество ресурсов, юнитов, зданий и т.д.), затем на основе этой и другой информации вырабатывает свою стратегию ведения войны. Притом каждый юнит имеет свой локальный ИИ, который помогает ему выполнять приказы глобального (перемещаться по локациям, обходя препятствия; распознавать вражеские юниты и т.д.). (В общем, как говорится, куда пехотинцев ни пошли, обязательно зайдут на поле с тибериумом и сгорят. Куда харвестер ни пошлешь, обязательно нарвется на огнеметные танки :(— прим. Лозовского.)

Другой пример. Игровой уровень сам по себе может обладать неким подобием ИИ: если игрок уничтожит N врагов, поглумится над тремя трупами и дважды подпрыгнет на левой ноге, то откроется проход на следующий уровень.

Теперь рассмотрим некоторые самые популярные технологии создания игрового ИИ.

автоматы, которые не стреляют

Первая технология, которая несмотря на свою простоту оказывается очень мощной и применяется во многих играх, — это конечные автоматы (finite state machine). Для определения такой системы нужно задать два ключевых элемента: множество состояний и множество правил перехода системы из одного состояния в другое. Состояния имеют вид наподобие: иду, стою, стреляю, бездельничаю и т.д. Правила перехода: если ИДУ и вижу врага, тогда начинаю СТРЕЛЯТЬ; если ИДУ и устал, тогда нужно немного поСТОЯТЬ, отдохнуть.

ДЛЯ БОЛЬШЕЙ НАГЛЯДНОСТИ Я НАПИСАЛ НЕБОЛЬШУЮ ПРОГРАММУ, КОТОРАЯ РЕАЛИЗУЕТ ПРИМЕР С ПАТРУЛИРУЮЩИМ СОЛДАТОМ. ТЫ НАЙДЕШЬ ЕЕ НА ДИСКЕ. ЗДЕСЬ ФИОЛЕТОВОЙ ТОЧКОЙ ОБОЗНАЧЕН ИГРОК (УПРАВЛЕНИЕ: W, A, S, D). КРАСНЫЙ ЦВЕТ — КОНТРОЛЬНЫЕ ТОЧКИ И СОЕДИНЯЮЩИЕ ИХ ОТРЕЗКИ, ЗЕЛЕНЫЙ — СОЛДАТЫ И ИХ ПОЛЕ ЗРЕНИЯ.

Рассмотрим небольшой пример. Пусть нам нужно придумать ИИ для солдата, который держит дубину и патрулирует местность. Для простоты будем считать, что на местности нет никаких препятствий, а солдат ходит по четко определенному маршруту (ключевым точкам). Итак, какие действия он может выполнять? Первое, конечно — ходить по прямой от одной контрольной точки (waypoint) к другой. Введем еще состояние поворота от одной точки по направлению к следующей. Плюс, конечно же, каждый уважающий себя солдат должен уметь преследовать жертву и молотить ее дубиной.

Теперь нужно разобраться с правилами изменения состояний. Ходьба: если увидел врага — переход в состояние преследования; если пришел в нужную точку — начать поворот к следующей. Поворот: увидел врага — преследовать; если повернулся на достаточный угол — начать ходьбу. Преследование: врага не видно — разворот к нужной точке (возврат к патрулированию); подошел достаточно близко к противнику — начать бой. Бой: враг

мнение редактора

Андрей Каролик/ Andrusha — выпускающий редактор

СОВЕРШЕННО ОЧЕВИДНО, ЧТО СЛИШКОМ СЛОЖНЫЕ ИГРЫ НЕ ИНТЕРЕСНЫ ГЕЙМЕРАМ. ЕСЛИ НА ПРОХОЖДЕНИЕ ИГРЫ ТРАТИТСЯ НЕОПРАВДАННО МНОГО ВРЕМЕНИ, ТО ВЫХОДА ДВА: УЗНАТЬ В СЕТИ, КАК ПРОЙТИ ЗАТЫК В ИГРЕ, ЛИБО БАНАЛЬНО ЗАБРОСИТЬ ВСЮ ИГРУ. НО ПАРАДОКС СОСТОИТ В ТОМ, ЧТО СЛИШКОМ ЛЕГКИЕ ИГРЫ ТОЖЕ МАЛОИНТЕРЕСНЫ. ТАК ЧТО ОСНОВНАЯ ЗАДАЧА — СДЕЛАТЬ НЕ СВЕРХУМНЫЙ AI, КАК ОШИБОЧНО ПОЛАГАЮТ МНОГИЕ, А ОПТИМАЛЬНЫЙ AI, ОРИЕНТИРУЯСЬ НА НЕКИЙ СРЕДНИЙ УРОВЕНЬ СЛОЖНОСТИ, ПРИЕМЛЕМЫЙ ДЛЯ БОЛЬШИНСТВА ГЕЙМЕРОВ. В ИТОГЕ БОЛЬШИНСТВО ПОПУЛЯРНЫХ ИГР — ЭТО НЕКИЙ ИДЕАЛЬНЫЙ, ЕСЛИ ЕГО ТАК МОЖНО НАЗВАТЬ, AI, НО С ЯВНЫМИ БРЕШАМИ, КОТОРЫЕ ЗАЛОЖЕНЫ В НЕГО ИЗНАЧАЛЬНО. И ВСЯ ИГРА СВОДИТСЯ К ПОИСКУ И ОСОЗНАНИЮ ЭТИХ БРЕШЕЙ, ЧТОБЫ ИСПОЛЬЗОВАТЬ ИХ И ПОБЕДИТЬ ПРОТИВНИКА.

отошел дальше — преследовать; враг исчез (умер, сбежал) — поворот. Вот и все. Когда уже определены все состояния и переходы, закодить такую систему не составит большого труда.

Теперь немного о преимуществах и недостатках. Из преимуществ можно назвать относительную простоту в понимании, написании и отладке небольших систем, а также высокую производительность в работе. Недостатки: при большом количестве состояний и переходов реализация превращается в настоящий кошмар; такие системы трудно расширять — логика состояний и переходов жестко зашита в коде системы.

Конечные автоматы являются, пожалуй, одной из самых старых (и проверенных временем) техник создания ИИ, особенно в FPS. Наши любимые Doom, Quake и Wolfenstein, — все имели ИИ, построенный на основе конечных автоматов.

добавим перцу

Сейчас посмотрим, как можно усовершенствовать конечные автоматы, чтобы системы на их основе выглядели еще лучше. Для начала замечу, что конечные автоматы бывают полностью детерминированными и недетерминированными. Различие состоит в том, что в недетерминированных переходы между состояниями являются не такими жесткими и имеют элемент случайности, непредсказуемости. Если в примере выше «научить» солдата с 10% вероятностью не замечать врага, а в случае исчезновения врага в 50% случаев возвращаться к патрулированию и еще в 50% — начинать поиски сбежавшей жертвы, то такая система уже становится недетерминированной.

Создавать конечный автомат с большим количеством состояний и правил перехода — задача не из легких. Иногда бывает проще разделить систему ИИ на несколько параллельно действующих и (почти) не связанных конечных автоматов, каждый из которых имеет свои состояния. Например, один автомат отвечает за патрулирование/преследование/атаку, а второй — за систему ведения огня: стрелять, не стрелять, перезарядить и т.д.

Еще более удивительных результатов можно добиться при синхронизации конечных автоматов, работающих параллельно. Допустим, в нашей игре будут присутствовать отряды из трех солдат, которые охраняют некоторые локации. Конечно, каждый солдат может реализовывать свою логику ведения боя, но представь, насколько интереснее они смотрятся, если действуют слаженно. Например, один из солдат, заметив противника, сообщает об этом по рации (записывает это в общую память отряда) и принимает бой. Получив из общей памяти информацию о происходящем, другой солдат прикрывает товарища издалека («сообщая» об этом в ту же общую память). Наконец, третий боец начинает закидывать врага гранатами. Ну что, ничего не напоминает? (Даю наводку: Гордон Фримэн, монтировка, бригады солдат из первой Half-Life...) То-то же!

жизнь по правилам

Есть ситуации, в которых довольно проблемно выразить поведение игрового агента в терминах конечных автоматов. Например, когда из каждого состояния агент может переходить во все другие состояния или когда агент может пребывать в двух состояниях одновременно («идет» и «стреляет»). В таких случаях можно применить другую технологию — систему правил (rule system). Все поведение агента описывается набором простых правил вида: условие->действие. Сначала выбираем первое правило. Если условие в нем истинно, выполняем указанное действие и выходим (прекращаем проверку последующих условий). В противном случае переходим к следующему правилу, если таковые имеются. Как всегда, пример. Попробуем описать с помощью системы правил поведение юнита из одной RTS:

1. ВРАГ БЛИЗКО-> ДАТЬ ЕМУ ПО МОРДЕ
2. РАССТОЯНИЕ К ВРАГУ> ДВУХ МЕТРОВ && МЫ СИЛЬНЕЕ-> ДОГНАТЬ
3. РАССТОЯНИЕ К ВРАГУ> ДВУХ МЕТРОВ (ОН СИЛЬНЕЕ)-> ОЙ, ПОРА БЕЖАТЬ!
4. НАШИХ БЬЮТ-> ПОЙТИ НА ПОМОЩЬ
5. СТОЯТЬ И ЛЮБОВАТЬСЯ ПЕЙЗАЖАМИ

Замечу, что в подобных системах порядок указания правил имеет большое значение. Например, если поменять местами правила №2 и 3, юнит начнет убегать и от сильных, и от слабых врагов. (Такое поведение системы объясняется тем, что условие в правиле №3 включает в себя условие из правила №2.) Также замечу, что пятое правило вообще не имеет условия. Оно задает действие по умолчанию и равноценно правилу: true-> стоять и любоваться пейзажами.

На практике системы правил реализуются двумя способами. Первый — дерево принятия решений (decision tree), которое выглядит приблизительно так:

```
if ( условие1 )
  действие1;
else
  if ( условие2 )
    действие2;
  else
    // и так далее...
```

Другой способ реализации системы правил опирается на скриптовые языки и является более гибким, так как в этом случае вся логика работы ИИ отделена от основного кода движка. Скриптовые системы правил применяются шире всего в стратегических играх, например Age of Empires.

Две следующие технологии более современные и берут свое начало из биологии — совокупности наук о живой природе. И правда, где еще можно узнать столько о существах, которых мы так старательно пытаемся смоделировать?

по стопам дедушки дарвина

Применять генетические алгоритмы в играх начали сравнительно недавно. Они хорошо подходят в ситуациях, в которых нужно создать разнообразие су-

Системы правил в действии

Отряд из Half-Life

ществ, несколько схожих по внешнему виду и/или поведению. Примером тому могут служить пешеходы в гоночных симуляторах или население некого виртуального города/мира (как в GTA или The Sims).

Из школьного курса биологии нам всем известно, что в каждом из живых существ зашит некий генетический код (ДНК помнишь?). Причем существа в пределах одного вида имеют схожий по структуре код, разный по своим составляющим единицам — генам. Каждый ген кодирует некое свойство живого организма, например цвет глаз, тип волос. Генетический код каждого индивида является некой комбинацией генов его родителей плюс небольшой процент мутаций (вообще-то в биологии имеются более строгие правила наследования, но нам это не так важно). Основываясь на теории генов, Чарльз Дарвин вывел свою знаменитую теорию эволюции (что-то подсказывает мне, что Дарвин вообще ничего не знал о генах: когда Чарльз писал свои книги, Мендель еще не опубликовал даже свои опыты :) — прим. Лозовского). Суть теории Дарвина известна всем: выживает сильнейший, то есть самый приспособленный. У одних видов показателем «силы» может быть скорость, у других — устойчивость к морозам, у третьих — собственно физическая сила. На основе этих базовых концепций и создают ИИ в некоторых играх. Для начала зададим структуру ДНК как массив генов и их возможных значений. Не забудем учесть, что здесь понятие гена может быть намного шире и с его помощью удобно кодировать не только биологические свойства, но и, например, стиль одежды, черты характера, прическу и т.д. (см. таблицу).

Дальше генерируем первое поколение из N представителей данного вида выбором случайных значений из множества возможных (для каждого гена). Следующая процедура называется тестом на пригодность (fitness testing), она же является одной из самых сложных. Сгенерированных индивидов помещаем в реальный игровой мир и выбираем наиболее «способных». К примеру, для бойцов можно устраивать виртуальные поединки, чтобы решить, кто лучше. Потом генерируем второе поколение, создавав N индивидов как линейной комбинации ДНК отобранных. Также можно добавить небольшую степень мутаций. Дальше — снова тест на пригодность и т.д. В конечном счете получается, что чем больше номер поколения, тем лучше особи. И еще мысль: если продолжать «профотбор» и во время игры, то получится, что каждый следующий противник будет более приспособлен к конкретному игроку. Возникает некое подобие самообучения системы ИИ.

кто к нам в сети попадет Нейронные сети являются, пожалуй, одной из самых продвинутых и перспективных технологий создания ИИ. Применять нейронные сети в играх начали сравнительно недавно из-за их сложности и высокой требовательности к вычислительным ресурсам. Суть заключается в попытке математического моделирования работы головного мозга человека, который состоит из множества нейронов, соединенных между собой. Процесс моделирования состоит из двух основных этапов: обучение и собственно использование. Два самых распространенных способа обучения: 1) на вход системы подают сигналы и результаты, которые нужно получать при таких входах; 2) на вход подаются сигналы определенного вида, которые система должна самостоятельно структурировать и научиться отличать от других (самообучение). На математическом уровне обучение происходит путем подбора системой нужных весовых коэффициентов (weights) в уравнениях. После некоторого времени обучения нейронная сеть уже готова к использованию, а также последующему обучению.

Революционность нейронных сетей применительно к игровому ИИ состоит в том, что теперь виртуальные соперники получают возможность самообучаться. К примеру, вражеский солдат, получив несколько пулевых ранений во время отчаянной попытки лобовой атаки, запомнит, что от пуль больно и еще раз «так» лучше не делать. Или он сможет по чуть-чуть перенимать тактику ведения боя у игрока. Вот это действительно захватывающе!

«теперь и ты знаешь кунг-фу» Вот, в общем, и все. Мы рассмотрели несколько самых мощных технологий создания ИИ в играх, и тебе остается только посмотреть на наш диск, где мы заготовили несколько приятных программерских бонусов. Ну и, конечно же, дам парочку полезных советов напоследок :).

Прежде всего стоит помнить, что разработка плохого ИИ часто начинается с ошибочной постановки целей. Подумай над тем, какие цели поставлены перед системой, для чего конкретно она создана, а главное — каких результатов ты ждешь от нее. Например, было бы глупо и страшно расточительно биться в муках несколько дней, прикручивая супертехнологичный ИИ к тупому зомби в новом клоне Crimsonland: ему потребуется только знать, где находится игрок, и двигаться в этом направлении :)

а что дальше?

ИТАК, ЧЕГО ЖЕ СЛЕДУЕТ ОЖИДАТЬ ОТ ИСКУССТВЕННЫХ ПРОТИВНИКОВ УЖЕ ЗАВТРА? ПРОИЗВОДИТЕЛЬНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ РАСТЕТ СТРЕМИТЕЛЬНО, БЮДЖЕТЫ НА ПРОИЗВОДСТВО ИГР УВЕЛИЧИВАЮТСЯ ПОСТОЯННО, ЧТО ПРИВЕДЕТ В ПЕРВУЮ ОЧЕРЕДЬ К ПРИМЕНЕНИЮ БОЛЕЕ СЛОЖНЫХ ТЕХНОЛОГИЙ И АЛГОРИТМОВ, КОТОРЫМ ТРЕБУЮТСЯ БОЛЬШИЕ ВЫЧИСЛИТЕЛЬНЫЕ МОЩНОСТИ. УЖЕ СЕГОДНЯ НАЛИЧИЕ «ЧЕСТНОЙ ФИЗИКИ» СИЛЬНО УСЛОЖНЯЕТ ЖИЗНЬ СОЗДАТЕЛЮ ИГРОВОГО ИИ. ТАКЖЕ В ПОСЛЕДНИЕ ГОДЫ НАМЕЧАЕТСЯ ТЕНДЕНЦИЯ К РАЗВИТИЮ ТАКИХ НАПРАВЛЕНИЙ, КАК САМООБУЧЕНИЕ, УМЕНИЕ ВЗАИМОДЕЙСТВОВАТЬ С ИНТЕРАКТИВНЫМ ОКРУЖАЮЩИМ МИРОМ И НАЛАЖИВАНИЕ ЧЕТКОГО ВЗАИМОДЕЙСТВИЯ МЕЖДУ АГЕНТАМИ. ВСЕ ЭТО СЛУЖИТ ТОМУ, ЧТОБЫ ИГРЫ СТАЛИ ИНТЕРЕСНЕЕ И УВЛЕКАТЕЛЬНЕЕ.

Набор генов уличного бойца

генотип	возможные значения
Прическа	лысый, короткая стрижка, панк, косичка
Одежда	шорты, белая майка и синие джинсы, кимоно, спортивный костюм
Техника	бокс, кикбоксинг, каратэ, кикбоксинг, самбо, тайский бокс
Скорость	5–10
Сила	1–10
Спец. удар	нет, вертушка, аперкот снизу

рецепты lua!

СОВЕТЫ ПО ПРОГРАММИРОВАНИЮ ИГР

LUA (www.lua.org) — ОДИН ИЗ СКРИПТОВЫХ ЯЗЫКОВ, ИСПОЛЬЗУЕМЫХ В ИГРАХ ЧАЩЕ ВСЕГО. ЭТОТ ЯЗЫК ПОПУЛЯРЕН НАСТОЛЬКО В ОСНОВНОМ БЛАГОДАРЯ ПРОДУМАННОМУ ДИЗАЙНУ И РЕАЛИЗАЦИИ ЯЗЫКА |АЛЕКСАНДР ГЛАДЫШ (WWW.STEPGAMES.RU)

«ВСЕ О БЕТА-ТЕСТИРОВАНИИ LUA 5.1 — НА WWW.INF.PUC-RIO.BR/~ROBERTO/TALKS/NOVELTIES.PDF»

Первая версия языка Lua была создана в 1993 году сотрудниками научно-исследовательской группы по технологиям компьютерной графики (Computer Graphics Technology Group, TeCGraf) Бразильского папского католического университета (Pontifical Catholic University, PUC-Rio) в Рио-де-Жанейро. Над языком работали три человека: Роберто Иерусалимский (Roberto Ierusalimschy), Луиш Энрике де Фигуиредо (Luiz Henrique de Figueiredo) и Вальдемар Челеш (Waldemar Celes). Название языка на португальском означает «луна» и произносится как «луа» (LOO-ah). Истории развития языка посвящена специальная страница его официального сайта — **www.lua.org/versions.html.**

В 2003 году выпущена версия языка 5.0.2, последняя на момент написания статьи (следующая версия 5.1 находится на стадии бета-тестирования) — Lua обрел зрелость и стабильность. Его интерпретатор, реализованный на основе регистровой (register-based) виртуальной машины, очень эффективен и по производительности, и по объему занимаемой памяти. В сборке со всеми стандартными библиотеками языка он добавляет к исполняемому файлу программы всего порядка 150 Кб (100 Кб без библиотек).

Интерпретатор Lua написан на «чистом» (clean) C (подмножестве языков ANSI C и C++), его можно собрать на любой платформе, для которой существует компилятор, поддерживающий стандарт ANSI C. Высокая портируемость (portability) языка обеспечена!

Изначально Lua был задуман как язык расширения функциональности приложений (extension language), и поэтому он обладает достаточно удобным в этом плане интерфейсом (так называемый Lua C API).

гибкий синтаксис Lua обладает несложным синтаксисом, близким к традиционному и несколько напоминающим Pascal. Часто элементы синтаксиса носят необязательный характер, что позволя-

сборка версии 5.0.2

LUA С API ВЕРСИИ 5.0.2 ПО УМОЛЧАНИЮ СОСТОИТ ИЗ ДВУХ СТАТИЧЕСКИХ БИБЛИОТЕК: ЯДРА И НАБОРА СТАНДАРТНЫХ БИБЛИОТЕК ЯЗЫКА (В ВЕРСИИ 5.1 ЭТИ БИБЛИОТЕКИ УЖЕ ОБЪЕДИНЕНЫ В ОДНУ). ДИСТРИБУТИВ LUA НАХОДИТСЯ НА ОФИЦИАЛЬНОМ САЙТЕ ЯЗЫКА (**www.lua.org**). МОЖЕШЬ СКАЧАТЬ ПРЕКОМПИЛИРОВАННУЮ ВЕРСИЮ LUA С САЙТА LUAFORGE (**luabinaries.luaforge.net**).

ет писать на Lua и в C-подобном стиле, и в стиле, который при определенном старании можно приблизить к некоему подобию формализованного человеческого языка благодаря отсутствию избыточных синтаксических конструкций. Традиционный пример Hello world на Lua состоит из одной строчки:

```
print(“Hello, world!”);
```

Синтаксис языка позволяет при желании опустить скобки и точку с запятой. Таким образом, вот эта конструкция эквивалентна первой:

```
print “Hello, world”
```

Обычно сердцу программиста милее конструкция со скобками: ее легче воспринимает взгляд, натренированный работой с C-подобными языками. Однако язык Lua удобно использовать в качестве языка, специализированного под конкретную задачу (Domain-Specific Languages, DSL), например как язык описательного характера, такой как «язык» конфигурационного файла программы. Свобода в синтаксисе Lua бывает очень полезна для повышения выразительности подобных языков.

В Lua также удобно хранить табличные данные, причем их можно экспортировать непосредственно из Excel, написав маленький скрипт-кодогенератор на Visual Basic. Значит, больше не нужен лишний парсер данных (например, в формате CSV), парсер и интерпретатор Lua сделают все за тебя.

на полную мощность Часто при разработке архитектуры скриптовой системы и при непосредственной разработке на Lua программисты допускают ошибку — пытаются писать на этом языке так, как будто он некое бледное подобие C++. Lua — мультипарадигменный язык, гибкий в той степени, чтобы допускать разработку в стиле C++. Однако удобнее и эффективнее рассматривать его как самостоятельную сущность и вести разработку исходя из всего богатства возможностей, в том числе работать с возможностями, которых или нет в C++, или использование которых затруднительно.

Функции в Lua — значения первого класса. В Lua есть замыкания (closures). Функция может возвращать список значений... Если ты помнишь обо всем этом, то значительно облегчишь свой труд по разработке проекта. Подход с использованием корутин (coroutines) дает хорошее преимущество в объеме и прозрачности кода (если сравнивать с классическими системами взаимодействия с движком, основанными исключительно на обратных вызовах (callbacks) и/или событиях (events). Благодаря встроенным в язык возможностям рефлексии (например, функции type и функциям работы с таблицами), на Lua очень удобно реализуются связанные с ней задачи, скажем сериализация (пример такой реализации есть в книге Programming In Lua). Когда в движке требуются рефлективные возможности (для сериализации или автоматической генерации интерфейса редактирования данных), не пытайся прикрутить рефлексию к C++, а используй возможности Lua.

консоль игры В поставку Lua, помимо исходников библиотеки для программной работы с интерпретатором (Lua C API), входят исходники интерактивного интерпретатора (lua.exe), очень удобного для изучения языка, быстрого тестирования и отладки небольших кусков кода. Пользуясь им, ты видишь результат выполнения написанного кода не отходя от кассы. Очень удобно реализовать функциональность, аналогичную той, которой обладает интерактивный интерпретатор Lua, в консоли игры. Тогда ты и твои дизайнеры смогут эффективно писать код в контексте твоего движка и немедленно просматривать то, как изменения в коде влияют на игровой мир.

перенаправление вывода Посмотрим такой пример. В Lua нужно переопределить функцию print, чтобы она выводила текст не в стандартный поток вывода, а в игровую консоль. Функция print в Lua является частью базовой библиотеки языка и принимает список аргументов переменной длины. Она преобразует каждый из аргументов в строку при помощи функции tostring() и выводит ее разделяя символами табуляции.

Функции в Lua — значения первого класса (first-class values), их можно присваивать переменным, передавать в качестве аргументов и возвращать из других функций. Благодаря этому переопределение любой функции в Lua — не больше чем присвоение переменной с именем этой функции нового значения. Если у тебя есть функция __console_print, которая принимает строку и выводит ее в консоль, то переопределение системной функции print в Lua будет выглядеть, например, вот так:

```
print = function(...)
for i = 0, arg.n -1 do
__console_print(tostring(arg[i]) .. "\t")
end
__console_print(arg[arg.n] .. "\n")
end
```

Если нужна высокая гибкость или производительность, ты можешь написать свою функцию __console_print на C/C++, которая работает непосредственно с виртуальным стеком интерпретатора, по образцу print из базовой библиотеки функций (luaB_print() в файле lbaselib.c исходников Lua). Фактически ты только продублируешь ее исходный код, заменив в нем вызов fputs() на вызов нужной функции. Соответственно, для замены стандартного print в Lua достаточно одной строчки:

```
print = __console_print
```

При необходимости аналогичным образом переопределяешь и другие стандартные функции, работающие с вводом-выводом.

дополнительные источники информации

WWW.LUA.ORG — ОФИЦИАЛЬНЫЙ САЙТ ЯЗЫКА LUA. ДИСТРИБУТИВ, ОФИЦИАЛЬНОЕ РУКОВОДСТВО LUA 5.0 REFERENCE MANUAL, А ТАКЖЕ КНИГА РОБЕРТО ИЕРУСАЛИМСКОГО PROGRAMMING IN LUA, В КОТОРОЙ ИЗЛОЖЕНЫ ОСНОВНЫЕ ПРИЕМЫ РАБОТЫ С ЯЗЫКОМ.
WWW.LUA-USERS.ORG — САЙТ СООБЩЕСТВА ПОЛЬЗОВАТЕЛЕЙ LUA. СОДЕРЖИТ LUA USERS WIKI, В КОТОРОЙ МОЖНО НАЙТИ МАССУ ПОЛЕЗНЫХ СТАТЕЙ, НАПИСАННЫХ КАК РАЗРАБОТЧИКАМИ ЯЗЫКА, ТАК И ПРОСТЫМИ ПОЛЬЗОВАТЕЛЯМИ.
WWW.LUAFORGE.NET — СОБРАНИЕ ПРОГРАММ, НАПИСАННЫХ НА LUA, ИСПОЛЬЗУЮЩИХ LUA ИЛИ СВЯЗАННЫХ С НИМ.
WWW.INF.PUC-RIO.BR/~ROBERTO — ОФИЦИАЛЬНАЯ WEB-СТРАНИЦА ОДНОГО ИЗ АВТОРОВ ЯЗЫКА РОБЕРТО ИЕРУСАЛИМСКОГО. СОДЕРЖИТ, В ЧАСТНОСТИ, ТЕКСТЫ ЕГО ДОКЛАДОВ ПО ЯЗЫКУ.
WWW.KEPLERPROJECT.ORG — ПРОЕКТ АВТОРОВ ЯЗЫКА ПО СОЗДАНИЮ ДВИЖКА ДЛЯ РАЗРАБОТКИ WEB-САЙТОВ НА LUA. МОЖНО РАЗДОБЫТЬ ПОЛЕЗНЫЕ МОДУЛИ, ТАКИЕ КАК LUASOCKET, LUAPROFILER И LUALOGGING.

подключение игры к движку Вместе с языком для подключения языка к твоей программе поставляется библиотека на C (Lua C API). Для взаимодействия с языком C интерпретатор Lua использует виртуальный стек. Когда из скрипта вызывается функция на C, она получает чистый экземпляр стека, который независим от стеков других функций, вызванных на данный момент, и содержит аргументы, переданные в данную функцию. Значения, возвращаемые функцией, тоже передаются через стек. Большинство функций Lua C API работают и с виртуальным стеком. Для удобства эти функции не следуют строгой стековой дисциплине, когда можно использовать только операции push и pop, но позволяют обращаться к значениям, содержащимся в стеке, по их абсолютной или относительной позиции. Подробности о работе с виртуальным стеком интерпретатора Lua расписаны в Lua Reference Manual.

Чаще всего удобнее оказывается работа на более высоком уровне, который предоставляют различные библиотеки-обертки. Тем не менее полезно уметь работать с интерпретатором непосредственно на уровне Lua C API, хотя бы чтобы понять, что происходит на самом деле. К тому же для некоторых специфических задач бывает просто недостаточно функциональности, предоставляемой библиотеками-обертками.

Язык Lua специально создавался как язык расширения функциональности (extension programming language), поэтому достаточно легко ор-

ганизовать взаимодействие кода на Lua с кодом на C (и, следовательно, C++).

Помимо Lua C API, существует некоторое количество оберток для этой библиотеки, призванных повысить удобство работы с Lua из C и C++. Такие обертки делятся на две основные группы: 1) генерируют прослойку межъязыкового взаимодействия автоматически на основе данных, подготовленных специальным образом; 2) служат для облегчения ручного создания такой прослойки.

Несмотря на обилие оберток для подключения Lua, стоит изучить Lua C API — получишь самое полное представление о том, что происходит в программе. К тому же при всей своей мощи сам API достаточно небольшой. Необходимую информацию ищи в Lua Programming Manual и Programming With Lua.

На сегодня из библиотек для обеспечения межъязыкового взаимодействия C++ с Lua наиболее развиты:

- TOLUA, TOLUA++ (**www.tecgraf.puc-rio.br/~celes/tolua, www.codenix.com/~tolua**);
- SWIG (**www.swig.org**);
- LUABIND (**luabind.sourceforge.net**).

И toLua, и SWIG генерируют код регистрации по специально подготовленным входным файлам (SWIG умеет генерировать такой код не только для Lua, но и для множества скриптовых языков).

Luabind использует шаблоны C++ для генерации кода регистрации твоих типов в Lua, поэтому он избавляет от промежуточного этапа генерации кода сторонними средствами за счет некоторого оверхеда в скорости компиляции и объемах конечного исполняемого файла. По моему опыту, удобнее всего пользоваться библиотекой Luabind в сочетании с прямым использованием Lua C API (чтобы реализовать отдельные тонкие моменты).

вопросы производительности

При разработке программ на Lua нужно иметь в виду, что компилятор этого языка не обладает широкими способностями по оптимизации. С другой стороны, интерпретатор Lua реализован очень эффективно и чаще всего обеспечивает весьма приличную скорость работы. Если же тебе нужна сверхвысокая производительность, ты должен сам следить за тем, чтобы в коде не было множества лишних конструкций. Добиться этого будет легче, если будешь придерживаться нескольких простых правил при написании кода на Lua (подробнее о подходах к оптимизации кода на Lua — в материалах, опубликованных в Lua Users Wiki).

Доступ к локальным переменным в Lua несколько быстрее, чем к глобальным. Если требуется интенсивный доступ к глобальной таблице, функции или корутине, лучше завести локальную, присвоить ей значение глобальной и только потом начинать использовать. Эта техника не несет накладных расходов на копирование, так как в Lua данные таких типов копируются как ссылки, а не как значения.

По возможности в коде, критичном по производительности, не создавай множество объектов Lua, управляемых сборщиком мусора. Такие объекты создаются при склеивании строк, вызове конструкторов таблиц (например при вызовах функций с переменным числом аргументов), при объявлении любых функций, при выполнении команд dofile/dostring.

Компилятор Lua способен выполнять элементарную оптимизацию. Например, он оптимизирует простые выражения с участием переменных и констант. Не обязательно заменять, скажем, выражение:

```
a = 2 * 34567 + b
```

на

```
a = 69134 + b.
```

Перевод скриптового кода в код на C++ — один из действенных методов оптимизации. Если тебе вдруг показалось, что какой-то участок кода на Lua нужно подвергнуть интенсивной оптимизации, может быть, ты обнаружил первый сигнал к тому, что этот кусок кода нужно переписать на C++.

Заботясь о производительности кода, остерегайся преждевременной оптимизации. По возможности соблюдай рекомендации по написанию оптимального по производительности кода — писать гарантированно медленный код не стоит. Однако, как известно, затраты, скажем, на оптимизацию куска кода, который выполняется не более 1% времени, не уменьшат временные затраты на выполнение программы больше чем на 1%.

Сначала реализуй код самым удобным (по твоим ощущениям) способом. Если понадобится, собирай статистику с помощью таких инструментов, как Lua Profiler (**www.keplerproject.org/luaprofiler**). И оптимизируй только то, что действительно должно быть оптимизировано.

обработка ошибок

Важно помнить, что при небрежном отношении к обработке ошибок в Lua, после возврата ошибки из lua_pcall(), скриптовая система может оказаться в невалидном состоянии.

Безопаснее всего (по крайней мере, в стадии активной разработки) прекращать выполнение программы при возникновении ошибок в скрипте. Естественно, это не относится к консольному интерпретатору, в котором для комфортной работы нужно обеспечивать толерантность к возможным ошибкам и опечаткам. Тем не менее желательно принимать меры к тому, чтобы ошибки в коде, выполняемом с консоли, не приводили к неявной «порче» игрового мира.

чтобы ошибок было меньше, вот несколько простых правил на заметку:

- ПРИ ВОЗНИКНОВЕНИИ ОШИБКИ В ФУНКЦИЯХ НА LUA ВМЕСТО ВЫЗОВА БИБЛИОТЕЧНОЙ ФУНКЦИИ ERROR() ВОЗВРАЩАЙ NIL И ТЕКСТ СООБЩЕНИЯ ОБ ОШИБКЕ.
- ФУНКЦИИ, КОТОРЫЕ МОГУТ ВЫЗВАТЬ ERROR(), ВЫЗЫВАЙ (ГДЕ ВОЗМОЖНО) ЧЕРЕЗ БИБЛИОТЕЧНУЮ ФУНКЦИЮ PCALL(), ДЛЯ ЧЕГО ЛУЧШЕ ИСПОЛЬЗОВАТЬ ИДИОМУ PROTECT, ОПИСАННУЮ В СТАТЬЕ ДИЕГО НЕХАБА (DIEGO NEHAB) FINALIZED EXCEPTIONS (**www.lua-users.org/wiki/FinalizedExceptions**). ТАКЖЕ ПОЛЕЗНО ПОЛЬЗОВАТЬСЯ ИДИОМОЙ NEWTRY (СМ. ТУ ЖЕ СТАТЬЮ).
- ПРОВЕРЯЙ ПЕРЕДАННЫЕ ФУНКЦИИ ПАРАМЕТРЫ НА NIL. ЕСЛИ, НАПРИМЕР, ФУНКЦИЯ, ОБЪЯВЛЕННАЯ С ТРЕМЯ ПАРАМЕТРАМИ, ВЫЗЫВАЕТСЯ С ДВУМЯ, ТО ТРЕТЬЕМУ ПАРАМЕТРУ БУДЕТ ПРИСВОЕН NIL.
- ЕСЛИ ТВОЯ ФУНКЦИЯ РАБОТАЕТ С ГЛОБАЛЬНЫМИ ДАННЫМИ, ОСОБЕННО С ОПИСАНИЯМИ УРОВНЕЙ, ТЕКСТАМИ ДИАЛОГОВ И Т.П., ПО ВОЗМОЖНОСТИ ПРОВЕРЯЙ ЭТИ ДАННЫЕ НА NIL.
- ОЧЕНЬ ЧАСТО ОШИБКИ ВОЗНИКАЮТ ИЗ-ЗА ТОГО, ЧТО ПРОГРАММИСТ ЗАБЫЛ НАПИСАТЬ КЛЮЧЕВОЕ СЛОВО LOCAL ПЕРЕД ПЕРВЫМ ПРИСВАИВАНИЕМ ЛОКАЛЬНОЙ ПЕРЕМЕННОЙ И ЗАТЕР ТАКИМ ОБРАЗОМ ГЛОБАЛЬНУЮ. СЛЕДИ ЗА ОБЛАСТЬЮ ВИДИМОСТИ ТВОИХ ПЕРЕМЕННЫХ. В НАЗВАНИЯХ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ ИСПОЛЬЗУЙ УНИКАЛЬНЫЙ ПРЕФИКС, НАПРИМЕР G_. ПРИМЕНЯЙ ИНСТРУМЕНТЫ ТИПА LUA LINT ДЛЯ ПРОВЕРКИ ИСПОЛЬЗОВАНИЯ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ. В ДИСТРИБУТИВ LUA 5.1 ВХОДИТ ПРИМЕР НА LUA, КОТОРЫЙ НАСТРАИВАЕТ МЕТАТАБЛИЦУ ТАБЛИЦЫ, СОДЕРЖАЩЕЙ ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ ("_G"), ТАКИМ ОБРАЗОМ, ЧТО ПРИ ДОБАВЛЕНИИ ИЛИ ЧТЕНИИ НЕЗАРЕГИСТРИРОВАННЫХ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ ВО ВРЕМЯ ВЫПОЛНЕНИЯ СКРИПТА ВЫДАЕТСЯ СООБЩЕНИЕ ОБ ОШИБКЕ.
- СТАРАЙСЯ МИНИМИЗИРОВАТЬ ИСПОЛЬЗОВАНИЕ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ. СОБЛЮДАЙ МОДУЛЬНОСТЬ КОДА. СЧИТАЕТСЯ ХОРОШИМ ТОНОМ, ЕСЛИ ОДИН МОДУЛЬ СОЗДАЕТ ТОЛЬКО ОДНУ ГЛОБАЛЬНУЮ ПЕРЕМЕННУЮ. ПОДРОБНЕЕ О ПОДХОДАХ К ОРГАНИЗАЦИИ МОДУЛЕЙ В LUA МОЖНО ПРОЧИТАТЬ В LUA TECHNICAL NOTE 7: MODULES & PACKAGES РОБЕРТО ИЕРУСАЛИМСКОГО (**www.lua.org/notes/ltn007.html**).
- ИЗБЕГАЙ СОЗДАВАТЬ НА СКРИПТОВОМ ЯЗЫКЕ СЛИШКОМ СЛОЖНЫЕ СИСТЕМЫ. ДИНАМИЧЕСКАЯ ТИПИЗАЦИЯ И ПРОЧИЕ ОСОБЕННОСТИ, ПРИ ВСЕМ СВОЕМ

УДОБСТВЕ, ПЛОХО ВЛИЯЮТ НА СТАБИЛЬНОСТЬ, НАДЕЖНОСТЬ И ПРОИЗВОДИТЕЛЬНОСТЬ КОДА. СКРИПТОВЫЕ ЯЗЫКИ ВООБЩЕ И LUA В ЧАСТНОСТИ ХОРОШИ ДЛЯ «СКЛЕЙКИ» И НАСТРОЙКИ ФУНКЦИОНАЛЬНОСТИ ИГРОВОГО ДВИЖКА. ЕСЛИ ТЫ ВИДИШЬ, ЧТО КАКОЙ-ТО МОДУЛЬ НА LUA СТАНОВИТСЯ СЛИШКОМ СЛОЖНЫМ, ПОСТАРАЙСЯ ПЕРЕНЕСТИ ЕГО ФУНКЦИОНАЛЬНОСТЬ НА C++, ОСТАВИВ LUA ТОЛЬКО ВЫСОКОУРОВНЕВОЕ УПРАВЛЕНИЕ ЭТИМ МОДУЛЕМ.

управление памятью Язык Lua автоматически управляет памятью при помощи сборщика мусора (garbage collector): интерпретатор периодически вызывает сборщик мусора, удаляющий объекты, для которых была выделена память (таблицы, userdata, функции, потоки и строки) и которые стали недоступными из Lua («мертвые» объекты).

Интерпретатор Lua задает предел объема памяти (threshold), занимаемого данными. Как только занятый объем достигнет этого предела, запускается алгоритм, освобождающий память, занятую накопившимися «мертвыми» объектами. Затем устанавливается новый предел, равный двухкратному объему памяти, занимаемой после очистки.

Чтобы запустить сборку мусора немедленно, нужно программно установить предел занимаемой памяти в 0 (вызвав lua_setgcthreshold() из C или collectgar-bage() из Lua). Чтобы остановить сборку мусора, устанавливаем этот предел в достаточно большое значение.

Замечено, что в некоторых случаях память, занимаемая данными Lua, проявляет тенденцию к разрастанию, что может негативно сказаться на производительности программы. Чтобы избежать этого, лучше периодически вызывать сборку мусора принудительно.

При написании программ на Lua обязательно учитывай то, каким образом интерпретатор Lua управляет распределением памяти. Сборка мусора в версии 5.0.2 — относительно затратная по производительности операция (используется алгоритм non-incremental mark and sweep). Она инициируется автоматически, когда объем выделенной памяти превышает двукратный объем памяти, оставшейся выделенной после предыдущей сборки мусора (объем выделенной памяти, при превышении которого произойдет следующая сборка мусора, также можно задавать программно). Значит, фактически, в достаточно большой динамической системе сборка мусора может быть запущена в произвольный момент времени, вызвав просадку по производительности. Чем больше памяти выделено под объекты Lua, тем дольше происходит сбор мусора.

Итак, если ты, например, читаешь в Lua данные из файла построчно и склеиваешь их в одну строку линейно, скорее всего, процесс будет продвигаться страшно медленно: после каждой следующей операции склейки объем занятой памяти, как минимум, удваивается и, соответственно, запускается процесс сборки мусора.

Код создает новую строку (и присваивает ее переменной some_string), а строка, хранившаяся в some_string до склейки, остается в памяти до следующей сборки мусора:

```
some_string = some_string .. "a"
```

Такая проблема и способы ее решения описаны в статье Роберто Иерусалимского Lua Technical Note 9: Creating Strings Piece by Piece (**www.lua.org/notes/ltn009.html**). В Lua 5.1 реализована инкрементальная сборка мусора, позволяющая распределить нагрузку по производительности от процесса сборки мусора во времени

проблемы переносимости

НА МОБИЛЬНЫЕ УСТРОЙСТВА

РЫНОК МОБИЛЬНЫХ ИГР РАСТЕТ ВЗРЫВООБРАЗНО, А ВМЕСТЕ С НИМ РАСТЕТ И ИНТЕРЕС ПРОГРАММИСТОВ. ЛАКОМЫЙ КУСОК РЫНКА! ПРАКТИЧЕСКИ ВСЯКИЙ, КТО ХОТЯ БЫ ОДИН РАЗ СОЗДАВАЛ ИГРУ ДЛЯ ПК, НАЧИНАЕТ ПОГЛЯДЫВАТЬ В СТОРОНУ СОТОВЫХ ТЕЛЕФОНОВ. РАЗРАБОТКА ИГР ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ — ЗАНЯТИЕ ДЕЙСТВИТЕЛЬНО НЕСЛОЖНОЕ, НО В НЕМ ЕСТЬ НЮАНСЫ | **КРИС КАСПЕРСКИ АКА МЫЩЪХ**

Многообразие «железа» под IBM PC доставляет множество неприятностей и буквально сводит разработчиков с ума, но это ничто по сравнению с миром мобильных устройств. Меняется не только архитектура процессора, объем памяти, глубина цветности, но даже геометрия дисплея и расположение кнопок, что ужесточает требования к дизайну — чертовски трудно создать игру, которая сохраняла бы свою привлекательность на устройствах всего спектра. Никакой стандартизацией здесь и не пахнет.

Карманные компьютеры, позиция которых еще вчера казалась несокрушимой, сегодня сходят со сцены, уступая места «смартфонам» — сотовым телефонам с функциями КПК. Мощность процессоров и объемы памяти растут не по дням, а по часам. Цветной дисплей и графический акселератор — уже не редкость, а насущность и необходимость. Прошли времена, когда пользователь был доволен тем, что установил на сотовый телефон «Пасьянс» или «Дурак», и теперь народ требует трехмерных игр с активной динамикой.

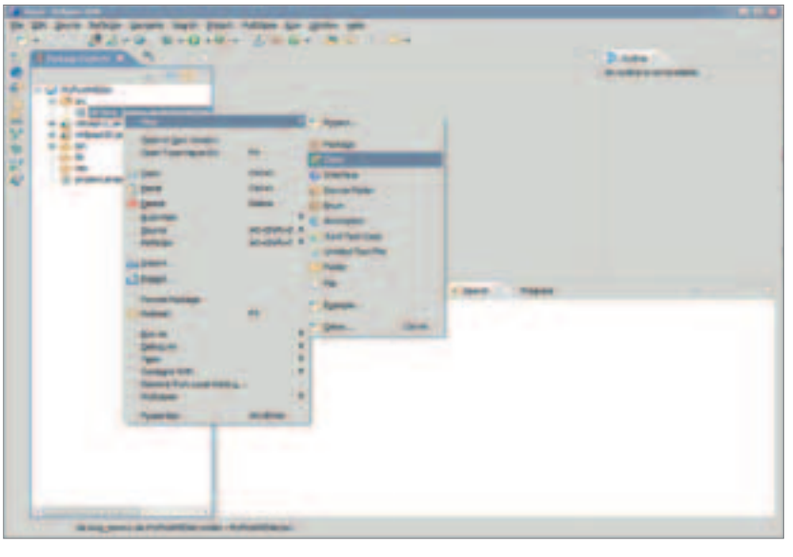
Основная проблема в том, что жизненный цикл мобильных устройств очень невелик, а их популяция крайне разнообразна. Так что разработка игры для одной конкретно взятой модели, скорее всего, не окупится и придется заняться ее портированием на другие платформы. Так что ставим себе задачу провернуть достаточно хитрую и коварную операцию, причем планируем ее заранее.

ПЛАТФОРМЫ ДЛЯ РАЗРАБОТЧИКОВ

j2me

Подавляющее большинство мобильных приложений (по некоторым оценкам, аж до 70%) пишутся на Java (а точнее J2ME, расшифровывается как Java 2 Micro Edition) — урезанной версии языка, ориентированной на маломощные системы и поддерживающей рекордное количество мобильных устройств. Вместо «живого» машинного кода сотовому телефону подсовывают так называемый «байт-код», который исполняется на виртуальной машине (Virtual Machine, сокращенно VM). Разработкой VM занимается компания Sun (создатель языка Java) и сторонние поставщики, так как спецификации байт-кода открыты, а SDK (пакет программ и библиотек для разработчиков) бесплатен.

Теоретически, игра, написанная для одного сотового телефона, пойдет на любом устройстве, для которого есть соответствующая реализация VM. Даже на пылесосе и домофоне! На практике же выходит иначе. Прежде всего, производительность J2ME-приложений крайне невелика, и вычислительной мощности даже самых быстрых телефонов может попросту не хватить для разработки динамичной игры. Чтобы преодолеть это ограничение, производители телефонов выпускают специальные высокопроизводительные библиотеки (попутно реализующие дополнительные возможности по работе с дисплеем и музыкой), несовместимые, однако, со стандартной виртуальной машиной, и поэтому непереносимые. Естественно, Sun не стоит на месте и продолжает наращивать потенциал мобильной Java. Достаточно сказать, что в MIDP 1.0 не было поддержки работы со спрайтами, и вплоть до появления MIDP 2.0 разработчикам приходилось делать это самим. Однако время уже упущено — на мобильную сцену ринулись конкуренты.



Популярная среда разработки Eclipse под J2ME

brew

Двоичная среда исполнения для беспроводных устройств (Binary Runtime Environment for Wireless, сокращенно BREW) — альтернативная программная платформа, разработанная Qualcomm. В наших мозгах эта компания упорно ассоциируется с аббревиатурой CDMA, где, собственно, и зародился BREW, а на GSM-телефоны он был перенесен значительно позднее. Как понятно по названию, это двоичная среда! В состав бесплатно распространяемого SDK входит полноценный C/C++-компилятор ARM Builder, который генерирует высокопроизводительный код (несомненный плюс).

Какое-то время назад все были убеждены, что Qualcomm завоюет мир. Темпы, которыми развертывалось BREW, вселяли оптимизм. Смущало лишь то, что данная технология поддерживает лишь семь моделей телефонов, в то же время Qualcomm требует обязательной сертификации всех приложений. С одной стороны, так гарантировалось высокое качество кода, с другой — разработчики высаживались на геморрой, поэтому позднее развитие BREW зашло в тупик.

execution engine

Реалистичная трехмерная игра, разработанная с помощью Execution Engine

Execution Engine — разработка компании In-Fusio, еще один конкурент на голову Sun, которая попыталась обойти ограничения, накладываемые J2ME. Здесь представлен не двоичный код (как в BREW), а виртуальная машина (как в Java), только более производительная. В среднем быстродействие выше в 10-15 раз, а на некоторых операциях — в 30 раз! Кроме того, Execution Engine поддерживает ряд жизненно важных графических функций (масштабирование, панорамная прокрутка, трассировка лучей, вращения), которые J2ME-программисты вынуждены реализовывать самостоятельно на медленном байт-коде. Execution Engine делает это непосредственно из машинно-зависимых библиотек, специальным образом оптимизированных под каждую поддерживаемую модель телефона. Вот только поддерживается совсем немного моделей: Philips Fisio 530/825, Alcatel OT 735/535/531, Panasonic X70/G60, Sagem myX-5/myG-5/myX-6... Добавлю и то, что SDK далеко не бесплатен и это отталкивает многих разработчиков.

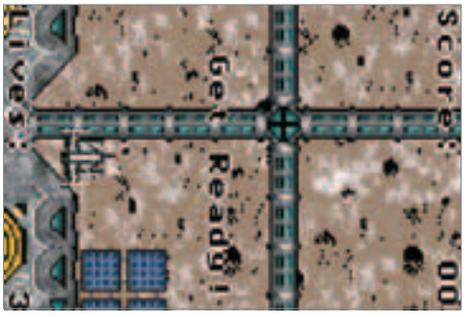
mophun

Весьма перспективная платформа. Разработана компанией Synergenix и позиционируется как «карманная консоль на базе ПО». Производительность — выше всяких похвал. Там, где J2ME показывает 400 KIPS, Mophun выдает 60 MIPS, что в 150 раз быстрее. Хорошо, конечно, но ассортимент поддерживаемых устройств, как легко догадаться, невелик. Плюс Synergenix требует обязательной сертификации всех приложений.

«ACE» — игрушка, реализованная на Mophun

wge

Wireless Graphics Engine (сокращенно WGE) от компании TTPCom содержит удобный API, ориентированный на создание двух- и трехмерных игр. Программировать на нем — одно удовольствие, к тому же SDK распространяется на бесплатной основе. Единственным недостатком остается ограниченный круг поддерживаемых телефонов.



Игра, разработанная с применением платформы Wireless Graphics Engine

Вывод: J2ME — единственная платформа, которая обеспечивает хоть какую-то переносимость. Для разработчиков неторопливых логических игр — оптимальный вариант, но динамическая графика по производительности уже не тянет. Альтернативные платформы решают проблему производительности нанося ущерб совместимости и налагая на разработчиков множество различных ограничений (типа лицензирования), что только отталкивает.

Если переносимости нет, зачем держаться за враждебные платформы, когда весь необходимый функционал можно реализовать самостоятельно? На ассемблере. Получишь высокую скорость и лучшую поддержку конкретного оборудования, соответственно, сможешь создать быструю и динамичную игру с кучей наворотов, которые отсутствуют у конкурентов, использующих J2ME.

Лучшие игры (типа Fight Hard 3D) создаются именно так. Естественно, о переносимости в этом случае придется забыть. Серьезно потрудившись, ты сможешь перенести свое творение на парутройку популярных телефонов, но не более того. Иначе затея окажется нерентабельной и к моменту завершения переноса возникнут новые технологии, появятся новые идеи, а игра, казавшаяся когда-то вершиной научной мысли, станет никому не нужным барахлом. В этом и состоит главное отличие приложений для сотовых телефонов от, например, картин.

Россыпь проблем совместимости:

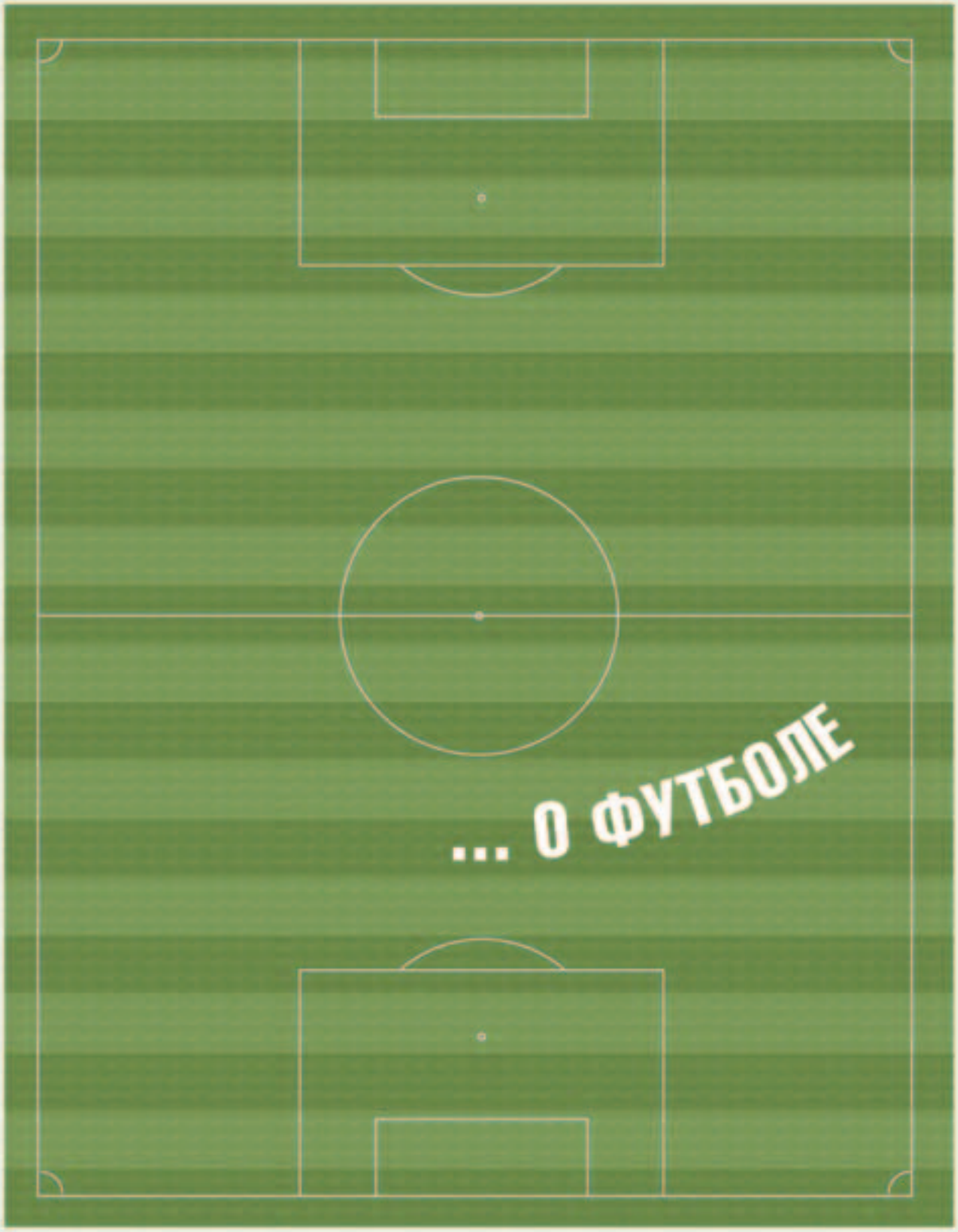
- РАСКЛАДКИ КЛАВИАТУРЫ РАЗЛИЧАЮТСЯ В ЗАВИСИМОСТИ ОТ УСТРОЙСТВА. ИНОГДА УДОБНО, НО ЧАЩЕ ВСЕГО — НЕ ОЧЕНЬ. ЗНАЧИТ, ИГРА НЕ ТОЛЬКО ДОЛЖНА ДОПУСКАТЬ ПЕРЕОПРЕДЕЛЕНИЕ КЛАВИШ, НО И ПРЕДОСТАВЛЯТЬ ТЩАТЕЛЬНО ПРОДУМАННЫЙ ИНТЕРФЕЙС, КОТОРЫЙ ПОЗВОЛИТ УПРАВЛЯТЬ ОДНОЙ РУКОЙ И С МИНИМУМОМ ТЕЛОДВИЖЕНИЙ.
- ВСТРЕЧАЮТСЯ САМЫЕ РАЗНЫЕ ГЛУБИНА ЦВЕТНОСТИ И РАЗРЕШЕНИЕ ДИСПЛЕЕВ, И УЛОЖИТЬСЯ В ОДНУ ЦВЕТОВУЮ ПАЛИТРУ ВСЕ РАВНО НЕ ПОЛУЧИТСЯ. НЕ ОСТАЕТСЯ НИЧЕГО ДРУГОГО, КРОМЕ КАК СОЗДАТЬ НЕСКОЛЬКО ВЕРСИЙ ГРАФИЧЕСКОГО ОФОРМЛЕНИЯ, КАК ЭТО БЫЛО ВО ВРЕМЕНА CGA-, EGA- И VGA-МОНИТОРОВ.
- МОЩНОСТЬ МОБИЛЬНЫХ ПРОЦЕССОРОВ И ОБЪЕМ ДИНАМИЧЕСКОЙ ПАМЯТИ ВАРЬИРУЕТСЯ В ШИРОКИХ ПРЕДЕЛАХ. ЧТОБЫ ПРИ ПОРТИРОВАНИИ НЕ ПЕРЕПИСЫВАТЬ ИГРУ ЗАНОВО, НЕОБХОДИМО ИЗНАЧАЛЬНО ЗАЛОЖИТЬ В НЕЕ МОДУЛЬНУЮ СТРУКТУРУ, ОБЕСПЕЧИВАЮЩУЮ НЕСКОЛЬКО УРОВНЕЙ «РЕАЛИСТИЧНОСТИ». — СООТНОШЕНИЕ МОЩНОСТИ ПРОЦЕССОРА И ОБЪЕМА ПАМЯТИ ТАКЖЕ ПОДВЕРЖЕНО ВАРИАЦИЯМ, ПОЭТОМУ В НЕКОТОРЫХ СЛУЧАЯХ ПРЕДПОЧТИТЕЛЬНЕЕ ТАБЛИЧНЫЕ ВЫЧИСЛЕНИЯ, А В НЕКОТОРЫХ — ОНИ ПРОСТО НЕ ВМЕЩАЮТСЯ.
- ЭНЕРГОЕМКОСТЬ АККУМУЛЯТОРОВ ТАКЖЕ РАЗЛИЧНА. ПРИ ПОСТОЯННОЙ ЗАГРУЗКЕ ПРОЦЕССОРА, ДИСПЛЕЯ И ЗВУКА ПОТРЕБЛЕНИЕ ТОКА ВОЗРАСТАЕТ ВЕСЬМА ЗНАЧИТЕЛЬНО. НЕОБХОДИМО ПРОЕКТИРОВАТЬ ИГРУ ТАК, ЧТОБЫ ИГРОВОЙ ЦИКЛ ВМЕЩАЛСЯ ХОТЯ БЫ В ПОЛНОСТЬЮ ЗАРЯЖЕННЫЙ АККУМУЛЯТОР. ЕСЛИ УЧЕСТЬ ТО, ЧТО ТЕЛЕФОН — ЭТО НЕ ТОЛЬКО ИГРУШКА, НО И СРЕДСТВО СВЯЗИ, ЖЕЛАТЕЛЬНО ПРЕДУСМОТРЕТЬ НЕСКОЛЬКО УРОВНЕЙ «ЭНЕРГЕТИЧЕСКОЙ ПРОЖОРЛИВОСТИ».
- МОБИЛЬНЫЕ УСТРОЙСТВА СОВЕРШЕННО НЕ ПРИСПОСОБЛЕНЫ К ПРОГРАММИРОВАНИЮ, ПОЭТОМУ ОСНОВНОЙ ЦИКЛ РАЗРАБОТКИ ПРИХОДИТСЯ ВЕСТИ НА ЭМУЛЯТОРЕ, ОДНАКО ПОВЕДЕНИЕ ЭМУЛЯТОРА МОЖЕТ СУЩЕСТВЕННО ОТЛИЧАТЬСЯ ОТ ПОВЕДЕНИЯ «ЖИВОГО» СОТОВОГО ТЕЛЕФОНА ИЛИ КПК.
- НЕ НА ВСЕ СОТОВЫЕ ТЕЛЕФОНЫ МОЖНО НАЙТИ СПЕЦИФИКАЦИЮ И ЭМУЛЯТОРЫ. ЕСЛИ НЕТ СПЕЦИФИКАЦИИ, О ПРОГРАММИРОВАНИИ НА АССЕМБЛЕРЕ МОЖНО СРАЗУ ЗАБЫТЬ. ПРИХОДИТСЯ ПРОГРАММИРОВАТЬ ПОД J2ME, ОСТАВАЯСЬ В ПОЛНОМ НЕВЕДЕНИИ НАСЧЕТ ДОСТУПНОЙ ПАМЯТИ И ПРОИЗВОДИТЕЛЬНОСТИ.
- ТЕСТИРОВАНИЕ ИГР ЗАТРУДНЕНО ТЕМ, ЧТО МОБИЛЬНЫХ УСТРОЙСТВ ОЧЕНЬ МНОГО, ОНИ СТОЯТ ДОРОГО, УСТАРЕВАЮТ БЫСТРО И ПРИОБРЕТАТЬ ВЕСЬ АССОРТИМЕНТ ОБОРУДОВАНИЯ, ИМЕЮЩЕГОСЯ НА РЫНКЕ, МОЖЕТ ПОЗВОЛИТЬ СЕБЕ ТОЛЬКО КРУПНАЯ ФИРМА.
- СЛОЖНОСТЬ ПЕРЕНОСА ЧАСТО ПРЕВЫШАЕТ ТРУДОЕМКОСТЬ РАЗРАБОТКИ ПРОГРАММЫ С НУЛЯ. К ТОМУ ЖЕ В ЭТОМ СЛУЧАЕ МЫ МОЖЕМ ЗАДЕЙСТВОВАТЬ СПЕЦИФИЧЕСКИЕ ОСОБЕННОСТИ КОНКРЕТНОЙ МОДЕЛИ ТЕЛЕФОНА.
- ОТСУТСТВУЕТ КАКОЙ БЫ ТО НИ БЫЛО СТАНДАРТ НА ЯЗЫК И API МОБИЛЬНЫХ УСТРОЙСТВ. КОМПИЛЯТОРЫ ДВОИЧНОГО И БАЙТ-КОДА ОТ РАЗНЫХ ПРОИЗВОДИТЕЛЕЙ СОВЕРШЕННО НЕСОВМЕСТИМЫ МЕЖДУ СОБОЙ И ОБЕСПЕЧИВАЮТ ПЕРЕНОСИМОСТЬ ТОЛЬКО В РАМКАХ ЗАРАНЕЕ ВЫБРАННОГО МОДЕЛЬНОГО РЯДА, КОТОРЫЙ ПОСТОЯННО ПОПОЛНЯЕТСЯ, НО ДВИЖЕТСЯ В ЗАРАНЕЕ НЕ ПРЕДСКАЗУЕМОМ НАПРАВЛЕНИИ И ОЧЕНЬ ЧАСТО ПАДЕТ ВНИЗ. А ПОСКОЛЬКУ ИСХОДНЫЙ КОД КОМПИЛЯТОРА И ВСЕХ СОПУТСТВУЮЩИХ ЕМУ БИБЛИОТЕК ЯВЛЯЕТСЯ СОБСТВЕННОСТЬЮ ФИРМЫ-ПРОИЗВОДИТЕЛЯ, РАЗРАБОТЧИКАМ ИГР СОВЕРШЕННО НЕ ГАРАНТИРУЮТ ОТСУТСТВИЕ «ИЗМЕНЫ».
- МОБИЛЬНЫЙ МИР ОЧЕНЬ ИЗМЕНЧИВ И НЕПОСТОЯНЕН. ЗА ВРЕМЯ, ПОТРАЧЕННОЕ НА ПЕРЕНОС, КОНКРЕТНАЯ МОДЕЛЬ ТЕЛЕФОНА МОЖЕТ СТАТЬ ПОРЯДОЧНО УСТАРЕВШЕЙ ИЛИ ОКАЗАТЬСЯ ВЫТЕСНЕННОЙ ДРУГОЙ, СОВЕРШЕННО ИНОЙ ПО ХАРАКТЕРИСТИКАМ. И ТОГДА ВЕСЬ ПРОЦЕСС ПРИДЕТСЯ ПРОДЕЛЫВАТЬ ЗАНОВО...

Реалистичная трехмерная игра, разработанная с помощью Execution Engine

Интерес к мобильным платформам вызван в первую очередь тем, что сотовый телефон дает программистам возможность разработать достойную игру за короткое время (от трех месяцев до года) и со скромными вложениями. На ПК игры выродились в монстров, над созданием которых работают огромные коллективы. Чтобы создать нечто конкурентоспособное на ПК усилиями нескольких человек, требуются принципиально новые идеи. А у кого они есть?

Сотовый телефон хорош тем, что народ толпами радуется питону, арканоиду или тетрису, а от дрочки, реализованной в стиле ZX-SPECTRUM, — буквально бьется в экстазе. К сожалению, в отличие от старика спектрума, век мобильных игрушек очень недолог. Зато народ привык покупать их (через мобильные службы, конечно), а не пионерить естественным образом в ближайшем ларьке. Уровень пиратства на мобильной арене намного ниже, чем на ПК, поэтому заработать деньги на программировании игр вполне реально, даже если мы учтем все сложности переноса на другие платформы.

Очень хочется надеяться, что через несколько лет ситуация нормализуется и появится если не стандарт, то хотя бы какие-то общие черты, объединяющие все мобильные устройства воедино


... О ФУТБОЛЕ

TOTAL FOOTBALL

!

НОВЫЙ ЖУРНАЛ О ФУТБОЛЕ

В КАЖДОМ НОМЕРЕ DVD С ЛУЧШИМ ФУТБОЛЬНЫМ КОНТЕНТОМ

В МАРТОВСКОМ НОМЕРЕ:

ЭКСКЛЮЗИВ
Чудо-братья Березуцкие

ТЕМА НОМЕРА
История всех российских чемпионатов. Великое и смешное

ФАНАТСКИЙ ГИД
Бригады, фирмы, мобы, банды

БУДЬ В ФОРМЕ
Как правильно выбрать футбольную экипировку. Вратарские перчатки

ПОСТЕР
Календарь Чемпионата России 2006 и откровенная фотосессия болельщиц "Зенита"

УНИКАЛЬНЫЙ КОНКУРС
Суперприз – поездка на финал Лиги Чемпионов 2006!

сэкономь деньги — закажи журнал в редакции

ВЫГОДА

Цена подписки до 15% ниже, чем в розничной продаже
Бонусы, призы и подарки для подписчиков
Доставка за счет редакции

ГАРАНТИЯ

Ты гарантированно получишь все номера журнала
Единая цена по всей России

СЕРВИС

Заказ удобно оплатить через любое отделение банка
Доставка осуществляется заказной бандеролью или курьером

КАК ОФОРМИТЬ ЗАКАЗ

1 Заполнить купон и квитанцию
2 Перечислить стоимость подписки через наш банк: ЗАО ММБ
3 Обязательно прислать в редакцию копию оплаченной квитанции с четко заполненным купоном любым из перечисленных способов:

— по электронной почте: **subscribe@glc.ru**;
— по факсу: (495) 780-88-24;
— по адресу: 119021, Москва, ул. Тимура Фрунзе, д. 11, стр. 44-45, ООО «Гейм Лэнд», отдел подписки.

Внимание!
Подписка оформляется в день обработки купона и квитанции.

— купоны, отправленные по факсу или электронной почте, обрабатываются в течение пяти рабочих дней.
— купоны, отправленные почтой на адрес редакции, обрабатываются в течение 20-ти дней.

Рекомендуем использовать электронную почту или факс. Подписка производится с номера, выходящего через один календарный месяц после оплаты. Например, если произвести оплату в сентябре, то подписку можно оформить с ноября.

ПОДПИСКА ДЛЯ ЮРИДИЧЕСКИХ ЛИЦ

Москва: ООО «ИНТЕР-ПОЧТА», (495) 500-00-60, www.interpochta.ru

Для получения счета на оплату подписки нужно прислать заявку с названием журнала, периодом подписки, банковскими реквизитами, юридическим и почтовым адресом, телефоном и фамилией ответственного за подписку лица.

подписной купон

стоимость заказа на Хакер Спец + CD

6 месяцев — 900 руб. 00 коп. | **12** месяцев — 1740 руб. 00 коп.

стоимость заказа на комплект Хакер Спец + Хакер + Железо

6 месяцев — 2550 руб. 00 коп. | **12** месяцев — 5040 руб. 00 коп.

прошу оформить подписку:
☐ на журнал Хакер Спец + CD
☐ на комплект Хакер Спец + Хакер + Железо
на _____ месяцев
начиная с _____ 200 г.
☐ Доставлять журнал по почте на домашний адрес
☐ Доставлять журнал курьером на адрес офиса (по г. Москве)
Подробнее о курьерской доставке читайте ниже*
(отметьте квадрат выбранного варианта подписки)
Ф.И.О.
дата рождения
адрес доставки:
индекс
область/край
город
улица
дом _____ корпус
квартира/офис
телефон ( )
e-mail
сумма оплаты

*Курьерская доставка осуществляется только по Москве на адрес офиса. Для оформления доставки курьером укажите адрес и название фирмы в подписном купоне.

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ЗАО ММБ
р/с № 40702810700010298407
к/с № 30101810300000000545
БИК 044525545 КПП - 772901001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200_ г. (месяц)	

Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ЗАО ММБ
р/с № 40702810700010298407
к/с № 30101810300000000545
БИК 044525545 КПП - 772901001
Плательщик
Адрес (с индексом)

Назначение платежа	Сумма
Оплата за «_____» с _____ 200_ г. (месяц)	

Ф.И.О.
Подпись плательщика

Кассир

ПО ВСЕМ ВОПРОСАМ, СВЯЗАННЫМ С ПОДПИСКОЙ, ЗВОНИТЕ ПО БЕСПЛАТНЫМ ТЕЛЕФОНАМ: **780-88-29** (ДЛЯ МОСКВИЧЕЙ) И **8-800-200-3-999** (ДЛЯ РЕГИОНОВ И АБОНЕНТОВ МТС, БИЛАЙН, МЕГАФОН). ВСЕ ВОПРОСЫ ПО ПОДПИСКЕ МОЖНО ПРИСЫЛАТЬ НА АДРЕС: **info@glc.ru**

dead can dance

СКЕЛЕТНАЯ АНИМАЦИЯ ОПТОМ И В РОЗНИЦУ

СКЕЛЕТНАЯ АНИМАЦИЯ — НЕ ПРОСТО ДАНЬ МОДЕ, А УДОБНЫЙ И ДОСТАТОЧНО ПРОИЗВОДИТЕЛЬНЫЙ МЕТОД СОЗДАНИЯ РЕАЛИСТИЧНОЙ АНИМАЦИИ ДВИЖЕНИЯ ЛЮДЕЙ И ЖИВОТНЫХ. ЕСЛИ СОЗДАТЬ СЕТКУ (MESH), МАКСИМАЛЬНО ПРИБЛИЖЕННУЮ К РЕАЛЬНОСТИ, И ПРАВИЛЬНО ПРИВЯЗАТЬ К ТАКОЙ СЕТКЕ КОСТИ, ТО АНИМАЦИЯ НА ОСНОВЕ ЭТОГО СКЕЛЕТА ДЕЙСТВИТЕЛЬНО БУДЕТ ПОХОЖА НА РЕАЛЬНУЮ. НО ВОТ ТЫ СЕЛ ЗА КЛАВИАТУРУ И СТОЛКНУЛСЯ С МНОЖЕСТВОМ ПРОБЛЕМ, КОТОРЫЕ НУЖНО РЕШИТЬ. О НЕКОТОРЫХ ПРОБЛЕМАХ И ПОГОВОРИМ | **ФЛЕНОВ МИХАИЛ АКА HORRIFIC**

моделирование Все начинается с создания персонажа. С помощью программы 3D-моделирования мы должны создать 3D-модель человека/монстра, подлежащего анимированию. Я привык к классике — 3D Studio Max (в народе 3DS). Эта классика тяжеловесная и стоит дорого, но очень удобна и люба моему сердцу еще со времен MS DOS-варианта.

Сама программа 3D Studio Max позволяет моделировать сетку персонажа, но тут возможности не заканчиваются. Далее должны быть созданы кости, к ним должны быть привязаны вершины сетки — когда двигается какая-нибудь кость, все связанные с ней вершины также должны перемещаться и изгибаться. Для создания скелета в 3DS есть модуль Character Studio, который еще недавно был отдельной программой. Сейчас он встроен непосредственно в оболочку в меню Character. Чтобы создать реалистичный персонаж, будет маловато знаний одних программных пакетов, потребуются еще и хорошие знания в области анатомии и физики, иначе никто не поверит в реалистичность движений твоего персонажа.

Мы не будем изучать процесс создания моделей и скелетов, так как подобному вопросу должна быть посвящена отдельная статья или даже книга. Если хочешь разобраться в этой теме поглубже, рекомендую прочитать книгу «3DS MAX 6 и Character Studio 4. Анимация персонажей» (авторы Ю. Кулагин и Б. Морозов — **http://bhv.ru/books/book.php?id=3888**). Отличная книга! Но чтобы понять ее, ты должен иметь начальные знания по самой программе 3DS. Авторы описывают только то, что касается анимации персонажей, и опускают базовые понятия моделирования.

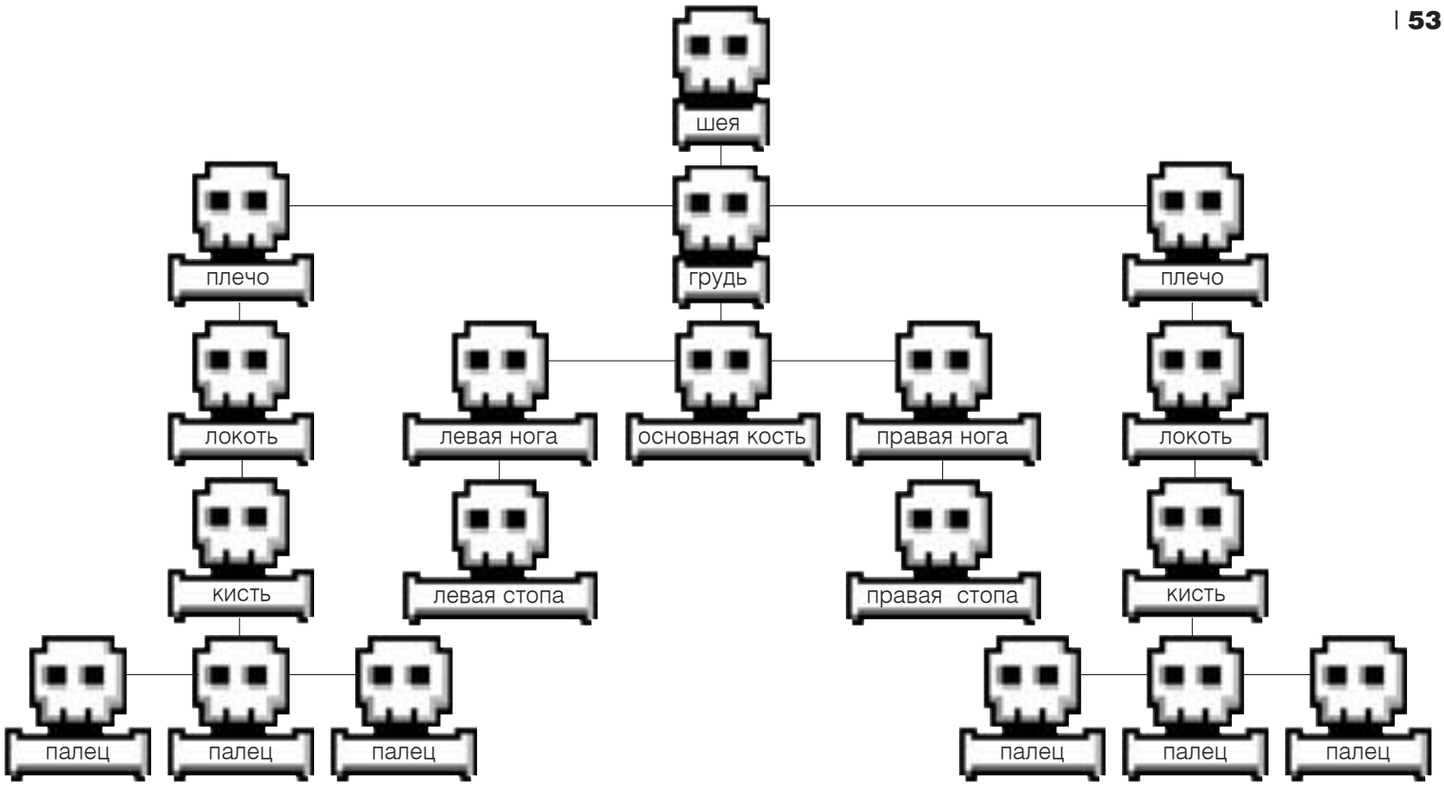
Зачем же я завел разговор о моделировании, если не собираюсь рассматривать его? Создание персонажа — это отдельная тема, а сейчас нам нужно понять, что должно быть создано и, соответственно, чем мы будем оперировать в собственной программе.

> «ПРИМЕР СКЕЛЕТНОЙ АНИМАЦИИ ЕСТЬ В СОСТАВЕ DX SDK В ДИРЕКТОРИИ DXSDK\SAMPLES\C++\DIRECT3D\MESHES\SKINNEDMESH»

скелет Посмотрим, из чего состоит скелет. Для иллюстрации и примера я взял один из файлов книги «3DS MAX 6 и Character Studio 4. Анимация персонажей». Найди иллюстрацию в этой статье с дамочкой и стоящим рядом скелетом. Дама — это то, что слева, а скелет — справа. По крайней мере, мне так кажется :). Скелет 3D-модели похож на настоящий скелет человека и состоит из костей, связанных между собой. Если повернуть плечо, то повернется вся рука вплоть до кончиков пальцев. Когда поворачивается кисть руки, поворачиваются и все ее пальцы.

Когда вращаются кости, движутся и связанные точки сетки персонажа. Одна точка сетки может быть привязана к нескольким костям. В этом случае точка сетки смещается в соответствии с указанным весом точки. Если вершина принадлежит двум костям, то по умолчанию вес в обоих случаях будет равен 0,5, то есть при вращении кости на 90 градусов вершина изменит положение только на 45. Так происходит только по умолчанию. В идеале мы можем настраивать вес самостоятельно (если позволит используемая программа 3D-графи-

Plug-in для экспорта модели, скелета и анимации в .x-файл

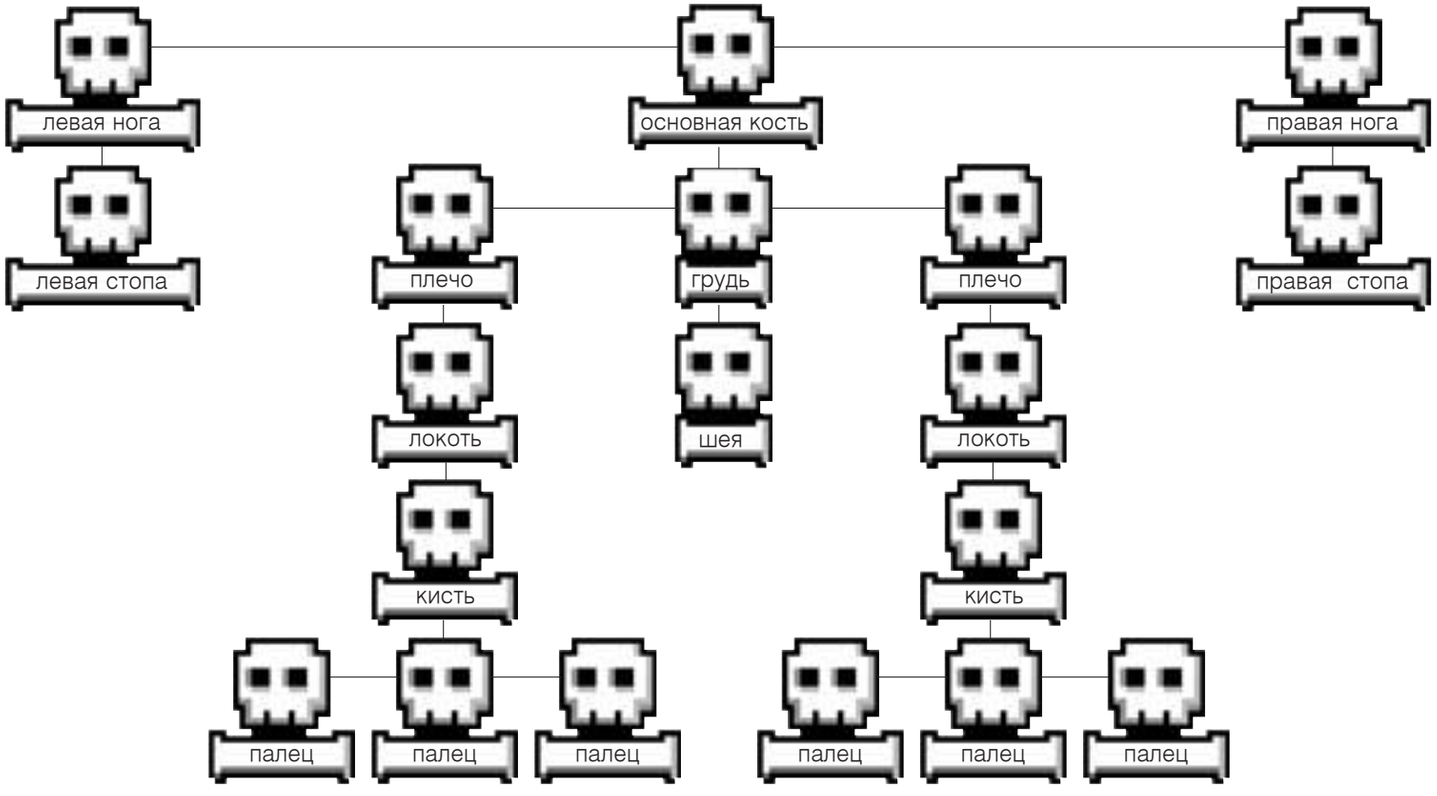


ки). Если вершина принадлежит одной кости, то вес ее преобразований равен 1, то есть она будет трансформироваться на все 100%.

Сейчас отвлекусь от темы, чтобы дать один совет: никогда не двигай кости для создания анимации. Все кости только вращаются относительно начала (точки крепления с главной или вышестоящей по иерархии костью). Например, кость руки крепится к плечу, и чтобы рука была протянута вперед, происходит вращение кости относительно плеча. Притом все остальные части руки (локоть, кисть, пальцы) движутся автоматически — перемещать их не нужно, если только ты не хочешь, чтобы от туловища оторвалась рука или нога. Попробуй подвигать пальцем — и наглядно убедишься, что любые движения пальца — это вращение относительно кисти. Если попытаться двинуть по прямой, то можно выбить палец из сустава. Конечно, встречаются индивидуумы с развитыми суставами, которые производят небольшие прямолинейные движения, но они небольшие (в пределах сустава) и это уже к Лозовскому на прием. Пусть я не врач, мне кажется, что это неизлечимо, в любом случае ненормально и требует обращения в больницу.

экспорт сетки Создав сетку персонажа, скелет и анимацию, сохрани их для последующей загрузки из собственной программы. Вот тут возникает проблема выбора: как сохранить? Формат .max позволяет хранить всю необходимую информацию, но он слишком сложный для применения в собственных программах. Можно использовать старый формат .3ds, но возникает проблема с сохранением информации о костях и самой скелетной анимации. Эти проблемы можно обойти, но тогда натолкнешься на другие большие проблемы. Придется писать собственные анализаторы и загрузки данных, потом создавать классы или функции для поддержки всего загруженного в программе.

Зачем выдумывать проблемы себе, когда можно использовать формат файла .x, который разработан Microsoft и является открытым, уни-



хорошие советы

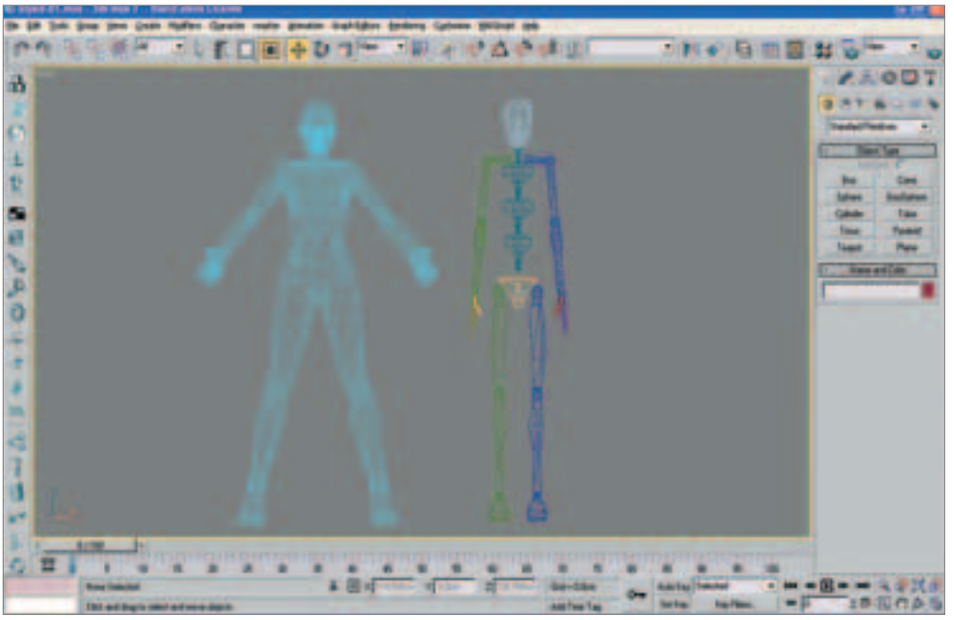
1 ПОСЛЕ МОДИФИКАЦИИ МАТРИЦЫ ПРЕОБРАЗОВАНИЙ ОДНОЙ КОСТИ У ТЕБЯ МОГУТ ВОЗНИКНУТЬ ПРОБЛЕМЫ С ПОЛУЧЕНИЕМ ПЕРВОНАЧАЛЬНОГО СОСТОЯНИЯ — МОЙ ПЕРВЫЙ СОВЕТ ПРИЗВАН ИЗБАВИТЬ ТЕБЯ ОТ ЭТОЙ НАПАСТИ. ДЕЛО В ТОМ, ЧТО В ЛЮБОМ СЛУЧАЕ СУЩЕСТВУЕТ КАКАЯ-ТО ПОГРЕШНОСТЬ, КОТОРАЯ СО ВРЕМЕНЕМ НАЧИНАЕТ ВЛИЯТЬ НА КАЧЕСТВО СЦЕНЫ. НАПРИМЕР, НОГИ ЧЕЛОВЕКА НЕОЖИДАННО ОТКАЖУТСЯ ВЫРАВНИВАТЬСЯ В ТОТ МОМЕНТ, КОГДА ТЫ ПОПЫТАЕШЬСЯ ОСТАНОВИТЬ СВОЕГО ПЕРСОНАЖА И ПОСТАВИТЬ ЕГО РОВНО. ЛИКВИДИРОВАТЬ ТАКОЙ ЭФФЕКТ ЛЕГКО: МОЖНО СОЗДАТЬ СВОЙ ВАРИАНТ СТРУКТУРЫ D3DXFRAME, ТО ЕСТЬ РАСШИРИТЬ ЕЕ, ДОБАВИВ ЕЩЕ ОДНО ПОЛЕ ТИПА D3DXMATRIX. В ЭТОМ ПОЛЕ МОЖНО ХРАНИТЬ МАТРИЦУ ПРЕОБРАЗОВАНИЙ КОСТИ НА ЭТАПЕ НАЧАЛА АНИМАЦИИ И ПРИ КАЖДОМ ОТОБРАЖЕНИИ СЦЕНЫ ПЛЯСАТЬ ИМЕННО ОТ ЭТОГО ЗНАЧЕНИЯ.

2 ДЛЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ДАМ ЕЩЕ ОДИН СОВЕТ: ЗАРАНЕЕ РАССЧИТАТЬ КЛЮЧЕВЫЕ КАДРЫ, ПО КОТОРЫМ БУДЕТ СТРОИТЬСЯ ДАЛЬНЕЙШАЯ АНИМАЦИЯ. ТАК ТЫ ЗАРАНЕЕ ПОДГОТОВИШЬ ТРАЕКТОРИИ ОСНОВНЫХ ТЕЛОДВИЖЕНИЙ: ХОДЬБЫ, БЕГА, ПРИСЕДАНИЯ, ПРЫЖКОВ И Т.Д. ВО ВРЕМЯ ИГРЫ В ЗАВИСИМОСТИ ОТ СИТУАЦИИ ОСТАНЕТСЯ ТОЛЬКО ВОСПОЛЬЗОВАТЬСЯ УЖЕ РАССЧИТАННЫМИ ТРАЕКТОРИЯМИ, КОТОРЫЕ НЕ ЗАНИМАЮТ МНОГО МЕСТА И МОГУТ БЫТЬ ИСПОЛЬЗОВАНЫ ДЛЯ РАЗЛИЧНЫХ ПЕРСОНАЖЕЙ ОДНОВРЕМЕННО. ОДНАКО ПОМНИ, ЧТО У ВСЕХ ЛЮДЕЙ РАЗНЫЕ ПОХОДКИ, И ОСОБЕННО ОТЛИЧАЮТСЯ МУЖСКИЕ И ЖЕНСКИЕ ПОХОДКИ. ТЕМ НЕ МЕНЕЕ ДАЖЕ ЕСЛИ ТЫ ЗАКРОЕШЬ ГЛАЗА НА ЭТО, В ДИНАМИЧНОЙ СЦЕНЕ ИГРОК НЕ ЗАМЕТИТ ПОДВОХА.

3 ТАК КАК ФОРМАТ .X-ФАЙЛА ОТКРЫТЫЙ И ЛЕГКО ПОДДАЕТСЯ РАСШИРЕНИЮ, ТО ЗАРАНЕЕ ПРОСЧИТАННУЮ АНИМАЦИЮ МОЖНО ХРАНИТЬ В НЕМ. ДРУГОЕ ДЕЛО, ЧТО ДЛЯ РЕДАКТИРОВАНИЯ ФАЙЛА И СОХРАНЕНИЯ АНИМАЦИИ МОЖЕТ ПОТРЕБОВАТЬСЯ ДОПОЛНИТЕЛЬНАЯ ПРОГРАММА И НЕМНОГО НАПРЯГА В КОДИНГЕ, НО ДЕЛО СТОИТ ТОГО.

версальным и легко расширяемым? Расширять возможности файла нам не понадобится, потому что все необходимое для хранения сетки и костей в нем уже есть. Сама 3DS Max не может сохранять или экспортировать данные в формат .x, но можно установить специальный plug-in, разработанный самой Microsoft. Этот plug-in можно найти в составе DX SDK, причем в исходных кодах. Нам остается только скомпилировать его и установить.

.x-файл обладает еще одним преимуществом: в составе функций Direct3D есть возможность загрузить сетку с помощью одной-единственной функции и с помощью еще одной — скелет. Корпорация Microsoft здесь очень хорошо позаботилась о нас как о программистах. Однако о функциях читай чуть позже. Сейчас еще немного мысленно пощупаем скелет.



Слева сетка человека, похожего на женщину :), справа — скелет

внутреннее пространство

Посмотрим, из чего состоит скелет в .x-файле и что нам предстоит загружать. Для ясности я набросал небольшую схему, ее достаточно для понимания материала — описывать тело в подробностях я не стал. В центре этой схемы находится основная кость, вокруг которой множество других. «Основная» — это не просто любая или центральная, а именно основная кость. При ее перемещении или вращении вращается абсолютно вся сцена. У человека эту роль выполняет позвоночник, а у гуманоида — что захочешь :). Насчет гуманоидов не могу посоветовать ничего, потому что не знаю, какую игру ты планируешь и с какой планеты прилетели твои пришельцы.

Все остальные кости крепятся к основной, создавая иерархическое дерево. Если поворачивается какая-нибудь часть этого скелета, перемещаются все дочерние кости. Такая схема создана только для удобства, чтобы она хоть немного напоминала человека. Если взглянуть на иерархию скелета в памяти, то она покажется похожей на дерево с множеством уровней, а не на человека. Если выстроить скелет таким, как он находится в памяти, то может получиться, что нога находится на одном уровне с грудью.

Может показаться, что работать с костями адски неудобно, на самом же деле ничего сложного нет. Создавая вторую схему скелета, я всего лишь изменил положение элементов схемы, а связи не изменялись. Все связано в той же последовательности.

загрузка скелета

Для загрузки информации из .x-файла с учетом скелета можно воспользоваться функцией D3DXLoadMeshHierarchyFromX, в общем виде она выглядит следующим образом:

```
HRESULT D3DXLoadMeshHierarchyFromX(
 LPCTSTR Filename,
 DWORD MeshOptions,
 LPDIRECT3DDEVICE9 pDevice,
 LPD3DXALLOCATEHIERARCHY pAlloc,
 LPD3DXLOADUSERDATA pUserDataLoader,
 LPD3DXFRAME* ppFrameHeirarchy,
 LPD3DXANIMATIONCONTROLLER*
ppAnimController
);
```

Коротко пробежимся по параметрам этой функции.

Filename — что-то подсказывает мне, что это путь к загружаемому .x-файлу.

MeshOptions — опции загрузки. Чаще всего будет нужна опция D3DXMESH_MANAGED, чтобы можно было управлять загруженными данными вершинного буфера. Если управление не нужно, то можно вообще ничего не указывать, чтобы использовать значения по умолчанию.

pDevice — указатель на интерфейс IDirect3DDevice. Я думаю, тут не нужно пояснять, что это такое и зачем.

pAlloc — указатель на интерфейс ID3DXAllocateHierarchy, методы которого вызываются во время загрузки информации из .x-файла. Этот интерфейс необходим для выделения и освобождения фреймов и контейнера объектов.

pUserDataLoader — указатель на интерфейс ID3DXLoadUserData для загрузки пользователь-

ских данных. Да, .x-файл является свободным, гибким, расширяемым, и поэтому может содержать и пользовательские данные, которые можно обработать указав здесь собственный экземпляр интерфейса ID3DXLoadUserData.

ppFrameHeirarchy — возвращает указатель на иерархию загруженных фреймов в виде массива структур D3DXFRAME.

ppAnimController — указатель на интерфейс ID3DXAnimationController, в котором будет размещена информация об анимации. Да, в .x-файле может оказаться еще и заранее подготовленная анимация. Следовать ей необязательно, но иногда очень удобно.

фреймы

Остановимся подробнее на шестом параметре функции D3DXLoadMeshHierarchyFromX, где мы получаем указатель на массив структур D3DXFRAME. Что это такое? На самом деле каждая такая структура описывает определенную кость в скелете и выглядит следующим образом:

```
typedef struct _D3DXFRAME {
 LPTSTR Name;
 D3DXMATRIX TransformationMatrix;
 LPD3DXMESHCONTAINER pMeshContainer;
 struct _D3DXFRAME *pFrameSibling;
 struct _D3DXFRAME *pFrameFirstChild;
} D3DXFRAME, *LPD3DXFRAME;
```

Что же есть в этой структуре и для чего «оно» предназначено?

Name — имя элемента скелета. Каждая кость может иметь свое имя, по которому достаточно удобно находить именно ту часть скелета, которая требует вращения. Зная имя кости, ты легко найдешь ее структуру, перебрав весь скелет. Указатели на наиболее часто используемые структуры лучше сохранить в глобальных переменных, и тогда ты не будешь каждый раз перебирать весь скелет, следовательно, повысишь скорость анимации.

TransformationMatrix — матрица преобразований. Определяет положение данной косточки.

pMeshContainer — указатель на контейнер сетки.

pFrameSibling — указатель на следующую структуру D3DXFRAME, которая находится на том же уровне. Например, все пальцы находятся на одном и том же уровне иерархии скелета, через этот указатель можно последовательно получать доступ к ним.

pFrameFirstChild — указатель на первую кость, которая находится на уровень ниже.

в движении

Перейдем к алгоритму, который позволит двигать скелет. Во время движения чаще всего необходимо изменять матрицу преобразования не только определенной косточки (например предплечья), но и всех дочерних (всех костей, которые входят в состав руки). Приходится создавать функцию, которая должна выполнять два действия: для начала должно изменяться положение текущей кости, а после этого — вызываться рекурсивная функция. Внутри рекурсивной функции нужно изменять матрицы положения не только дочерних, но и родственных (находящихся на одном уровне) костей. Например, когда движется кисть, должны двигаться и все ее пять пальцев, которые находятся на одном уровне, а не один, первый попавшийся палец. Функция может выглядеть примерно так:

рекурсивная функция

```
void Преобразовать
(D3DXFRAME фрейм, D3DXMATRIX матрица)
{
 // Перемножаем матрицы, т.е. объединяем
    фрейм.TransformationMatrix *= матрица;

 // Если есть родственники,
    то преобразовать их фреймы
 if (фрейм.pFrameSibling)
 Преобразовать(фрейм.pFrameSibling, матрица);
 // Если есть дочерние,
    то преобразовать их фреймы
 if (фрейм.pFrameFirstChild)
 Преобразовать(фрейм. pFrameFirstChild,
 фрейм.TransformationMatrix);
}
```

Внимание! Во время вызова функции обновления родственных костей в качестве второго параметра передается неизмененная матрица, а при обновлении дочерних костей — уже трансформированный вариант.

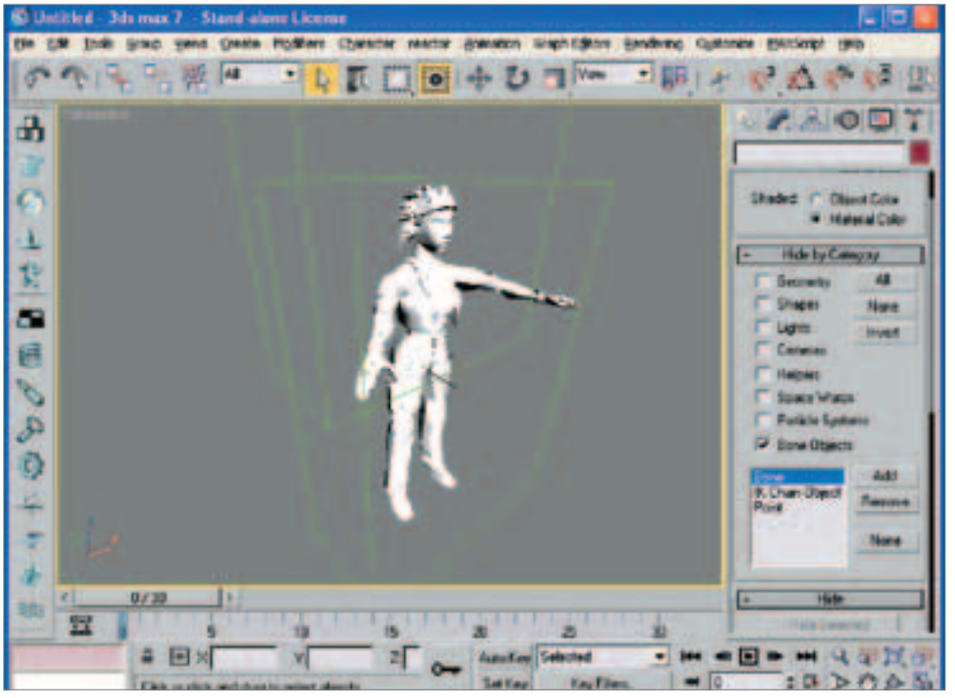
итого

Скелетная анимация позволяет создать анимацию движения персонажа игры легко и быстро. Если персонаж качественно смоделирован, то движения будут максимально реалистичны и пользователи по достоинству оценят твой труд.

Даже такая большая статья, как эта, не может вместить всю информацию. Я постарался показать основы, заинтересовать тебя и убедить в необходимости использовать эту технологию. Дальнейшее совершенствование знаний и умений оставляю на твоей совести. Думаю, через некоторое время я вновь подниму эту тему на моем сайте. Если возникли вопросы — пиши.

Напоследок загляни на сайт **http://cherb.web.ur.ru/mal.html** — увидишь танцующего мальчика в памперсе. Это рисованная анимация, но есть и демка, в которой тот же самый мальчик танцует под хорошую музыку, причем персонаж реализован в виде сетки mesh и скелета. Я видел эту демку примерно три года назад, а сейчас что-то не смог найти. Жаль

Модель человека tiny.x, который идет в составе с DirectX SDK для тестирования скелетной анимации



шаг в прошлое

ИГРЫ, КОТОРЫЕ МЫ НЕ ВОЗРОДИЛИ, ХОТЯ МОГЛИ

ЭТО БЫЛО НЕ ТАК ДАВНО. ВСЕГО ЛИШЬ 10-15 ЛЕТ НАЗАД. ТОГДА МЫ УЧИЛИСЬ В ШКОЛЕ, ПИЛИ ПИВО НА ПЕРЕМЕНАХ, ДЕРГАЛИ ДЕВОЧЕК ЗА КОСИЧКИ, ВЕРИЛИ ВО ЧТО-ТО ПРЕКРАСНОЕ И В ОДИН ПРЕКРАСНЫЙ МОМЕНТ ЗНАКОМИЛИСЬ С ЭВМ. КАКОЙ ИМЕННО МОМЕНТ? КОМУ КАК ПОВЕЗЛО. КОМУ-ТО ОТЕЦ ПРИНЕС С РАБОТЫ РАЗДОЛБАННУЮ БК'ШКУ, КТО-ТО УЖЕ В СРЕДНЕЙ ШКОЛЕ ПОЗНАКОМИЛСЯ С 286-МИ ТАЧКАМИ, КТО-ТО ОБЩАЛСЯ С ИМПОРТНЫМ СПЕКТРУМОМ ИЛИ С ОДНИМ ИЗ ЕГО МНОГОЧИСЛЕННЫХ СОВЕТСКИХ КЛОНОВ. МЫ ВСЛУШИВАЛИСЬ В ШИПЕНИЕ ДИНАМИКОВ И ПОСТЕПЕННО СРОДНЯЛИСЬ С МЕХАНИЧЕСКИМ ГОЛОСОМ, КОТОРЫЙ ШЕПТАЛ НАМ: «РЕДАКТОР И АССЕМБЛЕР…РЕДАКТОР ТЕКСТОВ.. КЛАД…ПЭКМЕН…» |

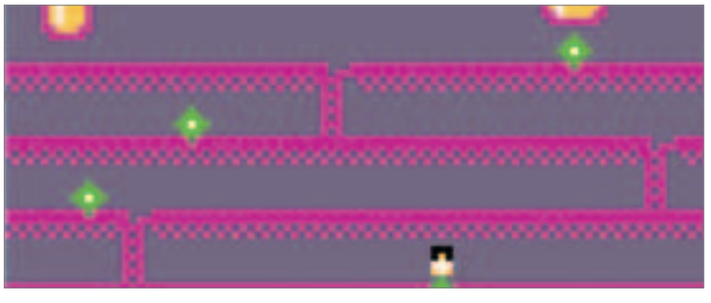
ЛОЗОВСКИЙ АЛЕКСАНДР (ALEXANDER@REAL.XAKEP.RU)

лестница

В далеком 1989 году я впервые увидел компьютер. Мне купили «Микрошу» — суперсоветскую ЭВМ, которая не может существовать без телевизора и магнитолы. На многократно зажеванной и разглаженной пленке кассеты MK60, помимо кучи полезных программ, были и игры, а среди них — та самая «Лестница». Идея ее проста: человечек идет снизу вверх по лабиринту (точнее, даже не по лабиринту, а просто по уровням, соединенным лестницами, причем игровое поле открывается взору полностью). Наверху же игрового поля находится один (несколько) источников, из которых вываливаются скачущие камни. Камни (в виде символа «0») катятся вниз по уровням и лестницам, подпрыгивают и норовят раздавить игрока. Цель — долезть до верха. Уровней, причем самого разного дизайна, было куча. Я, к примеру, дошел до 14-го и нисколько не растерял игровой интерес :). Не знаю, как эта игра будет восприниматься сейчас, но тогда она затягивала просто страшно. Кстати, как ты, наверное, уже читал в Интро, именно эту игру я пытался портировать, но в итоге посеял исходники из-за проблем с Чубайсом.

ЧТО НАС ПОКОРИЛО?

КНОПОЧКА «ГТ» НА КЛАВИАТУРЕ. ЕЕ НАЗВАНИЕ РАСШИФРОВЫВАЕТСЯ ВОВСЕ НЕ «ГЛУТАМИЛТРАНСПЕПТИДАЗА» ИЛИ «ГОТОВ К ТРУДУ», КАК МОЖЕТ ПОКАЗАТЬСЯ КОМУ-НИБУДЬ :), А НЕИЗВЕСТНО КАК. ГЛАВНОЕ — ТО, ЧТО ОНА ВВОДИТ СИМВОЛ, ИЗОБРАЖАЮЩИЙ ЧЕЛОВЕЧКА.



быстрый счет

Не имеющая аналогов в мире игра. Есть две беговые дорожки (последовательность знаков «====» тогда называлась границей дорожки :)). В верхней — наш ГТ-человечек, в нижней — вражеский. Вражеский идет медленно, но неумолимо вперед. Наш двигается быстро, широкими скачками, но… он не может без допинга. А допингом является правильное устное решение арифметического примера, который высвечивался вверху экрана. Решать нужно было в уме, пользоваться калькулятором — читинг, к тому же тогда калькуляторы были роскошью :(. На уроках математики — и вовсе за использование мажорной «Электроники» могли надавать линейкой по рукам :). Игра имела кучу уровней сложности и недетски развивала способность к устному счету. В общем, правильная социалистическая игрушка. Must have, особенно для однозадачных «Пальмов» :).

ЧТО НАС ПОКОРИЛО?

КОММЕНТАРИИ В КОНЦЕ. ХУДШИЙ РЕЗУЛЬТАТ — «НЕ ХОДИТЕ В МАГАЗИН — ОБМАНУТ». ЛУЧШИЙ — ЧТО-ТО ВРОДЕ «ТАКОМУ ЯЙЦЕГОЛОВОМУ ДАСТ ЛЮБАЯ EX-СССР-ДЕВУШКА, ДАЖЕ ЕСЛИ У НЕГО НЕТ МАЙКИ ОТ ADIDAS» (ВОЗМОЖНО, ПОСЛЕДНЯЯ ФРАЗА РОЖДЕНА МОЕЙ ФАНТАЗИЕЙ И НЕ МОЖЕТ ЯВЛЯТЬСЯ ОСНОВАНИЕМ ДЛЯ ВЧИНЕНИЯ ИСКА.

порнотетрис, sexonix

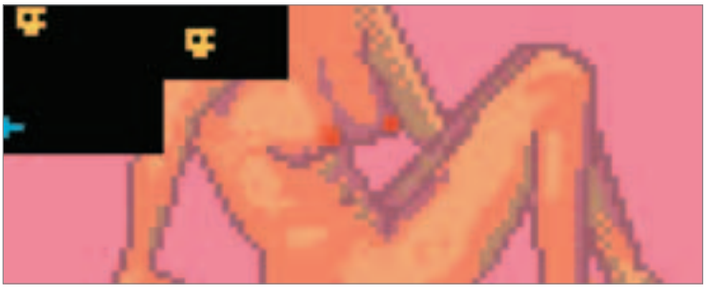
Об этих играх я бы не написал ничего, если бы не настойчивые просьбы коллеги, пожелавшего остаться неизвестным (назовем его условно — «Sky»). Итак, в те далекие времена, когда кибер-онанистов было немного и они были не такие пресыщенные, как сейчас :), им приходилось довольствоваться малым, а именно — вот такими простыми и безыскусными играми. Например, задача «Порнотетриса» — с каждым заполненным уровнем не начислять очки, а открывать часть порноклипа (плохого качества, конечно же). В SeXonix'е же мы отвоевываем у кариозных монстров не просто абстрактную территорию, а эротическую фотографию. В принципе, правильно. Нужно же бороться за что-то светлое :).

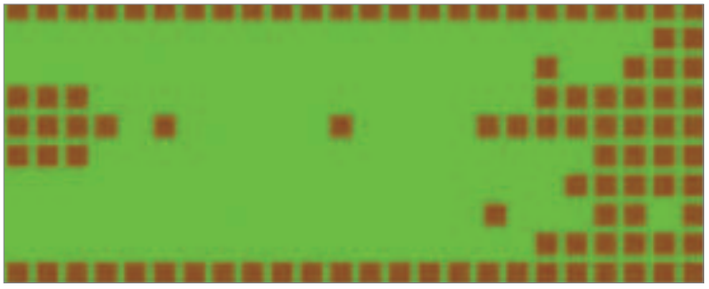
МОРАЛЬ:

XONIX НУЖНО ПОРТИРОВАТЬ. АССОРТИМЕНТ КАРТИНОК ПОПРОШУ ПОРАЗНООБРАЗИТЬ!

ЧТО НАС ПОКОРИЛО:

ИГРА ОДНОЙ РУКОЙ ;).





игра без названия

Внизу карты ползает танк. Сверху на него медленно, но верно плывут фигуры и наша задача — ДОСТРОИТЬ каждую фигуру до ровного прямоугольника выстрелами из танка. Видимо, именно тогда, достигнув критической массы, она уничтожится сама. В качестве бонуса — три бомбы, очищающие весь экран от вражеских фигур. Также, естественно, постепенно фигуры ускоряются.

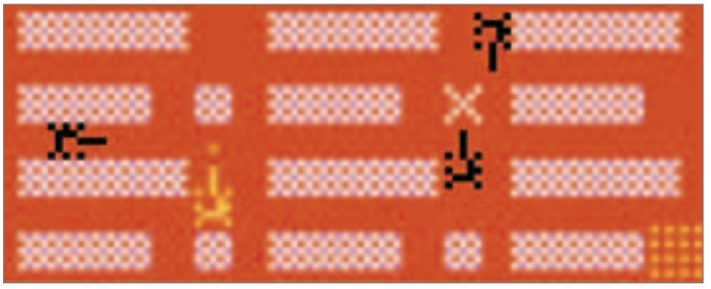
МОРАЛЬ:
ИНТЕРЕСНЫЙ СЮЖЕТ, НЕИЗВЕСТНЫЙ АВТОР, СУПЕРИНТЕРЕСНАЯ ИГРА. ЗАВЕРНИТЕ!

boomberman

Игра-король! Куча клонов, куча модификаций. Маленький анархист с бомбой ©, бегающий по лабиринтам, взрывающий и убивающий все и вся — практически супергерой поколения 90-х. Дюк Нюкем, Терминатор, Коммандер Кен и Рэмбо — все это дети по сравнению с стареньким низкорослым тротиловым парнем. Игра постоянно переделывалась всеми кому не лень, и в результате превратилась в трехмерную музыкальную порнографию (в худшем смысле этого слова), в которую часами уже не поиграешь.

Как BoomberMan может быть трехмерным? Как ходячие лампочки-враги могут быть супердетализированными? Не стоит повторять ошибки таких разработчиков. Возрождать игры нужно в том виде, в котором мы их полюбили. Только улучшать их уже некуда. Они идеальны. В свете последних политических событий, наверное, эта игра будет запрещена, а человек, собравшийся возродить ее — расстрелян или переправлен на базу Гуантамо для уточнения связей с мировым терроризмом. Но если ты не боишься — дерзай, для покета будет супер. Главное — не забыть бонусы в духе взрыва бесконечной длины (весь коридор) и подрыва по кнопке (а не по таймеру: это так нравилось нам). BoomberMan уже есть на Java, но получилось неидеально ;).

ЧТО НАС ПОКОРИЛО:
В-О-О-ОМ! ЭТА ХОДЯЧАЯ ЛАМПОЯКА СЕЙЯАС СДОХНЕТ!



танки

Эта игра, скорее всего, восходит к временам «Денди». На PC она пришла позже, и сюжет ее прост: в середине карты стоит база из кирпича. Около базы стоит пара танков: твой и друга-человека. Задача — оборонять базу и убивать другие танки. В свою очередь, вражеские танки могут раздолбать кирпич базы и уничтожить ее, организовав нам game over. Из хороших бонусов запомнились силовое поле, способность плавать по воде, способность пробивать бетон (кирпич можно уничтожать и базовым оружием) и абсолютная тупость AI: глупые или пьяные танкисты только ездили и палили в белый свет, но, поскольку их было много, шансов разнести игрока оставалось порядочно. Игра — класс, так почему же не сделать ее мультиплеером по Bluetooth'у? Погонять на лекции в танки — неплохая идея :). Кстати, несколько лет назад я видел реализацию этой игры на Delphi с открытым сорцом.

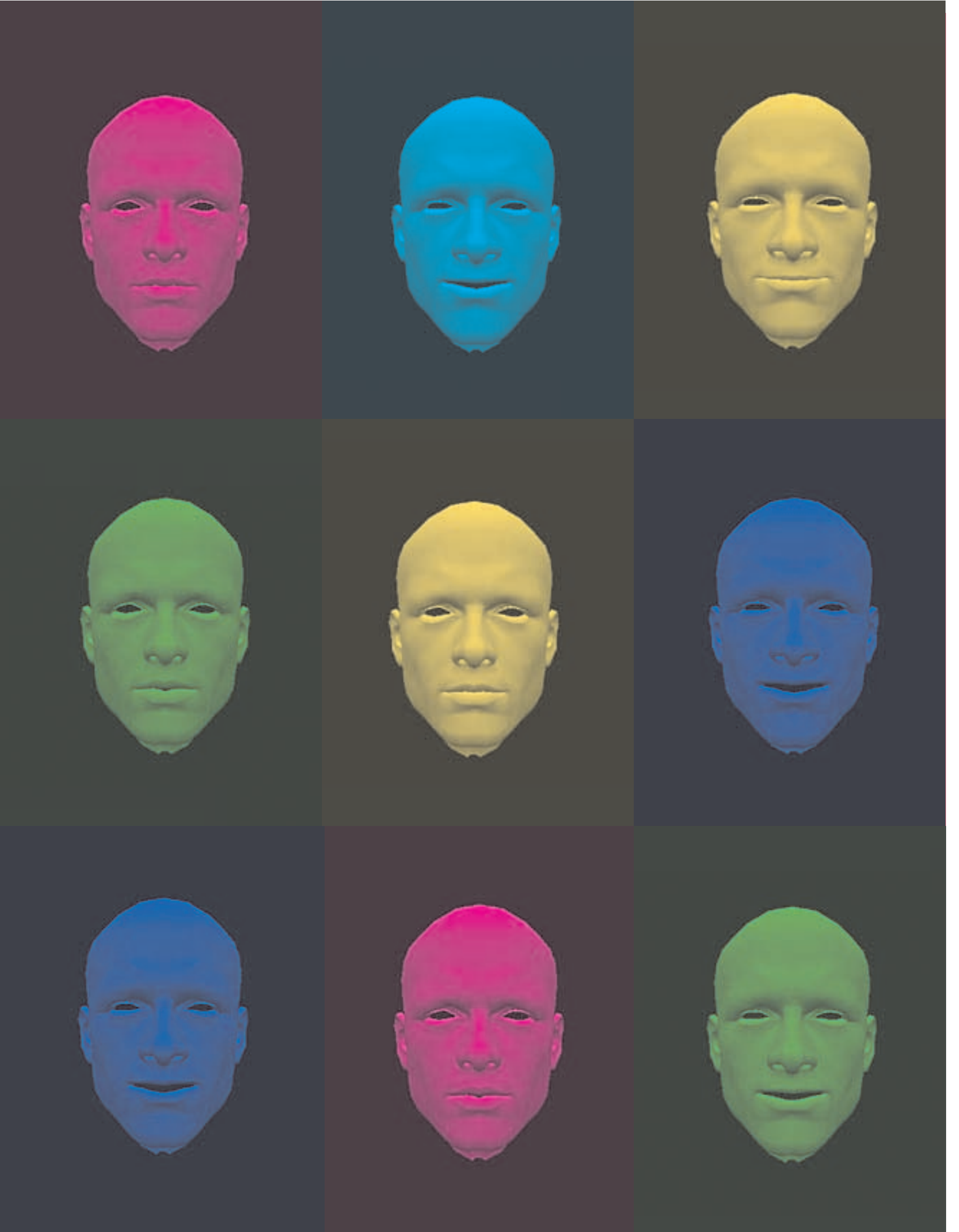
castle war & scorched earth

Идея проста, как заповеди Мао Цзедуна из его красной книжечки. Есть две крепости. Есть три клавиши. Регулируем силу выстрела и траекторию, стреляем…и не попадаем во вражескую крепость. Как быть? Корректируем, стреляем. Попадание! Часть крепости снесено, но на этом игра не заканчивается: она пошаговая и оппонент имеет шанс отомстить. Что хорошего в этой игрушке? Она простая, она затягивает, в нее можно играть вдвоем (по Bluetooth? ;)) и стилуса хватит с лихвой :).

Scorched Earth — намного более продвинутая коммерческая игра. Несколько танков (и несколько игроков, управляющих ими) ведут смертельный бой с помощью кучи вооружений от пушки до мини- и макроядерной боеголовки. Хотя нет, ядерная боеголовка — не предел. Есть еще Рука Смерти — ужасный зонтик, который накрывает огромное пространство! Правда, его элементы могут рикошетить от края карты и угробить самого бойца. Силовое поле не спасет, Рука уничтожает землю, и смерть наступит от падения в котлован (если не купить парашют)

МОРАЛЬ:
CASTLE WAR — ЭТО ЦАРСКАЯ ИГРА. ПОРТИРОВАТЬ ЕЕ БЫСТРО, ДЕШЕВО, А ИГРА — СУПЕР! С «ВЫЖЖЕННОЙ ЗЕМЛЕЙ» ПОСЛОЖНЕЕ, НО ЭТА ИГРА — СТОПРОЦЕНТНЫЙ УБИЙЦА WORMS'ОВ ДЛЯ ПОКЕТА. КАЧАЙ ОРИГИНАЛ С HTTP://386.BY.RU/GAME/CASTLE.ZIP И ПРИОБЩАЙСЯ.

ЧТО НАС ПОКОРИЛО:
ЭХ, КАК ОН ПОПАЛ ПРЯМО В ЦЕНТР? Я ЖЕ ЕГО ПОЧТИ СРЫЛ… НУ ВОТ СЕЙЧАС…



проблемная рожа

ЛИЦЕВАЯ МОРФИРУЮЩАЯ АНИМАЦИЯ

ЛЮБОЙ УВАЖАЮЩИЙ СЕБЯ ТРЕХМЕРНЫЙ ПЕРСОНАЖ СОВРЕМЕННОЙ КОМПЬЮТЕРНОЙ ИГРЫ ДОЛЖЕН УМЕТЬ ГОВОРИТЬ И ВЫРАЖАТЬ СВОИ ЭМОЦИИ МИМИКОЙ И ЖЕСТАМИ. ЗАСТЫВШИЕ ЛИЦА-МАСКИ ВПИСЫВАЮТСЯ ДАЛЕКО НЕ В КАЖДЫЙ СЮЖЕТ |

АЛЕКСАНДР ГЛАДЫШ, АЛЕКСАНДР ФЕДОРА (WWW.STEPGAMES.RU)

Идеальное в плане кинематографичности решение того, как поставить диалоги в игре, — позволить аниматорам смоделировать каждую реплику персонажа в 3D-редакторе как отдельный трек анимации, со всеми жестами, мимикой и положениями губ, соответствующими записанному тексту. Совсем как если бы делалась не компьютерная игра, а анимационный ролик.

Похожее решение применяется в большинстве игр — в пререндеренных видеовставках. Однако видеовставки не всегда приемлемы в обычных игровых диалогах, потому что они не обеспечивают достаточной интерактивности. Например, что делать, если нужно отобразить интерактивный диалог главного героя фэнтезийной ролевой игры и одного из рядовых персонажей, причем если главный герой одет в произвольную комбинацию из всех представленных в игре видов брони?.. Таких диалогов в игре может быть очень много. Создание интерактивного видеоролика для каждого диалога обошлось бы слишком дорого. Чтобы решить эту проблему, для отображения диалогов пользуются возможностями движка игры.

Однако если экспортировать каждую реплику как анимационный трек, размеры моделей раздуются до неприемлемых масштабов и работа с ними в движке серьезно осложнится. Кроме того, если игра насыщена диалогами, станет необходимо анимировать каждую реплику персонажей в игре, соответ-ственно, аниматоры получат на свои плечи непосильную нагрузку. Другими словами, необходим компромисс между кинематографичностью отображения диалогов персонажей в игре и ограничениями по производительности движка, а также по объему работы, нужной для создания этих диалогов. Как решать проблему? Автоматически генерировать анимацию персонажа средствами движка в диалоге — на основе текста и озвучки реплики.

речь и эмоции Речь на низком уровне делится на морфемы — минимальные элементы, несущие смысл высказывания (упрощенно — деление на слова). Морфемы, в свою очередь, состоят из фонем — минимальных элементов, получаемых линейным членением речи (упрощенно — состоят из звуков и слогов).

> «НА ДИСКЕ ТЫ НАЙДЕШЬ ПРИМЕР РЕАЛИЗАЦИИ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ ЛИЦЕВОЙ АНИМАЦИИ, СИНХРОНИЗОВАННОЙ С РЕЧЬЮ. ПРИМЕР НАПИСАН С ИСПОЛЬЗОВАНИЕМ DIRECTX FRAMEWORK»

Выражение лица, соответствующее данной эмоции, определяется естественным положением мышц лица, которое соответствует эмоциональному состоянию человека в данный момент времени. Список выражений лица можно составлять очень долго. Для практических целей обычно достаточно четырех основных: радость, злость, ужас и удивление. Договоримся называть выражение лица, характерное для данного элемента речи или эмоции, виземой.

разбиение речи на виземы Синхронизация положения губ персонажа с озвучиванием называется английским термином lipsync (lip — губа, sync — от synchronization, «синхронизация»). В качестве русского термина обычно используется прямая транслитерация — липсинк. Чтобы анимировать говорящего персонажа, нужна комбинация анимации его эмоций и анимации его речи. Необходимо поставить визему эмоции и визему речи в соответствие каждому моменту произношения реплики.

Виземы речи можно вычислить анализируя звуковой файл и текст реплики. Чтобы вычислить виземы эмоций, нужно попросить сценариста специальным образом разметить текст реплики. Обычно он все-таки делает это для удобства актеров, и достаточно только формализовать процесс так, чтобы конечные файлы поддавались распознаванию программой.

Уровень детализации, на котором представлены виземы речи, можно изменять в зависимости от того, анимацию какого уровня детализации ты хочешь получить в конечном итоге. Опыт подсказывает, что если модели персонажей в игре располагаются в основном относительно далеко от камеры, можно ограничиться разбиением на уровне морфем — расстояние скроет мелкие огрехи в синхронизации. Также большая детализация не нужна, если ты не можешь позволить себе такое количество полигонов в голове модели, которое достаточно для ее отображения. Если же ты хочешь, например, давать крупные планы высокополигональных голов моделей, то ради реалистичной анимации, скорее всего, придется перейти на уровень фонем.

разбиение речи на уровне морфем

Рассмотрим упрощенный подход с разбиением речи на уровне морфем, который, судя по практике, несмотря на свою простоту дает приемлемые результаты, если детализация анимации будет достаточно низкой. Задача по такому разбиению (как, впрочем, и по разбиению при более высокой детализации) значительно упрощается: анализируемый звуковой поток обычно записан на студии и содержит речь одного человека без фона и серьезных шумовых помех. Благодаря этому можно просто считать сигнал выше заданного уровня морфемой, ниже этого уровня — паузой между морфемами.

При таком подходе для анимации речи персонажа проще всего использовать две виземы. Первая — нейтральное лицо без всякого выражения, рот закрыт. Вторая — лицо без выражения, но рот открыт. Между морфемами воспроизводится лицо с закрытым ртом. Во время звучания морфемы воспроизводится циклический переход из виземы нейтрального лица в визему с открытым ртом, и так имитируются движения губ во время речи. Обрати внимание на то, что такой циклический переход не обязательно должен быть построен по синусоидальному закону. Чтобы привнести реализма, в не-

го можно добавить хаотичность. Главное — чтобы анимация начиналась и заканчивалась фазами с закрытым ртом, чтобы в результате плавно состыковать с участками тишины между морфемами.

Для анимации эмоций обычно достаточно использовать одну визему на одну эмоцию. Если для большей реалистичности нужно заставить персонажа моргать, можно выбрать один из двух простых способов. Первый — считать моргание «эмоцией» и с некоторой периодичностью вставлять ее между другими эмоциями. Второй способ — сделать третью визему для речи с закрытыми глазами и, например, закрытым ртом. Затем периодически подменять на нее соответствующую визему с открытыми глазами.

реализация анимации модели

Непосредственно анимацию модели можно реализовать при помощи морфирующей либо скелетной анимации со скиннингом.

Скелетная анимация — вид анимации, при котором основным рычагом управления аниматора являются кости анимируемого им скелета. Скиннинг (англ. skin — «кожа») — метод привязки вертексов модели к костям скелета, трансформированным при помощи скелетной анимации. Если используется этот метод, на положение одного вертекса могут влиять кости в количестве больше одной штуки. Когда скелетная анимация со скиннингом реализуется в реальном времени, обычно накладывается ограничение по максимальному количеству костей, к которому может быть привязан вертекс модели (обычно не более четырех).

Морфирующая анимация — вид анимации, при котором итоговое состояние модели в данный момент просчитывается при помощи интерполяции нескольких ключевых кадров. Каждый кадр такой анимации — это анимируемая модель целиком. Итоговое положение данного вертекса модели при морфинге определяется как среднее взвешенное положений этого вертекса в рассматриваемых ключевых кадрах. Интерполяция кадров может проводиться как по времени, так и по каналам анимации.

Обычно скелетная анимация поддерживается движком игры вне зависимости от того, как именно реализована лицевая анимация персонажей. В то же время скелетная анимация сложна для адекватного представления лицевой анимации, особенно с учетом ограничения количества костей на вертекс: для эффективной передачи мимики требуется использовать анимационные кости как аналоги множества мышц лица. Морфинг же позволяет анимировать лицо произвольными, удобными художнику средствами, не привязываясь к мизерному набору возможностей его анимационного пакета, который поддерживается движком.

практическая реализация

Посмотрим некоторые вопросы реализации, приведенной в примере (ищи на диске).

Используется морфирующая анимация. Для создания анимации используются три канала (в примере используется общий буфер). При необходимости число каналов может быть увеличено за счет некоторых потерь производительности.

В каждый момент времени в каждом из каналов выбран один из кадров морфирующей анимации. Кадры, выбранные в каналах, смешиваются согласно весам, вычисленным по данным синхронизации с речью. Таким образом мы получаем конечный кадр для данного момента времени. Текущий кадр задается как положение содержащегося в нем меша в общем вертексном буфере. Для случаев, когда видеокарта пользователя не поддерживает указание смещения для вертексного буфера, можно предусмотреть такой вариант, когда используется один вертексный буфер на каждый кадр.

В первом (основном) канале всегда находится кадр с нейтральным выражением лица и с закрытым ртом. Во втором — текущая визема положения губ. В нашем случае с делением на уровне морфем — в этом кадре всегда находится визема «открытый рот». В третьем — визема текущей эмоции. Текущие веса смешивания задаются для всех неосновных каналов (второго и третьего). Вес первого канала вычисляется исходя из того, что сумма весов всех каналов должна равняться единице и каждый вес должен быть строго больше нуля.

Итак, положение заданного вертекса меша модели в конечном кадре вычисляется так: суммируем положения этого вертекса в каждом канале, взяв их с весами, заданными для данного момента времени:

```
pos = pos1 * (1 — weight2 — weight3) + pos2
* weight2 + pos3 * weight3.
```

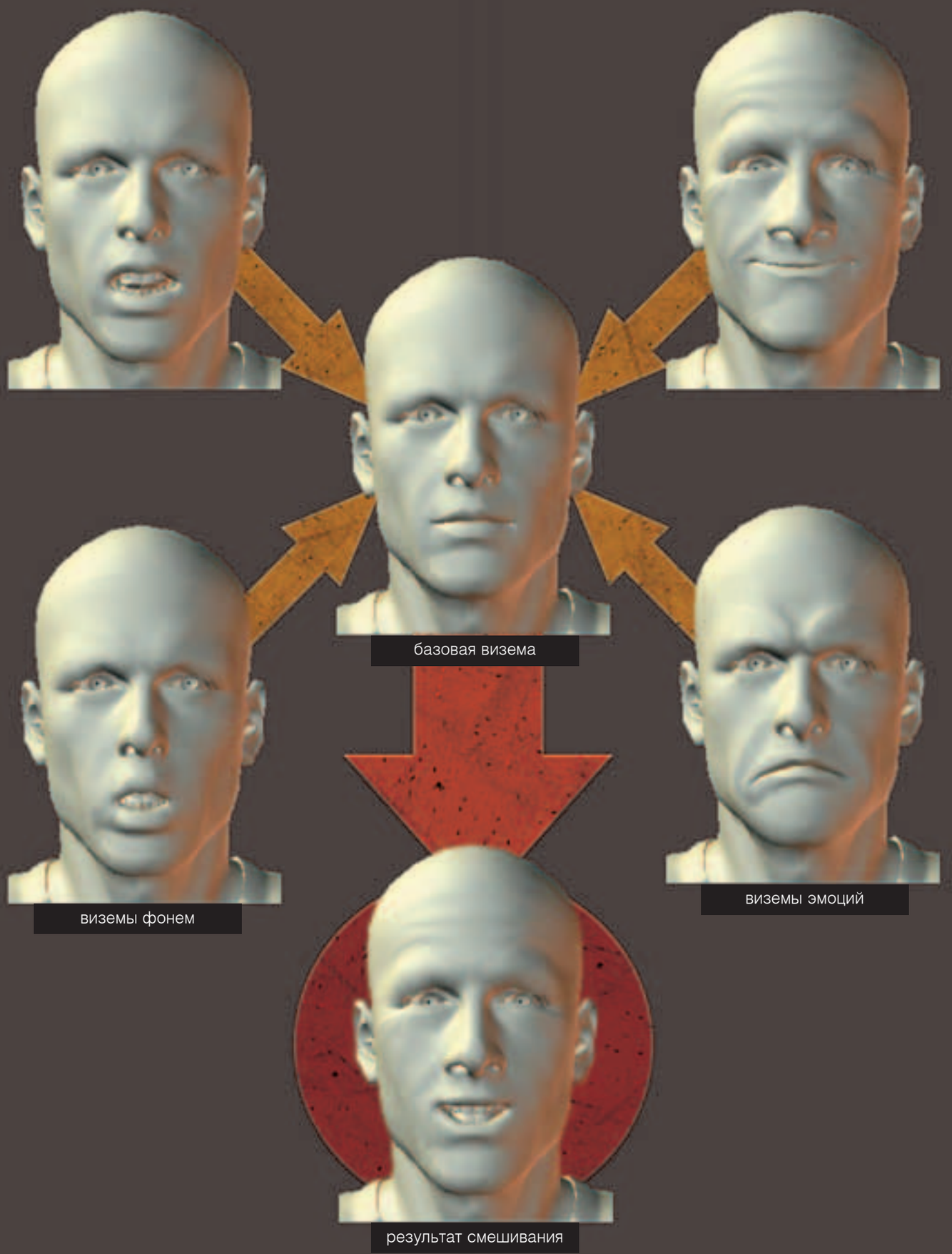
Вычисление конечного кадра производится в вертексном шейдере. Вертексы каждого кадра передаются в шейдер как поток вершин через вызов функции SetStreamSource() с соответствующим индексом потока, где и смешиваются по указанной формуле. Текущие веса каналов передаются в шейдер как константы. После того как конечное положение вертекса вычисляется, с ним производятся те же операции, что и обычно (например, рассчитывается освещение модели).

полный код вертексного шейдера на HLSL для морфирующей анимации из трех каналов (без работы с текстурами)

```
// Матрица преобразования.
float4x4 WorldViewProjection :
WORLDVIEWPROJECTION;

float3 weight; // Веса анимаций
      // weight.x — вес основного кадра
      // weight.y — вес кадра «открытый рот»
      // weight.z — вес кадра с эмоцией

// Вершинный шейдер
void vshader(
  float4 vPos0 : POSITION0,
// вершина из первого потока
  float3 vNormal0 : NORMAL0,
// нормаль из первого потока
  float4 vPos1 : POSITION1,
// вершины из второго потока
  float3 vNormal1 : NORMAL1,
// нормаль из второго потока
  float4 vPos2 : POSITION2,
// вершины из третьего потока
  float3 vNormal2 : NORMAL2,
// нормаль из третьего потока
  out float4 oPosition : POSITION,
// результирующая позиция вершины
  out float4 oDiffuse : COLOR0
// цвет вершины
 )
 {
  // Здесь смешиваются три кадра анимации
  в соответствии с заданными весами
  // Позиция:
  float4 cPos = vPos0 * weight.x +
        vPos1 * weight.y +
        vPos2 * weight.z;
  // Нормаль:
  float3 cNor = normalize(
    vNormal0 * weight.x +
    vNormal1 * weight.y +
    vNormal2 * weight.z
   );
  // трансформируем полученную позицию
  матрицей преобразования.
  oPosition = mul( cPos, WorldViewProjection );

  // Расчет примитивного освещения
  float3 lightPos = float3(0, 0, 100);
  float3 invL = normalize(lightPos — cPos);
  float NdotL = dot(cNor,invL);
  oDiffuse.xyz = NdotL + 0.2;
  oDiffuse.w = 0;
}
```

в итоге

В статье мы описали простой метод того, как можно реализовать лицевую морфирующую анимацию, с разбиением речи на виземы на уровне морфем. Этот же метод вполне годится для игр с низкополигональными моделями. Если введешь небольшие изменения, ты дополнишь синхронизацию, чтобы поддерживалась детализация на уровне фонем. Если хочешь заставить персонажа моргать во время речи, попробуй добавить новый специальный поток анимации для век или периодически заменяй некоторые виземы на их версии с закрытыми глазами модели. Можно еще оптимизировать логику работы с каналами, используя не занятые в данный момент каналы анимации, чтобы обеспечить более плавные переходы между анимациями


базовая визема
виземы фонем
виземы эмоций
результат смешивания

сетевые игры

РЕШЕНИЕ ПРОБЛЕМ РЕАЛИЗАЦИИ

СЕГОДНЯ КАЖДАЯ УВАЖАЮЩАЯ СЕБЯ ИГРА ОБЯЗАНА ПОДДЕРЖИВАТЬ МУЛЬТИПЛЕЕР — РЕЖИМ, ПРЕДНАЗНАЧЕННЫЙ ДЛЯ НЕСКОЛЬКИХ ИГРОКОВ, СВЯЗАННЫХ ЛОКАЛЬНОЙ СЕТЬЮ, МОДЕМОМ ИЛИ ИНТЕРНЕТОМ. НА ПЕРВЫЙ ВЗГЛЯД, ПРОСТАЯ ОБЯЗАННОСТЬ, НО ПОПЫТКА ПРАКТИЧЕСКОЙ РЕАЛИЗАЦИИ ОТКРЫВАЕТ ЦЕЛЫЙ ВОРОХ ПРОБЛЕМ, КОТОРЫЕ ИНОГДА ТРЕБУЮТ НЕОБЫЧНЫХ РЕШЕНИЙ |КРИС КАСПЕРСКИ АКА МЫЩЪХ

когда игровые миры соприкасаются Прежде чем погружаться в тонкости межсетевого взаимодействия, вспомним старые игры. В пошаговых стратегиях участники обычно управляли игрой по очереди. В играх реального времени такой подход был уже неприемлем и приходилось делить одну клавиатуру на двоих (хорошо если под руку попадался джойстик).

Идея: чтобы организовать сетевую игру, достаточно загнать весь ввод-вывод в магистральный кабель, реализовав удаленный монитор и клавиатуру (джойстик/мышь). Получится точно так же, как и раньше, только намного круче. С клавиатурой все просто, но чтобы осуществить передачу видеоизображения в реальном времени, даже если применить компрессию, понадобится локальная сеть или, по меньшей мере, DSL.

игра для двух игроков

Рассмотрим простейший случай — игру для двоих игроков. Один компьютер будет «ведомым», второй — «ведущим». Ведущий компьютер обсчитывает игровое пространство, управляет движением монстров (если они есть), обрабатывает столкновения и т.д. Для синхронизации игровых миров все события игрового пространства (перемещение объектов, взрывы мин, полеты торпед) передаются ведомому компьютеру, который отображает события на экране и отмечает их на своей карте. При этом неявно предполагается, что обе карты (ведомого и ведущего компьютеров) абсолютно идентичны. Если же в игровом пространстве присутствуют случайные элементы (например аптечки, расставленные в разных местах), они должны быть переданы ведомому компьютеру, что достаточно сделать всего один раз — перед началом игры. При желании можно даже передать всю карту целиком (нужно учитывать, что ведущий компьютер может использовать дополнительные или «самопальные» миссии, которых нет у ведомого). Ведомый компьютер, в свою очередь, управляет локальным игроком, отслеживая нажатия клавиш и отправляя эту информацию ведущему.

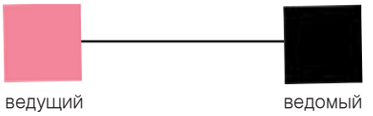


Схема взаимодействия ведомого и ведущего компьютеров в мире двух игроков

из локальной сети в интернет

Кажется, что все просто и никаких граблей здесь нет. В локальной сети — может быть, но передача данных через интернет сопряжена с периодичными задержками. Если не предпринять дополнительных мер, монстры будут двигаться как припадочные. Как же быть? А что если передавать не сами перемещения, а их предполагаемый сценарий? Обычно движение монстров подчиняется набору шаблонов, и компьютеру достаточно передать что-то типа «Двигайся от меня и до обеда/упора», «Атакуй по сценарию D» или «Уклоняйся по сценарию A». Количество передаваемой информации резко сокращается, и для обеспечения синхронизации становится достаточно периодически (скажем раз в секунду) передавать «квитки», сигнализирующие о пересечении объектом некоторой клетки игрового поля. Очевидно, что такой протокол передачи устойчив даже к длительным задержкам, что немаловажно при работе на сильно загруженных каналах.

ЛЮБИМАЯ СЕТЕВАЯ ИГРА. МНЕНИЯ С ФОРУМА

http://forum.xakep.ru/view.asp?topicID=64268

F1EX: ДЛЯ МЕНЯ ЛЮБИМОЙ СЕТЕВОЙ ИГРОЙ ОСТАЕТСЯ QUAKE III. ДОМА ПОСТОЯННО ИГРАЮ С СОСЕДЯМИ В ЭТУ ИГРУ ПО ЛОКАЛКЕ. ДАЖЕ НЕ МОГУ ОТОРВАТЬСЯ ОТ НЕЕ В СВОБОДНОЕ ВРЕМЯ — ИГРАЮ У СЕБЯ НА КПК, ОЧЕНЬ СИЛЬНО ПРИВЛЕКЛА ДИНАМИКА, ГЕЙМПЛЕЙ... И, ПОГРУЗИВШИСЬ В НЕЕ ПОЛНОСТЬЮ, ПЕРЕСТАЕШЬ ОБРАЩАТЬ ВНИМАНИЕ НА ГРАФИКУ (СКОЛЬКО ЛЕТ ОТ СОЗДАНИЯ ПРОШЛО). А ЕЩЕ МОЖНО ОТЫГРАТЬСЯ НА ДРУЗЬЯХ!

CYBERMIND: STARCRAFT — СТРАТЕГИЯ ВСЕХ ВРЕМЕН И НАРОДОВ! ХОТЯ ВЫПУЩЕНА ОНА В ДАЛЕКОМ 98-М ГОДУ, В НЕЕ И СЕЙЧАС ИГРАЕТ КУЧА НАРОДА. STARCRAFT ОЧЕНЬ ДИНАМИЧЕН, И, НА МОЙ ВЗГЛЯД, ДО СИХ ПОР НИ У КОГО ЕЩЕ НЕ ПОЛУЧИЛОСЬ ЕГО ПРЕВЗОЙТИ, ХОТЯ ПОПЫТОК СОЗДАТЬ ПОХОЖУЮ ИГРУ БЫЛО ОГРОМНОЕ МНОЖЕСТВО. В МУЛЬТИПЛЕЕР ИГРАТЬ ОЧЕНЬ ПРИЯТНО И НЕНАПРЯЖНО. ИГРАЮ УЖЕ ЧЕТЫРЕ ГОДА, И ДО СИХ ПОР ОН МНЕ НЕ НАДОЕДАЕТ! НАВЕРНОЕ, ИЗ-ЗА АБСОЛЮТНОЙ НЕПРЕДСКАЗУЕМОСТИ: СЕГОДНЯ Я БАТЯ, ЗАВТРА МЕНЯ РВУТ.

MCDAUN: DOOM 2 — СКОЛЬКО ЛЕТ УЖЕ, А Я ВСЕ В НЕГО РЕЖУСЬ.

SERJ: WARCRAFT III TFT. В ОТЛИЧИЕ ОТ СТРЕЛЯЛОК, МОЖНО ПОИГРАТЬ ПО ИНЕТУ ДАЖЕ ПО МОДЕМУ, ИБО В СТАРТЕГИЯХ ПИНГ НЕ ТАК КРИТИЧЕН, КАК ДЛЯ QUAKE III ИЛИ ЖЕ КС. У ЭТОЙ ИГРЫ ВЕСЬМА СКРОМНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ, ТАК ЧТО МОЖНО ПОИГРАТЬ ДАЖЕ НА СЛАБЕНЬКИХ КОМПАХ. В ЭТОЙ ИГРЕ МОЖНО ПОКАЗЫВАТЬ НЕОРДИНАРНЫЕ ТАКТИКИ, ИМЕННО ПОЭТОМУ ОНА ЦЕПЛЯЕТ ПРАКТИЧЕСКИ ВСЕХ, КТО ХОТЬ РАЗ ЕЕ ВИДЕЛ!

С игроками дела обстоят значительно сложнее. Допустим, два горячих мудреца стоят супротив друг друга и каждый пускает по торпеде. Если информация о перемещении одного из игроков хоть чуть-чуть запоздает, очень может случиться так, что с точки зрения ведомого компьютера торпеда пройдет мимо, а ведущий увидит, как противника разорвало в клочья. Первое, что приходит на ум, — осуществлять перемещение только после подтверждения. Игрок нажимает на стрелку, компьютер А (неважно, ведущий или ведомый) отправляет уведомление компьютеру В (первая стадия), компьютер В обновляет свое игровое пространство и посылает подтверждение компьютеру А (вторая стадия), компьютер А принимает его и перемещает игрока (третья стадия). Только в жизни... Игрок давит на клавишу, а фигурка на экране остается неподвижной (задержка передачи данных по сети). Что делает игрок? Давит на клавишу еще и еще! Если же случится задержка на третьей стадии, в игровых мирах вновь произойдет рассинхронизация и дело закончится лесом. Следовательно, возникает необходимость четвертой стадии, которая обеспечила бы дополнительный уровень подтверждений, но… Как же тогда все будет тормозить!

«быстрая» и «медленная» синхронизация

Можно использовать полуэвристический алгоритм. Когда игроки находятся в разных концах игрового поля, они перемещаются без подтверждений, но как только их игровые миры соприкоснутся (один игрок увидит другого или в поле их зрения попадет общий монстр), автоматически активируется режим «обмена подтверждениями».

Зная направление и скорость движения игрока/монстра/торпеды, удаленный компьютер может с высокой степенью достоверности рассчитать, произойдет ли столкновение на данном временном участке. Даже самый прыткий игрок не может менять направление своего движения несколько раз в секунду, поэтому достаточно подтверждать лишь изменение направления. Конечно, если возникнет задержка в сети, фигурка на экране отреагирует на действие игрока не сразу, но, во всяком случае, она не будет тупо стоять, а продолжит движение. Если перевести в субъективное восприятие, в этом случае поведение компьютера покажется человеку более реалистичным.

борьба с жульничеством

Еще одна серьезная проблема — читерство. Ведущий компьютер может мухлевать как угодно — код и данные игрового процесса находятся в полном ведении игрока. Что ему стоит, слегка повозившись с отладчиком, приобрести божественное здоровье или нескончаемые патроны? Некоторые программисты просто отмахиваются от этой проблемы, делая вид, что ее не существует: «Ты либо доверяешь ведущему компьютеру, либо не доверяешь».

На самом деле решение лежит на поверхности! Пусть ведомый и ведущий компьютеры периодически меняются ролями — на программном уровне реализуется достаточно просто и надежно защищает от взлома.

Конечно, хакер может исправить свой экземпляр программы так, чтобы пули не кончались, но подобный трюк сработает только тогда, когда его компьютер станет ведущим. Даже такой «половинчатый хак» дает огромное преимущество персонажу в игре, однако оба игрока находятся в равных полевых условиях и «хакерствовать» в свое удовольствие может каждый. Уже нет компьютера, владелец которого приравнивается к Богу...

GROKINN: OGAME.RU — КОСМИЧЕСКАЯ СТРАТЕГИЯ, ОТЛИЧНОЕ СООТНОШЕНИЕ ЭКОНОМИКИ И ФАЙТИНГА, ЛЕГКА В ОСВОЕНИИ НОВИЧКАМИ ОНЛАЙН-ИГР (КСТАТИ, ДЛЯ НОВИЧКОВ ТАМ ЕСТЬ ЗАЩИТА, НА НИХ НЕЛЬЗЯ НАПАДАТЬ). Я, НАПРИМЕР, ДО ЭТОГО НИКОГДА НЕ ИГРАЛ ВО ВСЯКИЕ COMBATS.RU (НЕИНТЕРЕСНО БЫЛО), А ТУТ ЗАЦЕПИЛО, УЖЕ НЕСКОЛЬКО МЕСЯЦЕВ РЕЖУСЬ. КРОМЕ ТОГО, ИГРА БРАУЗЕРНАЯ И БЕЗ FLASH, ИДЕТ В РЕАЛЬНОМ ВРЕМЕНИ (ИГРОК СПИТ — РЕСУРСЫ КОПЯТСЯ). ОСОБО МНОГО ТРАФИКА НЕ ЖРЕТ, ПОКАЗЫВАЕТ НЕБОЛЬШОЙ РЕКЛАМНЫЙ БАННЕР. В ОБЩЕМ, МЕНЯ ПРИКОЛОЛО.

SAMID: COMBATS.RU! ОСОБО ЧАСТО В ИГРЫ НЕ ИГРАЮ, НО ЕСЛИ ЕСТЬ ВРЕМЯ, ТО ТОЛЬКО ЭТА.

][EAL: НЕ ЗНАЮ, ПОЧЕМУ, НО МНЕ НИКОГДА НЕ НРАВИЛИСЬ РОЛЕВЫЕ ИГРЫ. ПОЭТОМУ Я ПРЕДПОЧИТАЮ ВСЯКИМ ЛИНЕЙДЖАМ И WOW ЭКШЕНЫ ТИПА КС (ИГРАЛ В НЕЕ ДВА ГОДА, ПОТОМ БРОСИЛ), БАТЕЛФИЛД (ВЗЯЛ ВТОРУЮ ЧАСТЬ ЭТОЙ ЗАМЕЧАТЕЛЬНОЙ ИГРЫ, ПОПРОБУЮ) И, ЕСТЕСТВЕННО, ТРЕТЬЮ QUAKE (В КОТОРУЮ Я ИГРАЛ ПО СЕТИ С САМОГО МОМЕНТА ЕЕ ВЫПУСКА, СЕЙЧАС ТОЖЕ НАДОЕЛО).

PUPKIN-ZADE: ЛУЧШАЯ СЕТЕВАЯ — BATTLEFIELD 2. ОПТИМАЛЬНОЕ СООТНОШЕНИЕ ТЕХНИКИ И ЛЮДЕЙ, СТРАТЕГИИ И ТАКТИКИ В СОВРЕМЕННОЙ ВОЙНЕ. ДО ЭТОГО БЫЛ JOIN OPERATION, ТОЖЕ НЕПЛОХО. НУ И COUNTER-STRIKE НА КРАЙНИЙ СЛУЧАЙ.

GACKT: ИЗ СТРАТЕГИЙ — STARCRAFT. ЭТО УЖЕ КЛАССИКА, В КОММЕНТАРИЯХ НЕ НУЖДАЕТСЯ. ИЗ ММОРПГ — LINEAGE2. ГРАФИКА, ГЕЙМПЛЕЙ, ДИНАМИКА (КАК ДЛЯ ОНЛАЙНОВОЙ ИГРЫ) — ВСЕ НА ВЫСОТЕ, ОСОБЕННО ЕСЛИ ИГРАТЬ НА ОФИЦИАЛЬНОМ СЕРВЕРЕ. WOW — ТОЖЕ НИЧЕГО, НО ХУДОЖНИКИ ПРОГНАЛИ, СЛИШКОМ ДЕТСКАЯ ПОЛУЧИЛАСЬ. ОСТАЛЬНЫЕ ФЭНТЕЗИЙНЫЕ ММОРПГ — ПРОСТО ЖАЛКАЯ ПОПЫТКА СКОПИРОВАТЬ ЛИНЕЙКУ :). А ИЗ ТУПЫХ СТРЕЛЯЛОК — UT2004, ГРАФИКА ХОРОШАЯ :). ОСТАЛЬНЫЕ НЕ ПРИЗНАЮ ВООБЩЕ.

RUSSO_TURISTO: ПОЧТИ ПОЛГОДА ИГРАЛ В ОТЕЧЕСТВЕННУЮ ММОРПГ «СФЕРУ». ЖЕСТКО ЗАТЯГИВАЮЩАЯ ИГРУШКА, КАК И ЛЮБАЯ ОНЛАЙН. НО КОЛИЧЕСТВО БАГОВ И НЕДОРАБОТОК НАСТОЛЬКО ВЕЛИКО, ЧТО БУКВАЛЬНО ВИДИШЬ АЛЬТЕРНАТИВНЫЕ ПУТИ РАЗВИТИЯ ПЕРСОНАЖЕЙ

три и более игроков

Играть вдвоем довольно скучно, в какой-то момент возникает желание привлечь третьего. А где трое, там и пятеро. И тут всплывают свои проблемы... Ведущий компьютер может обслуживать множество ведомых игроков, количество которых (в теории) ограничивается пропускной способностью канала и мощностью процессора. Найти мощный процессор — не проблема. Если учесть то, что передаются не сами перемещения, а изменения направления, с лихвой хватит и хлипкого модемного канала. Камень преткновения в другом: что произойдет, если ведущий компьютер внезапно отвалится (разорвется соединение или владелец просто устанет и пойдет спать). В ситуации с двумя игроками никакой проблемы не возникнет, так как если один из игроков «исчезает», другой автоматически переходит в режим одиночной игры. С тремя игроками такая стратегия уже не срабатывает, и получается, что один отдувается за всех! А оно ему надо?

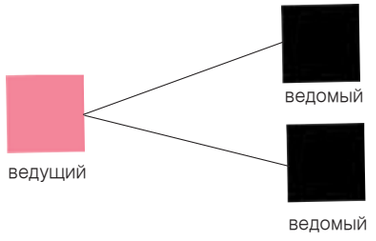


Схема взаимодействия компьютеров в игре из трех игроков

Кто-то наверняка предложит установить выделенный сервер, но тогда тройка-пятерка игроков натолкнется на решение, требующее расточительства. Так пусть все игроки устанавливают соединение не только с ведущим компьютером, но и друг с другом! Пусть по очереди становятся то ведомыми, то ведущими. В результате, если ведущий компьютер исчезнет, он просто-напросто будет выброшен из очереди и ничего катастрофического не произойдет.

Как вариант, можно совсем отказаться от концепции «ведомого» и «ведущего». Каждый компьютер управляет движением «своего» игрока и «своих» монстров, рассылая информацию о перемещении всем остальным. Такая схема существенно упрощает проектирование и реализацию игры, снимая множество проблем.

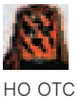
самоорганизующиеся системы с большим количеством игроков

Описанная схема для трех-пяти игроков имеет два серьезных недостатка. Если игроков очень много, объем трафика увеличивается настолько, что перестает вмещаться в любые каналы и появляются дикие тормоза, особенно в тот момент, когда игроки сходятся в смертельном поединке. Правило «семеро одного не ждут» тут не срабатывает и динамика игры определяется самым медленным компьютером в сети — все остальные терпеливо ждут, пока придет подтверждение. Приходится усложнять протокол и запрашивать подтверждение только у того, «против кого дружим». Допустим, игрок А пускает в игрока Б торпеду, а игрок С за этим внимательно наблюдает — интересно же :). Так вот чтобы не допустить рассинхронизации, компьютеры А и Б вынуждены обмениваться подтверждениями, в то время как компьютер С может и отдохнуть.

Проблема множественности соединений снимается, если создан своеобразный «поезд». Вместо того чтобы рассылать данные всем узлам, компьютер А посылает их компьютеру В. «В» посылает их (вместе со своими перемещениями) компьютеру С и т.д. Естественно, чтобы восстановить цепочку в случае «падения» одного из узлов, компьютер А должен знать адреса компьютеров С и D, чтобы при необходимости установить соединение с ними. На самом деле структура сети не обязательно должна быть линейной (это худший вариант). Наиболее типичный случай из жизни: компьютеры А и С находятся в локальной сети, а компьютер В — где-то далеко в интернете. За каким чертом мы будем гонять трафик через В, когда логичнее соединить компьютеры так: В=> А=> С? Сделать «так» действительно возможно!

И ГУБИТСЯ ВЕСЬ ГЕЙМПЛЕЙ. НА ОДНОМ БАГЕ, ДАЮЩЕМ НЕОГРАНИЧЕННОЕ КОЛИЧЕСТВО ИГРОВЫХ ДЕНЕГ, НЕКОТОРЫЕ ИГРОКИ ЖИЛИ КАК НА ЗАРПЛАТЕ, ОБМЕНИВАЯ ПОТОМ ИГРОВЫЕ ДЕНЬГИ НА «ЯНДЕКСДЕНЬГИ» ДРУГИМ ИГРОКАМ, КОТОРЫЕ НЕ ЮЗАЮТ БАГИ.

BELOZ: WOW ОДНАЗНАЧНО РУЛИТ! МНЕ ВСЕГДА НРАВИЛАСЬ СЕРИЯ WARCRAFT, А ТУТ BLIZARD ВЫПУСТИЛ WOW. Я ПОПРОБОВАЛ, И ОЧЕНЬ ПОНРАВИЛОСЬ. ХОТЯ ГРАФИКА В ИГРЕ ДЕЙСТВИТЕЛЬНО КАКАЯ-ТО ДЕТСКАЯ НЕМНОГО... ИГРАЮ СЕЙЧАС ТОЛЬКО В НЕЕ, КОГДА ЕСТЬ СВОБОДНОЕ ВРЕМЯ.

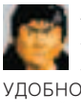


QUESTO: РУБЛЮСЬ В WOLFENSTEIN ENEMY TERRITORY. СКАЧАТЬ МОЖНО ОТСЮДА: **http://strimarena.ru/games/et**. ВЕРСИЯ ПОД *NIX'Ы БЫЛА НА КАКОМ-ТО DVD ИЗ «ХАКЕРА», ТОГДА Я НА НЕЕ И ПОДСЕЛ :).

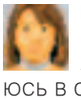
SUGAR: А Я ПОСЛЕДНЕЕ ВРЕМЯ ЗАВИС В LINEAGE2 — ВОТ ЭТО ТЕМНАЯ ИГРУШКА. ТРАФИКА ПОЧТИ НЕ ЕСТ. ГРАФИКА ПРЕКРАСНА. РУБИЛ РАНЬШЕ В WOW, НО СЛИШКОМ МУЛЬТЯШНА. А ЛИНЕЙКА ХОРОША. ЗАЛИТЬ МОЖНО С **www.l2online.ru**, ДИСТРИБУТИВ БЕСПЛАТЕН, ВХОД ТОЖЕ. У МЕНЯ ТАМ ХОРОШАЯ БОТОАРМИЯ :). НАДО БЫ МОБИЛЬНОЕ УПРАВЛЕНИЕ ИМ СДЕЛАТЬ...

HO@XER: КАЗАКИ! ПЕРВАЯ ЧАСТЬ И ВТОРАЯ. ПРОСТО ПОМЕШАН! С ДРУГОМ ПОЧТИ КАЖДЫЙ ДЕНЬ НОЧЬЮ СИДИМ КАК НЕНОРМАЛЬНЫЕ. УЖЕ ГОДА ДВА НЕ ПЕРЕСТАВАЯ ИГРАЮ. ПРИЧЕМ НА КОМПЕ НЕТ ДРУГИХ ИГР — ОДНА ЭТА!

KREEZZIS: ЛЮБИМЫЕ ИГРЫ — ЭТО WARCRAFT III TFT, QIII, CS. НО СЕЙЧАС СОВСЕМ НЕ ИГРАЮСЬ, ПОДНАДОЕЛО ЭТО ДЕЛО... А ВООБЩЕ НЕКОТОРОЕ ВРЕМЯ НРАВИЛИСЬ «КАЗАКИ», STARCRAFT, UT2004, HEROYES M&M 4 И СЕРИЯ NEED FOR SPEED.

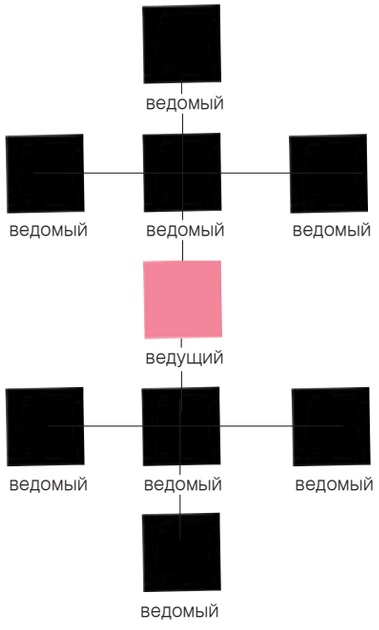


ANDRUSHA: А МНЕ НРАВИЛАСЬ ИГРА КРЕСТИКИ-НОЛИКИ «ПЯТЬ В ДЛИНУ» :). УДОБНО БЫЛО СОВМЕЩАТЬ С РАБОТОЙ, РАЗВИВАЛО МЫШЛЕНИЕ, ПЛЮС СЛУЧАЙНЫЕ ЗНАКОМСТВА С ИГРОКАМИ, ИНОГДА ДЕЙСТВИТЕЛЬНО ИНТЕРЕСНЫМИ...



MAD HAMSTER: ПОСЛЕДНИЕ АНРИЛЫ! И В ПОСЛЕДНЕЕ ВРЕМЯ ВЛИВАЮСЬ В САНАНДРЕАС ПО НЭТУ!

динамическое балансирование нагрузки



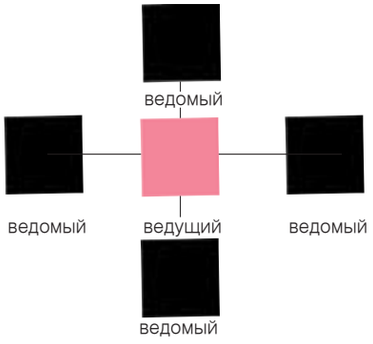
Система обмена в игре с множеством игроков

Выбираешь самый быстрый компьютер с мощным процессором (выбираешь автоматически — по скорости передачи данных и времени отклика на ping), подключаешь к нему чуть-чуть менее быстрые компьютеры, которые будут нести на своих плечах еще менее быстрые и т.д. Теперь весь трафик ложится на плечи узлов, висящих на быстрых каналах. Эта схема не является заданной раз и навсегда, она может (и должна) динамически обновляться. Главное — выбрать протокол обмена так, чтобы равномерно распределить трафик на все узлы.

Чтобы игроки (их количество в этой схеме измеряется десятками) не толпились на крошечном пятачке, необходимо создать достаточно большое игровое пространство, которое не сможет обсчитать отдельно взятый Pentium. Значит, потребуется распределенная система, нарезающая «объекты» игрового мира на куски и раскидывающая их по наименее загруженным машинам.

Использованная ранее схема «каждый компьютер обрабатывает своих монстров» уже не котируется, так как когда некоторый объект (монстр или игрок) попадет в поле зрения «чужого» объекта (монстра или другого игрока), для синхронизации игровых миров требуется задействовать механизм подтверждений, а эти тормоза вполне терпимы в случае трех-пяти игроков, но неприемлемы в мире, где их десятки. Допустим, ты модернизируешь протокол обмена и решил, что пусть каждый компьютер обрабатывает только тех монстров, которые находятся в его игровом пространстве (то есть если монстр уходит из «своего» игрового пространства, его обработка передается другому компьютеру). Будет работать замечательно до тех пор, пока все монстры не соберутся в пространстве одного игрока, и тогда его компьютер конкретно ляжет, будучи не в состоянии обрабатывать эту армию в реальном времени. Так что без распределенной системы с автоматическим балансированием нагрузки никуда! К сожалению, подобные системы очень сложны в реализации и отладке, в общем случае они не оправдывают себя и оказывается выгоднее установить выделенный сервер.

выделенный сервер



Игровой мир с выделенным сервером

Выделенный сервер — фактически тот же самый ведущий компьютер, но без заморочек. В отличие от обыкновенного компьютера, сервер не отваливается внезапно, чем сразу же снимает целый комплекс проблем. Задача синхронизации в подобной централизованной структуре решается сама собой, поскольку клиент отображает на экране только то, что сказал ему сервер, а уж игровой мир сервера всегда поддерживается в синхронном состоянии с самим собой. Есть и недостатки. Поддержка постоянно работающего сервера — удовольствие не из дешевых. Обычного хостинга за $8 в год с Perl, PHP и MySQL тут явно недостаточно. Потребуется прямой доступ к машине с возможностью выполнять двоичные файлы, написанные под XP (или что у тебя там), и очень мощный процессор, способный поддерживать огромный игровой мир в подвижном состоянии.

Что самое неприятное, подобная «принудительная самосинхронизация» означает, что клиенты лишаются какой бы то ни было самостоятельности и вынуждены передавать все свои передвижения на сервер, ожидая подтверждения. Даже если игрок решил заняться онанизмом в тихом дальнем углу, он все равно будет двигаться рывками и слегка притормаживать!

Плюс игроки (живые люди, а не фигурки) находятся в полной зависимости от сервера, компании-разработчика и своего провайдера. Уже нельзя собраться в кучу и поиграть по локалке. Нужно обязательно выходить в интернет или… устанавливать свой собственный сервер. Только как его установишь, если в открытом доступе его нет?

Очень остро встает и проблема читерства. Чем больше игроков, тем выше вероятность обмана. И хотя код, управляющий игровым пространством, находится на сервере, непосредственно залезть в который никак нельзя, хакеры могут повесить бот, то есть написать скрипт, управляющий движением игрока и с нечеловеческой меткостью и быстротой мочащий все, что попадается на глаза. Увы, защититься от этого никак нельзя! Единственное, что остается, — распознавать обманщиков по слишком большому количеству трупов, оставленных в единицу времени, и ставить им бан. Однако всегда найдется игрок, который поднимет вселенский визг, возмущаясь, «за что» его «так»?! Он же играл по всем правилам, а меткость и реакция, как известно — не порок.

В принципе, обработку игровых миров можно поручить и клиентам, а сервер будет только коммутировать потоки данных и распределять нагрузку по узлам. Короче говоря, приходим к той же самой самоорганизующейся системе, но только с центральным сервером, что снимает требование к вычислительной мощности сервера и отменяет принудительную синхронизацию, однако вызывает много путаницы с «отрубавшимися» клиентами, а серверу приходится постоянно перепроверять, был ли обработан данный блок игрового пространства. При огромном множестве клиентов это нереально, поэтому на практике часто используются гибридные схемы. Весь игровой мир хранится на сервере, но максимум перемещений обрабатывается локально: схватка двух игроков неизбежно происходит через сервер, а разборки игрока с монстром может обработать и сам клиент, доложив серверу конечный результат (кто кого порешил). Естественно, в этом случае со стороны клиента возможно наглое читерство...

От централизованного сервера не так уж и много пользы. Не стоит думать, что наличие сервера решит все проблемы. Проблем будет ненамного меньше, чем в самоорганизующейся распределенной системе, и это при всех недостатках, присущих серверу!

защита игр от взлома

ПРАКТИЧЕСКИЕ ПРИЕМЫ

НАПИСАТЬ ИГРУ — ПОЛДЕЛА, НУЖНО И ЗАЩИТИТЬ ЕЕ ТАК, ЧТОБЫ НЕ ВЗЛОМАЛИ (ИЛИ ВЗЛОМАЛИ, НО НЕ СРАЗУ). ЗАЩИТА ДОЛЖНА БЫТЬ МАКСИМАЛЬНО ПРОСТОЙ И БЕЗГЛЮЧНОЙ. НА ЭТУ ТЕМУ НАПИСАНО МНОЖЕСТВО СТАТЕЙ, НО ОЧЕНЬ МАЛО «РЕЦЕПТУРНЫХ» СПРАВОЧНИКОВ, ДОХОДЧИВО И БЕЗ ЛИШНЕГО, БЕЗ ВОДЫ ОБЪЯСНЯЮЩИХ, КУДА ИДТИ И ЧТО ДЕЛАТЬ | **КРИС КАСПЕРСКИ АКА МЫЩЪХ**

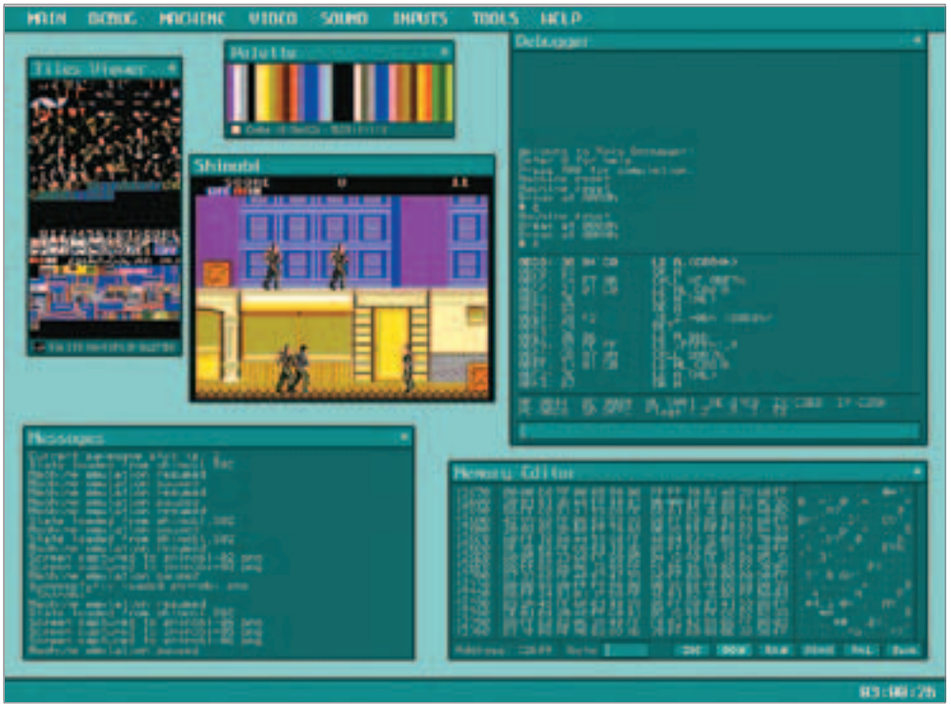
«НА 100% ИГРУ НЕ ЗАЩИТИТЬ, НО МОЖНО ОСЛОЖНИТЬ ЕЕ ВЗЛОМ, ОТБИВ ТЕМ САМЫМ ЖЕЛАНИЕ ЕЕ ВЗЛАМЫВАТЬ В ПРИНЦИПЕ!»

Обычно о защите вспоминают в последний момент, перед самой сдачей проекта. Дописывают код впопыхах, а потом удивляются, почему взломанные версии доходят до рынка раньше официальных. Защита должна быть глубоко интегрирована в программу и должна разрабатываться заранее. То же самое скажет любой эксперт. Однако если к защите подходят вот таким, должным образом, к багам самой программы добавляются глюки защиты и проект рискует не уложиться в срок. Кроме того, неясно, как отлаживать программу, если она доверху нашпигована антиотладочными приемами и активно сопротивляется отладчику.

Лучше всего реализовать защитный механизм в виде модулей, готовых к интеграции в программу в любой момент. Рабочая версия вместо защитных функций вызывает процедуры-пустышки, заменяемые в финальной версии реальным кодом. О том, как проектировать эти модули, и поговорим.

арсенал защиты Вот три кита, на которых держатся защитные механизмы: машинный код, шифровка и p-код. Электронные ключи и прочие экзотические технологии этого типа здесь не рассматриваются. Массовая продукция именитых фирм давно взломана, а разработка собственного ключа с нуля — занятие не для слабонервных, к тому же он будет выгодным только при серийном производстве, иначе защита даже не окупится.

Аппаратная защита — совсем другой разговор. При желании в микрочип можно перенести программу даже целиком, и тогда никто не сможет скопировать (если, конечно, выбрать правильный чип). Однако процесс разработки и отладки усложняется в десятки и даже сотни раз, «железное обеспечение» получается крайне негибким, неудобным в тиражировании и распространении, не говоря уже о невозможности исправить обнаруженные ошибки. Даже если чип имеет перепрошиваемое ПЗУ, разрешение на запись равносильно разрешению на чтение, так как самое ценное



Любая игра — это код!

в железе — это прошивка. Скопировать железо намного проще, чем выдрать из защищенного чипа прошивку, если же она будет свободно распространяться как обновление... Можно, конечно, распространять не всю прошивку, а только измененные модули или даже их части. Что сделает хакер? Он просто создаст слегка модифицированный модуль, считывающий содержимое ПЗУ и выводящий его наружу контрабандным путем.

Защиты, о которых будем говорить, не требуют никакого дополнительного железа, они системно независимы, работают на прикладном уровне и не требуют прав администратора. Никаких драйверов! Никакого ассемблера! Возможно ли эффективно противостоять опытным хакерам в таких условиях? Возможно! Большинство защит ломаются из-за досадных ошибок и мелких конструктивных просчетов, допущенных разработчиком, который никогда не заглядывал в дизассемблерный листинг и не пытался взломать свое творение.

тайники машинного кода

Сокрытие алгоритма в машинном коде — широко распространенный, но неэффективный защитный прием. Понятно, что дизассемблерный листинг — это не исходный код и одна строка на языке высокого уровня может транслироваться в десятки машинных команд, часто перемешанных. К тому же в двоичном файле нет комментариев, зато все остальное очень даже встречается: структура классов, имена функций/переменных...

текстовые строки

ASCII- и UNICODE-строки несут в себе очень богатую информацию: и текстовые сообщения, выводимые на экран при регистрации/окончании trial-срока/неправильном вводе пароля, и ветви реестра, и имена ключевых файлов, иногда и сами серийные номера/пароли. С паролями все ясно. Хранить их в открытом виде нельзя, нужно хэшировать их. А что плохого в текстовых сообщениях? Обнаружив их в теле программы, хакер по перекрестным ссылкам очень быстро найдет тот код, выводящий их, — вот что плохо. То же самое относится и к именам ключевых файлов с ветвями реестра.

текстовая строка «wrong s/n» с перекрестной ссылкой, ведущей к процедуре sub_401000

```
.text:00401016      call      sub_401000
.text:0040101B      add       esp, 4
.text:0040101E      test      eax, eax
.text:00401020      jz        short loc_40102F
.text:00401022      push      offset aWrongSN ;
«wrong s/n\n» ; указатель на строку
.text:00401027      call      _printf
...
.data:00406030 aWrongSN  db 'wrong
s/n',0Ah,0 ; DATA XREF: 00401022h^o
```

Чтобы затруднить анализ, необходимо либо зашифровать все строки, расшифровывая только перед непосредственным употреблением (если расшифровать сразу же после запуска программы, хакер просто снимет дамп и увидит их в виде прямого текста), либо поместить их в ресурсы и грузить через LoadString — это программируется легче, но в то же время легче ломается. Хакеру достаточно открыть файл в любом редакторе ресурсов, найти нужную строку, запомнить ее идентификатор, запустить отладчик и установить условную точку останова. Далее — дождаться вызова LoadString с нужным uID, определить адрес буфера lpBuffer, установить на него точку останова и... если lpBuffer выводится не сразу, а передается через цепочку промежуточных буферов, хакер матерится, но ничего не делает, потому что не может сделать ничего. Аппаратных точек останова всего четыре, и если защита использует не последний буфер в цепочке, то отследить момент реального обращения к строке становится невозможно (на самом деле возможно: с помощью секретного хакерского оружия NO_ACCESS на страницу, только об этом не все знают).

В дизассемблере же отследить перемещение строки по локальным буферам практически невозможно (во всяком случае, автоматически). Глобальные буфера (в которые компилятор пихает константные строки, в том числе зашифрованные) отслеживаются сразу, так что LoadString по некоторым позициям очень даже рулит!

отладочная информация

По умолчанию компилятор генерирует файл без отладочной информации, и она попадет туда только в исключительном случае, но все-таки попадет. Такая участь настигает не только начинающих программистов, не знающих, чем отличается Debug от Release, но и «маститые» фирмы, выпускающие достойные программные продукты. Допустим, у нас имеется глобальная программа IsRegistered, тогда смысл пары машинных команд будет ясен и без комментариев.

дизассемблерный листинг исполняемого файла с отладочной информацией

```
.text:00405664      cmp       _IsRegistered, 0
.text:0040566B      jz        short loc_40567A
```

динамические библиотеки

Имена неэкспортируемых функций уничтожаются компилятором, экспортируемые же по умолчанию остаются «как есть». C++-компиляторы в дополнение «замангляют» имена, дописывая к ним «зашифрованный» прототип функции, но IDA PRO с легкостью возвращает их в исходный вид. Если защитный модуль реализуется в виде динамической библиотеки (очень часто случается именно так), наличие символьных имен (причем с готовыми прототипами) значительно упрощает анализ. Например, OO Software (создатель одноименного дефрагментатора) любит таскать за собой библиотеку oorwiz.dll (очевидно, расшифровывается как «OO Registration Wizard»), экспортирующую всего три функции, но зато какие...

библиотека oorwiz.dll от OO Software экспортирует функции, говорящие сами за себя

```
3 0 00001FD0 RegWiz_InitLicMgr
1 1 000019D0 RegWiz_InitReadOnly
2 2 00001D00 RegWiz_InitTrial
```

rtti

Динамические классы, тесно связанные с механизмом RTTI (Runtime Type Identification) и активно используемые компиляторами DELPHI/Borland C++ Builder, сохраняют в откомпилированном файле не только свою структуру, но и символьные имена! Если брать как пример результат работы утилиты DEDE, реконструировавшей структуру классов программы Etlin HTTP Proxy Server, сразу в глаза бросится класс TfrmRegister, который соответствует форме fRegister и обрабатывает нажатие кнопки «OK» процедурой bOKClick, расположенной по адресу 48D2DCh. Благодаря динамическим классам сердце защитного механизма было локализовано всего за несколько секунд!

обфускация Код, генерируемый компилятором, очень громоздок, и разобраться в нем крайне непросто, но возможно. Чтобы помешать злоумышленникам, некоторые протекторы используют «запутывание», или обфускацию (англ. obfuscation). В простейшем случае автоматический кодогенератор, свинченный с полиморфного движка, внедряет в код огромное количество незначащих команд типа NOP, XCHG EAX, EBX/XHG EBX, EAX, нашпиговывая ими программу как рождественскую утку/гуся. Более совершенные генераторы используют разветвленную систему условных переходов, математические операции и присвоения, результат которых никак не используется, и другие полиморфные технологии.

Можно сгенерировать хоть миллион команд, это легко. Проанализировать их намного сложнее, если вообще возможно. Назначающие команды и заведомо никогда не исполняющиеся условные переходы типа XOR EAX, EAX/JNZ trg сможет отсеять и компьютер (достаточно написать простенький плагин к дизассемблеру IDA PRO). Освободиться от ненужных вычислений значительно сложнее. Как минимум, необходимо загнать все команды на граф, отображающий зависимости по данным, и убрать замыкающиеся ветви. Некоторые хакерские команды уже решили эту задачу (например, группа Володи с wasm'a), однако готовых инструментов в публичном доступе что-то не наблюдается, значит, юные взломщики, наткнувшись на обфускаторный код, скорее обломаются, чем взломают его. С точки зрения разработчика программы очень хорошо!

Высаживаться на разработку собственного обфускатора совершенно необязательно, есть готовые — как коммерческие, так и бесплатные. Например, .NET Obfuscator — **http://blogs.msdn.com/obfuscator/default.aspx**. Забавно, но большинство обфускаторов не используют обфускацию для защиты самих себя от взлома! А все потому, что в программах, критичных к производительности (к ним, например, относятся трехмерные игры), обфускация вызывает значительные тормоза и запутывать можно только редко вызываемые модули, например код защитного механизма. Однако здесь возникает угроза: хакер просто «выломает» защитный механизм из программы не анализируя его устройство. Как правило, для этого достаточно проанализировать код материнской процедуры (которая не подвергалась обфускации) и удалить вызов «запутанной» защитной функции, подсунув «правильный» код возврата, который ожидает вызывающая функция. Чтобы помешать подобным действиям хакера, защитная процедура, кроме проверки аутентичности копии программы, должна делать что-то полезное, такое, без чего программа не сможет работать. Но и в этом случае шансы хакера на взлом остаются высокими: шпионаж за API-функциями и реестром дает богатую пищу для размышлений, которая часто избавляет от необходимости анализировать машинный код.

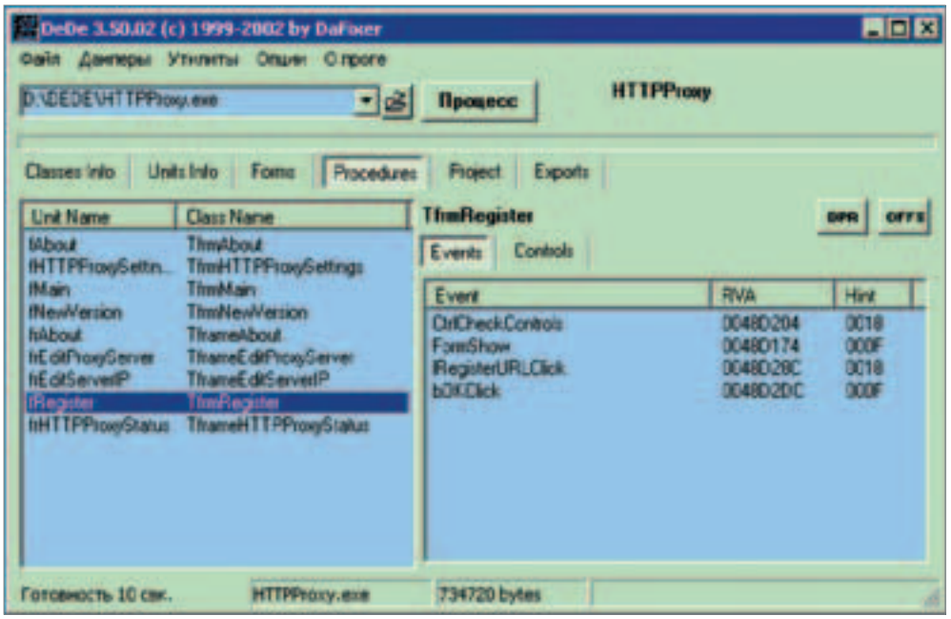
фрагмент программы, защищенной протектором armadillo

```
.00434000: 60                pushad
.00434001: E800000000        call    .000434006 -------- (1)
.00434006: 5D                pop     ebp
.00434007: 50                push    eax
.00434008: 51                push    ecx
.00434009: EB0F              jmps    .00043401A -------- (2)
.0043400B: B9EB0FB8EB        mov     ecx,0EBB80FEB ;«e?0e»
.00434010: 07                pop     es
.00434011: B9EB0F90EB        mov     ecx,0EB900FEB ;«e?0e»
.00434016: 08FD              or      ch,bh
.00434018: EB0B              jmps    .000434025 -------- (3)
.0043401A: F2                        repne
.0043401B: EBF5              jmps    .000434012 -------- (4)
.0043401D: EBF6              jmps    .000434015 -------- (5)
.0043401F: F2                        repne
.00434020: EB08              jmps    .00043402A -------- (6)
.00434022: FD                        std
.00434023: EBE9              jmps    .00043400E -------- (7)
.00434025: F3                        repe
.00434026: EBE4              jmps    .00043400C -------- (8)
.00434028: FC                        cld
.00434029: E959585051        jmp     051533887
.0043402E: EB0F              jmps    .00043403F -------- (9)
```

шифровка Это мощное оружие против взлома, оно бьет точно в цель и высаживает хакера на конкретный геморрой. Есть два вида шифровки: статическая и динамическая.

При статической зашифрованный код/данные расшифровываются один-единственный раз, на самой ранней стадии инициализации программы, после чего расшифрованному коду передается управление. Это программируется просто и ломается еще проще: дождавшись завершения расшифровки, хакер снимает с программы дамп и дизассемблирует его.

Динамические шифровщики расшифровывают код по мере возникновения необходимости в нем и, когда управление будет возвращено, тут же зашифровывают его вновь. Чем меньшие порции кода (данных) расшифровываются за раз, тем лучше (если извернуться, можно расшифровывать по одной машинной команде или байту данных). Таким образом, при динамической шифровке в каждый момент времени в памяти присутствуют только крохотные куски расшифрованного кода/данных и снятие дампа дает немного пользы.



Утилита DEDE восстанавливает RTTI-информацию из программы Etlin HTTP Proxy (видны надписи: fRegister, CtrlCheckControls, FormShow, IRegisterURLClick, bOKClick и др.)

Широкому внедрению динамической шифровки препятствуют следующие проблемы. Во-первых, сложность разработки и трудоемкость отладки. Во-вторых, производительность, точнее, полное отсутствие таковой (что очень критично в играх). В-третьих, возможность снять дамп руками самого расшифровщика, которому последовательно передаются адреса всех зашифрованных регионов. Есть другой способ, при котором легким битхаком его тело исправляется таким образом, чтобы он только расшифровывал, но ничего не зашифровывал. Конечно, способы сопротивления существуют. Например, используют перекрывающие шифроблоки, которые могут быть расшифрованы только по очереди, но не все сразу, или многослойную шифровку типа «луковицы», при которой один шифровщик плюс немного полезного кода вложен в другой, а тот в третий и т.д. Шифровщики как бы перемешаны с кодом, и «отодрать» чистый дамп невозможно. Очень надежно, но реализуется очень и очень сложно.

Бесспорных лидеров среди алгоритмов шифровки нет. Выбор предпочтительного метода защиты не столь однозначен, и статическая шифровка при всех своих недостатках продолжает удерживать прочные позиции, она не собирается сдаваться.

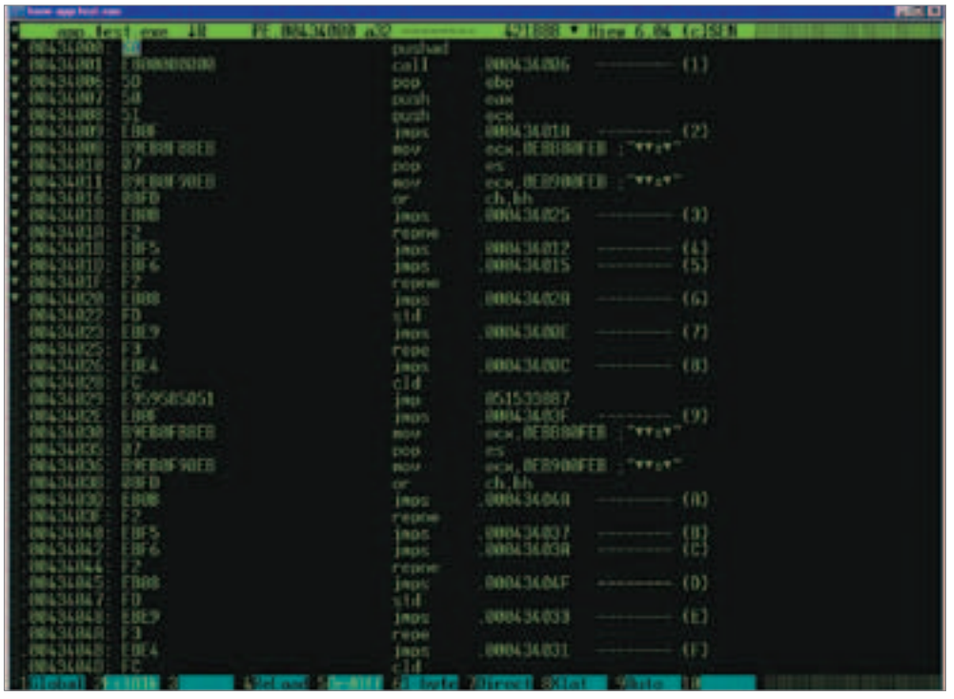
p-код Термин p-код восходит к интерпретатору Visual Basic'a, но в хакерских кругах означает нечто большее и подразумевает любой интерпретируемый код произвольной виртуальной машины (не путать с VMWare). Непосредственное дизассемблирование в этом случае становится невозможно, и приходится прибегать к декомпиляции, для чего необходимо проанализировать алгоритм работы интерпретатора, написать (и отладить!) декомпилятор, что требует пива и времени. Правда, разработка (и отладка) интерпретатора тоже не обходится даром. Плюс разработка языка тянет за собой кучу вспомогательного инструментария (в первую очередь понадобится отладчик). Иначе как на нем программировать?..

Все это выливается в солидный проект, который может быть использован всего один раз, в одной-единственной программе, причем желательно слегка изменять ядро интерпретатора после выхода нескольких версий, чтобы написанный хакером декомпилятор перестал работать. Что поделаешь, защита требует жертв и больших вложений. Плюс производительность интерпретируемого кода плетется в самом хвосте, отставая от динамической расшифровки и обфускации, но... Может быть, есть возможность реализовать на p-коде только защитные модули? Нет! Тогда их отломают! Не глядя! На p-коде должна быть реализована вся программа целиком, в том числе защитный механизм, тогда без декомпилятора его будет не хакнуть. Но это все теория. На практике полностью загнать программу в p-код не удается из-за производительности и критичные к быстродействию функции пишутся на языке высокого уровня или даже ассемблере. Зато вызываются они уже из p-кода, в котором сосредоточена основная логика по типу «если нельзя, то все-таки».

Кстати, большинство игровых миров управляются своим собственным скриптовым языком, который описывает движение облаков, поведение мячика и характер разных монстров. На чистом C не запрограммируешь никакую игру сложнее «тетриса»! Если мы уже имеем скриптовый язык, то почему бы не включить в него несколько защитных функций? И в этом случае, чтобы взломать программу, хакеру придется разобраться в работе скриптового движка.

Чаще всего разработчики используют Pascal- или Basic-подобные языки — не самый лучший выбор в плане защищенности. Программа на p-коде при этом представляет собой последовательность ключевых слов (операторов языка) с аргументами и декомпилируется очень просто. На другом конце шкалы сложности находится Машина Тьюринга, Сети Петри, Стрелка Пирса и прочие примитивные виртуальные машины, которые реализуют фундаментальные логические операции, в результате чего даже такая конструкция, как «IF THEN ELSE», распадается на сотни микрокоманд! Прежде чем хакер проанализирует их, солнце успеет погаснуть, или… Может, все-таки не погаснет? Суще-

«НИКОГДА НЕ ОСТАВЛЯЙ ОТЛАДОЧНУЮ ИНФОРМАЦИЮ В ОТКОМПИЛИРОВАННОЙ ПРОГРАММЕ!»



Фрагмент программы, защищенной протектором armadillo

«НЕ ИСПОЛЬЗУЙ RTTI В ЗАЩИТНЫХ МЕХАНИЗМАХ ИЛИ НЕ ДАВАЙ ФОРМАМ И ОБРАБОТЧИКАМ “ОСМЫСЛЕННЫЕ” ИМЕНА!»

ствует множество продвинутых способов наглядной визуализации таких алгоритмов, к тому же мы опять забываем о производительности... Реализовать crackme на базе Машины Тьюринга еще можно, но коммерческое приложение — едва ли.

Хорошая идея — приложить к этому делу Форт. Его преимущества: простота реализации Форт-машины, компактность p-кода, довольно высокая производительность, сложность декомпиляции и ни-на-что-непохожесть. Форт стоит особняком от всех языков, совсем не стремясь соответствовать человеческим привычкам, «здравому смыслу» и «логике». Разобраться в его идеологии с нуля непросто. Хакеру придется нарыть кучу литературы и запастись терпением. Даже если ему не наскучит, разработчик защиты получит хорошую фору по времени...

дизассемблерный листинг Форт-машины (из программы see.exe, поставляемой вместе с Interrupt List'ом)

```
seg000:1F29 loc_1F29:
; CODE XREF: start+39B9vj
seg000:1F29     call    sub_1F04
seg000:1F2C     mov     word_A5C, bx
seg000:1F30     mov     si, dx
seg000:1F32     mov     bp, [bx+38h]
seg000:1F35     mov     sp, [bx+36h]
seg000:1F38     lodsw
seg000:1F39     mov     bx, ax
seg000:1F3B     jmp     word ptr [bx]
```

Другая хорошая идея — использовать готовые интерпретаторы малораспространенных языков (Пролог, Хаскель, Оберон), под которые еще не написаны декомпиляторы. Многие из них разработаны в недрах исследовательских институтов и поставляются в исходных текстах, что упрощает модификацию ядра интерпретатора и позволяет легко адаптировать чужой язык к своим целям.

Круче всего — нарыть в Сети эмулятор малоизвестной ЭВМ, под которую есть компилятор с языка высокого уровня (C, ФОРТАН), и... Хакеру придется очень худо, придется осваивать неизвестный ему машинный язык, искать дизассемблер (или, скорее, писать собственный), прилагая титанические усилия для взлома. Разработчик тем временем будет пить пиво и кодить в привычной для него среде. Самое замечательное — то, что в следующей версии программы можно использовать эмулятор от «другой» ЭВМ, перекомпилировав весь код без адаптации и каких-либо изменений. Точнее, совсем без изменений, конечно, не получится, но трудоемкость переноса не сравнить с тем объемом работы, который предстоит проделать взломщику!

секреты привязки Рассуждая о методах защиты, мы не касались вопроса привязки, то есть такой характеристики, которая бы отличала одну копию программы от другой. Самые надежные защиты основаны на привязке к носителю. Они же — самые капризные и глюкавые.

Практически нереально создать защиту, которая безошибочно распознавалась бы любым приводом (включая что-то сильно раздолбанное или слишком нестандартное), но в то же время не копировалась бы ни одним известным копировщиком. Лучший результат дает привязка к геометрии спиральной дорожки (защиты типа CD-COPS, Star-Force) — ее невозможно скопировать, но легко проэмулировать или… подобрать диск с похожими характеристиками. Для предотвращения эмуляции разработчикам Star-Force пришлось очень глубоко зарыться в систему, в результате чего получился полный глюкодром, а установка очередного пакета обновлений на Windows требует параллельной установки соответствующего обновления для Star-Force. Так что лучше похоронить эту тему раз и навсегда и даже не пытаться возвращаться к ней. Иначе столкнешься с таким количеством проблем, осилить которое сможет только крупная компания, да и то…

серийные номера Самое простое — защитить программу серийным номером, который элементарно программируется и не вызывает никаких конфликтов ни с чем. Конечно, это не помешает Пете скопировать понравившуюся ему программу у Васи или даже выложить серийный номер в Сеть напоказ всем желающим. Соответствующий способ борьбы — использовать черный список «засвеченных» серийных номеров, проверяя их на соответствие при установке обновлений. Если защищенная программа предполагает взаимодействие с удаленным сервером, процедура проверки валидности серийных номеров значительно упрощается и нелегальный пользователь просто не получит доступ к серверу (очень актуально для сетевых игр!).

Несетевые программы также могут периодически ломиться в Сеть, передавая свой серийный номер. Если этот номер не подтверждается, регистрация считается недействительной и работа программы прекращается. При этом возникают следующие проблемы: поддержка выделенного и постоянно работающего сервера стоит денег, к тому же такой сервер сам по себе представляет весьма нехилую мишень для атаки. Что если хакеры завесят его или, еще хуже, проникнут внутрь и украдут легальные номера? Значит, необходимо содержать толкового администратора и разрабатывать протокол проверки серийных номеров с учетом возможного перехвата, например использовать несимметричную криптографию — тогда

украденная база не позволит восстановить ни один легальный серийный номер. Впрочем, что-то я все о мелочах да о мелочах.

Как быть с персональными брандмауэрами, популярность которых постоянно растет? Пользователь не выпустит программу в Сеть! Хотя любой брандмауэр легко обойти установкой собственного драйвера, так поступать нельзя. Журналисты тут же поднимут шумиху и обольют программиста грязью. Требовать обязательного наличия интернета нельзя. До сих пор он есть не у всех (если все-таки требовать, необходимо как минимум уметь работать через прокси). Правда, можно прибегнуть к одному очень хитрому маневру: через функцию InternetGetConnectedState определить, доступен ли в настоящее время интернет, и если сетевое соединение действительно присутствует, потребовать от пользователя открыть брандмауэр (или что там у него есть). Популярная программа Macro Express поступает именно так, но и этот путь не свободен от проблем. Если выход осуществляется через локальную сеть, функция InternetGetConnectedState не сможет сказать, имеется ли выход оттуда в интернет. С удаленным доступом по модему намного проще, однако не стоит забывать, что пользователь может подключаться не только к провайдеру, но и к своему приятелю, у которого стоит импровизированный «сервер», не предоставляющий доступа в интернет... Однако InternetGetConnectedState этого не объяснишь! Наконец, системную библиотеку WININET.DLL, которая экспортирует функцию InternetGetConnectedState, очень просто хакнуть в контексте памяти ломаемого приложения, и тогда защите настанут кранты.

криптография Вместо серийных номеров некоторые разработчики предпочитают использовать криптографические ключи и прочие системы цифровой подписи типа сертификатов, усиленно продвигаемые на рынок компанией Microsoft и прочими гигантами. Все чушь собачья! Цифровая подпись хорошо работает только в связке «клиент-сервер», но на локальной машине она бессмысленна. Да, сгенерировать «левый» ключ в этом случае невозможно, но если как следует пошурудить hiew'ом в исполняемом файле, никакие ключи уже не понадобятся. Так что цифровая подпись должна использоваться совместно с шифровкой кода и проверкой целостности!

Можно (и нужно) шифровать ключевым файлом критические модули защищаемой программы, но главное тут — не перестараться! Независимо от алгоритма реализации, такая защита вскрывается при наличии одного-единственного ключа. Кроме того, она не защищена от распространения ключа.

оборудование физическое и виртуальное Привязка к аппаратуре — это грязный и пакостный трюк, отравляющий жизнь легальным пользователям и все-таки не спасающий от взлома, поскольку любая привязка обнаруживается опытным хакером элементарно. Либо эмулируется, либо вырезается. Виртуальные машины типа VM Ware представляют собой большую проблему, на варезных серверах уже появились программы с примечанием «Запускать под VM Ware». Достаточно зарегистрировать программу один раз, и дальше можно тиражировать ее сколько угодно.

Распознать виртуальные машины довольно легко, они несут на своем горбу довольно специфичный набор оборудования. Однако уже появились патчи, которые скрывают присутствие VM Ware, и, что хуже всего, многие легальные пользователи предпочитают запускать второстепенные утилиты из-под виртуальных машин, чтобы не засирать основную систему. Защита не может (не имеет ни морального, ни юридического права!) отказывать VM Ware в исполнении, иначе пользователи снесут защиту к чертям, мотивировав это «производственной необходимостью». Они будут правы (правда, так они не освобождаются от регистрации). Некоторые даже подадут в суд за умышленное ограничение функциональности, не отраженное на упаковке. Перейдем к практике...

Лучше всего привязываться к жесткому диску. Как показывает практика, он меняется реже всего. Чтобы прочесть серийный номер, совершенно не обязательно писать собственный драйвер, достаточно воспользоваться библиотекой ASPI (но она не установлена по умолчанию) или SPTI (есть только в NT/XP и требует прав администратора). Есть и вот такой трюк. Вызываешь GetVolumeInformation и смотришь на размер тома в байтах. Не слишком уникальная информация, но на «другой» машине с вероятностью, близкой к единице, она будет иной. Плюс такая проверка реализуется очень просто, работает стабильно на всем семействе Windows-подобных систем (в том числе эмуляторы) и не требует никаких прав.

> «ОБФУСКАЦИЯ — НЕ ПАНАЦЕЯ. ИСПОЛЬЗОВАНИЕ ГОТОВЫХ ОБФУСКАТОРОВ ЛИШЬ УВЕЛИЧИВАЕТ ОБЪЕМ ЗАЩИЩАЕМОЙ ПРОГРАММЫ И УХУДШАЕТ ПРОИЗВОДИТЕЛЬНОСТЬ, НО ДАЛЕКО НЕ ВСЕГДА ЗАТРУДНЯЕТ ВЗЛОМ!»

компиляция on demand При наличии ресурсов можно установить сервер, генерирующий программы индивидуально для каждого пользователя, специально под его регистрационные данные. Пользователь скачивает крошечный инсталлятор. Инсталлятор извлекает с компьютера ключевую информацию, необходимую для привязки (например размер тома), спрашивает имя пользователя и передает эту информацию серверу, который жестко прописывает все это «хозяйство» в исходном тексте, перекомпилирует его, зашифровывает любым понравившимся навесным упаковщиком и отсылает назад. Что выигрывает разработчик?

Во-первых, «отломать» защиту становится намного сложнее, поскольку она не спрашивает никаких серийных номеров, не требует ключевых файлов и хакеру просто не за что зацепиться. Во-вторых, даже если программа будет взломана, придется распространять ее только в исполняемом файле (в который легко внедрить «водяные знаки», идентифицирующие владельца, например зашифрованный MAC-адрес его сетевой карты). К исполняемым файлам, добытым противоестественным путем (то есть скачанным из ненадежных источников), народ испытывает традиционное недоверие, и далеко не каждый рискнет запускать их.

как затруднить распаковку Откомпилированная программа обычно подвергается упаковке. Цель здесь совсем не в уменьшении размера, а в затруднении анализа. Упаковщик набивает программу антиотладочными приемами и прочей бодягой, которая затрудняет пошаговую трассировку или даже делает ее невозможной. На самом деле хакер и не собирается ничего трассировать, а только снимает с работающего приложения дамп и дизассемблирует его (реконструировать exe-

файл для этого необязательно). Надежно противостоять снятию дампа на прикладном уровне невозможно, а спускаться на уровень драйверов как-то не хочется. Некоторые защиты искажают PE-заголовок, гробят таблицу импорта и используют другие грязные трюки, с помощью которых затрудняют дампинг, но не предотвращают его в принципе.

глобальные инициализированные переменные

Предлагаю альтернативный путь — не препятствовать снятию дампа, а сделать полученный образ бесполезным, для чего достаточно использовать глобальные инициализированные переменные, «перебивая» их новыми значениями.

защитный механизм, предотвращающий снятие дампа с работающей программы путем использования инициализированных глобальных переменных

```
char *p = 0;
// глобальная переменная 1
DWORD my_icon = MY_ICON_ID;
// глобальная переменная 2
...
if (!p) p = (char*) malloc(MEM_SIZE);
my_icon = (DWORD) LoadIcon (hInstance, my_icon);
```

Задумайся, что произойдет, если сбросить дамп с работающей программы. В переменной p окажется указатель на когда-то выделенный блок памяти, условие (!p) обломится и новая память не будет (!) выделена, а при обращении по старому указателю произойдет исключение. Другими словами, хакер уже не сможет изготовить исполняемый файл из дампа! Как минимум, придется восстановить значения всех глобальных переменных — геморрой :(. Ладно, изготовить исполняемый файл из дампа нельзя, но, может, дизассемблировать его? А вот и нельзя...

После выполнения функции LoadIcon переменная my_icon будет содержать не идентификатор иконки, а ее обработчик. Хакер не сможет установить, что это за иконка (строка, битмап или другой ресурс), и ему придется обращаться к отладчику, противостоять которому намного проще, чем дизассемблеру. Кстати, такой прием экономит память и широко используется во многих программах. Например, в стандартном «Блокноте» — попробуй снять с него дамп и обломайся :).

стартовый код

Единственная надежда хакера — отловить момент завершения распаковки и тут же сбросить дамп, пока защита не успела нагадить в глобальные переменные. Пошаговая трассировка исключается (противостоять ей очень легко), и остается... стартовый код, который варьируется от компилятора к компилятору.

типичный представитель стартового кода (в случае с Microsoft Visual C++ MFC)

```
.text:00402A82  push   ebp
.text:00402A83  mov    ebp, esp
.text:00402A85  push   0FFFFFFFFh
.text:00402A87  push   offset unk_403748
.text:00402A8C  push   offset loc_402C06
.text:00402A91  mov    eax, large fs:0
.text:00402A97  push   eax
.text:00402A98  mov    large fs:0, esp
.text:00402A9F  sub    esp, 68h
...
.text:00402BAA  call   ds:GetModuleHandleA
.text:00402BB0  push   eax
.text:00402BB1  call   _WinMain@16
; WinMain(x,x,x,x)
```

точка останова на GetModuleHandleA

Вызов API-функции GetModuleHandleA сразу же бросается в глаза. Если хакер установит сюда точку останова, отладчик/дампер «всплывет» в start-up-коде еще до передачи управления WinMain (также можно поставить точки останова на GetVesion/GetVersionEx, GetCommandLine, GetStartupInfo и т.д.). Если точка останова программная, распаковщик может обнаружить ее по наличию CCh в начале API-функции и, с некоторой долей риска, снять ее. Если второй байт функции равен 8Bh, то перед нами, очевидно, предстает стандартный пролог, первый (оригинальный) байт которого равен 55h. Получаешь права на запись через VirtualAlloc, меняешь CCh на 8Bh и продолжаешь распаковку в обычном режиме. Пусть хакер крякнет! Правда, в последующих версиях Windows пролог API-функций может быть модифицирован, и тогда этот трюк не сработает.

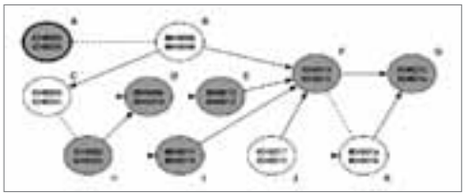
отладочные регистры

Аппаратную точку останова можно обнаружить чтением регистров Drx. Команда mov eax,DrX на прикладном уровне приводит к исключению, кроме того, отладчик (теоретически) может отслеживать обращение к отладочным регистрам, чтобы маскировать свое присутствие, — x86 процессоры предоставляют все необходимое для этого. Но если распаковщик прочитает свой контекст, он сможет дотянуться и до Drx, причем не только на чтение, но и на запись! Получается, что можно не только обнаружить точки останова, но и обезвредить их. Весь вопрос в том, как получить контекст. Чтение SDK выявляет API-функцию GetThreadContext, которая как раз для этого и предназначена, однако пользоваться ей нельзя, иначе хакер установит сюда точку останова и защита проиграет войну.

Нужно действовать так. Регистрируешь из распаковщика собственный обработчик SEH, возбуждаешь исключение (делишь на ноль, обращаешься по недействительному указателю) и получаешь контекст в одном из аргументов структурного обработчика. Остается установить точку останова на fs:0, где и хранится указатель на SEH-обработчик (но до этого додумается не каждый хакер).

структурные исключения

Кстати о fs:0. Первое, что делает стартовый код, — это регистрация собственного SEH-обработчика, поэтому установка точки останова на fs:0 позволяет хакеру всплыть сразу же после завершения распаковки, следовательно, распаковщик должен обращаться к этой ячейке как можно чаще. Десятки или даже тысячи раз, причем следует класть туда не что угодно, а именно ESP, иначе хакер установит условную точку останова (soft-ice это позволяет) и легко обойдет защиту.



Визуализация программы на графе

поиск по сигнатуре

Есть такой козырный хакерский трюк. Взломщик снимает дамп с работающей программы, находит там стартовый код по сигнатуре, определяет его адрес и ставит на его начало аппаратную точку останова (программную ставить нельзя, она будет затерта при распаковке). Разработчику защиты необходимо либо распознавать аппаратные точки останова и снимать их, действуя по методике, описанной выше, либо использовать модифицированный стартовый код, который не сможет распознать хакер.

контроль $pc

Еще один трюк. Большинство распаковщиков располагаются в стороне после распакованного кода и при передаче управления на оригинальную точку входа прыгают куда-то далеко. Хакер может использовать этот факт как сигнал о том, что распаковка уже завершена. Конечно, установить аппаратную точку останова на это условие уже не удастся, и придется прибегнуть к пошаговой трассировке (ей легко противостоять), но ради подобного случая хакер может написать и трейсер нулевого кольца. Несложно. Сложно определить, когда же заканчивается распаковка. Контроль на $pc (в терминологии x86 — «eip») — единственный универсальный способ, который позволяет сделать это без особых измен и, чтобы обломать хакера, распаковщик должен как бы «размазывать» себя вдоль программы. Тогда он победит!

боремся с отладчиком

Распространенный миф гласит, что штурмовать отладчик уровня soft-ice можно только из ядра, а надежно обуть его с приказного уровня невозможно. Неправда. В частности, эффективная отладка форт-программ (как и другого p-кода) под soft-ice невозможна. Отладчик просто вращается внутри форт-машины, наматывая мили на кардан, хакер матерится, выкуривает одну сигарету за другой, но не может предложить ничего конструктивного, кроме как написать декомпилятор, на что требуется время, которого никогда нет.

антиотладка

К слову, использовать антиотладочные приемы следует с большой осторожностью, лучше не использовать их вообще. То, что работает под 9x, часто не работает под NT, и наоборот. Обращение к недокументированным структурам операционной системы ставит программу в зависимость от левой пятки Microsoft, привязывая ее к текущей версии Windows. Кроме того, многие антиотладочные приемы, опубликованные в различных статьях, на проверку оказываются сплошной фикцией. Отладчик работает нормально и не ведется на них, зато глюки прилетают косяками.

Следует помнить и о существовании такой штуки, как IceExt — специальной примочки для soft-ice, которая скрывает его от многих защитных механизмов. Всегда проверяй все применяемые идеи на вшивость, и не только под soft-ice+IceExt, но и в альтернативных отладчиках типа OllyDbg. Совершенно недопустимо отказываться от работы в присутствии пассивного отладчика. Аргумент «soft-ice держат на компьютере только хакеры» идет лесом на хутор. Туда же посылаются и разработчики. Необходимо противодействовать лишь активной отладке. Нейтрализации точек останова обычно бывает вполне достаточно. Не нужно мешать отладчику, главное — сделать этот процесс максимально неэффективным.

вероятностное поведение программы

Пусть вызовы защитных функций следуют из разных мест с той или иной вероятностью, определяемой либо оператором типа rand(), либо действиями пользователя. Например, если сумма последних шести символов, набранных пользователем, равна 69h, происходит «внеплановый» вызов защитного кода. Если программа при каждом прогоне ведет себя слегка по-разному, реконструкция ее алгоритма чрезвычайно усложняется.

взлом по следам, взлом без следов

Основная ошибка большинства создателей защит заключается в том, что они дают хакеру понять, что защита распознала попытку взлома. Ни в коем случае не делать так! Пусть хакер сам догадывается, перелопачивая тонны машинного кода. Ах, если бы мы только могли не выводить диалоговое окно с надписью «неверный серийный номер/инвалидный ключ» — тогда хакеру остается только поставить точку останова и посмотреть на код, который выводит его, а защитный механизм сразу же будет пойман за хвост! Приходится идти в обход: вместо немедленного выполнения какого-либо действия программист создает список отложенных процедур (просто массив указателей на функции) и проверяет его в цикле выборки сообщений или во время простоя системы из отдельного потока. Один поток проверяет регистрационные данные и кладет сюда указатель на функцию, которую нужно осуществить вместе с другими функциями, выполняемыми программой. Цепь разомкнулась! Простая трассировка топит отладчик в цикле выборки сообщений, и хакеру приходится разбираться со всеми этими списками, очередями и т.д., что в отладчике сделать очень непросто. Для реконструкции алгоритма требуется помощь дизассемблера...

полезные советы россыпью

Проверяй серийный номер (пароль) не с первого байта (лучше всего с пятого), этим защита обломает начинающего хакера, который установил точ-

«ВСЕГДА УДАЛЯЙ СИМВОЛЬНЫЕ ИМЕНА ИЗ ЭКСПОРТА И ВЫЗЫВАЙ ФУНКЦИИ ТОЛЬКО ПО ОРДИНАЛУ»

WANTED:
WMF — БАГ ГОДА
15 ЛЕТ ZX-СЦЕНЫ
ВОЙНА МИРОВ:
EXT2 VS EXT3

Binary Your's

ку останова на начало. Также не проверяй последние символы серийного номера. И не проверяй середину! Проверяй не более половины символов вразброс. Смешно, конечно, но очень выручает. Никогда не считывай серийный номер (пароль, имя пользователя) из строки редактирования целиком! Хакер тут же найдет его в памяти и расставит точки останова, в которые угодит защитный механизм. Считывай только по одному вводимому символу за раз (через WM_CHAR или DDE) и тут же шифруй их (если скалывать считанные символы в локальный буфер, то получится тот же самый WM_GETTEXT, только реализованный своими руками).

боремся с дизассемблером

Оптимальный дизассемблер — IDA PRO. Действительно очень мощный дизассемблер, его не так-то просто взять на испуг. С классическими защитными приемами (типа прыжка в середину команды) он справляется даже не замечая, что тут было что-то защитное. Если на пути хакера встретится стена зашифрованного/упакованного кода, он напишет короткий (длинный) скрипт и расшифрует все, что нужно, даже не выходя из дизассемблера. Для p-кода будет написан отдельный плагин типа «докомпилятор».

Самым мощным антихакерским средством был и остается косвенный вызов функций по указателю, передаваемому в качестве аргумента. Хакер натыкается на что-то типа call eax и матерится в бессильной злобе, пытаясь определить, что содержится в eax на данный момент. Если материнская функция вызывается обычным способом, взломщику достаточно просто перейти по перекрестной ссылке, заботливо сгенерированной IDA, и подсмотреть, что передается функции. Однако если указатель инициализируется далеко от места вызова, хакеру придется раскрутить всю цепочку вызовов. Если же функция, вызывающая косвенную функцию, сама вызывается косвенным путем, то вообще наступают кранты! Остается запускать отладчик, устанавливать точку останова на call eax и смотреть ее значение вживую, однако точкам останова легко противостоять. К тому же в различные моменты времени eax может указывать на разные функции, тогда простое подглядывание eax ничего не даст. Дизассемблер ослепнет и оглохнет!

пример программы, вызывающей функции по указателю

```
sub_sub_demo(int a, void *p, void *d)
{
  // printf(«sub_sub_demo\n»);
  if (--a) return ((int(*)(int, void*, void*))p)(a, p, d);
  return 0;
}

sub_demo(int a, void *p, void *d)
{
  // printf(«sub_demo\n»);
  if (--a) return ((int(*)(int, void*, void*))d)(a, p, d);
  return 0;
}
demo(int a, void *p, void *d)
{
  // printf(«demo\n»);
  ((int(*)(int, void*, void *))p)(a, p, d);

}
main()
{
  demo(0x69,sub_demo, sub_sub_demo);
}
```

В исходном тесте все понятно. Функция main вызывает функцию demo, передавая ей указатели на sub_demo и sub_demo, которые поочередно вызывают друг друга, каждый раз уменьшая счетчик на единицу. Короче, мы имеем цикл. Но какой! Ты оцени его дизассемблерный код.

дизассемблерный листинг, демонстрирующий мощь косвенного вызова функций

```
.text:00401000 loc_401000:
; DATA XREF: _maino
.text:00401000          mov   ecx, [esp+4]
.text:00401004          dec   ecx
.text:00401005          jz    short loc_401018
.text:00401007          mov   eax, [esp+0Ch]
.text:0040100B          push  eax
.text:0040100C          mov   eax, [esp+0Ch]
.text:00401010          push  eax
.text:00401011          push  ecx
.text:00401012          call  eax
.text:00401014          add   esp, 0Ch
.text:00401017          retn
.text:00401020
.text:00401020 loc_401020:
; DATA XREF: _main+5o
.text:00401020          mov   ecx, [esp+4]
.text:00401024          dec   ecx
.text:00401025          jz    short loc_401038
.text:00401027          mov   eax, [esp+0Ch]
.text:0040102B          mov   edx, [esp+8]
.text:0040102F          push  eax
.text:00401030          push  edx
.text:00401031          push  ecx
.text:00401032          call  eax
.text:00401034          add   esp, 0Ch
.text:00401037          retn
.text:00401040
.text:00401060 _main    proc near
; CODE XREF: start+AF p
.text:00401060          push  offset loc_401000
.text:00401065          push  offset loc_401020
.text:0040106A          push  69h
.text:0040106C          call  sub_401040
.text:00401071          add   esp, 0Ch
.text:00401074          retn
.text:00401074 _main    endp
```

IDA Pro не смог распознать функции, хотя восстановил перекрестные ссылки на main — они еще ни о чем не говорят! Функции sub_demo и sub_sub_demo вызываются совсем не оттуда. Возьмем «функцию» loc_401000. Она принимает указатель в качестве аргумента и тут же вызывает его. А что это за указатель? Да хвост его знает! Перекрестной ссылки на материнскую функцию же нет. Единственный способ установить истину — проанализировать всю цепочку вызовов с самого начала программы, с функции main, но слишком трудоемко, так что даже не обсуждается (особенно если ломать реальную программу). Даже в таком случае приходится постоянно следить за указателями, которые пляшут как кони, причем, даже если развяжешь себе пупок, не сможешь понять, что находится в них в каждый конкретный момент...

Защита будет значительно усилена, если реализовать модель Маркова — функцию, возвращающую указатель на функцию. Такой прием программирования не слишком популярен, так как непривычен и лишен языковой поддержки. Язык C вообще не позволяет объявлять функции, возвращающие указатели на функции, поскольку такие определения рекурсивны и программисту приходится возиться с постоянным преобразованием типов, что не слишком украшает программу и является потенциальным рассадником ошибок. Тем не менее на автоматическое дизассемблирование моделей Маркова не способен ни один дизассемблер, в том числе IDA PRO.

боремся с мониторами

Рассмотрим пару способов борьбы с файловыми мониторами и мониторами реестра. Самое простое, что только можно придумать, — через FindWindow находить главное окно монитора и затем либо закрывать его к чертям, либо, посылая специальные Window-сообщения, удалять из списка «свои» обращения. Естественно, если хакер переименует окно (делается через FindWindow/SetWindowText), защита облома-ется по полной программе. Так что такой прием слишком ненадежен.

А что надежно? Хранить флаг регистрации вместе с настройками в одном ключе реестра в двоичном виде — вот тут его мониторинг ничего не даст. Хакер видит, что из такой ветви читается куча байтов, но ему неведомо, какой из них и за что отвечает. Выяснить можно только отладкой/дизассемблированием защищенной программы, сопротивляться чему несложно.

Мы рассмотрели множество эффективных и одновременно системно независимых защитных техник, работающих на прикладном уровне и не сильно усложняющих реализацию. Видно, что даже в таких жестких условиях можно сделать достаточно много!

РАЗРАБОТКА ИГР

Лучшая игра 2005 года (все данные **3dgamers.ru**):

1. F.E.A.R.
2. NEED FOR SPEED: MOST WANTED
3. GTA: SAN ANDREAS
4. КОРСАРЫ 3
5. CALL OF DUTY 2
6. PRINCE OF PERSIA 3: TWO THRONES
7. SPLINTER CELL: CHAOS THEORY
8. FAHRENHEIT
9. QUAKE 4
10. BATTLEFIELD 2 + SPECIAL FORCES

Action/FPS 2005 года:

1. F.E.A.R
2. SPLINTER CELL: CHAOS THEORY
3. CALL OF DUTY 2
4. GTA: SAN ANDREAS
5. QUAKE 4

Русская игра 2005 года:

1. КОРСАРЫ 3
2. В ТЫЛУ ВРАГА: ДИВЕРСАНТЫ
3. EX MACHINA
4. АЛЬФА: АНТИТЕРРОР + МУЖСКАЯ РАБОТА
5. КАЗАКИ 2. НАПОЛЕОНОВСКИЕ ВОЙНЫ

Стратегия 2005 года:

1. CIVILIZATION IV
2. AGE OF EMPIRES 3
3. BLACK & WHITE 2
4. UFO: AFTERSHOCK
5. КАЗАКИ 2. НАПОЛЕОНОВСКИЕ ВОЙНЫ

Adventure/Quest 2005 года:

1. FAHRENHEIT
2. ПЕТЬКА 6: НОВАЯ РЕАЛЬНОСТЬ
3. KING KONG
4. MYST V: END OF AGES
5. МОР. УТОПИЯ

RPG 2005 года:

1. FABLE: THE LOST CHAPTERS
2. КОРСАРЫ 3
3. GOTHIC 2: NIGHT OF THE RAVEN

Simulation 2005 года:

1. BLACK & WHITE 2
2. X3: REUNION
3. SILENT HUNTER 3
4. EX MACHINA
5. SIMS 2 NIGHTLIFE

Racing Sim 2005 года:

1. NEED FOR SPEED: MOST WANTED
2. JUICED
3. LADA RACING CLUB

По данным **3dgamers.ru**, спортивная игра 2005 года — Pro Evolution Soccer 5

MMORPG 2005 года:

1. GUILD WARS
2. ULTIMA ONLINE: MONDAIN'S LEGACY
3. СФЕРА: МИР ИЗБРАННЫХ

tetris

СТРОИМ ДВУХМЕРНОЕ СЧАСТЬЕ

КАКОЙ ЖЕ БУДЕТ НОМЕР ЖУРНАЛА ОБ ИГРОПАЯНИИ, ЕСЛИ В ПРОЦЕССЕ РАБОТЫ НАД НИМ НЕ УБИТО НИ ОДНОГО КРО… — ТЬФУ! — НЕ НАПИСАНО НИ ОДНОЙ ИГРЫ?! ПЛОХОЙ НОМЕР! ВЫКИНУТЬ ЕГО! МУСОР! ВОПЛИ ВОЗМУЩЕННЫХ РЕДАКТОРОВ ЖУРНАЛА БЫЛИ ПРЕРВАНЫ РЕПЛИКОЙ ОДНОГО АВТОРА (ТО ЕСТЬ МЕНЯ): «НЕ ВОЛНУЙТЕСЬ, ГОСПОДА, СЕЙЧАС ВСЕ СТАНЕТ КРУТО… ПРЕДСТАВЛЯЮ ВАШЕМУ ВНИМАНИЮ XTREAS…» ПО ПРАВДЕ ГОВОРЯ, ЭТО САМЫЙ ОБЫЧНЫЙ И БАНАЛЬНЫЙ ТЕТРИС, НО ОН РАЗРАБОТАН СПЕЦИАЛЬНО ДЛЯ СПЕЦА И ЕГО ЧИТАТЕЛЕЙ И ДОСТУПЕН ДЛЯ НИХ ПО ОСОБОЙ ЛИЦЕНЗИИ :) |

ПАЛАГИН АНТОН (TONY@EYKONTECH.COM)

инструменты Для работы нам понадобятся некоторые инструменты. Во-первых, это 2D-движок, который используется в тетрисе, — я использовал PopCap Framework (см. ссылки). Чтобы получить этот SDK, необходимо зарегистрироваться «у них» на форуме и скачать файлик весом около 7 Мб. На базе этого инструмента сделана целая куча офисных развлекалок наподобие Zuma и Dynomite. Кроме движка, мы не обойдемся без библиотеки Boost (**www.boost.org**). Откомпилируй PopCap SDK и настрой свою среду разработки на его заголовочные и экспортные файлы. Также не забудь указать путь к Boost'у. Как среду разработки я использовал MSVC++ 6.0. Кроме нее, ты спокойно можешь использовать VC 7 и 8, достаточно просто конвертировать проекты и откомпилировать новую библиотеку PopCap.

архитектура виджетов Прежде чем углубляться в дебри кода, посмотрим на архитектуру приложения PopCap.

Точка входа

```
//Создаем ядро игры
GameApp* anApp = new GameApp();
//Инициализируем ресурсы игры
anApp->Init();
//Запускаем игру
anApp->Start();
//Завершаем работу ядра игры
delete anApp;
```

В точке входа в программу создается объект класса GameApp. Этот класс производится от базового класса SexyAppBase — ядра любого приложения PopCap. Он хранит в себе все игровые сущности и менеджеры, с его помощью происходит управление игровыми событиями, такими как: начало игры, пауза, просмотр дополнительной информации, завершение работы программы. Игровое поле реализуется в классе Board, который является производным от класса Widget и агрегируется классом GameApp. Игровое поле разделяется на две части: левую панель (меню) и правую панель — стакан, куда падают фигуры. Класс Board настраивает положение главного меню и игровой области (зоны), следит за освобождением памяти при завершении работы и обрабатывает нажатия кнопок: перемещение фигур влево-вправо (кнопки <влево>/<вправо>), поворачивание (пробел) и ускорение (кнопка <вниз>). Главное меню игры реализуется в классе LeftPanel, который является производным от классов Widget и ButtonListener, то есть кроме графической нагрузки несет ответственность за обработку событий, генерируемых кнопками главного меню игры. Агрегируется классом Board.

Игровая зона также является виджетом, но не обрабатывает никаких кнопок — используется только для отображения тетрисных фигур. Игровая зона реализуется классом PlayArea и агрегируется классом Board. Еще один виджет в нашей игре — заставка, этот экран демонстрируется пользователю перед началом игры (во время загрузки игровых ресурсов). Здесь можно демонстрировать пользователю информацию об игре и требовать ввести ключ разблокировки, если ты

пишешь условно бесплатную игру. UML-диаграмму, описывающую архитектуру этих классов, смотри на рисунке «Архитектура виджетов».

архитектура игровой логики Логика тетриса реализуется в классе GameDriver. Он воспринимает управляющие события от виджетов: начало новой игры, пауза в игре, управление тетрисными фигурами. Также этот класс рисует на экране фигуры тетриса, удаляет заполненные строки в стакане, генерирует новые фигуры, следит за окончанием игры. Подсчет очков, набранных пользователем, и повышение уровня сложности реализует класс GameLogic, агрегируемый классом GameDriver. Отображение тетрисных блоков в заданных координатах игровой сетки берет на себя класс GameView, также агрегируемый классом GameDriver. Класс GameView «знает» то, как переводятся координаты на игровой сетке (поле) в экранные координаты. Поведением блоков на игровой сетке размером 10x20 управляет класс Playfield. Каждая фигура описывается классом TetrisBlock, который умеет хранить конфигурации фигур и вращать их. Соответствующую UML-диаграмму смотри на рисунке «Архитектура игровой логики».

инициализация игры

```
//Инициализируем движок игры
SexyAppBase::Init();
//Парсим манифест ресурсов
(сценарии ресурсов)
LoadResourceManifest();
//Загружаем секцию ресурсов «Init»
if( !mResourceManager->LoadRe-
sources(«Init») )
{
  ShowResourceError(true);
  return;
}
if( !ExtractInitResources(mResourceManager) )
{
ShowResourceError(true);
  return;
}
//Загружаем ресурсы, нужные
для игровой заставки
if( !mResourceManager->LoadResources
(«TitleScreen») )
{
  ShowResourceError(true);
  return;
}
if( !ExtractTitleScreenResources
(mResourceManager) )
{
  ShowResourceError(true);
  return;
}
//Инициализируем заставку
m_titleScreen.reset(new TitleScreen(this));
m_titleScreen->Resize(0, 0, mWidth, mHeight);
m_titleScreen->Init();
mWidgetManager->AddWidget
(m_titleScreen.get());
mNumLoadingThreadTasks = mResourceMana-
ger->GetNumResources(«Game»);
//Загружаем фоновую музыку
mMusicInterface->LoadMusic(0, «Musics/1984-
Network.s3m»);
mMusicInterface->FadeIn(0, 0, 0.002, false);
```

Первый шаг, который совершают после создания ядра игры, — инициализация ядра. Инициализируем движок игры, затем загружаем в память манифест ресурсов игры. Манифест — это XML-документ, в котором описаны наборы (секции) ресурсов: шрифты, изображения и звуки. На рисунке «Манифест ресурсов» как раз приведен пример из XTreas. В секции Init содержатся ресурсы, нужные в течение всего игрового цикла. В секции TitleScreen — ресурсы, необходимые на этапе демонстрации заставки игры. Наконец, в секции Game — игровые ресурсы.

Манифест определяет для каждого ресурса его символическое (строковое) имя в контексте менеджера ресурсов PopCap, и, чтобы обратиться к нужной картинке или звуку из игры, достаточно сказать менеджеру ресурсов: «Дай мне картинку A» или «Дай мне звук B». После того как пользователь нажмет на заставке кнопку, вызовется код изображенный на листинге «Начало игры». Пользователь удалит ресурсы заставки, ставшие ненужными, создаст игровое поле и запустит игровой цикл.

начало игры

```
//Удаляем ненужные ресурсы заставки
mResourceManager->DeleteResources
(«TitleScreen»);
//Инициализируем игровое поле
m_board.reset( new Board(this) );
mWidth = IMAGE_LEFT_PANEL->GetWidth() +
320 + BORDER_SIZE;
m_board->Resize(0, 0, mWidth, mHeight);
mWidgetManager->AddWidget( m_board.get() );
SetFocusToBoard();
```

события игры Игра управляется событиями, которые генерируются пользователем с помощью мыши (выбор кнопок в главном меню) и клавиатуры. Все игровое поле делится на две части: главное меню и «стакан», куда падают фигуры. Обе части игрового поля, как и само игровое поле, являются виджетами, то есть элементами интерфейса. Левая часть (меню) — тоже виджет, но, кроме того, она может принимать команды от нажатий кнопок главного меню. Все элементы интерфейса инициализируются в виртуальном методе AddedToManager, удаляются в методе RemovedFromManager.

инициализация виджетов

```
void LeftPanel::AddedToManager(WidgetMana-
ger* theWidgetManager)
{
Widget::AddedToManager(theWidgetManager);
//Рассчитываем размеры кнопок
int width = IMAGE_BUTTON_AQUA->GetWidth();
int height = mHeight — IMAGE_BUTTON_AQUA-
>GetHeight();
//Создаем кнопку
m_newGame = CreateButton(BUTTON_NEW_GA-
ME, «New Game», mWidth/2 — width/2, height);
//Добавляем кнопку к менеджеру виджетов
theWidgetManager->AddWidget
( m_newGame.get() );
}
```

Если пользователь нажимает мышкой на кнопку-виджет, то генерируется соответствующее событие, которое обрабатывается методом ButtonDepress.

обработка событий виджетов

```
void LeftPanel::ButtonDepress(int theId)
{
switch(theId)
{
case BUTTON_NEW_GAME:
m_app->OnNewGame();
break;
case BUTTON_PAUSE:
```

«ИСХОДНИКИ, БИНАРНИКИ И МНОГОЕ-МНОГОЕ ДРУГОЕ — НА НАШЕМ ДИСКЕ»

```
m_app->OnPauseGame();
break;
}
}
```

Эти события являются глобальными событиями игры. Например, начинается новая игра, игровой процесс ставится на паузу и т.д. Управление движением фигур в стакане осуществляется при помощи клавиатуры. Эти события обрабатываются в виртуальном методе KeyDown (листинг «Обработка кнопок клавиатуры») и являются локальными событиями, которые влияют на логику игры: фигура крутится и перемещается влево-вправо.

обработка кнопок клавиатуры

```
void Board::KeyDown(KeyCode theKey)
{
Widget::KeyDown(theKey);
switch(theKey)
{
case KEYCODE_SPACE:
m_app->m_gameDriver.Rotate();
break;
case KEYCODE_LEFT:
m_app->m_gameDriver.Left();
break;
case KEYCODE_RIGHT:
m_app->m_gameDriver.Right();
break;
case KEYCODE_DOWN:
m_app->m_gameDriver.Drop();
break;
}
}
```

игровой цикл С того момента, когда пользователь нажимает кнопку New Game, начинается непрерывный игровой цикл, на каждой итерации которого падает текущая фигура, на «заполненность» проверяются линии в стакане и условия окончания игры, прорисовываются фигуры всех фигур в стакане. Эти итерации происходят в методе Update() класса GameDriver.

игровая итерация

```
//Если игра закончена или еще не начата,
то итерация не выполняется
if(!m_isNewGame && !(m_gameState &
GAME_OVER) ) return;
m_gameView.SetGraphics( g );
//Если есть падающая фигура
if( m_current )
{
  //Если наступил момент изменения кадра
и игра не на паузе
if( elapsed > m_gameLogic.GetSpeed()
&& !m_isPaused )
{
//Необходимо или оставить блок на поле
или опустить его на ячейку вниз
if( m_playfield.CheckBlock
(*m_current, m_currentX, m_currentY+1) )
{
//Опускаем блок на ячейку вниз
m_playfield.InsertBlock(*m_current, m_cur-
rentX, m_currentY);
//Проверяем, не заполнился ли наш стакан
if( IsGameOver() )
{
GameOver();
secStart = secEnd;
return;
}
m_current = 0;
//Проверяем, заполнена ли полностью
линия(и) стакана
m_playfield.GetFullRows(bottom, count);
if(!count)
{
//Если нет ни одной заполненной линии
стакана, то опускаем в стакан новую фигуру
ThrowNextBlock();
//Проигрываем специфический звук
gSexyAppBase->PlaySample
(SOUND_BLOCK_DOWN);
}
else
{
  //Устанавливаем состояние
для последующей анимации процесса
  //удаления заполненной линии стакана
m_animState = CLEAN_ROWS;
//Проигрываем специфический звук
gSexyAppBase->PlaySample(SOUND_FULL_LINE);
}
}
else
{
++m_currentY;
}
secStart = secEnd;
elapsed = 0;
}
}
//Если надо анимировать только падение фигуры
if(m_animState == NORMAL)
{
//То анимируем падающую фигуру
DrawNormal(elapsed / m_gameLogic.GetSpeed());
}
//Надо показывать «спец»Эффекты
else
{
  //Если игра не на паузе и надо удалить
заполненные линии стакана
if( elapsed > LINE_ANIM_SPEED && !m_isPaused )
{
  //Обновляем стакан
m_playfield.Update();
secStart = secEnd;
elapsed = 0;
m_animState = NORMAL;
//Выбрасываем новую фигуру
ThrowNextBlock();
//Обновляем набранные игроком очки
m_gameLogic.UpdateScore(count);
bottom=count=0;
return;
}
//Рисуем спецэффекты (удаление линии)
DrawSfx
(elapsed/LINE_ANIM_SPEED, bottom, count);
}
```

Вначале нужно опустить фигуру тетриса на ячейку вниз, затем — проверить, не закончилась ли на этом игра. Если игра не закончилась, то стоит посмотреть, есть ли заполненные линии. Если после того, как фигура опустилась, появились заполненные линии, то проигрывается звук удаления линии и отображается анимация удаления этих линий. Если заполненных линий нет, то проигрывается глухой звук, который соответствует простому опусканию фигуры. Далее проверяем, есть ли место в стакане (еще ниже), куда бы могла опуститься фигура. Если некуда — генерируется новая фигура. Теперь должно быть обновлено игровое поле (должны быть удалены линии в стакане) и очки, набранные игроком. Процесс обновления поля см. на листинге «Обновление стакана».

обновление стакана

```
void Playfield::Update()
{
SquareCoord bottom, count;
//Узнаем, есть ли заполненные линии в стакане
GetFullRows(bottom, count);
if(count)
{
  //Удаляем заполненные линии
RemoveRows(bottom, count);
}
```

```
}

void Playfield::GetFullRows(SquareCoord & bottom, SquareCoord & count) const
{
count = 0;
bottom = -1;
//Итерируемся по всем линиям
в указанном диапазоне
for(int i = m_rows.size()-1; i != 0 &&
!m_rows[i].IsEmpty(); --i)
{
  //Если линия заполнена
if(m_rows[i].IsFull())
{
  //То устанавливаем флаг наличия
заполненных линий
if(count == 0)
{
bottom = i;
count++;
}
else
count++;
}
else
  //Прерываем цикл поиска, если найдена
хотя бы одна заполненная линия
if(count != 0) break;
}
}

void Playfield::RemoveRows(SquareCoord
bottom, SquareCoord count)
{
//Проверяем, не наблудили ли наши
шаловливые ручки
assert(bottom>=0 && bottom<m_rows.size());
assert(count>=1 && count<=4);
assert(bottom-count>0 &&
bottom-count<=m_rows.size());
SquareCoord i;
//Итерируемся по заполненным линиям
for(i = bottom; i > bottom-count; --i)
{
//Удаляемая линия должна быть полной
assert( m_rows[i].IsFull() );
//Удаляем ее содержимое
m_rows[i].Reset();
}
//Сдвигаем верхние линии вниз на места
удаленных линий
for(i = bottom; i != 4; --i )
{
m_rows[i] = m_rows[i-count];
}
}
```

Сначала требуется определить, есть ли заполненные линии, — это делается в функции GetFullRows(). Если такие линии есть, грохнем их и опустим все верхние линии вниз на столько линий, сколько только что было удалено, — вытворяем это в функции RemoveRows(). Конфигурации новых фигур создаются случайным образом в функции ThrowNextBlock.

генерация нового блока

```
void GameDriver::ThrowNextBlock()
{
//Если игра не поставлена на паузу
if( !m_isPaused )
{
assert(m_current==0);
//Текущий блок становится будущим блоком
m_block = m_nextBlock;
//Создаем будущий блок
m_nextBlock = TetrisBlock();
//Случайным образом задаем его
конфигурацию
for(int i = 0; i < rand()%4; ++i )
m_nextBlock.Rotate();
  //Задаем начальное положение
в стакане нового блока
m_current = &m_block;
m_currentX = 3;
m_currentY = -2;
//Задаем скорость блока
if(m_pushedSpeed>0)
{
m_gameLogic.m_levelInfo.m_spe-
ed = m_pushedSpeed;
m_pushedSpeed = 0;
}
}
}
```

ваниях. Давай переменным говорящие имена, только не злоупотребляй! Читать сочинения на языке программирования — весьма сомнительное удовольствие. И комментируй, комментируй и еще сто тысяч раз комментируй. Готов поспорить, что через неделю ты посмотришь на свои 20 000 строк кода — и не поймешь, что там написано, если неделю назад не откомментировал.

откажись от указателей Использование указателей на объекты классов и данные в памяти несет в себе потенциальную опасность. Ты можешь забыть удалить выделенную память, переписать значение указателя и потом будешь тратить сутки на поиски тривиальных ошибок. Старайся использовать по максимуму статическую компоновку данных, механизм ссылок, контейнеры STL и Boost. Если нельзя обойтись без указателя, то воспользуйся умными указателями, например shared_ptr из библиотеки Boost.

«ТЕБЕ ПОНАДОБИТСЯ 2D-ДВИЖОК POP-CAP FRAMEWORK И БИБЛИОТЕКА BOOST»

отделяй котлеты от мух Как видишь, архитектурно код тетриса разделен на три части. Одна часть отвечает за отображение пользовательского интерфейса на экране и прием команд от пользователя. Этот код инициализирует виджеты и обрабатывает события, генерируемые ими. Другая часть кода отвечает за игровую логику, причем идет четкое разделение на код, рисующий игровые объекты, обновления игровых объектов, подсчета статистики, анализа правил игры и рисования «спецэффектов».

Вся игровая область (стакан) описывается игровой сеткой. Тетрисные фигуры двигаются строго по этой сетке, причем код логики оперирует только координатами фигур на игровой сетке — перевод в экранные координаты осуществляется только на этапе рисования каждой фигуры. Старайся разбивать программный код на функциональные блоки, используй рекурсию. Пусть у тебя будет множество методов в классах, зато ты не запутаешься в гигантских циклах и непонятных присва-

используй сценарии Для хранения связей между кодом и графическими и звуковыми ресурсами используй сценарии: Lua, Python, XML и т.д. Твой код станет чище и проще, а в довесок ты получишь возможность изменять игровые ресурсы без перекомпиляции программы.

используй готовые решения Если перед тобой стоит дилемма выбрать готовый инструмент или разработать свой, всегда выбирай первое. Сэкономишь массу времени и нервных клеток. В интернете можно найти множество бесплатных готовых движков, вспомогательных библиотек, реализаций различных алгоритмов и т.д.

трехмерные моторы

САГА О 3D-ДВИЖКАХ: КУРС МОЛОДОГО БОЙЦА

ЕСЛИ ПРОВЕСТИ АНАЛОГИЮ С АВТОМОБИЛЕМ, ТО ИГРОВОЙ ДВИЖОК — ЭТО НЕ СОВСЕМ ТО ЖЕ, ЧТО ДВИГАТЕЛЬ ВНУТРЕННЕГО СГОРАНИЯ. СКОРЕЕ, СОВОКУПНОСТЬ СИЛОВЫХ ДЕТАЛЕЙ КУЗОВА, ПОДВЕСКА И, ЕСТЕСТВЕННО, САМ ДВИГАТЕЛЬ. НА ПОЛУЧИВШИЙСЯ СКЕЛЕТ НАВЕШИВАЮТ ДЕТАЛИ ЭКСТЕРЬЕРА И ИНТЕРЬЕРА, КОТОРЫЕ И ПОСТУПАЮТ КОНЕЧНОМУ ПОТРЕБИТЕЛЮ |TONY (PORCO@ARGENTINA.COM)

бестиарий Часто случается так, что жизненный цикл игрового движка прекращается на первой же игре — уже в момент выпуска движок является морально устаревшим. Конечно, с точки зрения экономии ресурсов это абсолютно неправильно, но сложившиеся правила игрового бизнеса не позволяют перевести старый программный код на новые аппаратные рельсы. Так было не раз, и, думаю, так будет и в будущем. Ты, наверное, воскликнешь: «Как же, черт побери, хорошо работать на игровых консолях!» И в какой-то мере ты прав. Для повторного использования кода на консолях есть гораздо больше пространства, если учесть, что жизненный цикл игрового железа составляет не два-три года, как на PC, а в несколько раз больше. Однако в кустах нас поджидает парочка жирных и хищных «но». Во-первых, не так просто пробраться на консольный рынок, там нас ожидает строгий face-контроль платформодержателя, а игру ждут пытки, по сравнению с которыми испанская инквизиция покажется сборищем воспитательниц детского сада. Во-вторых, стоимость разработки под современную консоль значительно выше, чем под PC, хотя, конечно, можно сорвать больший куш, но, впрочем, это не тема статьи.

Ты тут же вспомнишь о RenderWare, Unreal Engine, Source и прочих монстрах геймдева. Как же так? Эти технологии имеют длительный жизненный цикл! Не забывай, что эти движки разрабатывались опытными командами от крупных компаний, которые зашибают основные деньги на лицензировании и поддержке своих продуктов. Примечательно, что на просторах нашей необъятной крупные игроки рынка наконец осознали, что могут зарабатывать деньги подобным же образом. Собственно, кроме предоставления технологий, они обеспечивают продюсерское участие в проектах молодых студий, что позволяет начинающим командам избежать типичных ошибок новичков. Конечно, пока качество отечественных движков оставляет желать лучшего . Будем надеяться, что рано или поздно наше отставание от капиталистов будет ликвидировано, особенные надежды возлагаем на недавние экспансии иностранных инвестиций в наш игропром.

рекомендуем движки

1. WWW.OGRE3D.ORG
2. WWW.NEBULADEVICE.ORG
3. IRRLICHT.SOURCEFORGE.NET
4. WWW.DEVMASTER.NET/ENGINES

ogre 3d

Впрочем, спустимся с небес на грешную землю и вернемся к нашим баранам. Для молодых команд (буржуи называют их Indie team — пока молодая и пока независимая компания) многие компании предоставляют солидные скидки на свои технологии. Кроме того, существует масса бесплатных открытых движков, которые вполне пригодны для создания современных игр. Типичный пример — Ogre 3D (Open source graphics engine), к использованию которого примеряются многие начинающие игроделы. К слову, существует русский ресурс, посвященный Ogre (**www.ogre3d.org.ru**). Еще одно «кстати»: недавно (в декабре 2005) питерская студия Lesta выпустила игру «Стальные Монстры», сделанную на этом движке. Выбор Lesta оправдан, поскольку Ogre — объектно ориентированный кросс-платформенный движок (Windows и Linux), который развивается и поддерживается разработчиками (сейчас они уже перевалили за версию 1.0.6). В качестве рендера в Ogre используются DirectX 7.0, DirectX 9.0 и openGL 1.5, что, с одной стороны, обеспечивает солидную гибкость, с другой — порождает рыхлый исходный код движка с массой лишних классов-wpapper'ов. Движок поддерживает различные шейдеры: Cg, HLSL, GLSL (к примеру, используется развитая идеология материалов). Для экспорта мешей из 3D-редакторов существует масса экспортеров, поддерживаются следующие редакторы: Milkshape3D, 3D Studio Max, Maya, Blender и Wings3D. Возможно исполь-

зовать скелетную анимацию и прогрессивные меши. Менеджмент сцены организован просто и в то же время гибко, имеются объекты сцены, предопределенные на многие случаи жизни, поддерживаются алгоритмы BSP и Octree. Реализовано несколько алгоритмов построения теней. Поддерживаются системы частиц и несколько алгоритмов отображения неба. Функциональность движка легко расширяется за счет того, что он реализован по объектно ориентированным принципам и поддерживает развитую систему плагинов.

Приятно, что движок качественно документирован и имеет обширное Community. К недостаткам можно отнести только отсутствие встроенного качественного редактора игрового мира (сцены), жуткие тормоза и (по большому счету) отсутствие моделирования физики сцены. Однако, как показывает практика, решать эти проблемы гораздо проще, чем писать движок с нуля.

Как видишь, на халяве можно спокойно ковать бабки

nebula device

Ogre — не единственный качественный бесплатный движок. Также заслуживает внимания продукт немецкой компании Radon Labs — Nebula Engine, построенный по принципам клиент-серверной архитектуры и являющийся объектно ориентированным движком, как и Ogre. В отличие от злобного людоеда, Nebula предоставляет более широкие возможности для построения различных сценариев. Любой объект движка имеет символьное имя, который позволяет получить C++-указатель на объект. Другими словами, можно свести всю разработку игры на базе Nebula к написанию скриптов и исключить участие кода на C++. Еще одно отличие от Ogre — то, что Nebula работает только под управлением Windows и только с DirectX. Также здесь имеются классы для реализации сетевых баталий. Поддерживаются шейдеры (HLSL и FX-файлы DirectX), скелетная анимация, системы частиц, тени. Аналогично Ogre, отсутствует физика и встроенный редактор (общее место всех бесплатных движков!), зато предоставляется приличная документация и развитая инфраструктура всевозможных утилит. Дата последнего обновления — июнь 2005 года, текущая версия 2.0.

Неанонсированный ролевой проект Radon Labs

irrlicht engine

Упоминания заслужил и бесплатный движок Irrlicht Engine. Этот кросс-платформенный движок (Windows и Linux) работает с DirectX (8.1, 9.0), openGL 1.5 и собственными программными рендерами. Поддерживает шейдеры и расширяемую библиотеку материалов, анимацию (скелетную и морфинг), собственную систему оверлеев. Как и большинство 3D-движков, этот движок написан на C++ и подчиняется всем законам ООП. Понимает меши всех основных форматов. Реализует системы частиц, стенсильные тени, небо и т.д. Предоставляется качественная документация, последнее обновление датировано декабрем 2005 года, а текущая версия 0.14.0.

Пример работы движка с непроизносимым названием Irrlicht Engine

unreal engine 3

Итак, мы сказали пару слов о доступных движках. Настало время вернуться к теме любимых «Мерседесов» :). Когда задумываешься о них, сразу же вспоминаешь компанию Epic Games с ее линейкой Unreal и соответствующими движками. Последним продуктом этой конюшни является Unreal Engine 3, который смело позиционируется как настоящее и будущее индустрии развлечений. Основные платформы для этого движка — это DirectX 9.0, Xbox 360 и PlayStation 3. Поддерживается графический конвейер, способный работать с 64-битным цветом (HDR), все современные техники освещения (в том числе параметризованная модель освещения Фонга), продвинутые динамические тени (от стенсильных до размытых мягких теней), — все это становится возможным благодаря развитой системе материалов. С помощью редакторов контента разработчики могут редактировать (не отходя от кассы) все эффекты и материалы вплоть до констант шейдеров. Неудивительно и наличие систем частиц, которые можно редактировать с помощью специализированного редактора эффектов. Неотъемлемая часть UE3 — физический движок на базе NovodeX (надеюсь, никто еще не забыл о компании Ageia, ждем в марте месяце 2006 года). Все материалы объектов сцены обладают также физическими характеристиками, например трением. Имеется редактор моделирования физики, который позволяет не только создавать физически реалистичные модели, но и симулировать, настраивать и оптимизировать поведение этих моделей. Возможности анимирования моделей практически не ограничены, с помощью средств движка можно создавать высококачественную скелетную анимацию, которая тесно связана с физической моделью поведения тел, известной как Rag-Doll. Специальный редактор AnimTree позволяет создавать и просматривать все аспекты анимирования модели.

В комплекте движка идет набор экспортных утилит для всевозможных 3D-редакторов. Стандартное системное окружение позволяет создать на базе движка игровую логику любого уровня сложности. Поддерживаются следующие обобщенные игровые объекты: игроки, NPC, инвентарь, оружие и триггеры. Подсистема искусственного интеллекта содержит готовые алгоритмы поиска пути (с учетом триггеров, лестниц и дверей), навигации (с учетом локальных тактических задач), принятия решений и командного «разума». Коллективный разум подходит для реализации шутеров от первого и третьего лица, а также тактических игр. В качестве сценарного языка используется визуальная скриптовая система UnrealKismet. Для разработки демок и заставок на движке используется система UnrealMatinee. Звуковой движок поддерживает все основные звуковые форматы для всех целевых платформ, в том числе объемный 5.1 звук и Dolby Digital. Поддерживается трехмерное позиционирование, разделение звука и смещение по Доплеру. В редакторе сцены UnrealEd работа со звуком осуществляется с помощью специального визуального инструмента.

Познакомься с весьма серьезным дяденькой

Естественно, полностью доступны сетевые игрища, которые являются лицом бренда Unreal. Сетевое взаимодействие организовано на базе протокола UDP, клиент-серверная система поддерживает до 64-х игроков, система без выделенного сервера — до 16-ти игроков. Естественно, возможны игрища между людьми, сидящими на разных платформах. Из анонсированных проектов, под которые лицензирован этот движок, в глаза бросается Duke Nukem Forever и Star Wars: Republic Commando. Список вышедших игр на базе технологий Unreal воистину необъятен (**www.unrealtechnology.com/html/powered/released.shtml**). На DTF есть русский перевод описания движка: **www.dtf.ru/articles/read.php?id=1102**. Стоимость лицензии на использование UE3 для одной платформы составляет $750 тыс., за каждую дополнительную платформу предлагается доплачивать по сто тысяч безусловно правильных денег.

renderware

Продукт компании Criterion (с аппетитом сожрана Electronic Arts), в отличие от UE3, является собой сегодняшний день. Основные платформы для этого движка — PlayStation 2, PlayStation Portable, Xbox, GameCube, PC и N-Gage. Основанный на компонентной архитектуре, RenderWare Studio 2.0 предоставляет комплексную систему для разработки игр — от прототипирования игры до тестирования и сборки конечных билдов. Вся эта система завязана на четкой формализации всех процессов разработки. На данный момент RenderWare является корпоративным стандартом для электроников и закуплен (вместе с Criterion) для дальнейшей унификации и формализации процесса разработки сверхунифицированных и формализованных игр от EA :). Черт побери, кто-нибудь знает, чем отличаются FIFA 2005 от FIFA 2006?.. Кто это сказал, что одним байтом?!

source engine

Еще одно имя, которое пока у всех на слуху, — это компания Valve со своим движком Source, на котором были сделаны Half-Life 2, Day of Defeat: Source, Counter Strike: Source. С использованием этой же технологии компания Troika Games (о ней нужно говорить «либо хорошо, либо никак») выпустила модненькую RPG Vampire the Masquerade: Bloodlines, a Smiling Gator Productions ваяет очередную «нетленную» MMORPG Twilight War: After the Fall, которую сами авторы именуют не много не мало как Extreme Online Roleplaying Game (XORG).

Теперь покопаемся в его кишочках… В отличие от описанных мной движков, Source предназначен только для Windows-платформы и только для DirectX. Язык разработки — C++. Как и любой уважающий себя движок, этот поддерживает солидную библиотеку материалов, системы частиц, несколько моделей освещения и затенения. Аналогично Unreal Engine 3, материалы Source несут в себе и физический смысл, то есть поверхности игровых объектов сумеют катиться, скользить и т.д.

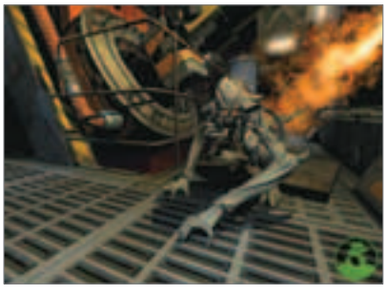
Source силен сетевым кодом, который отлаживается огромными толпами геймеров по всему миру начиная с 1999 года. Кроме обычной скелетной анимации, движок предоставляет возможности по анимированию лиц, симуляции речи, мимики лица и выражения глаз. В качестве физического движка используется самое современное на сегодня программное решение на базе Havok 2. Собственно, Half-Life 2 — это самый удачный пример активного использования физики в gameplay. Еще одной традиционно

HDR в действии

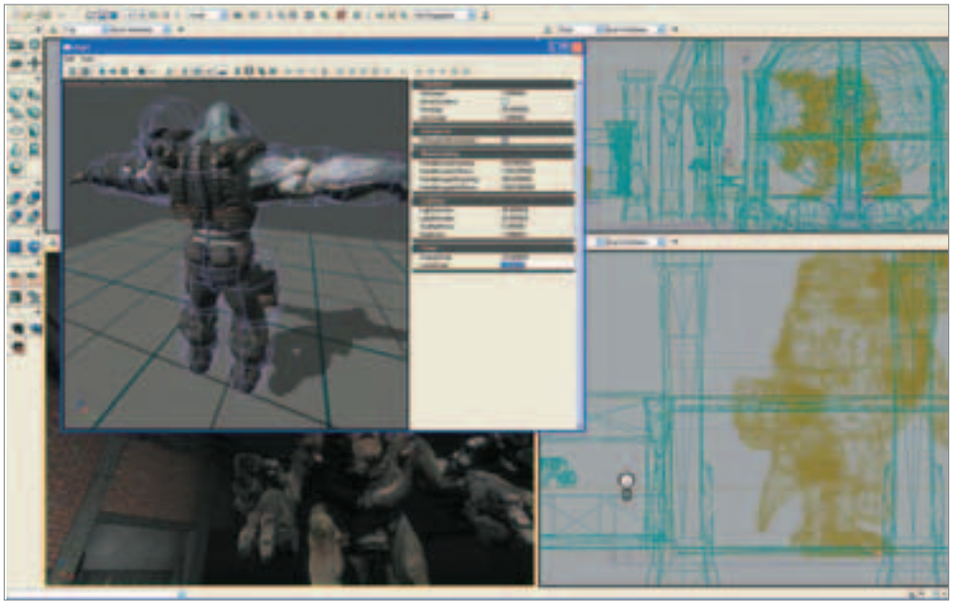
сильной стороной технологий Valve является искусственный интеллект. Персонажи в Source умеют принимать эффективные навигационные решения исходя из поставленных тактических задач, воспринимают окружающую информацию привлекая слух, зрение и нюх, умеют «работать» в команде. Звук в движке также тесно связан с физикой и логикой игрового мира, поддерживается трехмерный 5.1. звук и эффект Доплера. Имеется собственная система оверлеев. Все ресурсы, используемые Source, редактируются встроенными инструментами: моделирование лиц, игрового мира и игровой логики, оверлеи, всевозможные инструменты экспорта моделей из популярных 3D-редакторов. На сегодня перспективы Source, по сравнению с конкурентами, скрыты в тумане, что связано с пришествием консолей следующего поколения. Скорее всего, Valve не захочет терять столь привлекательный рынок, и компания сейчас ведет работы по созданию движка нового поколения, работающего на всех популярнейших платформах. Поживем — увидим.

doom 3 engine

Думаю, представлять банду Джона Кармака не нужно. Эти люди стояли у истоков шутеров и полигональных игр, поэтому просто необходимо сказать пару слов об их последнем движке. На нем были сделаны следующие игры: естественно, Doom 3, add-on к нему Doom 3: Resurrection of Evil, Quake 4, разрабатываемые в данный момент Pray, Enemy Territory: Quake Wars на движке с поддержкой технологии MegaTexture. Изначально планировалось доработать движок Quake 3 до потребностей сегодняшнего дня, однако позже команде Кармака стало ясно, что необходимо полностью переработать код. И вот, получившийся движок имеет очень мало общего с кодом для движков предыдущих Quake. Если раньше движки от ID были написаны на чистом C, то теперь D3 может похвастаться объектно ориентированными концепциями.



В остальной же функциональности движок вряд ли опережает современные решения на базе Unreal Engine 3. Да, есть освещение, есть тени, для будущих игр планируется сделать поддержку мягких теней, есть технология для анимации лица наподобие технологий Source. Есть даже технология громадных текстур, названная MegaTexture — она позволяет сделать текстуры земли и строений более детализированными и не повторяющимися. В целом же технология Doom 3 постепенно уходит в прошлое, а Кармаку и компании придется приложить значительные усилия, чтобы их технологии хотели лицензировать в будущем, — на одном имени далеко не уедешь.



Не двигайся, гаденыш. Сейчас я тебя…

ПОНЯТИЯ Как уже было сказано, движок — это совокупность программных абстракций. Конечно, разные движки отличаются друг от друга деталями, но все они оперируют одинаковыми понятиями и выполняют одинаковые функции. Следовательно, в общем они похожи. Для начала определимся с терминами данных, которые обрабатываются движком.

Самой высокоуровневой абстракцией в любой игре являются игровые объекты со своими игровыми характеристиками, такими как здоровье, энергия. Игровыми объектами управляет пользователь и подсистема ИИ. Такой объект подчиняется действию туземной физической модели. Случается, что функциональности игровых объектов, имеющихся в движке, не хватает для реализации всех фишек геймплея, соответственно, разработчики игры реализуют собственные игровые объекты. Фактически они моделируют логику игры. Конечно же, нам хочется идеала: чтобы единственным кодом, который пришлось бы писать в процессе разработки, был код геймплея. Жаль, но ничто в этом мире не идеально. Очень часто модели поведения игровых объектов реализуются на языке, отличном от языка движка (обычно C++). Для этого используются сценарные языки, такие как Lua и Python, в основном чтобы было возможно изменять и отлаживать игровую логику без перекомпиляции исходного кода движка (она занимает до черта много времени!). Плюс, конечно — для того чтобы упростить и без того сложный игровой код. Правда, зачем нужна небезопасная возможность работать напрямую с памятью, если при разработке логики игры оперируют только абстракциями игровых объектов? Жизненный цикл игровых объектов обеспечивается специальным менеджером, который также связывает игровые объекты с подсистемой и алгоритмами ИИ плюс с пользователем. Аналогично игровым объектам, особняком в движке стоят так называемые оверлеи, или сущности для реализации пользовательского интерфейса. Часто используются готовые решения, такие как Crazy Eddie's GUI. Оверлеи также описываются сценариями, причем широко используются сценарии XML для представления иерархий элементов пользовательского интерфейса. Оверлеи, конечно, не имеют никакого отношения к рендерингу сцены, зато с их помощью пользователь управляет игровыми объектами.

сцена Перейдем на один уровень абстракций ниже — где-то на этом уровне появляются абстракции, существующие в любом современном движке. Управление сценой возлагается на менеджер сцены, который хранит иерархию объектов сцены, производит проверку видимости объектов, управляет камерой, источниками света, небом, туманом и прочими объектами сцены. Рассмотрим пример «простейшей» сцены: автомобиль едет по дороге мимо фонаря. К корню (root) сцены присоединяется статичный (в процессе игры не передвигается) объект — дорога (плоскость), для этого объекта устанавливается материал — асфальт с текстурой двухполосной дороги. Дифференцирование всех объектов сцены на статичные и динамичные производится ради оптимизации. На обочине дороги устанавливается фонарный столб со своим материалом. На верхушке столба располагается фонарь — это нестатичный источник света, поэтому для него заранее можно рассчитать карту освещенности всех статичных элементов сцены: дороги и столба. Таким образом можно сэкономить при расчете освещения.

Теперь начинается самое интересное. На дорогу помещается кузов машины, к нему присоединяются четыре колеса и два направленных источника света — фары. Машина передвигается по дороге, вместе с ней передвигаются фары, то есть два источника света, поэтому для каждого кадра необходимо рассчитывать освещенность дороги и столба фарами, а также освещенность машины фонарем со столба. Прибавь к этому расчет тени от машины, которую она отбрасывает на дорогу от фонаря, расчет карты отражения: кузов лакированный, соответственно, в нем можно разглядеть дорогу и столб.

Посмотрим, как происходит типичный цикл рендеринга такой сцены. Сначала проверяются все динамичные объекты. Для тех объектов, которые изменили свое положение на сцене, пересчитываются их мировые координаты. Далее вызывается один из алгоритмов отбрасывания невидимых объектов (BSP, Octree). Затем для материалов объектов сцены, подвергаемых рендерингу, устанавливаются шейдерные константы для источников света, различных матриц преобразования. И только после этого графический процессор начинает обработку видимой геометрии.

Меши, модели и их материалы Мы плавно перешли к понятиям моделей и мешей. Изображение, которое ты видишь на экране, составлено из кучи полигонов (треугольников), они и образуют единые и неделимые наборы геометрии — меши. Например, кузов автомобиля и его колеса — это различные меши. Вместе они составляют модель автомобиля. Почему именно так, а не иначе? Предполагается, что в игре нельзя разбить кузов автомобиля, нельзя разрезать и разделить его на составные части, которые перемещались бы относительно друг друга. Колеса же могут перемещаться (например изменять свою ориентацию относительно двух осей), поэтому они реализованы в модели автомобиля как составные части. Каждый меш обладает своим материалом — способом отображения геометрических данных на экране.

Материал — это совокупность нескольких текстур, различных состояний графического конвейера, а также вершинная и фрагментная программы (шейдеры). Игровые объекты разбиваются на меши и модели исходя из соображений моделирования физики и логики игрового мира, а также способности графического конвейера отрендерить картинку. Конечно, хотелось бы иметь возможность оперировать десятками тысяч игровых объектов, взрывать стены и изменять земной ландшафт ковровыми бомбардировками, но современные графические технологии пока не позволяют реализовать все это в реальном времени. Пока все, на что мы способны, — это моделировать поведение сотни моделей, взаимодействующих друг с другом, и отображать со значительной аппроксимацией картинку. Так вот интересность геймплея достигается за счет таланта и работы дизайнеров и художников. В любой современной игре геометрия занимает наибольшее место в памяти. Для повышения скорости загрузки мешей их геометрия сохраняется в бинарных форматах, часто уникальных для каждого движка. Материалы же, напротив, часто хранятся в виде сценариев, причем дифференцируются следующим образом: отдельно хранятся шейдеры и состояния конвейера (например, в *.fx-файлах), отдельно — наборы текстур и цве-

известные движки

1 WWW.UNREALTECHNOLOGY.COM/HTML/TECHNOLOGY/UE30.SHTML
2 WWW.RENDERWARE.COM
3 WWW.HL2.RU/SOURCE
4 HTTP://EN.WIKIPEDIA.ORG/WIKI/DOOM_3_ENGINE
5 WWW.TOUCHDOWNENTERTAINMENT.COM/JUPITEREX.HTM

тов. В код движка их обычно не встраивают, чтобы оставалась возможность вмешаться оперативно и без перекомпиляции самого движка. На рисунке «Сценарий материала» ты можешь увидеть кусок сценария, описывающий материал стекла транспорта из Quake 4.

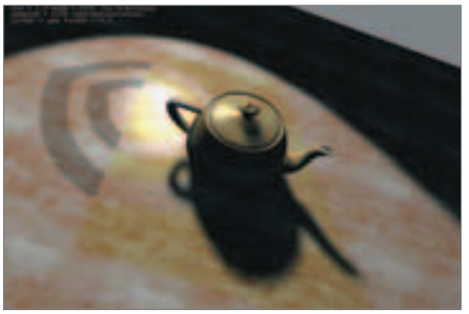
системы частиц Не все объекты сцены будет разумно представлять в виде мешей. Например, клубы дыма или пулеметные гильзы могут появляться на экране сотнями, тратить на них много памяти нецелесообразно. Для отображения таких объектов применяют системы частиц — абстракции, управляющие множествами спрайтов. На каждый спрайт тратится четыре вершины, то есть около 50-ти байт памяти. На вершины натягивается текстура. При моделировании систем частиц дизайнер изменяет их насыщенность, траектории, скорости и цвета, в итоге получаются огонь, дым, гильзы, снаряды и т.д. Параметры систем частиц сохраняются в специализированных сценариях — для оперативной правки. При создании систем частиц разработчики оперируют понятиями эмиттер и аффектор частиц.

Эмиттер — это сущность, из которой испускаются частицы, характеризуется скоростью испускания частиц, количеством частиц, плотностью и т.д. Аффектор — это сущность, которая воздействует на траекторию движения частиц, то есть некий модификатор, изменяющий на каждом кадре местоположение и цвета каждой частицы.

освещение Расчет освещения для всех моделей сцены — чертовски трудоемкая задача. Разработчики игр стараются максимально снизить подобные издержки, поэтому создают статические карты освещения, для того чтобы можно было не пересчитывать на каждом кадре кучу лишних источников света. Передвигаемых лампочек в любой игре одна-две, они действительно освещают модели в масштабе реального времени. Существуют следующие виды источников света: точечные (лампочка), направленные (солнце), споты (фары, прожектор).

Рядом с проблемой освещения стоит проблема построения теней, которые отбрасываются и принимаются объектами сцены, — самая трудоемкая часть рендеринга. Сейчас фотореалистичные мягкие тени строятся в реальном времени только в отдельных демонстрационных программах и примерах. В реальной игре разработчики стараются использовать предварительно рассчитанные (статические) тени, а для динамичных объектов строятся тени четкие либо упрощенные до максимума (например кружок на полу). Сравни качество мягких и четких теней на соответствующих рисунках — картинка стоит тысячи слов, особенно там, где указаны FPS.

Сравни мягкие тени



С четкими тенями

менеджмент ресурсов При создании движков разработчики стараются придерживаться концепций систем, управляемых данными (data driven system). Фактически получается, что различные сценарии (ресурсы) загружают другие ресурсы и управляют ими. При этом движок играет роль фабрики, на которой обрабатываются ресурсы, из которых получается конечный продукт — геймплей. Если ресурсы обрабатываются всеми подсистемами движка, то контроль над жизненным циклом ресурсов, а также процесс загрузки ресурсов в память возлагается на подсистему менедж-мента ресурсов. К таким подсистемам предъявляются следующие требования:

- ВОЗМОЖНОСТЬ ДЕКОДИРОВАТЬ ЗАГРУЖЕННЫЕ РЕСУРСЫ В ФОРМАТЫ, ПОНИМАЕМЫЕ ГРАФИЧЕСКИМИ API, ЗВУКИ, ТЕКСТУРЫ И Т.Д. ИЗ ОДНОГО ФОРМАТА В ДРУГОЙ;
- ЗАГРУЗКА РЕСУРСОВ В ПАМЯТЬ ИЗ ФАЙЛОВОЙ СИСТЕМЫ, ИЗ АРХИВА И ИЗ СЕТИ;
- ПАРСИНГ И ИСПОЛНЕНИЕ СЦЕНАРНЫХ РЕСУРСОВ.

В зависимости от используемых сценарных языков, реализуются различные механизмы передачи данных в движок и обратно. Кроме загрузки ресурсов, подсистема менеджмента должна контролировать правильность выгрузки ресурсов и освобождения памяти, иначе оперативная память будет исчерпана в рекордно короткие сроки.

диагностика На этапе разработки и, как показывает реальная жизнь, на этапе эксплуатации часто требуется проанализировать ошибки программы. Для решения подобных задач в движке задаются стратегии обработки ошибок, анализа сбоев, ведутся протоколы работы (логи). В некоторых случаях покупатели (!) игр участвуют в поиске ошибок: служба поддержки просит их выслать «некоторые файлики» — протоколы ошибок и всевозможные дампы. Диагностика ошибок игры в целом и движка в частности заметно осложняется условиями работы, а именно тем, что код работает в реальном времени. Именно по этой причине в играх редко используется многопоточность — она содержит потенциальную угрозу безопасной работе.

редактор Главными инструментами для любого разработчика игры являются всевозможные редакторы игрового контента. Для полноценного создания игры не хватает возможностей 3D Studio, Maya и т.д., в этих трехмерных редакторах создаются только модели и анимация к ним. Для создания игровых уровней, программирования игровой логики, программирования физических моделей, редактирования материалов, доведения до ума анимации, расстановки освещения и теней, создания систем частиц применяются редакторы, встроенные в движки.

коротко о главном Если тебя вдруг посетила идея взяться за написание движка, подумай сто раз, а лучше двести. Конечно, если ты не ставишь перед собой задач завоевать мир, вполне реально создать движок, а затем и игру на нем. Если ты четко представляешь себе цель (например самообразование, создание движка, который умеет текстурировать модельку и освещать ее непременно одной лампочкой), то ты ее достигнешь. В любом случае сразу забудь бредни по поводу кросс-платформенности кода, независимости от графического API — это нереально. Не думай, что кто-то в индустрии захочет пользоваться твоим шедевром. Пиши шедевр для себя, для удовольствия, для level_up'а наконец. Используй по максимуму готовые решения, не изобретай велосипед, изучи STL, Boost. Если же хочется не возиться с движком, а просто создать свой трехмерный тетрис, используй готовые практически бесплатные решения: движки Orge, Irrlicht или Nebula, звуковую библиотеку OpenAL, простенькую ODE или Tokamak-физику. И тогда вероятность, что твой проект станет завершенным, намного повысится

вспомогательные библиотеки

1 WWW.BOOST.ORG
2 WWW.LUA.ORG
3 WWW.PYTHON.ORG
4 WWW.CEGUI.ORG.UK
5 WWW.XMLSOFT.ORG
6 WWW.OPENAL.ORG
7 WWW.ZLIB.NET
8 WWW.ODE.ORG

мнение профессионалов

ПОЛНОЦЕННУЮ ИГРУ СОЗДАТЬ САМОСТОЯТЕЛЬНО НЕРЕАЛЬНО

СПЕЦ: СЛОЖНО ЛИ СДЕЛАТЬ СОБСТВЕННУЮ ИГРУ? С ЧЕГО НАЧАТЬ, ЕСЛИ ТЫ ПРОГРАММИСТ, НО НИКОГДА НЕ ДЕЛАЛ ИГРЫ?

АЛЕКСАНДР ФЕДОРА: Если программист имеет мало опыта, то самое главное — не браться за слишком сложный проект. Лучше сделать попроще, но закончить проект и получить драгоценный опыт, чем, переоценив свои силы, забросить проект и, потеряв энтузиазм, больше никогда не работать в геймдеве. Начинай с аркад, а не с MMORPG!

СЕРГЕЙ АЗАРОВ: Сделать собственную игру можно, вопрос только в том, какую игру делать. Для любого программиста, который никогда не делал игры, начинать надо с малого. Как правило, этим малым является тетрис :). Причем для многих данный шаг является психологически непреодолимым барьером, ибо почти каждый почему-то считает, что он может больше и напишет сразу «Убийцу Doom 3». Однако, даже прежде чем писать тетрис, необходимо подготовить теоретическую базу для этого процесса. В этом вопросе очень сильно помогут различные интернет-ресурсы, количество которых сейчас велико. Рекомендую **dev.dtf.ru**, **www.gamedev.ru** и **www.gamasutra.com**.

АНДРЕЙ БЕЛКИН: Если ты программист, но никогда не делал игры — лучше и не начинай, потому что один программист, конечно, может сделать собственную игру, но для этого ему надо заодно стать геймдизайнером, художником, аниматором и звукорежиссером. Даже если предположить, что один программист объединит в себе все эти способности и выберет самый простой жанр, то максимум, что он может сделать, — игру на уровне shareware первой половины 90-х годов прошлого века. И завоюет заслуженное уважение младшего брата, девушки, родителей и пары друзей, которые из вежливости скажут, что игра им понравилась.

МИХАИЛ ПИСКУНОВ: Прежде всего, игры делают не программисты, а геймдизайнеры. Это категория творческих людей вроде писателей, художников, музыкантов. Вот только геймдизайнерам приходится гораздо труднее. Если художнику нужны только краски, а писателю — редактор Word, то для того чтобы реализовать идею игры, нужно найти других людей: программистов, художников и т.д. Кстати, режиссеры фильмов такие же несчастные: им требуются операторы, сценаристы, актеры... Конечно, здорово, когда геймдизайнер умеет программировать или программист имеет задатки геймдизайнера. Тогда кое-что интересное можно сделать самостоятельно. Пример тому — тетрис :).

ДМИТРИЙ ЖУКОВ: Смотря какую игру. Можно написать простую, но гениальную игру типа тетриса. Сегодняшние игры — это результат работы художников, композиторов и т.д., — всех тех, кто создает ее содержание. Создать полноценную игру самому — нереально. С чего начать? В Сети есть множество форумов, имеет смысл зайти почитать...

СЕРГЕЙ ЗАГУРСКИЙ: Смотря о какой игре идет речь. По возрастанию сложности процесса, если грубо прикинуть: flash-игры, игры для мобильных телефонов, shareware-игры, PC-игры, игры для консолей. Сделать хорошую коробочную игру в одиночку и/или без финансирования практически нереально — слишком высока сложность технологий и велики объемы работ. И чем дальше, тем все меньше и меньше становится вероятность создания хорошей игры на энтузиазме. Лучше всего начинать с простого — с shareware-игр. Следует быть готовым к неудачам, так как бОльшая часть требований познается в процессе разработки и поиска издателя. Так что необходимо будет присутствие продюсерских центров, способных профессионально вести и координировать игровые проекты с самого начала и до конца.

АНДРЕЙ ТЕРТИЧНИКОВ: Сделать собственную полноценную игру одному человеку вообще очень сложно. Нужны и программирование, и графика, и звук. А начинать нужно с какой-нибудь общеизвестной несложной игры вроде тетриса. Самое главное тут — получить опыт, а потом искать единомышленников (либо искать уже существующие команды и вливаться в их ряды) и пытаться делать что-то серьезное. Все приходит с опытом, да и программирование игр — это ведь в первую очередь именно программирование. Поэтому, если человек хорошо разбирается в этой области, узнать специфику игростроения для него будет дело времени.

СПЕЦ: СКЛАДЫВАЕТСЯ ТАКОЕ ВПЕЧАТЛЕНИЕ, ЧТО БОЛЬШИНСТВО РАЗРАБОТЧИКОВ ПЕРЕХОДЯТ ОТ КАЧЕСТВА К КОЛИЧЕСТВУ. А НЕКОТОРЫЕ ИГРЫ ВЫХОДЯТ В СВЕТ ОТКРОВЕННО НЕДОРАБОТАННЫМИ...

АЛЕКСАНДР ФЕДОРА: Каждый разработчик, находясь в здравом уме, хочет, чтобы его игра получилась качественной. Но в большинстве случаев то, что задумали разработчики в начале, и то, что получилось в конце, не совпадает. Виной тому — ограниченность бюджета и прочие объективные причины, по которым игру необходимо выпустить в определенные сроки, чтобы она окупилась, даже если она еще не до конца готова и отлажена.

ЮРИЙ МАТВЕЕВ: Если про российские — причина на поверхности. Принятое радикальное решение борьбы с пиратством (снижение цен на лицензионный продукт до уровня пиратских дисков) привело к тому, что экономически стало невыгодно делать что-то «большое и светлое». В самом деле тот же Quake 5, сделанный в России, продавался бы по цене 200 рублей, как и какой-нибудь Штырлиц... По закону сохранения энергии (и финансов) выгоднее делать Штырлица, чем Quake 5. Очевидное — невероятное. Но это факт. С западными играми несколько сложнее, но действует, в общем-то, тот же принцип...

СЕРГЕЙ АЗАРОВ: В самом начале разработки никто на присутствие «багов» в конечном продукте не закладывается. В целом же можно сказать, что основная причина выхода недоработанных игр (в частности, в России) — перерасход бюджета игры и, соответственно, желание поскорее выпустить продукт, дабы его окупить. И вторая причина — крайне слабо поставленное тестирование игр.

МИХАИЛ ПИСКУНОВ: Создание игр — это борьба двух начал, творчества и бизнеса, как инь и ян. На данном этапе бизнес берет верх. Но если творчество будет брать верх, получим много- и долгострои, разоряющие издателей, — это исторический факт. В любом случае, борьба начал — колебательный саморегулирующийся процесс. Когда покупатели устанут от серий шаблонов-недоделок, маятник пойдет в обратную сторону.

СЕРГЕЙ ЗАГУРСКИЙ: Внутренняя сложность игр постоянно возрастает, а сроки становятся все более сжатыми. Индустрия должна будет претерпеть некоторое сущностное изменение, чтобы вывести планку качества на новый уровень, возможно, с уменьшением количества. Что-то подобное уже имело место в киноиндустрии в прошлом веке.

АНДРЕЙ ТЕРТИЧНИКОВ: Дело в том, что многие разработчики идут по пути наименьшего сопротивления: придумать что-то свое, оригинальное и интересное довольно сложно, а многим зачастую просто лень. Так как мобильные игры сейчас очень популярны, а срок их жизни довольно мал, то их надо производить очень быстро. Можно сделать игру с интересным геймплеем и соответствующей ему графикой, но не факт, что ее будут покупать. А можно просто взять какую-нибудь из популярных игр на PC и сделать ее «мобильную версию». Зачастую такие версии похожи на своего «прародителя» только заставкой...

СПЕЦ: МОЖЕТ ЛИ ТОТ, КТО ПИШЕТ ИГРЫ ДЛЯ «ПЕРСОНАЛКИ», РАЗРАБОТАТЬ И МОБИЛЬНУЮ ИГРУ? ИЛИ ЖЕ СУЩЕСТВУЮТ КАРДИНАЛЬНЫЕ РАЗЛИЧИЯ?

АЛЕКСАНДР ФЕДОРА: Есть различия и очень существенные. Сейчас при разработке игр для мобильных платформ стоят почти те же самые ограничения, какие были много лет назад на играх для ПК.

СЕРГЕЙ АЗАРОВ: Кардинальных различий (если судить с позиции кодирования игр) практически нет. Разница лишь в концептуальном подходе и контенте игр. Все мобильные игры из-за специфичности платформы накладывают свои уникальные требования на жанры и их реализацию.

АНДРЕЙ БЕЛКИН: Может. Но кардинальные отличия есть. Операционная система, процессорные мощности, объем оперативной памяти, размер экрана, специфический дизайн мобильных игр и многое другое.

МИХАИЛ ПИСКУНОВ: Различий, пожалуй, нет. Не надо забывать, что практически все, что имеется сейчас на мобильниках, раньше было реализовано на старинных персоналках :).

ДМИТРИЙ ЖУКОВ: Может. Кардинальных различий нет. Тот же самый язык C++, только сильно упрощенная операционная система и ограниченные ресурсы — маленькое количество памяти и небольшое разрешение экрана.

СЕРГЕЙ ЗАГУРСКИЙ: Безусловно, он может написать и мобильную игру, но не настолько быстро и качественно, как это сделает команда, уже поднаторевшая в создании мобильных игр. Кардинальные различия есть, причем как в способах разработки, так и в целевой аудитории.

АНДРЕЙ ТЕРТИЧНИКОВ: По большому счету, при наличии желания и времени можно разобраться в любой технологии. Однако различия, конечно же, есть. Во-первых, язык программирования. Для «персоналок» — в основном C++, для мобильников — Java (во всяком случае, в России другие технологии слабо развиты). Это не аксиома, но в большинстве случаев именно так. С другой стороны, если рассматривать именно сам процесс разработки, то он схож с разработкой игр для других платформ, просто имеет более упрощенную структуру. В частности, в разработке задействовано меньше людей и отсутствуют некоторые этапы, характерные для работы над PC-играми

В ПРОДАЖЕ С 22 ФЕВРАЛЯ

ЖУРНАЛ ДЛЯ МУЖЧИН, ЖИВУЩИХ В МИРЕ ТЕХНИКИ

ТАТУ В МОРГЕ: ЭКСКЛЮЗИВНО О ГЛАВНОМ СМЕРТЕЛЬНОЕ ОРУЖИЕ-XXI МЕЛ ГИБСОН ОТДЫХАЕТ КОЛЕСА ТВОЕЙ МЕЧТЫ НОВИНКИ ИЗ ДЕТРОЙТА ГАДЖЕТЫ НА ВСЕ СЛУЧАИ ЖИЗНИ

СИЛЬНЫЕ ЖЕНЩИНЫ
СМЕШНЫЕ МУЖЧИНЫ
И ГРЯЗНЫЕ СПОРТСМЕНЫ

как закалялся flash

СОЗДАНИЕ ОНЛАЙН-ИГРЫ THE OFFICE SPACE

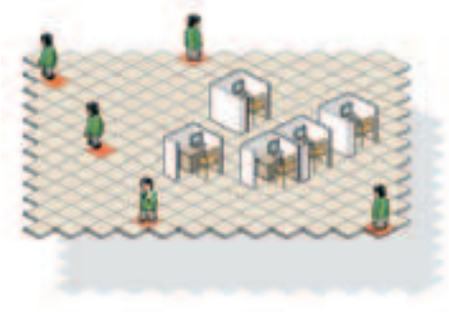
FLASH-ИГРЫ МЕГАПОПУЛЯРНЫ В НАШЕ ВРЕМЯ. РЕДКИЙ ОФИСНЫЙ СОТРУДНИК ХОТЬ РАЗ В ЖИЗНИ НЕ ПРОЖИГАЛ КАЗЕННОЕ ВРЕМЯ И ФИРМЕННЫЙ ТРАФИК В ПОДОБНОМ ВРЕМЯПРЕПРОВОЖДЕНИИ. В ЭТОЙ СТАТЬЕ МЫ РАССКАЖЕМ И ПОКАЖЕМ, КАК СОЗДАВАЛАСЬ ОДНА ИЗ ТАКИХ ИГР |**СЕРГЕЙ ПОДБЕРЕСКИЙ**

The Office Space

ИДЕЯ, РУКОВОДСТВО ПРОЕКТОМ, КОДИНГ И ДИЗАЙН — **СЕРГЕЙ ПОДБЕРЕСКИЙ**
АНИМАТОР — **АНДРЕЙ МАЦКО,** ТЕКСТЫ — **ERIN VAN SICKLE,** ЗВУК — **АЛЕКСАНДР ГЛАДКИЙ**

THE OFFICE SPACE, НЕБОЛЬШАЯ ИГРА, СОЗДАННАЯ С ПОМОЩЬЮ MACROMEDIA FLASH В 2003 ГОДУ, ДОВОЛЬНО БЫСТРО ЗАВОЕВАЛА ПОПУЛЯРНОСТЬ. ХОТЯ ЯДРОМ ИГРЫ ЯВЛЯЕТСЯ СЛОЖНЫЙ ИЗОМЕТРИЧЕСКИЙ ДВИЖОК (ENGINE), THE OFFICE SPACE ДОВОЛЬНО ПРОСТА И НЕ РАССЧИТАНА НА ДОЛГИЙ ИГРОВОЙ ПРОЦЕСС. ИГРА БЫЛА УДОСТОЕНА НЕСКОЛЬКИХ ИНТЕРНЕТ-ПРИЗОВ И СТАЛА СЕРЕБРЯНЫМ ПРИЗЕРОМ КОНКУРСА, ОРГАНИЗОВАННОГО ЖУРНАЛОМ I.D.MAGAZINE В 2004 ГОДУ, В КАТЕГОРИИ ИНТЕРАКТИВНОГО ДИЗАЙНА

ИГРА ДОСТУПНА НА WWW.THEOWORLDS.COM/GAMES/OS

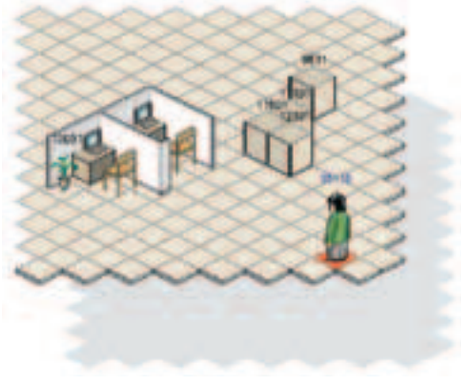


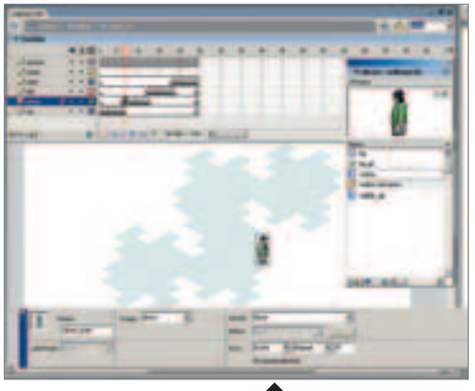
1 начало: идея Какую игру создать? Сначала для нас этот вопрос не казалось важным, но призадумавшись, я понял, что на самом деле не все так просто и нужно определиться. В первую очередь — с желаниями :). Мои желания были просты: создать небольшую игру, в которую можно поиграть во время обеденного перерыва, предварительно не читая длинные инструкции. Чтобы ограничить игровое время, было решено сделать игру из конечного числа уровней. На случай если понадобится расширить ее, мы запланировали наличие редактора карт. Так стали выявляться первые технические детали, а конечную идею нам подсказал фильм Office Space. Кто смотрел, тот поймет — комедия о самой обычной скуке и офисной рутине. В каждой компании есть какие-то особенности (они и описаны в фильме), иногда бывает полезно посмеяться над ними. Вот так на почве юмора и родился проект.

2 начало. разработка «движка» и определение задач Идея с уровнями/картами органично сочеталась с офисной обстановкой, поскольку нарисовать офис легко: мебель, кулеры, секретарши и прочие атрибуты :). Я поискал техническое решение и решил остановиться на изометрической проекции (вид сбоку-сверху) — выглядит не так плохо, кроме того, к тому времени появились аналогичные проекты в интернете, а значит, технически моя идея была реализуема. Итак, я сразу сел писать код actionscript (AS1 на Flash 5), чтобы создать базу движка и со временем построить проект на его основе. Я начал с простой карты, разделенной на квадраты (ромбы из нашего вида). Планировалось расположить все элементы по границам этих квадратов. Так они легче запоминаются в массиве (двумерный массив с ячейкой для каждого квадрата). Привязка к квадратам, а не к координатам очень облегчает просчет взаимодействий персонажей и предметов, упрощает определение z-порядка элементов (вопрос о том, какие объекты «ближе»), нужного для организации правильного перекрывания предметов.

Как создавался персонаж? Было решено сделать его с тремя фазами анимации: стоит на месте, правая нога поднята, левая нога поднята. Последние две фазы чередуются в то время, пока персонаж движется (самая элементарная анимация ходьбы :)). Герой/персонаж двигается в четырех направлениях, два из них зеркальные (можно просто перевернуть картинку, для этого персонаж рисуется симметричным). Итого — шесть разных картинок, то есть минимум, хотя если персонаж захочет скакать на одной ноге, можно обойтись и двумя фазами.

Однако трех фаз оказалось слишком мало, движения персонажа оказались чересчур резкими, так что пришлось добавить еще две фазы (итого десять картинок для каждого персонажа). Кстати, несмотря на то, что увеличение количества картинок с фазами обеспечивает достойную плавность и красоту, оно же приводит к увеличению времени работы аниматора и размера конечного файла. Так что тут, как и во многом в жизни :), нужно искать золотую середину. Кстати, профессиональные аниматоры знают много трюков по достижению достойной плавности движения с упрощенной анимацией. Мы же двигались методом проб и ошибок. Андрей Мацко занимался анимацией персонажей, и уже когда наш первый персонаж делал свои первые шаги, мы поняли, что анимация будет сложнее, чем ожидалось. Работы над персонажами начались рано, так как мне не хотелось





Первые шаги (персонажа) во Flash'е.

долго пользоваться временной заменой (куб подходящего размера) — нужно было определиться с размерами, скоростью анимации ходьбы и т.д.

В нашем проекте картинки анимации персонажа организованы по кадрам и слоям. Кроме простого кода передвижения, было добавлено пару строк для реагирования на клавиатуру. Еще несколько десятков строк — и персонаж начал двигаться, отвечая на команды. Хотя к тому моменту он еще не попадал точно в квадраты, выходил за карту и плохо реагировал на резкую смену направления, было приятно видеть первые визуальные результаты.

Я не буду описывать технические детали создания изометрического движка и сейчас объясню почему. Во-первых, в интернете накопилось очень много информации на эту тему. Для примера **www.kirupa.com/developer/actionscript/isometric_transforms.htm** — очень хорошая серия уроков. Во-вторых, технологии не стоят на месте, actionscript изменился, изменился и сам Flash. Кое-что я бы изменил, что-то поменял бы совсем. Не стоит повторять мои ошибки, пусть каждый делает как ему легче и понятнее. Главное — чтобы работало хорошо. Кроме того, этот текст не об actionscript'е, а о процессе создания игры в целом.

3 развитие проекта. создание игры

После нескольких экспериментов я решил перейти к организации самой игры. У нас было время обдумать общую структуру и то, как все части будут взаимодействовать друг с другом, чтобы потом не пришлось менять что-то в корне (хотя абсолютной гарантии, конечно, никто нам не давал).

На этом этапе я создал два основных actionscript-класса: класс персонажа и класс карты. И вот настало время работать над взаимодействием этих классов, вскоре анимация была налажена: персонаж двигался ровно по квадратам, в границах карты и хорошо реагировал на команды. Итак, мы уже создали персонажей. Но что они могут без мебели? :) Тут оказалось чуть больше работы. Нужно было расставить мебель в правильном порядке перекрывания (z-order), что оказалось довольно просто, поэтому была использована структура массивов. Правда, мебель у нас занимала больше одного квадрата, а точнее, до двух квадратов ширины/длины.

Для каждого предмета мебели пришлось определять занимаемую площадь и точку регистрации на карте. Оказалось, что кое-какие конфигурации мебели не перекрываются правильно, от таких пришлось отказаться. Все эти новые детали обсуждались с аниматором, который как раз параллельно рисовал оставшуюся графику.

С мебелью было покончено. Мы взялись за добавление остальных персонажей, которые, в отличие от мебели, двигаются по карте, а значит, изменяют z-order динамически.

Вот уже персонажи ходят по карте, случайным образом меняя направление при столкновении и обходя препятствия. Тут сами собой определились правила игры: убегать от других персонажей-монстров... и, скажем, собирать какие-то предметы. В терминах нашего офиса — «убегать от начальника» (добра от него не дождаться) и «собирать папочки» (такая вот рутинная работа). Кроме того, можно кинуть на карту какие-то бонусные предметы для дополнительных очков.

Создание редактора карт/уровней

Аниматор сел рисовать начальников («толстый и лысый менеджер...»), а я начал изменять алгоритм их движения со случайного на более осмысленный — «ловить работника» (то есть началась разработка простого AI для «монстров»). Для подобных целей обычно используется алгоритм поиска кратчайшего пути, и есть множество вариантов его реализации. Я решил ограничиться упрощенным решением: определять, в каком направлении находится цель, и по нему посылать движение. Этот алгоритм «прокручивался» при каждом новом цикле анимации. Можно было бы определить весь путь до цели, записать его, а дальше просто двигаться по намеченному пути. В этом случае нужно было бы прокручивать алгоритм только один раз для конкретного случая, но в нашей ситуации цель не стоит на одном и том же месте, то есть все время двигается, что свело бы эффективность решения на нет. Нам было нужно постоянно пересчитывать алгоритм, поэтому более простое решение было выбрано прежде всего по соображениям проблем скорости. После доработки алгоритма, особенно в области обхождения препятствий («мебели»), оказалось, что убегать от «монстров» очень сложно. Соответственно, я ограничил «зону видимости» менеджера до определенного радиуса, за которым он двигался в почти случайном порядке, внутри же радиуса — начинал преследовать свою цель. Кое-какие факторы случайности были добавлены для того, чтобы сделать преследование менее предсказуемым.

Величина «радиуса видимости монстров» напрямую влияет на сложность игры. Можно было бы увеличивать ее с переходом на новые уровни (карты), тем самым постепенно повышая сложность игры. Гораздо круче — прибавить нового «монстра» с более высоким уровнем интеллекта (то есть с большим «радиусом видимости»). Этих новых «монстров» можно было бы включать в карты на более сложных уровнях, добавляя новые элементы и новый интерес. Так появился «начальник отдела» — опасный тип в черном костюме и темных очках, убежать от которого довольно сложно.

Итого. Сложность игры можно варьировать, меняя количество папок, которые нужно собрать, количество «монстров» и их типы, а также сам рельеф карты. Факторов вполне достаточно. А главное — общий, но настраиваемый алгоритм AI для всех монстров.

Далее в игру были добавлены дополнительные детали. Появился подсчет очков и бонусы. Были рождены на свет «жизни» — бонусы, которые добавляют общее допустимое количество попыток закончить игру. Также мы придумали «магический бонус», который замораживает монстров на какое-то время. Когда были введены все эти довольно распространенные моменты, игра приобрела свой окончательный вид.

4 создание редактора карт/уровней

Когда детали игры были доработаны, пришла пора создать визуальные редакторы карт, причем в достаточном количестве, а карты стали довольно сложными (на этом этапе они включали три массива, содержащие данные о форме карты, местоположении мебели и персонажей).

Началась работа над редактором карт — это та часть, которую не видят игроки, но которая облегчает генерирование новых карт. На разработку редактора мы потратили времени больше, чем собственно на игру. На этом же этапе была изменена структура movie-клипов. Вся графика, использованная в игре, была перенесена в общий файл-библиотеку. Movie-клипы из самой игры и из редактора карт привязаны к этой одной внешней библиотеке. Теперь при изменении/добавлении графики мы модифицировали что-либо только в одном месте.

В редактор также были добавлены полезные функции, облегчающие быстрое создание карт, например «закрашивание» пола определенными квадратами.

Сами карты/уровни хранятся в отдельных внешних файлах. Есть несколько решений насчет того, в каком формате хранить данные такого типа: 1) как пары переменных/значений, 2) в XML-формате, 3) в форме массивов. Также можно хранить данные во внешних файлах или получать их из базы данных. Я решил использовать массивы во внешних SWF-файлах. Почему? Объяснение простое: хотя массивы содержат больше лишней информации и не столь компактны, как пары переменных/значений, работать с ними легче и быстрее. Кроме того, SWF-формат применяет свое сжатие, что обеспечивает небольшой размер файлов — около 2 Кб для карт The Office Space.

Сохраняя карты в отдельных SWF-файлах, можно подгружать их отдельно по мере перехода с уровня на уровень, благодаря чему мы сможем использовать неограниченное количество карт не увеличивая размер самой игры. Единственный недостаток — необходимость перекомпилировать карты в SWF при изменениях. Я также не использовал базу данных для записи-чтения карт и ограничивался простым copy/paste кода массивов, который генерировал на выходе редактор карт (так как Flash не может записывать файлы на диск самостоятельно). Нужно не забыть сказать несколько слов об интерфейсе нашего редактора: он был сделан для чисто «внутреннего» использования, не было особых причин делать его слишком user-friendly.

5 графика для игры

Что касается графики в целом, изначально мы решили придерживаться пиксельного стиля, то есть имитировать старые компьютерные игры. Во многом на наше решение повлиял вопрос скорости. Векторная графика ресурсоемкая и не дает преимущества в размере при высокой детализации, а в нашей компании многие до сих пор работают на старых компьютерах.

Для игры было создано всего три разных персонажа, около 15-ти разных текстур пола, около 70-ти предметов, таких как мебель, плюс бонусы, включая несколько анимированных. Комбинируя эти элементы, можно было создавать довольно много разнообразных карт, а поскольку каждая картинка экспортировалась в GIF-формат с наименьшим возможным количеством цветов, вся игровая графика занимает меньше ста килобайт.

6 сюжет игры

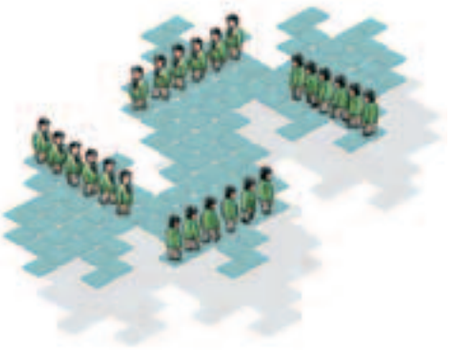
Когда настало время создавать карты для всей игры, естественно, возник вопрос: «Сколько же карт нужно сделать?» Я стал тестировать разные карты, прикидывая среднее время для прохождения одной карты, чтобы определить минимальное игровое время. Параллельно началась работа над интерфейсом: добавились подгрузчики, окна сообщений «Вы стали победителем!», «Вы проиграли!»...

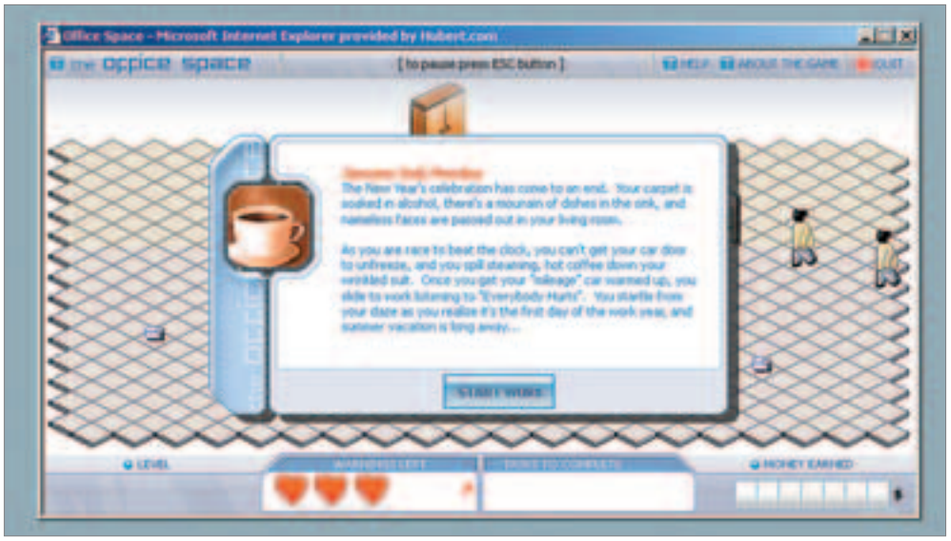
«Как-то чего-то не хватает», — подумал я. Не хватало главного — сюжета. Хотелось, чтобы игра была больше связана с идеей рутины и с прочими элементами офисной жизни, над которыми мы планировали посмеяться.

Так родился простой сюжет игры, и он объединил все части в единое целое. Было решено, что одна карта — это не просто абстрактный уровень, а рабочий день в офисе. Каждый новый уровень — это новый день офисной жизни. Игра начинается второго января, то есть сразу после новогоднего праздника. Главный герой — невзрачный клерк, которому уже давно надоела его работа и который просыпается с мыслью о горе немытой посуды, с усталостью после буйного веселья. С самого утра у него ничего не вяжется: он проспал звонок будильника, дверь машины примерзла, горячий кофе опрокидывается... Грустя, невзрачный клерк вспоминает, что начался всего лишь первый день долгого рабочего года, а каникулы так далеко... Цель игры (то есть цель жизни персонажа) — дожить до каникул, до далекого лета, точнее, до четвертого июля (чтобы добиться символичности, мы выбрали День независимости).

И вот рождается девиз игры: «дожить до каникул», пробиваясь через ежедневную рутину и избегая неприятностей с начальством (просто избегая его вообще). Каждый рабочий день начинается с короткого текста, который сообщает что-то об очередном дне из жизни нашего героя (даже если их никто не читает, они намного лучше обычного «следующий уровень номер такой-то»).

Серия коротких текстов, объединенных общей идеей (по количеству уровней), была создана Erin Van Sickle. Каждый день мы узнаем все больше о нашем главном герое и становимся свидетелями обычной жизни неудачника. Он пробует бросить курить, пытается похудеть, смотрит плохие фильмы в одиночку, от скуки меряет длину своего стола, предвкушает уикенды и справляет любые праздники (часто воображаемые)... И ждет, ждет, когда наконец настанет время каникул.





Дизайн интерфейса игры

Идея с рабочими днями и неделями также была совмещена с перезагрузкой игры: если игрок теряет все жизни, он возвращается не в самое начало, а в первый рабочий день после зарплаты (то есть каждый второй понедельник месяца).

Именно сюжет связал, наконец, все части воедино, придал смысл игре и добавил в нее юмора. Кстати, жюри I.D. Magazine особо отметило этот момент в своих отзывах, так что, видимо, наша идея достойна и хороша, а подход, выбранный нами, стоит перенять, если ты соберешься реализовать собственный проект :).

7 интерфейс игры Довольно часто небольшие онлайн-игры, аналогичные нашей, имеют недостаток — непривлекательный интерфейс. Наверное, после всех сотен и тысяч строк кода, исправления багов и оптимизации алгоритмов, эта часть перестает казаться важной. Главное — чтобы сама игра хорошо работала. Тем не менее игроки очень чувствительны к подобным деталям. В финальных стадиях проекта мы много работали над интерфейсом, стараясь улучшить его общий вид (правда, в игре не так много элементов интерфейса, и многие из них повторяются). Больше времени было уделено оптимизации самой графики и поиску решений, которые не слишком сильно увеличили бы размер игры.

8 оптимизация Финальная стадия разработки игры (параллельно с последними тестами) — это оптимизация. В случае с flash-играми у нас есть два основных типа оптимизации — оптимизация времени загрузки и оптимизация скорости работы (CPU нагрузки).

Оптимизация скорости загрузки предполагает в первую очередь уменьшение размера загружаемых файлов. Чем они меньше, тем быстрее грузятся. Общий размер игры The Office Space (в том числе музыка и звуки) — всего 278 Кб. Такой размер был обеспечен во многом благодаря оптимизации картинок, а точнее — применению GIF-формата с минимизацией количества цветов палитры. Полупрозрачность пикселей почти не использовалась, так как в противном случае пришлось бы применять PNG, который (даже если сжат в самом Flash'e) занимал бы больше места (к примеру, пришлось отказаться от легкого отражения предметов от пола, хотя смотрелось бы довольно красиво). Количество графики также пришлось ограничить. Где возможно — одна картинка применялась много раз. Самый простой пример — одна общая картинка для всех кнопок, с надписями в отдельном слое (как текст или как отдельные картинки). Другие примеры — зеркальное отражение одной и той же картинки, разделение картинок на части, если присутствуют какие-то общие элементы (к примеру, некоторые рабочие места на карте созданы из разных «кусков» рабочих мест и собраны вместе как Lego-модель).

Однако задача не сводится к оптимизации размера файлов. Есть много других способов сократить время ожидания игрока (или создать такую иллюзию). Загрузчик — самый распространенный способ. Расставив загрузчики в правильных местах, можно если не сократить время ожидания игрока, то хотя бы распределить это время по разным этапам игры. Например, предлагаю грузить все карты в самом начале игры и больше не применять никаких загрузок, но тогда начальное ожидание увеличивается. Так зачем это нужно? Грузить все карты с самого начала не имеет смысла, так как есть определенный шанс, что они и не понадобятся (не все игроки пройдут игру полностью сразу). Однако не следует злоупотреблять загрузчиками, если их много, — удобство в таком случае не повысится. В игре The Office Space каждая карта грузится отдельно перед своим уровнем. Так как карты занимают до 2 Кб, время загрузки очень мало. Другой способ — загрузка карт «блоками» (к примеру, можно было бы грузить по пять карт — одну рабочую неделю), что уменьшает количество загрузчиков и обеспечивает некоторый выигрыш в скорости (за счет одного запроса вместо нескольких).

Организация файлов также играет важную роль. Можно оставлять во внешних файлах какие-то данные и подгружать их по мере надобности (скажем, файл с музыкой).

Оптимизация CPU-загрузки решает проблему замедления игры на «слабых» компьютерах (понижение FPS), поскольку Flash довольно требователен к CPU-ресурсам, и это особенно важно для графической прорисовки (Flash не применяет графические оптимизаторы видеокарт).

The Office Space использует целую серию решений, обеспечивающих стабильную скорость даже на очень старых компьютерах. Прежде всего укажу на то, что игра работает на 8 FPS (восемь кадров в секунду), большинство компьютеров способно влиться в такой медленный темп. Несмотря на то, что эта частота кадров слишком низкая (12 FPS рекомендуется как самое низкое значение) и анимация перестает быть гладкой, для The Office Space это решение оказалось вполне органичным, поскольку лишь дополняло стиль старых компьютерных игр. Кроме того, The Office Space работает в низком качестве, без antialiasing ,что никак не влияет на качество картинок, так как все они пиксельные, а шрифты не применяют antialiasing.

Плюс ко всему, я избегал полупрозрачных movie-клипов (_alpha), векторных градиентов и масок — это очень ресурсоемкие операции. Для имитации полупрозрачной тени (скажем, для диалоговых окон) вместо полупрозрачного фона я применял непрозрачную картинку с «дырками» размером в пиксель, расположенными в шахматном порядке, — как в интерфейсах старых операционных систем.

Конечно же, как правило, незачем жертвовать столькими возможностями ради оптимизации. Компьютеры становятся все мощнее, а интернет — быстрее. На самом деле в мое стремление к оптимизации примешался спортивный интерес :). Я разрабатывал The Office Space в основном как эксперимент: «А насколько я смогу все “сжать и ускорить”?» Тем не менее никогда не стоит забывать об оптимизации. Будет очень жаль, если кто-то так и не оценит твою работу только из-за того, что пользователь не прождал лишние пару секунд...

9 а что дальше? The Office Space — довольно простая игра. Эта особенность становится очевиднее сейчас, когда прошло несколько лет после выпуска. Я стремился лишь рассказать о всех фазах разработки одной игры. Даже в такой простой игре, как The Office Space, их оказалось больше, чем можно представить навскидку. Я уверен, что любой, кто изучает Flash, сможет создать аналогичную игру за короткий срок. Главное — найти собственную оригинальную идею, часто такие лежат на поверхности. Используй свои знания в технологиях, чтобы имплементировать и развить идею, а не наоборот. Однако, естественно, универсального рецепта не существует


SPECIAL
DELIVERY

made in Russia

ИТЕРВЬЮ СО STEP CREATIVE GROUP

СТУДИЯ STEP CREATIVE GROUP СУЩЕСТВУЕТ УЖЕ МНОГО ЛЕТ И ИЗВЕСТНА СВОИМ СПЕКТРУМОВСКИМ ПРОШЛЫМ — НЕСОМНЕННО, БОЛЬШОЙ ОПЫТ, УСПЕХ СПЕКТРУМОВСКОГО «ЗН», ЖУРНАЛА SPECTROFON... НО ТЕПЕРЬ НА РЫНКЕ PC МНОГОЕ ПРИХОДИТСЯ ДЕЛАТЬ С НАЧАЛА

АНДРЕЙ КАРОЛИК (ANDRUSHA@REAL.XAKEP.RU)

ЮРИЙ МАТВЕЕВ, ГЕНЕРАЛЬНЫЙ ДИРЕКТОР СТУДИИ STEP CREATIVE GROUP: Поменялось многое: компьютер, платформа, технологии, команда, способы взаимодействия, конкуренция, отношения... Приходится, по сути, на ходу подстраиваться под новые реалии, под новые взаимоотношения. К тому же сейчас для нас геймдев — прежде всего бизнес. Романтический период (когда игры делались чуть ли не в одиночку, на одном вдохновении) давно прошел. Суровые будни бизнеса диктуют свои законы. Что касается выпущенных на PC проектов, о них можно прочитать на сайте **www.stepgames.ru**. А 2005 год стал самым плодовитым для нас — три проекта: детский квест «Путешествие Алисы», римейк «ЗН» и семейный квест по мотивам произведения Гоголя — «Вечера на хуторе близ Диканьки».

СПЕЦ: КАК ПОЯВЛЯЮТСЯ ИГРОВЫЕ ПРОЕКТЫ И ЦЕЛЫЕ КОМПАНИИ ПО РАЗРАБОТКЕ ИГР? ОЧЕВИДНО, ЧТО ТОЛЬКО ДРУЖЕСКИХ ОТНОШЕНИЙ И ЛЮБВИ К ИГРАМ НЕДОСТАТОЧНО.

ЮРИЙ МАТВЕЕВ: За всех ответить сложно. Но на сегодня главное — это желание заниматься этим видом деятельности плюс технологии, люди и средства. Вот те составляющие, что позволяют появляться на свет играм. Технологии нарабатываются годами, компания строится тоже не один год, деньги надо постоянно зарабатывать, привлекать инвестиции и проч. В общем, чтобы из точки А переместиться в точку Б, нужен, скажем, автомобиль, шофер и бензин. Если чего-то нет — останешься на месте. В наш век высоких скоростей и жесткой конкуренции все взаимосвязано. Старый автомобиль даже при наличии лучшего шофера будет плестись где-то в конце потока. Впрочем, так же, как и новый, современный автомобиль с неопытным шофером может потерпеть крушение на опасном повороте. Не говоря о том, что без бензина ни одна машина с шофером не сдвинется более чем на несколько метров (даже если лучший шофер будет толкать ее собственноручно — намек на команды энтузиастов, делающих дома «лучшую в мире MMORPG»).

СПЕЦ: ПО ИДЕЕ, ЕСТЬ ДВА СПОСОБА СУЩЕСТВОВАНИЯ. ПЕРВЫЙ — СОЗДАТЬ НЕКИЙ КОСТЯК, КОТОРЫЙ БУДЕТ ГЕНЕРИРОВАТЬ ПРОЕКТЫ, И ПОД КАЖДЫЙ ПРОЕКТ ИСКАТЬ «НЕГРОВ»-ПРОГРАММИСТОВ, ИМЕЯ КЛЮЧЕВЫХ СОТРУДНИКОВ В ШТАТЕ. ВТОРОЙ — ПОЛНОСТЬЮ УКОМПЛЕКТОВАТЬ ШТАТ ПОСТОЯННЫМИ СОТРУДНИКАМИ. КАКИЕ ПЛЮСЫ/МИНУСЫ? ЕСТЬ ЛИ АЛЬТЕРНАТИВЫ? КАКОЙ ПОДХОД НАИБОЛЕЕ ЖИЗНЕСПОСОБНЫЙ?

ЮРИЙ МАТВЕЕВ: Оба способа имеют право на жизнь. Но здесь, если вернуться к приведенному примеру с автомобилями и шоферами, получается следующее. Представим, что каждый проект — это один автомобиль, шофер и бензин. И если ты, скажем, обладаешь несколькими автомобилями, их можно сдавать в аренду другим шоферам, обеспечивать их бензином и отправлять на трассу. Это как раз вариант продюсерского центра, который позволяет использовать наработанные inhouse-технологии в сотрудничестве с привлеченными сотрудниками, работающими удаленно. Другой вариант — иметь штат отличных шоферов и искать возможность арендовать хороший автомобиль. Здесь возможны варианты с привлечением сторонних технологий, которые позволяют быстрее отправиться в путь. Но и в том, и в другом случае «бензин» необходим. И иногда это становится решающим фактором при принятии решения о разработке того или иного проекта. Есть возможность и комбинирования: часть проектов — inhouse, часть — на привлеченных технологиях. Так, например, поступаем мы. То есть иногда мы делимся своими технологиями с аутсорсерами, а иногда используем готовые технологии своими силами. Вообще все это очень ситуационно и зависит от текущего положения дел. Использование комбинированных методов позволяет подстраховаться от ошибок, и вероятность критической ошибки в масштабах компании снижается.

СПЕЦ: КАКИМ ОБРАЗОМ РЕШАЕТСЯ ТО, КАКОМУ ИГРОВОМУ ПРОЕКТУ БЫТЬ СЛЕДУЮЩИМ? КАК МОЖНО ПРОСЧИТАТЬ ПРИБЫЛЬНОСТЬ ПРОЕКТА И НАСКОЛЬКО ВЫГОДЕН ИГРОВОЙ БИЗНЕС?

ЮРИЙ МАТВЕЕВ: Все зависит от опыта предыдущих проектов. Сильно влияет и конкретная ситуация на рынке. Сегодня популярно фэнтези, завтра — научная фантастика. То же и по жанрам: то стратегии популярность набирают, то шутеры. Безусловно, оценивая и сравнивая продажи различных игр, приходится прислушиваться к мнению потребителей и двигаться в потоке. Впрочем, иногда неожиданный поворот в сторону от «накатанной» дороги позволяет найти более короткий путь к намеченной цели, а иногда (что чаще) — тупик или дорожный столб, о который можно просто-напросто разбиться. Думаю, аналогии понятны. Поэтому прибыльнее всего те проекты, которые оказываются в нужный момент в нужном месте. Иногда три года потраченных усилий, куча пролитого пота и нервов заканчиваются тем, что бюджет не отбивается, предприятие закрывается, а данный вид бизнеса объявляется убыточным.

ДЕНИС ТЕПЛЯШИН — ГЕЙМДИЗАЙНЕР
ОЛЬГА КОМАРОВА — PR-МЕНЕДЖЕР

Бывают и другие ситуации, когда, казалось бы, простая и несложная игра приобретает невероятную популярность и приносит ее создателям ощутимую прибыль. Честно скажу, подобное удается единицам. А в гонке участвуют сотни. Есть, безусловно, и промежуточные варианты, но я специально для наглядности остановился на крайностях, чтобы объяснить, что в нашем деле доля везения, мастерство «штурмана», способность предугадать развитие событий там («за поворотом») играют очень важную роль. Так что и прибыльность, и возможность просчитать ее заранее определяются опытом, способностью предвидеть возможные риски. На самом деле никто не скажет точно, сколько копий того или иного продукта будет продано. Есть такая поговорка у преферансистов: «Знал бы прикуп, жил бы в Сочи»... Чем ты ближе в предварительной оценке к реальному тиражу, тем проще формировать бюджет, оценивать риски, исходя из предположительных «финальных» данных и т.д. В последнее время, находясь в погоне за гарантированной прибылью, многие разработчики ударились в создание игр по фильмам — самый, на мой взгляд, безрисковый «маршрут». Игры по фильмам, которые смотрят люди. Фильмы по играм, в которые играют люди. То есть люди, которые смотрят фильмы, играют в игры. Все взаимосвязано, поэтому любая игра по известной кинолицензии однозначно обречена на коммерческий успех, если сделана вовремя и за разумные (или сверхсжатые) сроки. И, к сожалению, неважно как. Народ купит. Такие вот времена. Казалось бы, парадокс, но неоспоримый факт... Авторским играм в этом потоке гораздо сложнее. Но если они найдут свой путь к сердцам игроков — они обречены и на народную любовь, и на коммерческий успех в конечном счете. А это, согласись, приятнее, чем просто «бабла срубить» на очередной кинолицензии.

СПЕЦ: НЕ КАЖДЫЙ ИГРОВОЙ ПРОЕКТ УСПЕШЕН, БЫВАЮТ И ПРОСЧЕТЫ. КАКИМ ОБРАЗОМ КОМПАНИИ СТРАХУЮТСЯ ОТ ПОДОБНЫХ НАПАСТЕЙ? И ЧТО ДЕЛАТЬ, ЕСЛИ НЕПРИЯТНАЯ СИТУАЦИЯ СКЛАДЫВАЕТСЯ ВНОВЬ И ВНОВЬ?

ДЕНИС ТЕПЛЯШИН, ГЕЙМДИЗАЙНЕР СТУДИИ STEP CREATIVE GROUP: Застраховаться от неуспешных проектов можно разрабатывая одновременно несколько разных. Таким образом, если один из них будет не очень успешным, другие его подстрахуют. А если подобная ситуация складывается снова, то это уже стратегическая ошибка компании. Необходимо произвести переосмысление направления проектов, разобрать ошибки и т.п. В такой ситуации может помочь чтение отзывов игроков и прессы по уже выпущенным проектам.

СПЕЦ: КОГДА ИГРАЕШЬ, СЛАБО ПРЕДСТАВЛЯЕШЬ СЕБЕ, КТО И КАК СОТВОРИЛ ИГРУ. МОЖНО ЛИ ВЫДЕЛИТЬ ОСНОВНЫЕ ЭТАПЫ РАЗРАБОТКИ ИГРЫ? СКОЛЬКО ЧЕЛОВЕК ОБЫЧНО ПРИВЛЕКАЕТСЯ ДЛЯ РАЗРАБОТКИ КРУПНОГО ИГРОВОГО ПРОЕКТА? КТО И ЧТО ДЕЛАЕТ?

ДЕНИС ТЕПЛЯШИН: Конечно, существуют обычные этапы разработки, повторяющиеся при разработке любой игры. Первый и достаточно важный этап — препродакшн. Важность этого этапа часто недооценивается, однако именно на нем формируется видение всей игры: чем она может заинтере-

АНДРЕЙ ШКОЛЬНИКОВ — СЦЕНАРИСТ

совать игроков и может ли она вообще стать успешной. Далее идут различные внутренние технические этапы, такие как: технологическая демо-версия, играбельная демо-версия и т.п. Множество этапов зависят от конкретного проекта. А завершается это все так называемой бета-версией игры, в которую тестеры активно играют и в которой пытаются выловить как можно больше ошибок. После того как все найденные ошибки устраняются, создается так называемый мастер-диск, он отправляется на завод для печати копий игры. Эти копии уже и попадают на прилавки магазинов и полки игроков.

Состав команды различен для разных игровых проектов. Он может варьироваться от пятишести человек для создания небольшого дебютного проекта до нескольких сотен человек для создания мегахитов. Кроме того, не все члены команды работают на каждом этапе создания игры. Некоторые привлекаются лишь в определенные моменты. Из основных специальностей можно выделить: геймдизайнеров, программистов, художников, моделеров, аниматоров, музыкантов. Геймдизайнеры задают то, как будет выглядеть игра, создают диалоги и общее видение. Программисты создают то, что называется движком игры, и, собственно, ограничивают фантазию геймдизайнеров реалиями современных технологий. Художники придумывают, как это все будет выглядеть, рисуют локации, персонажей и т.п. Моделеры создают, собственно, трехмерный мир таким, каким его увидели художники, а аниматоры, в свою очередь, заставляют его жить и двигаться. Ну а без музыкантов не было бы атмосферы и передачи настроения, соответствующего различным эпизодам игры.

СПЕЦ: НЕРЕДКО МОЖНО ВСТРЕТИТЬ «НЕДОДЕЛКИ», КОТОРЫЕ ПОРОЙ НЕОЧЕВИДНЫ НА ПЕРВЫЙ ВЗГЛЯД, ХОТЯ ОТКРОВЕННОЙ ХАЛТУРЫ ТОЖЕ ХВАТАЕТ. КАКОВЫ ПРИЧИНЫ ПОДОБНОЙ НЕКОНДИЦИИ И НА ЧТО РАССЧИТЫВАЮТ ТЕ, КТО ЕЕ ДЕЛАЮТ? ПО ЛОГИКЕ ВЕЩЕЙ, ПРИ ПУБЛИЧНОЙ ОГЛАСКЕ ТАКИЕ ПРОЕКТЫ ЗАВЕДОМО ПРОВАЛЬНЫ И УБЫТОЧНЫ.

ОЛЬГА КОМАРОВА, PR-МЕНЕДЖЕР СТУДИИ STEP CREATIVE GROUP: Ни на что не рассчитывают. Поджатые сроками и выделенным бюджетом, разработчики вынуждены выпускать недоработанный (во всех смыслах) продукт. Особенно это касается «игр по лицензиям», то есть проектов, которые заказываются обладателем лицензии на кинофильм, литературное произведение, когда игру необходимо выпустить точно в срок — к премьере, например. И вот здесь ключевым будет слово «провальный», но не «убыточный». Как правило, выпуская «некондицию», и разработчик, и издатель знают, на что идут. Громкие имена (популярные названия, фамилии актеров, задействованных в озвучивании), маркетинг, реклама порой не только спасают от убыточности, но и обеспечивают неплохие продажи (тот же «Ночной Дозор», «Бой с Тенью», например).

СПЕЦ: НАСКОЛЬКО СИЛЬНО И ЧЕМ ОТЛИЧАЕТСЯ КЛАССНАЯ ИГРА ОТ КОММЕРЧЕСКИ УСПЕШНОГО ПРОЕКТА? ОЧЕВИДНО, ЧТО ЕСТЬ НЕПИСАНЫЕ ПРАВИЛА, КОТОРЫХ ВЫНУЖДЕНЫ ПРИДЕРЖИВАТЬСЯ РАЗРАБОТЧИКИ ИГР В УЩЕРБ ЗАДУМКЕ, ЧТОБЫ ПОЛУЧИТЬ НЕ ПРОСТО ХОРОШУЮ ИГРУ, НО И ПРИБЫЛЬНЫЙ ПРОЕКТ.

ОЛЬГА КОМАРОВА: Классная игра, как было прописано в небезызвестной мегапопулярной стратегии, становится частью анналов цивилизации. Коммерчески успешный проект может быть классным, но вполне может быть и халтурой. Но мы априори говорим о тех разработчиках, которые стремятся сделать высококачественный продукт и уложиться в отведенные средства. Первое неписаное правило — ликвидируем возможности, которые не влияют на геймплей. Например, в нашей игре «Звездное наследие» при нажатии одной из кнопок клавиатуры герой должен был издавать колоритный звук (такие фичи называют пасхальными яйцами — easter eggs). Однако в процессе разработки игры «яичко» кануло в Лету. Далее убираем задумки, которые влияют на геймплей незначительно. Но бывают и случаи провала планирования, когда приходится «отрезать» или «деформировать» коренные фичи проекта. После этого, как правило, следует пересмотр концепции игры.

СПЕЦ: ПОСТОЯННО СОВЕРШЕНСТВОВАТЬ ОДНУ ИГРУ И ДЕЛАТЬ НЕСКОЛЬКО ЧАСТЕЙ ВСЕГДА ВЫГОДНЕЕ — ПО СУТИ, БАЗА ВСЕГДА ОДНА, ЗАТРАТЫ МЕНЬШЕ. НО СЕРИЙНЫЕ ПРОЕКТЫ ИМЕЮТ СВОИ ПРОБЛЕМЫ И СЛОЖНОСТИ. ПРЕЖДЕ ВСЕГО, СЛОЖНО УДЕРЖАТЬ ИНТЕРЕС ГЕЙМЕРОВ, ОНИ ПОСТОЯННО ХОТЯТ КАРДИНАЛЬНО НОВОГО. ОДНАКО И СЕРИЙНЫЕ ПРОЕКТЫ НЕ БЕСКОНЕЧНЫ.

ОЛЬГА КОМАРОВА: Безусловно, «серийное производство» для компании гораздо удобнее. Название игры (тайтл) уже знакомо аудитории, высокое качество первого экземпляра дает гарантию качества второму. Плюс ко всему с большей точностью можно прогнозировать прибыль от продаж сиквела, add-on'а и тому подобного. Главное — не изжить себя. Идеи, идеи и снова идеи. Примеры того, к чему надо стремиться при разработке серии игр, у всех на слуху: HMM, Warcraft, Civilization, Prince of Persia, Sims.

СПЕЦ: ТАК НАЗЫВАЕМЫЙ PRODUCT PLACEMENT (РЕКЛАМА КОМПАНИЙ И ИХ ТОВАРОВ В СЮЖЕТЕ ИГРЫ) ВСТРЕЧАЕТСЯ ВСЕ ЧАЩЕ. С ЧЕМ СВЯЗАН ЭТОТ НАРАСТАЮЩИЙ ИНТЕРЕС? ИГРЫ ДЕЛАЮТ НЕ ПЕРВЫЙ ГОД, НО ТОЛЬКО СЕЙЧАС PRODUCT PLACEMENT СТАЛ ПОПУЛЯРНЫМ.

ОЛЬГА КОМАРОВА: Люди устали. Реклама в газетах, журналах, на радио и (тем более) на телевидении теряет в эффективности с каждым новым текстом, роликом и т.п. Product placement открывает новые возможности воздействия на потребителя. Например, особенно интересна в этом плане интерактивная реклама, то есть когда игрок использует продукт компании прямо в игре (машина, телефон, одежда). Однако есть и подводные камни. Наши игроки уже жалуются на обилие рекламы в «Ночном дозоре», «Адреналин-шоу». Не хочется думать, что скоро реклама и в играх будет восприниматься негативно. Пусть лучше она остается некой изюминкой.

СПЕЦ: КАК ВЫ БОРЕТЕСЬ С ПИРАТАМИ? БОРЕТЕСЬ ЛИ ВООБЩЕ? ПО СТАТИСТИКЕ, ПИРАТЫ ЛИШАЮТ РАЗРАБОТЧИКОВ ЛЬВИНОЙ ДОЛИ ПОТЕНЦИАЛЬНОГО ЗАРАБОТКА... НЕКОТОРЫЕ ИМЕННО ИЗ-ЗА ЭТОГО ВЫНУЖДЕННО ОРИЕНТИРУЮТ СВОИ ИГРЫ НА ЗАПАД ИЛИ ВЫПУСКАЮТ СПЕЦИАЛЬНЫЕ ЛОКАЛИЗАЦИИ. МЕНЯЕТСЯ ЛИ СИТУАЦИЯ К ЛУЧШЕМУ?

ОЛЬГА КОМАРОВА: С пиратами практически не боремся. Максимум, если на рынке встречаем нелицензионную копию, сообщаем номер продавца куда следует. Вообще-то борьба с пиратами — больше прерогатива издателя, чем разработчика.

Что касается ситуации в целом... Очевидно, что небольшими изменениями «порядок» не навести. Уберите пиратов — и большинство игроков окажутся неспособными покупать игры в том же количестве. А этот фактор может существенно повлиять на отечественное игростроение. Дело даже не в том, что останутся лучшие из лучших... Огромное количество идей и ресурсов останется невостребованным.

СПЕЦ: КАКОЙ ИЗ РЕАЛИЗОВАННЫХ ПРОЕКТОВ БЫЛ САМЫМ УСПЕШНЫМ И ПОЧЕМУ? ЧТО ОПРЕДЕЛЯЕТ УСПЕХ ЗАДУМАННОЙ ИГРЫ (ЕСЛИ ПЕРЕЧИСЛЯТЬ В ПОРЯДКЕ УБЫВАНИЯ)?

ОЛЬГА КОМАРОВА: Успех мы понимаем как соответствие результатов целям, а большой успех — превосходство первого над вторым. В этом смысле каждый из наших проектов успешен по-своему. «Звездное наследие 1: Черная Кобра» — римейк получился почти таким, как мы хотели. И игру можно назвать успешной: по предварительному голосованию, по отзывам в прессе, это неоднозначный проект, пройти мимо которого сложно. Причем «Звездное наследие» как раз из того рода игр, когда или «хорошо», или «плохо», а среднего не дано. В нашем случае игру на «отлично» оценили более 70% игроков. Рисованный квест «Вечера на хуторе близ Диканьки» оказался весьма интересным не только для русской аудитории, но и для зарубежной. «Путешествие Алисы» понравилось детям — и это главное.

СПЕЦ: ОТКУДА ЧЕРПАЕТЕ ИДЕИ НОВЫХ ИГР? МОЖЕТ ЛИ, К ПРИМЕРУ, ЧИТАТЕЛЬ НАШЕГО ЖУРНАЛА ЗАРАБОТАТЬ НА ТОМ, ЧТО ПРОСТО ПРИЙТИ К ВАМ С ИДЕЕЙ КЛАССНОЙ ИГРЫ? В КАКИХ СЛУЧАЯХ ДЕЛАЮТ РЕМЕЙКИ?

АНДРЕЙ ШКОЛЬНИКОВ, СЦЕНАРИСТ СТУДИИ STEP CREATIVE GROUP: Идеи, что называется, носятся в воздухе. Наверняка у каждой компании есть «в заначке» минимум полдюжины идей для игр и иногда очень неплохих. Далеко не все из них доживают до реализации. Заработать только на идее, то есть подкинуть ее, а дальше стричь купоны с проекта представляется не очень-то реальным. Увы, но тут наблюдается скорее ситуация «Спасибо, свои девать некуда». Скорее, человек, способный «генерить» интересные идеи, может попробовать стать сценаристом или геймдизайнером.

Что касается ремейков, идея с ними очень простая. Если, например, некая игра «Атака жукоглазых» пользовалась успехом десять лет, значит, хотя бы отчасти волшебная формула была угадана верно и что-то в этой игре заинтересовало людей. Соответственно, можно попробовать повторить успех на новом техническом уровне, сохранив «вечное» и обновив устаревшее. Понятно, что ремейк — всегда рискованный шаг. Во-первых, можно выплеснуть с водой ребенка — дообновляться до того, что ничего интересного в игре не останется. Во-вторых, интересы игроков меняются. Даже самый хороший текстовый квест или псевдографическая roguelike-RPG сегодня вряд ли завоюют массовую любовь. Так что универсальный рецепт тут едва ли возможен.

ЮРИЙ МАТВЕЕВ — ГЕНЕРАЛЬНЫЙ ДИРЕКТОР

СПЕЦ: ПЛАНЫ НА БУДУЩЕЕ И ОСНОВНЫЕ ВЫВОДЫ ЗА ВРЕМЯ СУЩЕСТВОВАНИЯ?

ЮРИЙ МАТВЕЕВ: Планы на будущее — дальнейшее развитие технологий, формирование команды и привлечение дополнительного финансирования. Иными словами, будем рады сотрудничеству как с командами, работающими удаленно, так и с новыми профессионалами в штате нашей компании. А новые проекты будут объявлены немного позже, следи за новостями на нашем сайте — **www.stepgames.ru**

обзор книг

ЧТО ПОЛИСТАТЬ

КАК МЫ ОТБИРАЕМ КНИГИ В ОБЗОР? БЕРЕМ СПИСОК КНИГ, КОТОРЫЕ ЕСТЬ НА СКЛАДЕ (НЕСКОЛЬКО ТЫСЯЧ НАИМЕНОВАНИЙ). ДЕЛАЕМ ВЫБОРКУ ПО ТЕМЕ НОМЕРА, ПОТОМ ОТБРАСЫВАЕМ УСТАРЕВШИЕ ЭКЗЕМПЛЯРЫ И ДУБЛИ. ЛУЧШЕЕ ПОПАДАЕТ В ЖУРНАЛ | **АНДРЕЙ КАРОЛИК**

Если заинтересовался, можешь заказать любую книгу из обзора (по разумным ценам), не отрывая пятой точки от дивана или стула, в букинистическом интернет-магазине os-книга» (**www.osbook.ru**). Книги для обзора мы берем именно там

Быстро и легко. Сетевые игры: в локальной сети, через модем, через Интернет

М.: Лучшие книги , 2003 / Матвеев Е.Е. / 400 страниц
Разумная цена: 120 рублей

Миром правят игры, однозначно! Зачем покупают компьютер домой? Прежде всего, чтобы поиграть. Может быть, просто об этом еще сами не знают :). Прикрываются необходимостью компьютера для работы, учебы и самообразования... В любом случае меганавороченные процессор, видео и звук нужны исключительно для игр.

Искусственный интеллект — как резиновая женщина. Очертания те же, а чувств и эмоций никаких... Так что все и всегда стремятся играть не с железкой, а друг с другом, через интернет или в локальной сети. Данная книга поможет настроить компьютеры для игры по сети в наиболее популярные игры. Быстро и легко :). Даже если не найдешь конкретную игру в перечисленных, будешь действовать по аналогичной логике подключения и шагам, разница только в нюансах и глюках, сугубо индивидуальных.

Создание игр во Flash MX

СПб.: БХВ-Петербург , 2005 /
Мельников С.В. / 304 страницы
Разумная цена: 148 рублей

Flash, наверное, к счастью, имеет собственный развитый язык программирования ActionScript, ближайшей аналогией которого, пожалуй, является JavaScript. Скрипты выполняют всю вычислительную работу и влияют на поведение клипа, нарисованного все в том же Flash. Отсюда возможность создавать интерактивные клипы, что в сочетании с векторной графикой Flash позволяет делать достаточно легкие онлайновые игрушки для web'а. К тому же Flash позволяет сделать вывод фильма не только в SWF-файл (этот формат и используется в web'е), но и в EXE-файл, который будет полноценным приложением для Windows или Macintosh. Чтобы фильм смог работать с файлами (надо же где-то рекорды хранить), понадобится и программа-оболочка на C или Delphi (кто как умеет). Обо всем написано в книге, только намного подробнее :).

Symbian OS. Программирование для мобильных телефонов на C++ и Java2 ME

М.: ДМК Пресс , 2005 / Горнаков С.Г. / 448 страниц
Разумная цена: 247 рублей

Все телефоны делятся на две категории: работающие на прошивке и работающие под управлением операционной системы, как обычная персоналка. Телефоны с операционной системой мощнее и имеют много встроенных программ, плюс (вот самое вкусное!) есть возможность инсталлировать дополнительные программы, написанные на C++, Java2 ME, OPL и даже Visual Basic. Раз есть возможность, обязательно нужно воспользоваться ей :). На телефонах с прошивкой такой прелести уже нет, хотя приложения на Java можно закачивать и на некоторые телефоны с прошивкой, но все-таки это сильные ограничения. Уже совсем «не то».

На рынке операционных систем для мобильных устройств существуют два крупных конкурента: Windows Mobile и Symbian OS. Ось Symbian изначально создавалась для работы на телефонах, а Windows Mobile оптимизирована под мобильные устройства и в основном используется для дорогих девайсов. Symbian же массова, и поэтому она популярнее среди программистов. В книге описаны доступные среды программирования под Symbian (CodeWarrior for Symbian и C++ BuilderX Mobile Stusio) и приведены наглядные примеры работы в этих средах. Рассматриваемые вопросы касаются программной архитектуры операционной системы, основных идиом программирования в Symbian, структуры и создания GUI, локализации, работы с меню, элементами пользовательского интерфейса, графикой, изображениями и т.д. Кому нужно, тот разберется...

DirectX 9 с управляемым кодом. Программирование игр и графика

М.: Издательский дом «КомБук» , 2005 / Том Миллер / 400 страниц
Разумная цена: 310 рублей

Эта переводная книжка (читай, ищи баги :) перевода) написана ведущим разработчиком Managed DirectX, что гарантирует актуальность информации. Что называется, сведения из первых рук. Здесь описаны основы программирования 3D-графики и более сложные манипуляции с mesh-объектами, использование языка шейдеров. В отличие от толстых талмудов типа «Библия разработчика» или «Полный справочник», здесь нет негарантированно полезного справочного материала по всему синтаксису (за ним обращаются к мануалам), зато есть множество наглядных примеров с комментариями. Введение в Direct3D, технология рендеринга, работа с Mesh, программируемые конвейеры, язык шейдеров HLSL, скелетная анимация, звук, 2D- и трехмерная графика, добавление сетевых возможностей и много всего другого по теме. Однако книга рассчитана на тех, кто уже знаком с C# (или Visual Basic). Для остальных очевидно далеко не все...

Текстурирование трехмерных объектов

М.: ДМК Пресс , 2004 /
Флеминг Б. / 240 страниц
Разумная цена: 236 рублей

Речь пойдет о том, как на практике делается рожа Страшилы Фрэнка, который был использован в известном 3D-мультфильме и демонстрировал явную фотореалистичность. Вроде бы что рисовать этому монстру? На самом деле сложное дело. Всякие шрамы, иллюзия освещения, пустота в глазах, ужасные зубы... Автор описывает уникальные методы, которые разработал сам. Проект, описанный в книге, полностью выполнен в Photoshop, так что воспроизвести описанные примеры не составит труда всем, кто работал с этой программой. Результаты сможешь загнать в 3DS Max, Maya, Light Wave или другой пакет трехмерной графики...

Flash MX 2004 и ActionScript 2.0: обучение на примерах

М.: Новое знание , 2004 /
Гурский Д.А. / 446 страниц
Разумная цена: 186 рублей

Если предыдущая книга вдохновила тебя на подвиги, то эта приятно дополнит твою приватную коллекцию по теме создания сценариев на языке ActionScript 2.0. Вся книжка состоит из десяти больших уроков, расположенных по возрастанию сложности. Сначала ты учишься рисовать банальное яблоко, потом елку, осваиваешь простейшую анимацию, к четвертому уроку — целый мультик, добавляешь звук и т.д. «В самом конце книги» ты сможешь сваять полноценный тетрис :). Книга удобна тем, что информация разжевана достаточно подробно и снабжена наглядными картинками. Отличный старт в мире ActionScript, а дальше все с опытом...

Анимация персонажей для игр в реальном времени

М.: ДМК Пресс , 2004 /
Стид П. / 416 страниц
Разумная цена: 347 рублей

Подробный анализ всех стадий работы над анимацией игровых персонажей различной степени сложности, особенно при выполнении типичных низкополигональных персонажей. Описывается создание каркаса Biped, задание весов вручную либо с помощью оболочек влияния, анимация с помощью только ключевых кадров или с использованием захвата движения. Тут же описан и экспорт персонажей на игровой движок. Книгу писал профи своего дела: разжеваны многие тонкости и оптимальные приемы, чего не встретишь в больших изданиях, которые годятся, скорее, как общие справочники. В теоретической части много наглядных примеров и иллюстраций. Вся приведенная практика — для 3DS Max 6 и Character Studio 4.

Звук в играх: технологии программирования

М.: Кудиц-Образ , 2004 / Мейсон МакКаски / 368 страниц
Разумная цена: 187 рублей

Создание игры без звука — трата времени. Чаще всего именно звук создает или дополняет уникальную атмосферу игры. Представь себе фильм, который классно сняли, сделали множество супертрюков, а в конце изгадили никакой озвучкой. Для игр этот пример более чем актуален. Иногда игра средненькая, но в нее хочется играть! Играть, чтобы просто окунуться в ту атмосферу, помедитировать наконец :). «Плюс» этой книги в том, что она знакомит с программированием звука с нуля, — явно идеально для начинающих. Шаг за шагом будешь разрабатывать собственный звуковой движок (а ты думал просто Dance Mashine использовать?).

Основной упор сделан на работу с DirectMusic, плюс ты познакомишься с OpenAL и различными вспомогательными библиотеками обработки и воспроизведения звука. Звук — это не только фоновая музычка, но и динамическое музыкальное сопровождение, и визуализация озвучки, и трехмерный звук, и звуковые спецэффекты...

Программирование трехмерных игр и приложений на Visual C# 2005 и DirectX 9.0c

М.: Жарков Пресс , 2006 / Жарков В.А. / 418 страниц
Разумная цена: нет данных

Одна из первых книг о разработке игр и приложений со звуком средствами нового языка программирования Visual C# 2005 в Visual Studio 2005 и новой технологии DirectX 9.0c. В начале книги немного теории: проектирование методами класса Mesh фигур, широко распространенных на практике, в сплошном и каркасном режимах визуализации, создание вращающихся сложных сетчатых объектов в виде файлов .x, использование эффект-файлов .fx, использование вершинного и пиксельного шейдеров на высокоуровневом языке шейдера (HLSL, high-level shader language), создание скелетной анимации на примере модели человека, воспроизведение звуковых файлов и т.д. Дальше идут методы программирования типичных игр: идет управление автомобилем, автомобиль едет по дороге, на дороге случайным образом появляются разные препятствия; танковые баталии по сети; спортивная гонка на микроавтомобилях и еще несколько других примеров.

Компьютерные игры: секреты бизнеса

М.: Кудиц-Образ , 2004 / Ларами Ф.Д. / 416 страниц
Разумная цена: 164 рубля

Для тех, кто решил заняться играми серьезно, то есть для продажи. Здесь собраны мысли издателей, разработчиков, продюсеров, дизайнеров и владельцев целых студий. Множество примеров из реальной жизни, чтобы показать все сложности и способы их решения на пути к поставленным целям. От работы розничного рынка до финансирования начала проекта и выбора бизнес-модели, подходящей для создаваемой игры. Правда, книжка не учитывает наших суровых реалий, так как переводная, однако все-таки она дает целостное представление хотя бы о том, как работает механизм на Западе. К примеру, одна из главных мыслей: «Не всегда все зависит от качества игры!» Очень важны связи и отношения с издателями, так как, в конечном счете, они помогают игре попасть на прилавки магазинов. Пираты, понятное дело, с распространением справятся и сами :), только вряд ли что-то с их продаж перепадет в твой кошелек...

Хакинг игровых приставок Xbox, Playstation, Nintendo, Atari и Gamepark 32

М.: ЗАО «Новый издательский дом» , 2005 / Гранд Дж. / 624 страницы
Разумная цена: 521 рубль

Под хакингом игровых приставок в данной книге понимаются способы улучшить технические характеристики, чтобы заставить приставку работать лучше, быстрее и по-новому. Описываются приставки от старенькой Atari до суперсовременной Xbox и Playstation. Удобно, что описание «примочек» подробное и с обилием картинок (именно хороших и наглядных картинок), так что подойдет и тем, кто копается в железе не один год, и тем, кто занялся им только «вчера». Сам посуди, персоналку мы уже разгоняли, а Playstation — еще нет... Круто было бы сказать: «А я вчера хакнул свой Nintendo» ;). Тут масса предложений: превратить приставку Atari 2600 в настоящий компьютер, загрузить коды в Playstation 2 напрямую из карты памяти, установить Linux (!) на Microsoft Xbox, избавиться от мерцающего голубого экрана на Nintendo NES, создавать собственные игры для всех существующих приставок и т.д. Перед тем как лезть внутрь любой приставки, прочитай курс молодого бойца (он в начале книжки): какие инструменты понадобятся, где их взять и, главное, как ими пользоваться. Кстати, пора прикупить себе приставку :).

Компьютер для геймера: сделай сам

СПб.: Питер , 2005 / Дж. Дарем-мл. / 191 страница
Разумная цена: 149 рублей

Геймеры — уникальный подвид пользователей компьютера, для них максимальная производительность — вопрос первостепенной важности, причем не всегда обоснованной :). Они вынюхивают любые возможности увеличить производительность системы, пусть даже на считанные проценты или доли процентов. Если ты настоящий геймер, то твой компьютер по определению должен быть самым высокопроизводительным и с самым стильным корпусом в мире (ну, хотя бы в мире твоих друзей). Можно взять много денег и купить брендовый собранный «черный» ящик, и когда-нибудь он морально устареет. От процесса покупки особых эмоций не останется. Не наш метод! Идешь на компьютерный рынок, выбираешь компоненты сам, тащишь домой, берешь отвертку и... Не знаешь, как подобрать железки, чтобы все было в ажуре? Не беда, в книге все расписано достаточно подробно. Теперь ты сборщик и дизайнер в одном лице!

Трехмерное моделирование и анимация человека, 2-е издание

М.: Издательский дом «Вильямс» , 2005 / Питер Ратнер / 272 страницы
Разумная цена: 437 рублей

Геймеры больше всего любят реалистичные игры, благо технологии и железо достигли соответствующего уровня... Однако, как ни странно, создать реалистичного человека в цифровом виде — достаточно сложная задача. Пока еще никому не удалось сделать с помощью компьютерной графики совершенное изображение человека, которое при увеличении было бы абсолютно реалистично и похоже на то, которое мы видим в кинофильмах и на фотографиях. Можно бесконечно восхищаться мельчайшими деталями и точностью представления фигуры человека, но все равно ЭТО невозможно принять за реальное существо. А чем реалистичнее созданное творение, тем выше его ценность в игровом мире. Как правило, нюансы поведения человека теряются при попытке имитировать их. В книге рассказаны многие нюансы того, как с помощью трехмерной графики получить максимум реалистичности. Уроки начинаются с простого моделирования, постепенно усложняются. Есть шанс дойти до последнего уровня :)

Программирование на Python

СПб.: Символ-Плюс , 2002 / Лутц М. / 1136 страниц
Разумная цена: 112 рублей

Python — это свободный (исходные тексты интерпретатора и библиотек полностью доступны для всех, в том числе для коммерческого использования) интерпретируемый («позднее связывание») объектно ориентированный (классическая ОО модель, включая множественное наследование) расширяемый (имеет строго определенные API для создания модулей, типов и классов на C или C++) встраиваемый (имеет строго определенные API для встраивания интерпретатора в другие программы) язык программирования сверхвысокого уровня (динамическая типизация, встроенные типы данных высокого уровня, классы, модули, механизм прерываний/исключений). Вся эта прелесть широко используется и в разработке игр, для программирования различных сценариев и web-приложений, для анимации игр и создания спецэффектов. К примеру, игра Minions of Mirth была написана за 11 месяцев одним программистом, он использовал недорогие и бесплатные технологии, а основным языком в проекте был именно Python, так как он отличается исключительной скоростью программирования задач в области своего применения. Кроме того, для Python имеются замечательные утилиты отладки, позволяющие обнаруживать ошибки не только разработчикам, но и тестерам, а также конечным пользователям

спроси эксперта!

МОЖНО ЛИ СДЕЛАТЬ ИГРУ САМОСТОЯТЕЛЬНО

НА F.A.Q ОТВЕЧАЕТ ЭКСПЕРТ ЭТОГО НОМЕРА АЛЕКСАНДР ГЛАДЫШ — ВЕДУЩИЙ ПРОГРАММИСТ КОМПАНИИ STEP CREATIVE GROUP (WWW.STEPGAMES.RU). ЗАДАТЬ ВОПРОС ЭКСПЕРТУ МОЖНО НА НАШЕМ ФОРУМЕ **forum.xakep.ru** ИЛИ ПО E-MAIL: **expert@real.xakep.ru**

СПЕЦ: ВСЕ, КТО ИГРАЕТ В ИГРЫ, ЗАДАЮТСЯ ВОПРОСОМ: «МОЖНО ЛИ СДЕЛАТЬ ИГРУ САМОСТОЯТЕЛЬНО?»

АЛЕКСАНДР ГЛАДЫШ: Конечно, можно! Только вопрос какую. Quake 4 с ходу не сделаешь. А что-нибудь несложное — запросто. Было бы желание. Не нужно забывать, что главное в играх — все-таки не навороченная графика или реалистичная физика. Если игра сделана с душой, если в нее приятно режутся, игрок легко простит и несложненькую графику, и отсутствие физики...

СПЕЦ: КАКИЕ ОБЩИЕ И СПЕЦИАЛЬНЫЕ ЗНАНИЯ НЕОБХОДИМЫ ДЛЯ НАПИСАНИЯ ХОТЬ КАКОЙ-ТО ИГРЫ? ЧТО СТОИТ ПРОЧИТАТЬ? МОЖЕТ БЫТЬ, СТОИТ ПОХОДИТЬ НА КУРСЫ? НА КАКИЕ ИМЕННО? УЧАТ ЛИ РЕМЕСЛУ РАЗРАБОТКИ ИГР В ВУЗАХ?

АЛЕКСАНДР ГЛАДЫШ: Для создания «хоть какой-то» игры с нуля важны не столько какие-то конкретные специальные знания, сколько навыки по их приобретению. В современных условиях для создания игры на PC больше не нужно знать ассемблер — нужно лишь изучить по имеющимся в изобилии в интернете источникам основы хотя бы того же Flash. Причем функциональности Flash вполне достаточно, чтобы создать свой некоммерческий хит. Нужно иметь желание делать игры, желание творить, желание узнавать что-то новое, желание самообучаться. В российских вузах, насколько я знаю, разработке игр пока еще не учат.

СПЕЦ: КАК ПОПАСТЬ В ЭТОТ ИГРОВОЙ МИР? С ЧЕГО ТЫ САМ НАЧИНАЛ И КАК САМОСОВЕРШЕНСТВОВАЛСЯ?

АЛЕКСАНДР ГЛАДЫШ: Как попал в игровой мир? Просто еще со школы поставил себе цель — стать игровым программистом. Начинал, как и большинство, с малого. Читал документы, форумы, задавал вопросы, писал программки... Здесь главное — накопить опыт, а негативный или позитивный опыт — не так уж важно. Не менее важно стремиться постоянно расти профессионально, получать новые знания. Не имея возможности получить профессиональное образование, люди постигают все набивая шишки.

СПЕЦ: ЛЮБАЯ ИГРА — ЭТО ОБЯЗАТЕЛЬНО ТРУД НЕСКОЛЬКИХ ЧЕЛОВЕК? ИЛИ ЖЕ ЕСТЬ ПРИМЕРЫ ТОГО, КАК ХИТЫ БЫЛИ СДЕЛАНЫ ОДИНОЧКАМИ?

АЛЕКСАНДР ГЛАДЫШ: Если говорить о коммерческих хитах, бестселлерах, то примеры, когда их делали одиночки, конечно, есть. Только большинство таких хитов выпущено уже 10/15/20 лет назад. Увы, романтическая эпоха рыцарей — одиночек игростроения, которые сами продумывали дизайн игры, создавали графическое и музыкальное оформление и писали код, давно прошла. Чтобы сделать современную, большую, коммерчески успешную игру, требуется упорный труд нескольких десятков человек самых разных специальностей: геймдизайнеров, художников, программистов, композиторов… С другой стороны, есть рынок shareware-игр. Такие хиты, как Zuma, Gish, вполне мог бы создать один достаточно талантливый и целеустремленный человек. Однако в современных условиях он бы, скорее всего, заказал звуковые эффекты и музыку, а то и графику на стороне, даже при наличии способностей, необходимых для их создания. Делать все самому — слишком долго, а время, как известно — деньги. А если посмотреть на flash-игры, там такие хиты сплошь и рядом. Вспомни, например, всем известные игры по забросу пингвинов на дальность: безусловный хит за счет удачного аркадного геймплея и хорошей графики. Реализовать геймплей такого уровня на Flash (с сильно упрощенной графикой, разумеется) при наличии определенных навыков — дело максимум пары выходных. Если увидишь, что игра получилась удачной, графику можно «вымучить».

СПЕЦ: КАКИЕ ИГРЫ ДЕЛАТЬ ПРОЩЕ? ДРУГИМИ СЛОВАМИ, С ЧЕГО НАЧАТЬ, ЧТОБЫ НЕ ОКАЗАТЬСЯ В ТУПИКЕ С САМОГО НАЧАЛА?

АЛЕКСАНДР ГЛАДЫШ: Главное — не начинать с заведомо невыполнимых задач. Не нужно выбирать в качестве жанра первой игры MMORPG или FPS. Идеальный вариант — аркады. Самое важное поначалу — как можно скорее получить отдачу, позитивный результат, иначе слишком велик шанс потерять энтузиазм посередине процесса. Каждая законченная игра добавляет значительно больше опыта, чем брошенная на полпути. Пусть делать «тетрисы» и «арканоиды» кажется менее романтичным, чем навороченные онлайн-RPG, зато есть реальный шанс сделать их и получить удовлетворение от выполненной работы, а не разочарование от неудачи. Как и в любой профессии, когда набираешься опыта, понимаешь, что разработка игр — это совсем не то море романтики, какое думалось. Работа над современным большим проектом в игровой индустрии, как и в любой другой отрасли производства, — это крупицы романтики в море рутины.

СПЕЦ: ПОРОЙ СЮЖЕТ ИГРЫ БЫВАЕТ НАСТОЛЬКО СЛОЖНЫМ И ХИТРОСПЛЕТЕННЫМ, ЧТО ОСТАЕТСЯ ТОЛЬКО ИГРАТЬ И ВОСХИЩАТЬСЯ. МНЕ ВСЕГДА БЫЛО ИНТЕРЕСНО, КАК РОЖДАЕТСЯ СЮЖЕТ...

АЛЕКСАНДР ГЛАДЫШ: Сюжет игры, особенно захватывающий, сложный, продуманный, рождается в муках. Даже если представление обо всей игре мгновенно возникнет в голове, как озарение, нужно будет потратить огромное количество сил на формализацию видения игры на бумаге. Причем в детализации, достаточной для того, чтобы остальные члены команды при его реализации не отклонились слишком сильно от оригинальной задумки.

СПЕЦ: ИНОГДА КАЖЕТСЯ: «ВСЕ, ЧТО ТОЛЬКО МОЖНО, УЖЕ ПРИДУМАНО...» НОВАЯ ИГРА — ЭТО ХОРОШАЯ ПЕРЕДЕЛКА ЧЕЙ-ТО ПЛОХОЙ РЕАЛИЗАЦИИ? МОЖЕТ БЫТЬ, ПОЧТИ ВСЕГДА РАЗРАБОТЧИКИ ИГР БУКВАЛЬНО БЕРУТ ЧИСТЫЙ ЛИСТ БУМАГИ И ПРИДУМЫВАЮТ ВСЕ С НУЛЯ?

АЛЕКСАНДР ГЛАДЫШ: Отказ от использования предыдущих наработок и от опыта конкурентов — неразумная трата ресурсов. Если проект технологически основывается на чем-то, что уже было сделано ранее, риск его провала значительно меньше. В программировании отказ от каких бы то ни было наработанных технологий, приемлемых для использования в новом проекте, — грех. Этот же принцип приемлем и для работы сценариста или геймдизайнера. Безусловно, речь идет не о прямом копировании сюжета выпущенной игры. Из опыта прошлого берутся подходы к разработке, приемы, технологии. Допустим, у тебя есть оригинальная геймплейная идея, которая нигде и никогда не использовалась и которая, как тебе кажется, станет изюминкой игры. Возможно, тебе повезет и ты сделаешь хит, откроешь новый подвид игрового жанра, а то и целый новый жанр. Но, с другой стороны, даже с большей вероятностью, тебе может не повезти. И окажется так, что идею не использовали просто потому, что она на практике ничем не привлекательна... Либо, что еще хуже, просто убивает игровой процесс, делая игру неудобной или скучной. Главное — отсеивать такие ситуации как можно раньше, в самом начале проекта. Чем дольше разрабатывается игра, тем

дороже обходятся любые незапланированные изменения. В программировании обычно наблюдается экспоненциальный рост таких затрат. Эта суровая правда жизни обычно отталкивает разработчиков от внесения в проекты экстремальных и оригинальных идей. Гораздо безопаснее двигаться по проторенной дороге, имея возможность оценивать риски разработки с заметно большей точностью.

СПЕЦ: КАК ДОЛГО ОБЫЧНО ПРИДУМЫВАЕТСЯ И РАЗРАБАТЫВАЕТСЯ ОДНА ИГРА? КОНЕЧНО, ЯСНО, ЧТО ВСЕ ЗАВИСИТ ОТ СЛОЖНОСТИ ПРОЕКТА, НО ХОТЯ БЫ С УСЛОВНЫМ ДЕЛЕНИЕМ...

АЛЕКСАНДР ГЛАДЫШ: «Придумать» игру можно за час — вопрос вдохновения. На так называемую стадию «препродакшена» (preproduction — отобранная идея прорабатывается, составляется сценарий, дизайн-документ и проводится планирование) нужно порядка месяца или двух. Однако, как правило, такая работа идет в фоновом режиме, параллельно с работой над текущими проектами. Так называемые «бюджетные» проекты (коммерческие игры среднего качества, предназначенные в основном для укрепления финансового положения компании без риска подрыва ее авторитета) обычно делаются шесть-девять месяцев. Три-четыре месяца разрабатываются проекты класса Trash and Cash, на которых можно заработать большие суммы, выпустив игру практически любого качества, но подгадав под определенные условия (такие игры часто ставят по фильмам и выпускают одновременно с выходом фильма на экраны). Высококачественные же «авторские» проекты, «игры мечты» могут создаваться годами — хватило бы финансирования.

СПЕЦ: ВО МНОГИХ ИГРАХ ЗАДЕЙСТВУЕТСЯ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ. КТО И КАК ПРИДУМЫВАЕТ/РАЗРАБАТЫВАЕТ AI?

АЛЕКСАНДР ГЛАДЫШ: «Придумывает» искусственный интеллект, то есть определяет основные критерии, которым он должен соответствовать, обычно геймдизайнер. В зависимости от жанра, реализация самого искусственного интеллекта либо выполняется самим же геймдизайнером на скриптовом языке (в случаях когда требуется элементарный AI, скажем, как в несложных аркадных играх), либо за дело берутся программисты (в случае если от AI требуется высокая производительность и/или высокий уровень интеллекта). Нередко искусственным интеллектом занимается специально выделенный для этого специалист, поскольку реализация AI в сложных случаях требует больших трудозатрат и специальных знаний. Много сил и времени уходит на тестирование и балансировку с целью придания искусственному интеллекту требуемого уровня сложности: игроку неинтересно играть ни со сверхсложным, ни с глупым до безобразия противником.

СПЕЦ: В ЧЕМ СПЕЦИФИКА РАЗРАБОТКИ ИГР ДЛЯ PC И ДЛЯ МОБИЛЬНЫХ ТЕЛЕФОНОВ? ЧТО ИЗ ЭТИХ ДВУХ ВАРИАНТОВ ПРОЩЕ?

АЛЕКСАНДР ГЛАДЫШ: Проще делать игры для PC, под Windows, где ситуация с технологиями достаточно стабильная и предсказуемая, а железо позволяет развернуться программистской, дизайнерской и художнической мысли по полной программе. В области мобильных платформ сейчас царит хаос: относительно большое количество конкурирующих стандартов, низкая производительность телефонов, заставляющая возвращаться лет на десять назад (по сравнению с технологиями на PC). Помимо этого, игру приходится индивидуально адаптировать под специфические особенности и глюки практически каждой модели сотового телефона индивидуально — у крупных разработчиков хранятся по несколько сотен экземпляров разных моделей телефонов, предназначенных для тестирования. Разработчикам игр не упрощают жизнь и жесткие ограничения операторов мобильной связи, продающих игры конечному потребителю.

СПЕЦ: КАКИЕ СПЕЦИАЛИСТЫ СЕГОДНЯ НАИБОЛЕЕ ВОСТРЕБОВАНЫ В ИГРОВЫХ КОМПАНИЯХ? КАКИЕ ЗНАНИЯ НЕОБХОДИМЫ, ЧТОБЫ ПОЛУЧИТЬ ТУ ДОЛЖНОСТЬ, КОТОРУЮ ХОТЕЛОСЬ БЫ ЗАНЯТЬ?

АЛЕКСАНДР ГЛАДЫШ: В принципе, высококвалифицированных людей любых специальностей всегда не хватает. Даже если в компании и нет вакансий по данному полю деятельности, грамотный руководитель всегда найдет рабочее место для грамотного специалиста. Вообще от человека требуются в первую очередь: способность работать в коллективе, чувство юмора и обучаемость. Работать с человеком, у которого плохо развиты эти навыки, очень сложно. Если у тебя нет опыта, но есть все остальное из перечисленного, то, наверное, самый простой способ попасть в индустрию — устроится тестером в одну из игровых компаний. Это позволит «повариться» в «кухне», лучше понять, в каком именно качестве ты хочешь работать над созданием игр

hard

ВНИМАНИЕ: INTEL ВНУТРИ!

ТЕСТИРУЕМ ПРОЦЕССОРЫ ПОД LGA 775

| ЕВГЕНИЙ ПОПОВ

Test_lab выражает благодарность за предоставленное на тестирование оборудование компании: «НИКС» — Компьютерный Супермаркет (тел. (495) 974-3333, **www.nix.ru**), международному рекламному агентству Image Media, а также российскому представительству компании Asus.

список тестируемого оборудования:

INTEL CELERON D 325	($75)	★★★★★
INTEL CELERON D 346	($88)	★★★★★
INTEL PENTIUM 4 505	($132)	★★★★★★
INTEL PENTIUM 4 506	($135)	★★★★★★
INTEL PENTIUM 4 521	($151)	★★★★★★
INTEL PENTIUM 4 660	($390)	★★★★★★★
INTEL PENTIUM 4 670	($400)	★★★★★★★
INTEL PENTIUM D 840	($520)	★★★★★★★★

введение Вполне возможно, что ты решил обновить своего железного друга или найти замену ему. Конечно, в числе комплектующих, которые ты будешь подбирать, есть и процессор. Если обратить внимание на линейку процессоров от AMD, становится заметно, что в последнее время они сильно подорожали. Их стоимость вплотную приблизилась к ценовому порогу процессоров от Intel. Так что в поисках сердца для твоей машины стоит обратить внимание и на извечные «пентиумы». Не будем агитировать тебя за определенного производителя, мы лишь постараемся определить приоритеты линейки ядер от Intel.

технологии Если полгода назад платформы на основе LGA775 не пользовались должным спросом, то сейчас можно с уверенностью заявить, что это самая успешная и развитая основа для системных плат под процессоры Intel. Благодаря поддержке довольно широкого диапазона процессоров — от слабеньких Intel Celeron D до мощнейших двуядерных монстров — успех обеспечен. В чем же основные преимущества таких платформ? Во-первых, цены на материнские платы с поддержкой LGA775 редко превышают $150 даже за топовые модели, а сами по себе процессоры не имеют легко ломающихся ножек. Во-вторых, LGA775 — это первая в мире десктопная x86-платформа с поддержкой памяти стандарта DDR2.

Замечу, что боксовый кулер, который мы решились использовать в тесте, не очень удачен по сравнению с тем же AMD BOX под 939 сокет. Медный сердечник радиатора покрывает крышку процессора не полностью, а само охлаждающее устройство крепится непосредственно к системной плате. Следовательно, повышается возможность повредить плату и процессор при установке кулера. Новые платформы базируются в основном на чипсетах формата Intel 915P Express и Intel 925X Express, которые в основном лишены поддержки AGP-шины и перешли на более скоростную PCI-Express x 16.

Некоторые процессоры Intel Pentium 4 LGA775 поддерживают 64-битную архитектуру EM64T, а также очень полезную и нужную технологию понижения множителя и напряжения процессора в моменты слабой нагрузки — EIST. Enhanced Intel Speed Step — это прямой аналог технологии Cool'n'Quiet от AMD. Технология Hyper-Threading в процессорах более позднего поколения — само собой разумеющееся.

методика тестирования В тесте участвовали самые разные процессоры — как по производительности, так и по набору функций и технологий, поэтому разумнее всего было бы включить в тест большой набор испытаний. В набор входят такие хиты, как утилита SuperPI версии 1.4 для подсчета числа «пи» до n-го знака после запятой, известный архиватор WinRAR со встроенным тестировщиком для проверки скорости кодирования, а также традиционные бенчмарки 3D Mark 2005 и 3D Mark 2003. Для данных программ были установлены стандартные настройки при разрешении 1024x768 без каких-либо фильтраций и антиалайзингов. В качестве игровых приложений выступили Doom 3 и HalfLife 2. Чтобы снизить влияние видеоподсистемы на производительность, мы использовали низкое (по современным меркам) разрешение 800x600. Кроме того, применялись такие тестовые пакеты CPU, как Hot CPU Tester Pro 4 и Dr. Hardware 2004 версии 5.5.0e. Поскольку в нашем обзоре присутствуют и решения с поддержкой технологии Hyper-Threading, и двуядерный процессор в лице Intel Pentium D 840, было решено проверить производительность подопытных экземпляров при запуске одновременно нескольких бенчмарков в различных вариациях.

тестовый стенд:
Кулер: Intel BOX
Материнская плата: Abit AW8-MAX
Память: AData Vitesta DDR2 533, 2x512 Mb
Видеоплата: ASUS Extreme N7800GTX TOP, 256 Mb DDR3
Винчестер: Maxtor 6Y120L0, 120 Gb, 7200 RPM, Ultra-ATA/133
Оптический привод: SONY DVD/CD-RW CRX300E
Блок питания: Thermaltake Power Station, 520W
Операционная система: Microsoft Windows XP Home Edition SP2

Intel Celeron D 325, 2,53 MHz

ЧАСТОТА, ГГЦ: **2,53**
ЯДРО: **Prescott**
ШИНА, МГЦ: **533**
ОБЪЕМ КЕША L2, КБ: **256**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **нет**
ПОДДЕРЖКА HYPER-THREADING: **нет**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Малайзия**

Самый слабый процессор в нашем тестировании изготовлен в Малайзии и имеет частоту 2,53 ГГц. Из достоинств стоит отметить только низкую цену. Данное решение отлично подойдет для персональных компьютеров, предназначенных для офисной работы и не рассчитанных на ресурсоемкие вычисления, тем более на многозадачную работу.

Что и говорить, в слабых тестовых результатах виноват не только низкий частотный порог, но и кеш второго уровня объемом всего в 256 Кб. Этот процессор не снабдили какими-либо функциями защиты наподобие Execute Disable Bit. О технологиях Hyper-Threading или поддержке 64-битной архитектуры вообще забудем. Зато будут работать стандартные инструкции MMX, SSE, SSE2 и SSE3.

В общем, неплохое бюджетное решение отрабатывает свои деньги, о чем нам и говорят результаты тестирования.

Intel Celeron D 346

ЧАСТОТА, ГГЦ: **3,06**
ЯДРО: **Prescott**
ШИНА, МГЦ: **533**
ОБЪЕМ КЕША L2, КБ: **256**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **да**
ПОДДЕРЖКА HYPER-THREADING: **нет**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Малайзия**

Intel Celeron D 346 изготовлен с применением 0,09-мкм технологии производства на «растянутом» кремнии с числом транзисторов, равным 125 млн. Площадь ядра составляет 112 кв. мм, а в набор SIMD-инструкций входят MMX, SSE, SSE2 и SSE3. Кстати, одной из причин перехода на ядро Prescott стало повышенное тепловыделение: процессор действительно мало греется даже при разгоне. Между тем, множитель CPU, зашитый в процессоре, составляет не много не мало x20.

Как известно, процессоры линейки Celeron всегда отличались дешевизной и скромной производительностью. Несмотря на внушительную частоту кристалла, кеш составляет всего 256 Кб. Технология Hyper-Threading также не поддерживается. Результаты тестирования, соответственно, весьма неутешительные. Никакого эффективного мультизадачного использования и быть не может! Однако подобное положение дел все-таки лучше, чем аналогичный вариант Intel Celeron D325.

Intel Pentium 4 505

ЧАСТОТА, ГГЦ: **2,66**
ЯДРО: **Prescott**
ШИНА, МГЦ: **533**
ОБЪЕМ КЕША L2, КБ: **1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **нет**
ПОДДЕРЖКА HYPER-THREADING: **нет**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Китай**

Единственный процессор в нашем обзоре, изготовленный руками наших китайских собратьев по разуму. Intel Pentium 4 505 отличается от процессоров линейки Celeron D сильно увеличенным объемом кеша второго уровня — целый мегабайт. Имеется поддержка стандартных инструкций SSE, SSE2, SSE3 и MMX. 20-кратный коэффициент умножения CPU порадует покупателей-оверклокеров. Процессор легко поддается разгону, однако серьезно греется, поэтому оверклокер должен действовать предельно аккуратно.

Устаревшее ядро устанавливает сильные ограничения на Intel Pentium 4 505. Прежде всего, пользователь лишен поддержки технологии Hyper-Threading. Отсутствуют даже аппаратная антивирусная защита и EM64T. В целом получаем хороший процессор, но без излишеств.

Intel Pentium 4 506

ЧАСТОТА, ГГЦ: **2,66**
ЯДРО: **Prescott**
ШИНА, МГЦ: **533**
ОБЪЕМ КЕША L2, КБ: **1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **есть**
ПОДДЕРЖКА HYPER-THREADING: **нет**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Коста-Рика**

Еще один явный представитель категории Middle-End, с частотой 2,66 ГГц, но произведенный в Коста-Рике. Сложно говорить о преимуществах этой модели перед Intel Pentium 4 505, так как разница проявляется только в результатах теста. В остальном процессоры полностью идентичны, если не считать такие новшества, как 64-битная архитектура и технология EVP (Enhanced Virus Protection), которые и позволили повысить рейтинг от 505 до 506. Здесь также поддерживаются дополнительные наборы инструкций: SSE, SSE2, SSE3. В остальном стандартно: объем кеша L1 составляет 16 Кб, а объем кеша второго уровня L2 имеется в размерах 1024 Кб. Коэффициент умножения частоты равен 20-ти.

Ясно, что такой процессор вряд ли предназначен для сверхмощных машин, хотя бы потому что технологией Hyper-Threading здесь и не пахнет.

Intel Pentium 4 660

ЧАСТОТА, ГГЦ: **3,6**
ЯДРО: **Prescott-2M**
ШИНА, МГЦ: **800**
ОБЪЕМ КЕША L2, КБ: **2x1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **есть**
ПОДДЕРЖКА HYPER-THREADING: **есть**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Коста-Рика**

Младший брат процессора Intel Pentium 4 670 произведен в Коста-Рике. Отличается от своего родственника только частотой, равной 3,6 ГГц, поэтому разница производительности заметна мало, но она есть. Само собой, пользователь гарантированно получает поддержку все той же технологии Hyper-Threading, 64-битной архитектуры и двухмегабайтный кеш L2. Пользователь также порадуется опциям Enhanced Speed Step и Execute Disable Bit.

В отличие от процессора Intel Pentium 4 670, в данном случае коэффициент множителя CPU уменьшился до x18.

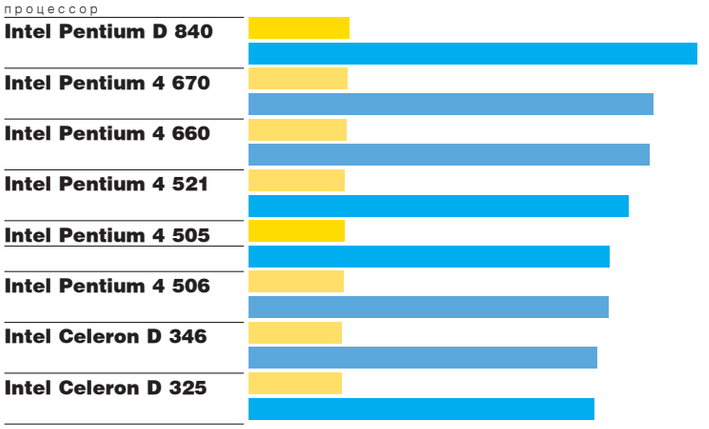
Несмотря на разницу всего $10 между Intel Pentium 4 660 и Intel Pentium 4 670, рекомендуем присмотреться скорее ко второму варианту. Правда, десять баксов бывают не лишние...

Intel Pentium 4 670

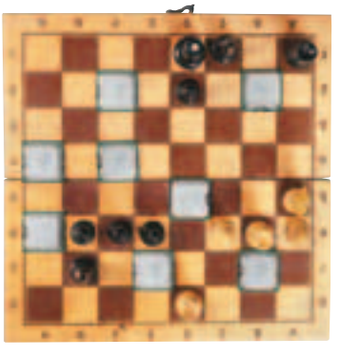
ЧАСТОТА, ГГЦ: **3,8**
ЯДРО: **Prescott-2M**
FSB, МГЦ: **800**
ОБЪЕМ КЕША L2, КБ: **2x1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **есть**
ПОДДЕРЖКА HYPER-THREADING: **есть**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Коста-Рика**

Представляем модель, действительно достойную называться Hi-End'ом. Имеется все, что только можно пожелать: и усовершенствованное ядро Prescott-2M, и двухмегабайтный кеш, и аппаратная антивирусная защита. Плюс ко всему празднику — технологии, ставшие традиционными: Hyper-Threading и EIST. Не стоит забывать и о полной поддержке 64-битной архитектуры. Пусть даже Intel Pentium 4 670 за счет Hyper-Threading дает пользователю мнимую двупроцессорность, он не уступает по производительности венценосному Intel Pentium D 840. Так что пользователь не будет обделен возможностью одновременно играть в игрушки и кодировать видео вместе с аудио с небольшими потерями. За право обладания таким мощным продуктом придется выложить не много не мало $400, что, согласись, равняется покупке бюджетного системника.

3D Mark05 + WinRAR



kb/s	баллы
395	
	6220
365	
	5611
357	
	5563
332	
	5270
313	
	5005
311	
	4997
296	
	4831
287	
	4789



А вот и тест на мультизадачность! Intel Pentium D 840 показал себя во всей красе

Intel Pentium D 840

ЧАСТОТА, ГГЦ: **3,2**
ЯДРО: **Smithfield (2xPrescott)**
ШИНА, МГЦ: **800**
ОБЪЕМ КЕША L2, КБ: **2x1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **есть**
ПОДДЕРЖКА HYPER-THREADING: **нет**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Малайзия**

Этот процессор является одним из самых мощных из линейки Intel Pentium D, доступных на сегодня в продаже. Построен на основе двух ядер Prescott, имеющих индивидуальный кеш и соединенных общей шиной. Ради топовой производительности в этот процессор было вложено 2 Мб кеш-памяти второго уровня. Все технологии, необходимые для такого процессора, на месте — EM64T и EIST готовы к использованию. Имеется и аппаратная антивирусная защита.

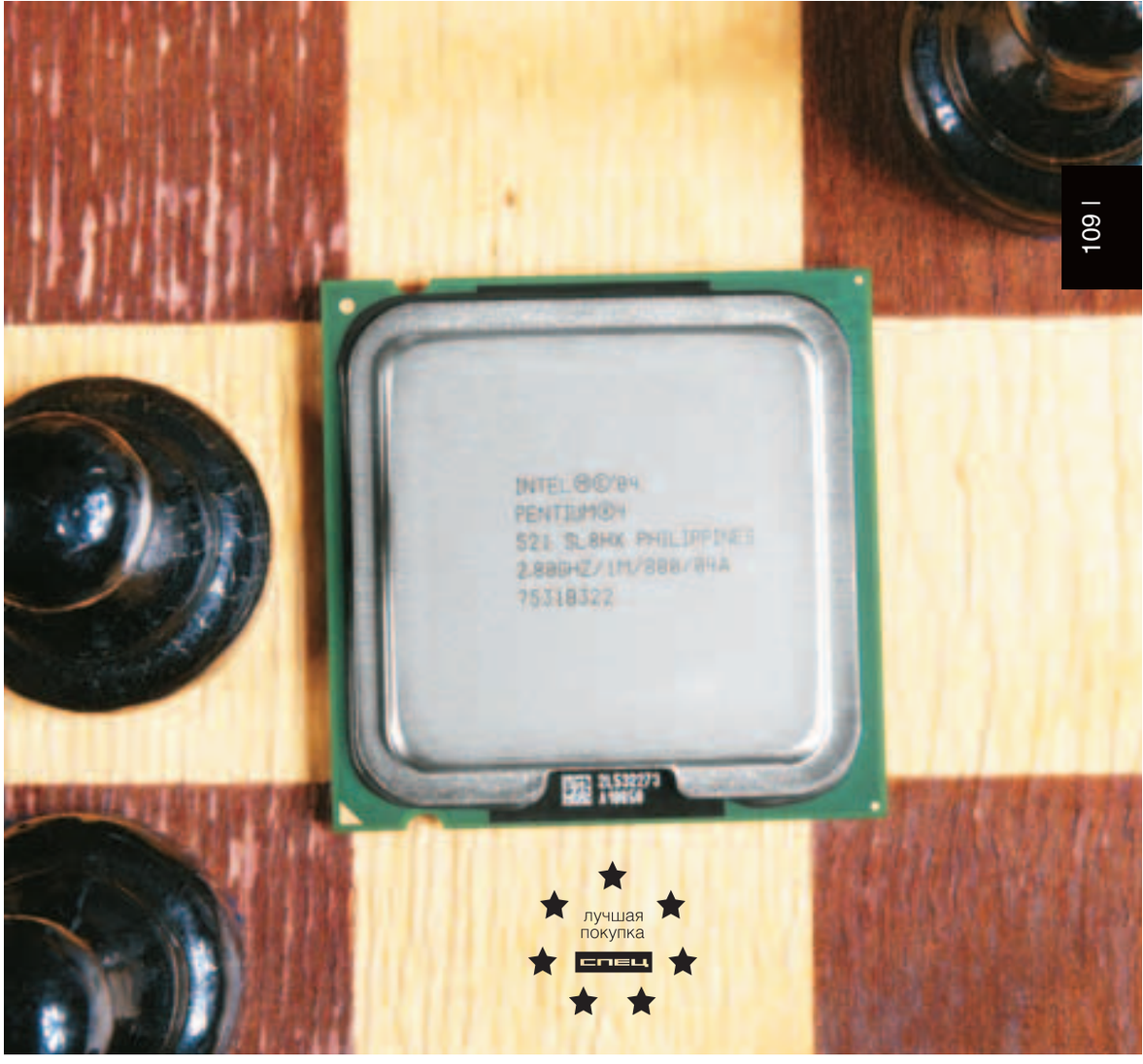
Сожалеем, но поддержка Hyper-Threading отсутствует. Возможность связывать четыре виртуальных процессора имеет только топовый Intel Pentium EE 840. Отмечу не только дороговизну продукта, но и очень высокую температуру при 50% нагрузке, так что для разгона вряд ли подойдет. В то же время частота в 3,2 ГГц уступает многим более дешевым одноядерным собратьям.

Intel Pentium 4 521

ЧАСТОТА, ГГЦ: **2,8**
ЯДРО: **Prescott**
ШИНА, МГЦ: **800**
ОБЪЕМ КЕША L2, КБ: **1024**
ТЕХПРОЦЕСС, МКМ: **0,09**
ПОДДЕРЖКА 64-БИТНЫХ ПРИЛОЖЕНИЙ: **есть**
ПОДДЕРЖКА HYPER-THREADING: **есть**
СТРАНА-ПРОИЗВОДИТЕЛЬ: **Филиппины**

Intel Pentium 4 521 не претендует на лавры двуядерного решения или сверхпроизводительного процессора. Обладая не самой высокой частотой в 2,8 ГГц, выполнен по 90-нм техпроцессу и содержит кеш внушительных размеров — 1024 Кб. При работе процессор практически не греется, так что возможностей для разгона хоть отбавляй, особенно если учесть x14 множитель CPU. Само собой, имеется и все необходимое в комплекте: 64-битная архитектура и технология Hyper-Threading. Функции безопасности будут держать твой процессор под надежной защитой. Имеется также и поддержка стандартных инструкций MMX, SSE, SSE2 и SSE3.

К нашему удивлению, данный процессор не обладает поддержкой Enhanced SpeedStep Technology, что показывает Intel Pentium 4 521 с невыгодной стороны. Так что с разгоном аккуратнее.



ВЫВОДЫ весьма неоднозначные. С одной стороны, в наших руках побывали процессоры, изготовленные с применением таких передовых технологий, как Hyper-Threading, предназначенных скорее для обработки нескольких потоков информации. С другой стороны, нам предлагают соблазнительные бюджетные варианты, которые прослужат долго не только на офисном, но и на домашнем компьютере. Так что тебе придется сделать выбор исходя только из своих потребностей и предпочтений. Если ты игрок по жизни и не мыслишь свою жизнь без виртуальной реальности, то процессоры марки 660 и 670 — твой вариант. Конечно, если ты не ограничен в средствах. Если же твоя работа связана с научной деятельностью или многопоточным проектированием, то советуем приобрести Intel Pentium D 840, который получает оценку «Выбор редакции». Этот двуядерный процессор показывает отличные результаты и в многопоточном тестировании, и в одиночном. Благодаря лучшему соотношению цены и производительности, награду «Лучшая покупка» заслуженно получает процессор Intel Pentium 4 521

Тесты на мультизадачность

3DMark 2005 + WinRAR

процессор	баллы	kb/s
Intel Celeron D 325	4789	287
Intel Celeron D 346	4831	296
Intel Pentium 4 506	4997	311
Intel Pentium 4 505	5005	313
Intel Pentium 4 521	5270	332
Intel Pentium 4 660	5563	357
Intel Pentium 4 670	5611	365
Intel Pentium D 840	6220	395

3DMark 2005 + WinRAR + Super PI

процессор	баллы	kb/s	c
Intel Celeron D 325	4058	203	85
Intel Celeron D 346	4122	216	82
Intel Pentium 4 506	4535	246	78
Intel Pentium 4 505	4532	251	78
Intel Pentium 4 521	4766	275	74
Intel Pentium 4 660	4989	298	69
Intel Pentium 4 670	5003	315	68
Intel Pentium D 840	5719	345	54

Процессорные тесты

Super PI, 1M, c

процессор	c
Intel Pentium 4 670	45
Intel Pentium 4 660	47
Intel Pentium D 840	55
Intel Pentium 4 521	59
Intel Pentium 4 506	59
Intel Pentium 4 505	60
Intel Celeron D 346	62
Intel Celeron D 325	65

WinRar

процессор	c
Intel Celeron D 325	397
Intel Celeron D 346	409
Intel Pentium 4 505	420
Intel Pentium 4 506	423
Intel Pentium 4 521	452
Intel Pentium 4 660	471
Intel Pentium 4 670	479
Intel Pentium D 840	481

Бенчмарки и игры

3DMark 2005

процессор	баллы
Intel Celeron D 325	5084
Intel Celeron D 346	5697
Intel Pentium 4 506	6451
Intel Pentium 4 505	6459
Intel Pentium 4 521	6570
Intel Pentium 4 660	7063
Intel Pentium 4 670	7116
Intel Pentium D 840	7431

Doom3 (800x600)

процессор	fps
Intel Celeron D 325	68
Intel Celeron D 346	75
Intel Pentium 4 505	121
Intel Pentium 4 506	123
Intel Pentium 4 521	130
Intel Pentium D 840	137
Intel Pentium 4 660	148
Intel Pentium 4 670	154

HalfLife2 (800x600)

процессор	fps
Intel Celeron D 325	78
Intel Celeron D 346	89
Intel Pentium 4 505	142
Intel Pentium 4 506	145
Intel Pentium 4 521	150
Intel Pentium D 840	156
Intel Pentium 4 660	163
Intel Pentium 4 670	171

3DMark05+WinRAR+SuperPI

процессор	с	kb/s	баллы
Intel Pentium D 840	54	345	5719
Intel Pentium 4 670	68	315	5003
Intel Pentium 4 660	69	298	4989
Intel Pentium 4 521	74	275	4766
Intel Pentium 4 505	78	251	4532
Intel Pentium 4 506	78	245	4535
Intel Celeron D 346	82	216	4122
Intel Celeron D 325	85	203	4058

Отрыв двуядерного Intel Pentium D 840 от одноядерных собратьев увеличивается при одновременной загрузке сразу трех бенчмарков!

WinRAR

процессор	kb/s
Intel Pentium D 840	481
Intel Pentium 4 670	479
Intel Pentium 4 660	471
Intel Pentium 4 521	452
Intel Pentium 4 506	423
Intel Pentium 4 505	420
Intel Celeron D 346	409
Intel Celeron D 325	397

Super PI, 1M, с

процессор	с
Intel Celeron D 325	65
Intel Celeron D 346	62
Intel Pentium 4 505	60
Intel Pentium 4 506	59
Intel Pentium 4 521	59
Intel Pentium D 840	55
Intel Pentium 4 660	47
Intel Pentium 4 670	45

Вычисление числа «пи» Intel Pentium D 840 снова не блещет

3D Mark 2005

процессор	баллы
Intel Pentium D 840	7431
Intel Pentium 4 670	7116
Intel Pentium 4 660	7063
Intel Pentium 4 521	6570
Intel Pentium 4 505	6459
Intel Pentium 4 506	6451
Intel Celeron D 346	5697
Intel Celeron D 325	5084

Процессоры распределились по рейтингу

HalfLife2 (800x600)

процессор	fps
Intel Pentium 4 670	171
Intel Pentium 4 660	163
Intel Pentium D 840	156
Intel Pentium 4 521	150
Intel Pentium 4 506	143
Intel Pentium 4 505	142
Intel Celeron D 346	89
Intel Celeron D 325	78

Популярный продукт от Valve никак не меняет сложившуюся картину. Количество FPS изменилось, позиции конкурентов — нет

Doom3 (800x600)

процессор	fps
Intel Pentium 4 670	154
Intel Pentium 4 660	148
Intel Pentium D 840	137
Intel Pentium 4 521	130
Intel Pentium 4 506	123
Intel Pentium 4 505	121
Intel Celeron D 346	75
Intel Celeron D 325	68

Качественное убийство марсианских монстров зависит не только от графической платы, но и от процессора. Заметим, что Intel Pentium D 840 занимает не передовые позиции

ВЫБИРАЕМ ДОМАШНИЙ КИНОТЕАТР

Тесты техники, советы по выбору и установке домашнего кинотеатра - ЖК-телевизоры, AV-ресиверы, DVD-плееры, акустика и многое другое.

Режиссер Роберт Лонго
В ролях Киану Ривз, Дина Майер,
Айс-Ти, Такеши Китано
ДЖОННИ-МНЕМОНИК (1995)*

*100% гарантия широкоэкранного анаморфного изображения; звуковые дорожки DD5.1.
DVD-приложения к журналу соответствуют уровню качества ПУЧШИХ мировых изданий!

A-DATA VITESTA DDR2-533

ТЕСТИРУЕМ ПАМЯТЬ СТАНДАРТА DDR2

Евгений Попов

Test_lab выражает благодарность за предоставленное на тестирование оборудование международному рекламному агентству Image Media.

СЕРИЙНЫЙ НОМЕР: **551XR (ADQKD1908)**
КОЛИЧЕСТВО МОДУЛЕЙ: **2**
ОБЪЕМ КАЖДОГО МОДУЛЯ, МБ: **512**
СКОРОСТЬ МОДУЛЯ: **PC2-4300**
ЧАСТОТА ШИНЫ, МГЦ: **266**
ПРОПУСКНАЯ СПОСОБ-ТЬ МОДУЛЯ, ГБ/С: **4,3**
ПРОПУСКНАЯ СПОСОБНОСТЬ В ДВУХКАНАЛЬНОМ РЕЖИМЕ, ГБ/С: **8,6**
ШИРИНА МОДУЛЯ, БИТ: **64**

тестовый стенд:
Процессор 1: Intel Pentium IV 505, 2, 66 GHz (Prescott, LGA775)
Процессор 2: Intel Pentium IV 670, 3, 80 GHz (Prescott-2M, LGA775)
Кулер: Intel BOX
Материнская плата 1: Abit AW8-MAX (чипсет i925X)
Материнская плата 2: ASUS P5GD1 Pro (чипсет i915P)
Видеоплата: ASUS Extreme N7800GTX TOP, 256 Mb DDR3
Винчестер: Maxtor 6Y120L0, 120 Gb, 7200 RPM, Ultra-ATA/133
Оптический привод: SONY DVD/CD-RW CRX300E
Блок питания: Thermaltake Power Station, 520W
Операционная система: Microsoft Windows XP Home Edition SP2

Так случилось, что в наши руки попал комплект памяти стандарта DDR2 от малоизвестного производителя A-Data. Перед тем как положить новинку на операционный стол, разберемся, что такое DDR2-память и в чем состоят ее преимущества.

Прежде всего, память типа DDR2 — это уже не новый стандарт, призванный заменить память DDR в настольной платформе Intel. Принцип функционирования нового поколения, по сравнению с DDR, остался прежним: данные передаются по шине и на восходящей части синхросигнала, и по его спаду, что обеспечивает модулям памяти удвоенную эффективную скорость передачи данных по отношению к частоте шины. Однако в DDR2 есть некие нововведения, которые обеспечили повышение пропускной способности на больших последовательных массивах данных. В основном DDR2 используется в современных системных платах под процессоры Intel.

Память, которую мы протестировали, является типичным представителем DDR2-поколения и поставляется в виде двух идентичных модулей, обладающих двусторонней алюминиевой системой охлаждения. Каждая из планок обладает объемом 512 Мб и способна работать в режиме Dual. Данный набор памяти характеризуется частотой передачи данных 533 МГц, а величина пиковой пропускной способности каждого модуля составляет 4,3 Гбита/с. На каждой планке имеется голографическая наклейка с серийным номером и спецификацией устройства для защиты от подделки. Сами охладители прижаты очень плотно к микрочипам, поэтому перегрев не обеспокоит пользователя.

Эксперимент проводился в два этапа — на двух разных процессорах. На первом «забеге» мы провели тест производительности при использовании схемы таймингов 4-4-4, установленной в BIOS материнской платы по умолчанию, то есть по SPD.

На втором этапе эксперимента был проведен тест стабильности памяти при работе в экстремальном режиме с максимально возможными таймингами 3-3-3 при напряжении 1,9 вольт. Цель эксперимента — найти минимально возможные значения, не приводящие к сбоям в работе подсистемы. Кстати, в процессе испытаний память устойчиво функционировала при задержке CAS#, равной трем, однако данный фактор зависит исключительно от системной платы. Все тесты проводились с помощью программы RightMark Memory Analyzer версии 3.58.

Из полученных данных мы можем сделать следующие выводы. Во-первых, память демонстрирует самую высокую стабильность и производительность на чипсете i925X. Модули достойно функционируют как со стандартными таймингами BIOS, так и в режиме высокой нагрузки, что немаловажно. Во-вторых, судя по результатам, данный набор памяти станет хорошим выбором при сборке современного настольного компьютера на основе процессора от Intel. Конечно, в тестировании многое зависело от материнской платы и процессора, но, несмотря на это, в качестве рассмотренного набора A-Data Vitesta DDR2-533 сомневаться не приходится. Остается только питать надежду на то, что компания-изготовитель не вздумает непомерно завышать цену на продукт и постарается сделать A-Data Vitesta DDR2-533 доступной для потребителя

КОДИНГ XXI ВЕКА

ИЗ СЛЕДУЮЩЕГО НОМЕРА ТЫ УЗНАЕШЬ:

О НОВОМ СТАНДАРТЕ C++
О СЕТЕВЫХ ПРОТОКОЛАХ, WEB-СЛУЖБАХ, XML, СЕТЕВЫХ ПРИЛОЖЕНИЯХ
ОБ ЭКСТРЕМАЛЬНОМ ПРОГРАММИРОВАНИИ
О ТОМ, ЧТО БУДЕТ С DELPHI
ВСЕ САМОЕ ГЛАВНОЕ О .NET
ЗАВОЮЕТ ЛИ JAVA МИР
О ВИЗУАЛЬНОМ МОДЕЛИРОВАНИИ
О VISUAL STUDIO 2005
А ТАКЖЕ ПОЙМЕШЬ, ПОЧЕМУ У MICROSOFT НЕТ БУДУЩЕГО, А У GOOGLE — ЕСТЬ!

СКОРО В СПЕЦЕ:

WINDOWS VISTA. ВЗГЛЯД ИЗНУТРИ. ПОДРОБНЫЙ АНАЛИЗ НОВОЙ ОС ОТ MICROSOFT. НОВЕЙШИЕ ТЕХНОЛОГИИ. УДОБСТВО, БЫСТРОТА РАБОТЫ.

САЙТОСТРОЕНИЕ. WEB-КОДИНГ: НОВЕЙШИЕ ТЕХНОЛОГИИ, ЯЗЫКИ, НЮАНСЫ. ДЕЙСТВЕННЫЕ СПОСОБЫ ПРОДВИЖЕНИЯ САЙТА. ПОРТАЛ СВОИМИ РУКАМИ.

BSD. УСТАНОВКА, НАСТРОЙКА, УПРАВЛЕНИЕ BSD-СИСТЕМАМИ.ИСТОРИЯ. ЕЗОПАСНОСТЬ.

Soft

NONAME

НАИСВЕЖАЙШИЕ ПРОГРАММЫ
ОТ NNM.RU | D()C (DOC@NNM.RU)

Opera 9 TP 2 (en)

Норвежская команда, разрабатывающая широко известный браузер Opera, выпустила Technology Preview 2 девятой версии своего продукта.

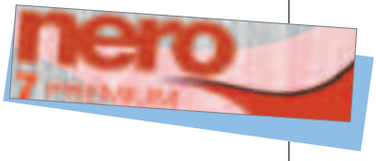
- ПОДДЕРЖКА ВИДЖЕТОВ;
- ВСТРОЕННЫЙ ТОРРЕНТ-КЛИЕНТ;
- ОТОБРАЖЕНИЕ УМЕНЬШЕННОЙ КОПИИ СТРАНИЦЫ В ТАБАХ.

Nero 7.0.5.4

Обновилась седьмая версия самой популярной программы для записи CD — Nero.

Один из лучших пакетов программ для записи на CD-R-, CD-RW- и DVD-диски, но не только. Теперь Nero — это многофункциональный набор средств для работы с данными, музыкой и звуком, фотографиями, телевидением и видео. Поддерживается работа с новыми форматами следующего поколения (Blu-Ray, HD-DVD) и многое другое.

Nero умеет писать CD-ROM (ISO, UDF, UDF/ISO, Boot, Hybrid), Audio CD, CD Extra, Mixed Mode CD, Video CD, Super Video CD, CD Copy. Nero 7 Premium объединил в себе целых 18 программ, включая такие новинки, как система управления телевизором, поиск медиаконтента, средства резервного копирования и восстановления данных. Возможности программ, включенных в пакет, адски широки, и вот только некоторые из них. Nero умеет писать одновременно на несколько приводов, возможна и потрековая запись, и копирование всего CD, имеется непосредственное преобразование MP3- и WMA-файлов в аудиотреки, есть встроенные аудиофильтры и WAV-редактор. Кроме всего прочего, имеются собственные средства для создания обложек CD.



Желающим просто «писать» стоит остановиться на последней версии шестой серии 6.6.1.14 . HD DVD-приводы пока не очень распространены, если вообще продаются где-то, так что отказ от новинки не станет очень большой потерей.

Right Click Image Converter 2.2.2

Программа для быстрой конвертации графических файлов. Весь процесс происходит в один клик. Выделяем нужные файлы (могут быть разных форматов) и в контекстном меню выбираем то, в какой формат будем конвертировать. Поддерживается конвертация из BMP, JPEG, GIF, ICO, PNG, TGA, TIFF, WBMP, PSD, CUT, IFF, DDS, PBM, PCX, PGM, PPM, RAS, XMB, XPM в JPEG, ICON, BMP, PNG, TGA, WBMP, TIFF, XPM, PBM, PGM, PPM.

DriveCrypt 4.20

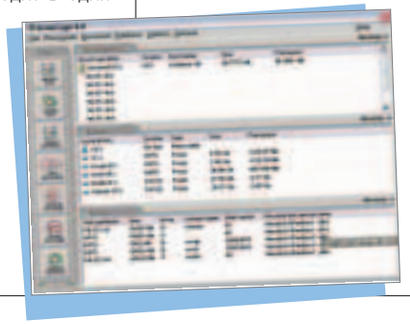
Программа для шифрования данных. Наследница довольно популярной программы ScramDisk. Позволяет не только зашифровать данные (для чего используются несколько алгоритмов, стойких к взлому, — AES, Blowfish, Tea 16, Tea 32, Des и Triple Des), но и скрыть их. Утаивание происходит с помощью метода стеганографии — сокрытие зашифрованных данных в музыкальных файлах, внешне ничем ни примечательных. DriveCrypt зашифровывает как отдельные файлы, так и весь жесткий диск, а также создает виртуальные диски — файлы-контейнеры для хранения данных, которые нужно скрыть от посторонних глаз.

Во всех этих случаях программа работает «прозрачно». Пользователю и программам, работающим с данными, кажется, что процесса шифрования-расшифровки как будто и нет вовсе. А он есть! Процесс протекает автоматически. Кстати, программное обеспечение, которое, как и любые другие данные, может быть размещено на зашифрованном диске, тоже будет работать как ни в чем не бывало.

В виде дополнительных возможностей DriveCrypt позволяет удалять любые данные (восстановить их уже невозможно), поддерживает управление горячими клавишами, блокирует компьютер на время временного отсутствия пользователя. Плюс обладает такой опцией, как два уровня паролей: один — для администратора, другой — для пользователя (полезно, если программа используется в организации).

Система защиты информации DriveCrypt предназначена для защиты конфиденциальной информации на рабочих станциях, персональных компьютерах и ноутбуках. Любая организация (от небольшой компании до международной корпорации) эффективно защитит бухгалтерию, бизнес-планы, списки клиентов — в общем, любую конфиденциальную информацию.

В чем же принцип работы? В процессе чтения данных с жесткого диска DriveCrypt автоматически раскодирует данные до их загрузки. При записи данные автоматически кодируются снова. Процесс кодирования-раскодирования абсолютно прозрачен для пользователя и любого приложения, поэтому пользователям не нужно изменять порядок работы с компьютером. Для дополнительной секретности в процессе работы раскодируются только те файлы, к которым обращается пользователь, — остальная часть диска остается закодированной.



Crypto 2.0

Универсальный шифратор. Использует алгоритм матрицы. С очень высокой скоростью обрабатываются любые файлы любого размера. Расшифровать файл можно только имея матрицу. Возможна генерация отдельной матрицы для каждого файла или для нескольких файлов. В новой версии добавлена упаковка файлов методом LZH.

Total Video Converter v2.4

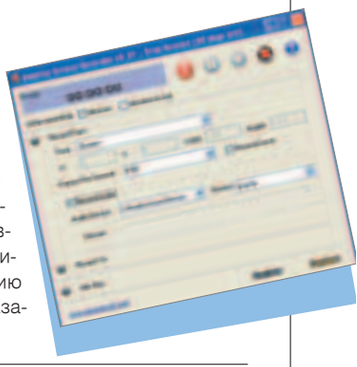
Total Video Converter поддерживает открытие-проигрывание большого количества видео- и звуковых форматов и их конвертирование в различные форматы. Total Video Converter имеет мощный алгоритм преобразования, который позволяет конвертировать файлы быстро и надежно. Форматы: 3GP, MP4, PSP, SWF, FLV, DVD, VCD и т.д.

Quiet Internet Pager (QIP) Build 7810 Alpha

Обновился бесплатный клиент для передачи мгновенных сообщений — альтернатива ICQ. Позволяет подключаться к различным общедоступным серверам.

Внешне QIP во многом напоминает ICQ последних версий. В отличие от стандартной «аськи», имеет множество дополнительных возможностей, таких как просмотр IP-адреса, программы-клиента, времени входа в Сеть собеседника, добавление любого пользователя в свой контакт-лист минуя авторизацию, причем QIP напрямую работает с контакт-листом на сервере. Дополнительные статусы (видимые только пользователям QIP). Удаление своего номера из контакт-листа другого пользователя (работает только для пользователей QIP). Шифрование сообщений. Мультиязычный интерфейс и многое другое.

AimOne Screen Recorder v1.31



AimOne Screen Recorder позволяет записывать происходящее на экране или на его участке в видеофайл (AVI или WMV).

Также может быть записан звук с любого источника. Программа особенно полезна для подготовки обучающих курсов, демонстрационных программ. По сравнению с uvScreen Camera, как мне показалось, тормозит меньше.

Miranda IM MDpack '07.02.06

Собственноручно собранный IM.

Умеет палить invisible'ов, удалять тебя из чужих списков, грузить аватары. Настроен удобно! Версия будет постоянно обновляться.

Unlocker 1.7.9

Вышла новая версия полезной утилитки. Unlocker позволяет «выкидывать» файлы и директории, которые не поддаются удалению обычным способом (Cannot delete file: Access is denied и т.п.). Интерфейс минималистичен: просто добавляется еще одна опция в контекстное меню. На мой взгляд, незаменимая вещь, очень сильно помогает не портить нервы :)

Light Alloy 3.5.5944

Программа для воспроизведения видео- и звуковых файлов под Windows. Проста в управлении, но содержит множество дополнительных настроек, то есть подходит как новичкам, так и профессионалам. Проигрыватель имеет небольшой размер и оптимизирован для быстрого запуска и минимальной загрузки системы.

Jet Audio 6.2.5 Plus VX

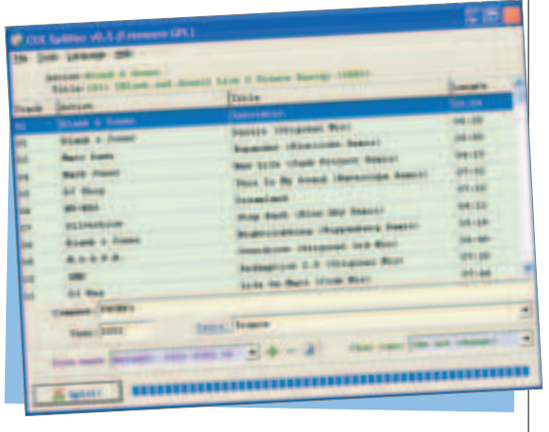
Популярная программа для воспроизведения аудио-видео. Поддерживает все форматы мультимедиа, редактирование тегов, синхронизацию с интернет-БД, скины, звуковые эффекты, запись CD и многое другое.

New Weather v1.0

Взбурлило множество разных споров по теме Weather Watchers, было предложено и несколько альтернативных программ... Честно скажу, я поддержал российского производителя. Программа для получения прогноза погоды по выбранному городу и курса различных валют — в режиме онлайн через интернет. Информер находится в системной панели возле часиков и показывает текущие сведения при наведении курсора мышки. Прогноз погоды для более 100 городов России, СНГ и Балтии. Курсы ЦБ РФ мировых валют и драгоценных металлов. Бесплатна и компактна...

CUE Splitter v0.5

Программка для нарезки дисков, рипанных non-stop'ом (одним большим MP3-файлом), с использованием CUE-файла. Стабильно работает с VBR-битрейтом



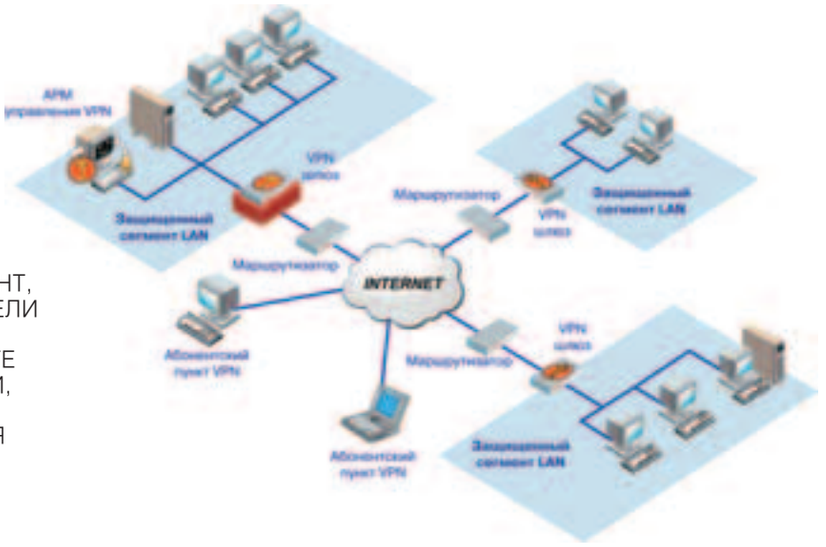
GIF Movie Gear 4.1.0

Если ты хочешь создать анимированные GIF, но боишься засорить свой компьютер такими монстрами, как Adobe Photoshop и ему подобные, то GIF Movie Gear создан для тебя.

Обладая маленьким размером (962 Кб), программа позволяет создавать полноценные анимированные GIF, а также оптимизировать их размер. В отличие от многих утилит похожего назначения, имеет простой интерфейс и позволяет выполнять задания быстро и без лишних действий. Программа позволяет открывать и использовать в качестве кадров для создаваемой анимации файлы следующих типов: GIF, AVIi, BMP, JPEG, PNG, PSD. Можно также работать со значками (ICO) и анимированными курсорами (ANI).

ADMINING: УСТАНОВКА VPN-СЕРВЕРА НА WINDOWS 2003 SERVER

НАКОНЕЦ-ТО НАСТАЛ ТОТ ДОЛГОЖЕЛАННЫЙ МОМЕНТ, КОГДА СЕТЬ НАСТРОЕНА, ДОВОЛЬНЫЕ ПОЛЬЗОВАТЕЛИ ЛОМЯТСЯ НА СЕРВЕР НАПРЯМУЮ И ПОЛУЧАЮТ ТО, ЧТО ДУШЕ УГОДНО. ОДНАКО ВСЕ ЧАЩЕ В ИНТЕРНЕТЕ ПОЯВЛЯЮТСЯ ВНУТРЕННИЕ ДОКУМЕНТЫ КОМПАНИИ, А НАЧАЛЬНИК НЕ МОЖЕТ ПОНЯТЬ, ПОЧЕМУ ОТЧЕТ НА ПРОДАЖУ, СДЕЛАННЫЙ ТОЛЬКО ЧТО, ПОЯВИЛСЯ В ОТКРЫТОМ ДОСТУПЕ СОВЕРШЕННО БЕСПЛАТНО. ЗА ЭТО ОН БОЛЬНО БЬЕТ ПО ЧЬЕЙ-ТО ГОЛОВЕ И КОШЕЛЬКУ |ЕВГЕНИЙ AKA SATURN (SATURN@LINKIN-PARK.RU)



средство от головной боли Удары судьбы стали нестерпимы? Корпоративная сеть все больше напоминает информационное решето, доступное всем? Пора задуматься о защите информации. Существует масса способов предотвратить утечки данных и несанкционированный доступ. Можно, например, зашифровать все файлы и открывать их только по требованию конкретного человека. Или использовать ставшую модной электронно-цифровую подпись (ЭЦП). Казалось бы, проще простого: подписываем файл средствами ЭЦП и пересылаем адресату. Он в свою очередь проверяет подлинность и расшифровывает документ. Но как объяснить это дяде Васе, которому компьютер нужен, чтобы «докУменты печатать»? И как заставить себя выполнять сотни операций «зашифровка-расшифровка» в день, когда друзья принесли ящик отличного пива? К счастью, на помощь способно прийти средство под названием VPN (Virtual Private Network), оно реализовано в нашем любимом Windows 2003 server.

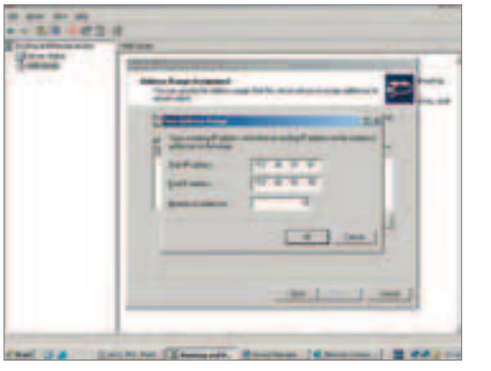
виртуальная? частная? что это такое? Строго говоря, VPN — это частный случай оверлейных сетей (Overlay Network) — логических сетей, создаваемых поверх другой сети (например интернет). Узлы оверлейной сети (и VPN в частности) могут быть связаны не только физическим соединением, но и логическим. Так что существует возможность создавать множество логических сетей одна поверх другой, добиваясь практически любого уровня защиты данных. За счет шифрования средствами VPN создается канал обмена информацией, закрытый от посторонних.

Практический смысл виртуальной сети состоит в следующем. Перед отправкой IP-пакета VPN-агент (всего лишь средство, реализующее алгоритмы шифрования и транспорта, а не сотрудник ГРУ) шифрует его целиком, включая заголовок, формирует новый заголовок (проводит инкапсуляцию), где указывает не физический адрес получателя, а координаты его VPN-агента. Этот механизм, во-первых, позволяет скрыть от злоумышленника реальные IP-адреса пользователей. Во-вторых, так достигается высокий уровень абстракции от аппаратных средств, используемых в сети.

При получении пакета проверяются сведения о VPN-агенте отправителя. В случае если такая информация отсутствует, пакет игнорируется. Если заголовок поврежден, его постигнет та же участь.

Как видишь, VPN — достаточно простое и в то же время надежное средство защиты информации, передаваемой по сетям общего пользования. Вот почему эта технология завоевывает все большую популярность и производители программного обеспечения постоянно упрощают ее настройку. Более того, сейчас многие крупнейшие вендоры (Cisco, Intel) встраивают поддержку VPN в маршрутизаторы и другие сетевые устройства.

Так выглядит сеть с VPN. Благодаря высокому уровню абстракции от аппаратных средств, связь между двумя сетями может быть представлена как связь между двумя компьютерами

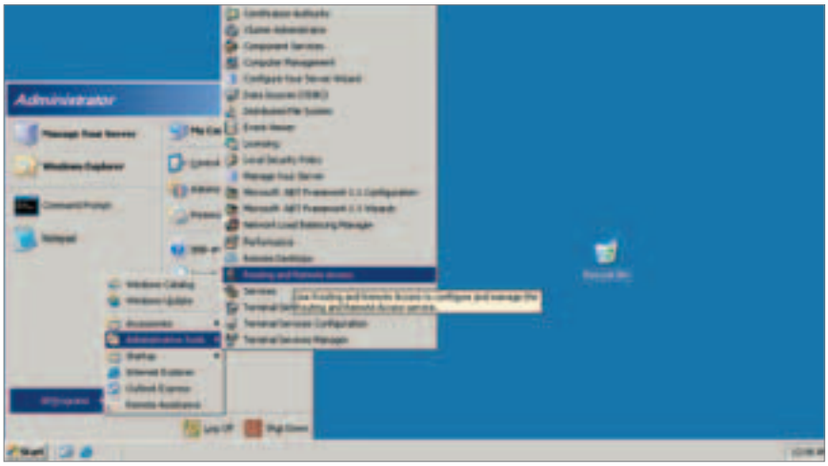


IP-адреса можно назначать с помощью внутреннего DHCP, этим же может заняться сам VPN-сервер

даешь vpn в родном колхозе! Итак, наша задача — сконфигурировать VPN-сервер таким образом, чтобы он умел «узнавать» пользователей и предоставлять им доступ к внутренней сети на основе прав. Приступим...

Из известного всем и вечно здравствующего меню Start выбираем Administrative Tools-> Routing and Remote Access. Если все произошло правильно, видим оснастку Routing and Remote access (проще говоря, окно с названным заголовком). Находим здесь имя сервера и щелкаем по нему правой кнопкой мыши. В появившемся меню нас больше всего интересует пункт со славным названием Configure and Enable Routing and Remote Access. После выбора этого пункта должен запуститься мастер Routing and Remote Access Server Setup Wizard.*

* Стоит сказать, что есть другой путь к сердцу мастера: Start-> Manage Your Server-> Add or remove a role. Появляется окно с возможными ролями сервера, здесь выбираем пункт Remote Access/VPN server.



С чего начинается VPN-сервер? Правильно, с пункта Routing and Remote Access (находится в Administrative Tools)

Теперь, ориентируясь на подсказки мастера, выбираем нужные пункты в каждом экране. Начинаем! В окне Common Configurations выбрать Virtual Private Network (VPN) server — таким образом мы сообщаем о намерении установить-таки сервер VPN :). Далее нас любезно просят подтвердить, что на сервере установлены все протоколы, нужные для нормального функционирования службы. Если в окне протоколов есть TCP/IP, смело нажать подтверждение (Yes, all of…). Далее выбрать сетевой адаптер, который связан с интернетом или, если нужно создать VPN во внутренней сети, адаптер, подключенный к сети. Теперь определимся с IP-адресацией нашей защищенной сети, на этом остановимся подробнее.

Существует два способа назначить IP-адреса клиентам VPN: 1) назначить с помощью основного DHCP-сервера (его настраивали в предыдущем номере); 2) их назначает VPN-сервер. В первом случае ты всего лишь «натравишь» VPN-сервер на его DHCP-собрата, проблем возникнуть не должно. Чтобы адреса назначались VPN-шлюзом (по-моему, этот вариант предпочтительнее), выбираем пункт From a specified range of addresses (экран IP Address Assignment). После нажатия заветной кнопки Next выбираем диапазон адресов, распределяемых VPN-сервером (например, Start IP: 172.16.0.10, End IP: 172.16.0.200). После выбора адекватного диапазона адресов нас будет ждать еще один, последний экран. Смело отвечаем «No, I don't...» и таким образом завершаем установку VPN-сервера! В последнем окне мы отказались от аутентификации с помощью сервера Radius (Remote Authentication Dial-In User Service). Отныне эта процедура будет производиться при помощи Active Directory. Как видишь, все просто. Теперь все легитимные пользователи смогут подключаться к интернету (или к внутренней сети — по желанию) при помощи защищенного канала и использовать для авторизации стандартную пару «логин-пароль». Таких пользователей ты сможешь контролировать средствами Active Directory.

Альтернативный путь к VPN лежит через Start-> Manage Your Server-> Add or remove a role

Напоследок напомню, что для каждого клиента необходимо создать VPN-соединение, в свойствах которого обязательно указать адрес сервера и, в общем случае, убрать требования по шифрованию заголовков

Е-МЫЛО

ПИШИТЕ ПИСЬМА!
SPEC@REAL.XAKEP.RU | НА ПИСЬМА ОТВЕЧАЛ SKYWRITER

ОТ: Андрей [crash@e1.ru]
ТЕМА: впечатление от С.Хакера (02/2006)

здравствуйте, spec.

1 БЛАГОДАРНОСТЬ: ЧИТАТЬ СТАЛО ИНТЕРЕСНЕЙ, ЖУРНАЛ ЗА ФЕВРАЛЬ 2006 ГОДА ПРОСТО ПОРАДОВАЛ.

2 ЗАМЕЧЕННЫЕ НЕДОСТАТКИ (МОЕ МНЕНИЕ): МАЛО МАТЕРИАЛА ДЛЯ АДМИНИСТРАТОРОВ. ОН, КОНЕЧНО, ЕСТЬ. НИКТО НЕ СПОРИТ, НО НОСИТ БОЛЕЕ ОБЗОРНЫЙ ХАРАКТЕР, ХОТЕЛОСЬ БЫ ВИДЕТЬ РАЗОБРАННЫМ ЕГО ДО ПОСЛЕДНЕЙ КОСТОЧКИ.

3 ОЧЕВИДНОЕ-НЕВЕРОЯТНОЕ: СОДЕРЖАНИЕ CD (02/2006) РАСПЕЧАТАНО НА ОТДЕЛЬНОЙ СТРАНИЦЕ, НО ВЫРЕЗАТЬ ЕГО И ВСТАВИТЬ В КОРОБКУ CD/DVD НЕ ПОЛУЧИТСЯ. ИЛИ ЭТО ТАКАЯ СВОЕОБРАЗНАЯ РУБРИКА ПРИКОЛЮХА/РЕБУС/ЗАПАДЛОСТРОЕНИЕ?

4 ДАЕШЬ РЕКЛАМУ В МАССЫ: А ИМЕННО. НА CD/DVD НАВЕРНЯКА МОЖНО ВПИХНУТЬ КРАСИВЫЕ ОБОИ/СКРИНСЕЙВЕРЫ И ПР. С ЛОГОТИПАМИ С.ХАКЕРА. ПОСТЕР/КАЛЕНДАРЬ НА СТЕНУ ТОЖЕ БЫ НЕ ПОМЕШАЛ.

ОТВЕТ:

Привет, Андрюха!

1 СПАСИБО ТЕБЕ ЗА БЛАГОДАРНОСТЬ, ДОБРЫЙ ЧЕЛОВЕК! МЫ ВСЕ ДРУЖНО НАД НИМ КОРПЕЛИ, СТАРАЛИСЬ И ТРУДИЛИСЬ. ОЧЕНЬ-ОЧЕНЬ РАДЫ, ЧТО ХОТЬ ОДИН ЧИТАТЕЛЬ ОЦЕНИЛ НАШИ УСИЛИЯ... (ДОКТОР ПРОСЛЕЗИЛСЯ.)

2 ИМЕННО ДЛЯ ЭТОГО МЫ И ОРГАНИЗОВАЛИ РУБРИКУ «АДМИНИНГ». ЧИТАЙ, ПРЕДЛАГАЙ ТЕМЫ НОВЫХ СТАТЕЙ — МЫ ЛИШЬ РАДЫ ДАРИТЬ ЗНАНИЯ!

3 ДА, ЭТО ТАК СПЕЦИАЛЬНО СДЕЛАНО! ПОМНИШЬ, В «ХАКЕРЕ» БЫЛИ ТАКИЕ КОНКУРСЫ: «ЗАХАКАЙ МЕГАПРОГУ ВАСИ ПУПКИНА И ПОЛУЧИ ПРИЗЫ»? У НАС ТО ЖЕ САМОЕ. ЖДЕМ ВЫКРОЕК ;).

4 ЧЕСТНО ГОВОРЯ, У МЕНЯ, КАК У РЕДАКТОРА CD, ТАКАЯ ИДЕЯ ПОЯВИЛАСЬ ДАВНО, НО СКОЛЬКО Я НИ ПЫТАЛСЯ ЗАПИХНУТЬ КАЛЕНДАРЬ В ДИСК — НУ НИКАК ОН ВО СТОЛЬКО РАЗ НЕ СВОРАЧИВАЕТСЯ :(. ДА ЧТО ТАМ КАЛЕНДАРЬ! ДАЖЕ ПОСТЕР НЕ ЛЕЗЕТ! УМА НЕ ПРИЛОЖУ, ЧТО ДЕЛАТЬ...

5 И ВСЕ-ТАКИ УДАЧ ТЕБЕ, АНДРЕЙ.

ОТ: КолДуН [tranceriver@mail.ru]
ТЕМА: яЗык :)

Hi there spec, может хватит пальцегнутия? Информации вы даете достаточно, но продираться через дебри жаргонизмов, чтобы достать крупицу нужной информации это что-то :). Так что чтение вашего журнала это тоже в какой-то мере хак :).

ОТВЕТ: Ы, в натуре, здорово, КолДуН! Чисто, о каких пальцАх ваще идет речь?! Мы за чистую (Понял? Чиста-ю) информацию! И никаких жаргонизмов. Наши хакерские братаны и знать не знают, что такое «жаргонизмы»!

ОТ: DEViANCE [DEViANCE1989@mail.ru]
ТЕМА: помогите, пожалуйста

Вас беспокоит Макс. У меня к Вам огромная просьба, уважаемый ХАКЕР. У нас на Сахалине в городе Поронайск есть всего один провайдер **www.card.sakhalin.ru** для пользователей Сахалина. Так как провайдер один и люди не могут выбирать себе провайдера, то он поднимает цены на инет (хотя качество соединения очень плохое, в нашем городе самая большая скорость 33.6 kb/s и то редко). Помогите пожалуйста. Мне нужно чтобы вы взломали этот сервер **www.card.sakhalin.ru**, там защита я думаю очень слабая. Большая просьба взломайте этот сервер и подскажите мне пароль на инет (так как я не горю желанием платить 30 рублей в час за такое соединение).
ЗАРАНЕЕ ОГРОМНОЕ ВАМ СПАСИБО, УВАЖАЕМЫЙ ХАКЕР!!

ОТВЕТ: Нет, Макс нас не беспокоит. Немножко ноги перед сном чешутся, а Макс — нет, не беспокоит, слава Богу... Впрочем, как говорили классики, не зарекайся, да? :)
Кстати, *у нас* на Сахалине тоже только один провайдер. Да и с ценами та же проблема. В общем, мы с тобой товарищи по несчастью. И вот тогда, когда мы до конца осознали свое ничтожество, мы обратились к Великому Хакеру! Припали десницами к его ногам и воззвали к Халявному Инету. Ну, тут, как полагается, гром, молния и прочие катаклизмы. Глядь, а прямо перед носом лежит табличка с высеченными буквами: «Login: fsb password: fsb123».
Ввели мы их в компьютер и горя не знали, всем Порнорайском пользовались халявой, данной Хакером. Но недолго счастье народное по городам и весям гуляло — пришла беда в Порнорайск: угрюмые дяди с надписью «ФСБ» на спинах приехали и увезли всех порнорайчан туда, где не светит солнце, вручили им ледорубы и стали нещадно заниматься эксплуатацией. Долго и горько плакали порнорайчане, но делать нечего — УК суров, и над всеми нами довлеет «неотвратимость наказания». Тут и сказке конец.
Сказка ложь, да в ней намек — добрым молодцам урок ;).

От: Smolsky Vycheslav [SmolskyVV@mail.ru]
Тема: не раскрыта
Здравствуйте, авторы любимого мной журнала!!! Пишу Вам первый раз и хочу задать такой вопрос. Я сам из Ставрополя. Подключаюсь к инету через мобильник (Билайн GPRS). Так вот, я хотел бы узнать. Если я захочу что нибуть взломать, то как меня могут выследить или поймать??? (SIM-карта зарегистрированая не на меня, вообще даже не знаю на кого.) Прошу Вас, если Вас это не затруднит, ответьте мне. (И еще один маленький вопросик: вообще хватит ли скорости моего соединения для такого дела??) Заранее огромное спасибо!!! Желаю Вам удачи!! ВАШ ЖУРНАЛ ПРОСТО СУПЕР!!!
GooD...GooD...GooD…

Ответ: Здравствуйте, читатели любимого вами всеми журнала! Мы, кстати, тоже обратили внимание на то, что ты нам еще не писал: у нас уже, так сказать, сложившаяся аудитория и мы знаем в лицо всех многочисленных читателей. Как КГБ.
Что касается надежности взлома через мобильник, это 100% надежно! Ну, представляешь, сидишь ты где-нибудь в глухом лесу с ноутбуком. Кто тебя там будет искать? Там же холодно и страшно :(...
Впрочем, я уверен, что там тебя согреют и успокоят, посадят в теплую машину с решетками на окошках и отвезут в будущий дом :)! Так что все плюсы налицо.
Насчет скорости — верное замечание, помнишь фильмы про хакеров? Для взлома нужна скорость, которая в состоянии обеспечить тебе, по крайней мере, полноэкранное потоковое видео с хорошим качеством, так что нечего даже и пытаться ;(.
BaD.. BaD.. BaD..

От: Некит [fs_noffls@mail.ru]
Тема: –
Доброе время суток! Буду краток! Случайно прочитал Хакер Спец за октябрь 2005. Очень понравилась рубрика Мобильный взлом! Сам вопрос вот: собираетесь ли вы делать отдельно журнал по мобильникам и прочей современной технике (КПК, смарты) по взломам, прошивкам, софтам и др.!? А то на данный момент нет достойного журнала! Или расширить рубрику!
P.S. Ответ пришлите :)! Удачи!

Ответ: Э-э-э, это ты зря, Некит! «Буду краток» уже вносит политический оттенок, так недолго и дзюдо начать заниматься ;).
Что касается «Могильного взлома». Да, эта тема была (уже дважды, по-моему), есть и будет, но до того, чтобы делать отдельный журнал — вряд ли. По крайней мере, наверное, не по нашей части. Так вот насчет того, что «нет достойного журнала». Ты наши братские «Мобильные компьютеры» не обижай! Они, конечно, не мегахаксорские, но все-таки практически у руля мировой индустрии ;).
P.S. Ответ прислали. Тираж около 40 тыс. экз.

От: Jons [Jons.90@mail.ru]
Тема: Дорово,][ацкеры!!!
Дорово,][ацкеры!!!
Ваш журнал очень Cool!!! Читать очень интересно! Хочу спросить, как можно получить на халявщину инет. Например прогу, через которую буду выходить через другой номер, но через свою линию.
Заранее благодарен.

Ответ: Здоровей видали, Jons!
Спасибо за заслуженную (еще бы!) похвалу, мы очень гордимся и улыбаемся от счастья, когда нас хвалят ;). И за это мы тебе раскроем секрет «крэкера интернета». В последнее время очень сложно найти настоящий крэкер. Итак, методика: заходишь на **www.google.com** и набираешь «крэкер интернета скачать бесплатно», дальше жмешь «Найти» и жмешь по первой ссылке. Дальше тебе нужно запустить скачанный exe'шник. Все! Теперь интернет полностью бесплатен!
Не за что. Побольше бы таких добрых и отзывчивых читателей!

От: Новрузов Али [novruzov81@mail.ru]
Тема: –
Здраствуйте! Когда уже придет январский СПЕЦ? Уже скоро февраль месяц, а его все нет.

Ответ: Уж Герман полнится, а близости все нет... Али, получив твое письмо, мы очень обеспокоились: вроде бы и готовили номер долго и упорно, ночами сидели, в типографию резидентов отправляли и даже сигнальный экземпляр видели, а в продажу тираж не поступил :-(! Стали спрашивать у издателей, у распространителей — те делают вид, что ничего не заметили и ничего не произошло... А что самое страшное (для SkyWriter'а, ественно — прим. AvaLANche) — это то, что номер-то был с диском! Неужели диска тоже нет? Это наверняка провокация шпионов с загнивающего капиталистического Запада! Они хотят помешать нам раздуть мировой пожар антиглобализма, антикопирайта и свободы совести в цифровую эпоху. Но мы-то тоже не просто так, тоже не лыком SHIT'ы!
А тем временем это письмо уже пойдет в мартовский номер...
Где же январский? :(Если до сих пор нет, а ты на него подписан, обратись в наш отдел подписки: info@glc.ru или по горячей линии 935.70.34 для москвичей или 8 (800) 200.3.999 для жителей других регионов и абонентов сотовой связи (звонок бесплатный).

От: voron-19851 [Voron-19851@bk.ru]
Тема: ufls
Здравствуй, увожаемый журнал Хакер Спец. Довожу до вашего сведения, что вы должны мне 60 центов, потому что в вашем ноябрьском номере неправильно указан код для получения пароля к сюрпризу. По моей личной соображалке я понял, что необходимо изменить последнюю цифру в коде вернуть средства я прошу через Яндекс.Деньги с учетом моральных затрат на этот лицевой счет 4100144**0**2.

Ответ: Привет, уважаемый читатель! Поздравляем! Ты стал настоящим Хакером! Ты хакнул эту замечательную головоломку. Причем быстро и практически без хлопот! А ты знаешь, сколько машинного времени ушло у жадного до 60 центов населения на подбор 32-буквенного пароля от архива с сюрпризом?! Хочется плакать! Если бы все это время — да на борьбу, скажем, со СПИДом... Или на какой-нибудь идейный DDoS.
Очень жаль, что ты сделал этот хак именно таким не вселяющим оптимизм способом, и пришлось потратить целых 60 центов :(. А с другой стороны, положа руку на сердце, скажи, ведь было необязательно слать SMS, так? Можно было взломать сервер оператора, получающего эти сообщения, и взять код оттуда. Но это, как говорят в моих любимых магазинах на диване, еще не все!
Ты ведь мог заодно и незаметно перевести на лицевой счет 4100144*0** (узнаешь номер ;-)) все средства оператора контента. Ладно... Не расстраивайся. Все еще впереди, перед тобой огромный мир непознанных возможностей — дерзай и новых тебе свершений!
P.S. Главное — оставайся добрым и отзывчивым человеком ;-).
Искренне твои, Спецы

Story

иллюстрации Анна Журко

ПРИНЦИП МАКЛАУДА

... — И ЭТОГО ЧЕЛОВЕКА МЫ... ТО ЕСТЬ ВОТ ЕМУ МЫ ДОЛЖНЫ ДОВЕРИТЬ СВОЮ СУДЬБУ? — ШЕПНУЛ МОЛОДОЙ ГОСТЬ ПОЖИЛОМУ. — ПО-МОЕМУ, МЫ ЛИБО ОЧЕНЬ ОПРОМЕТЧИВО ПОСТУПАЕМ, ЛИБО НАШИ СВЕДЕНИЯ О НЕМ ЧЕРТОВСКИ УСТАРЕЛИ.
— ПОВЕРЬ МНЕ, ОПАСНОСТЬ — ЛИШЬ ИЛЛЮЗИЯ, — БЫЛ ОТВЕТ. — И... МЫ ВСЕГДА МОЖЕМ ОТКАЗАТЬСЯ |NIRO

— Да уж, можем... — разочарованный взмах руки, прикушенная губа. — Вот только что в таком случае ждет нас и наше дело?

— По крайней мере, мы останемся в живых, если повернемся сейчас и уйдем отсюда, не произнеся ни слова, — в словах пожилого человека чувствовался опыт прожитых лет. — А если откроем рот и поделимся своими планами, нас, скорее всего, пристрелят.

— Не очень верится в подобный расклад, — молодой старался не повышать голос, но высокие нотки невольно срывались с его губ. — Но я придерживаюсь такого же мнения. Если этот человек получит от нас хоть какую-то информацию, мы покойники. Но вы же видите сами — он способен провалить любое мероприятие!

На полу что-то мерзко хрустнуло. Молодой отступил в сторону и увидел под ботинком раздавленные в пыль осколки лампочки. Поднял глаза к потолку, разглядел прямо над собой болтающийся на проводе патрон, в нем прикипевший цоколь, разочарованно — в очередной раз! — покачал головой и глубоко вздохнул.

Пару минут назад они вошли в эту квартиру и, когда только открыли дверь, стали улавливать что-то гнетущее в обстановке — и это еще мягко сказано. Они сделали всего несколько шагов здесь и уже стали замечать признаки того, что здесь живет человек пьющий, причем пьющий всерьез и надолго.

Обшарпанные стены со следами старой штукатурки и обоев. Банка из-под горошка, пристроенная в качестве пепельницы у входной двери (еще пара таких банок стояли в разных углах той комнаты, которая имела право называться большой, но никак не гостиной, поэтому мысль о том, что хозяин выходит курить на лестничную площадку, отпала практически сразу).

Вешалка в углу прихожей была явно холостяцкой: вертикальная стойка с торчащими в разные стороны крючками, на которых были развешаны два выцветших плаща и несколько хозяйственных сумок. Сразу за раскрытой входной дверью по звуку и запаху ощущалось присутствие трясущегося старого холодильника «ЗИЛ» или «Бирюса» — никакого другого. Никогда не поймешь, то ли он так работает, то ли стремится выйти из квартиры и сброситься с лестницы вниз, чтобы прекратить свое механическое существование.

Где-то там, где предполагалось быть кухне, бормотало радио: новости жизни, экономики и прочей чепухи под периодически вылезающие рекламные ролики. Оттуда же слышалось журчание крана («Незавернутый кран — это же такая пытка...» — «Для соседей особенно...»).

Полное отсутствие ковровых дорожек и любого покрытия пола. Только обшарпанный паркет с непонятной геометрией. И вдоль всего этого великолепия — две длинные колеи, каждая шириной с велосипедное колесо... По коридору, поворот в кухню. Еще две полосы уходят туда, где находился сейчас хозяин квартиры и слышались звуки, которые издает телевизор.

— Инвалидная коляска, — глядя себе под ноги, сказал пожилой. — Я был в курсе...

— Понятно, для чего все те пакеты, которые в машине остались. Куча еды ему дороже денег, в магазин ведь выйти целая проблема.

— Совершенно верно. Я все ждал, Михаил, чтобы ты раньше спросил. Ну да бог с ним. Надо проходить. Похоже, он открыл дверь и даже сам не понял кому. Говорить по большей части буду я. Ты же понимаешь, что у меня это выйдет гораздо дипломатичней. Внимательно следи за развитием разговора: вполне возможно, что он не очень адекватен, может воспринять наш приход как угрозу, как вторжение, как опасность для жизни...

— Все так серьезно с ним, Павел Григорьевич?

— Я, скажу по секрету, читал его историю болезни. Есть еще у меня знакомые такого ранга, что достать для меня документы, составляющие тайны разного рода, не так уж и трудно... Так вот, фабула там еще та... Мне иногда кажется, что все, кто занимается Сетью на том уровне, на каком он умеет... или умел, черт его знает... Все они потенциальные клиенты психиатрических клиник.

Из комнаты раздался кашель, невнятное бормотание. Скрипнули колеса, наверное, от инвалидного кресла. Спустя несколько секунд потянуло дымком.

— Господи, что он курит? — скривился Михаил. — Что за мерзость... Неужели нет ничего поприличнее? Надеюсь, там, в ваших пакетах, есть хотя бы «Парламент»?!

— «Парламент»? Вы с ума сошли... Неужели вы никогда не нюхали этот запах — вот он стелется сейчас по всей квартире? Я, едва вошел сюда, сразу понял, откуда черпает силы несчастный. Это же марихуана. Конопля. Трава. Ферштейн?

— Натюрлих, — ответил Михаил, продолжая кривить лицо. — Наркота, черт побери. Еще один плюс в мою пользу. Надо уходить отсюда, пока есть гарантия, что он не запомнил наши лица.

— Сомневаюсь, что он запомнит их даже после получасовой беседы, — Павел Григорьевич невесело усмехнулся. — Живые люди — это совсем не то, что ему интересно. Неужели ты раньше никогда не сталкивался с такими, как он, Миша?

— Не приходилось, чего уж греха таить... Потому мне немного не по себе. Думаю, это видно невооруженным глазом. И еще. Все это чертовски похоже на что-то, что я уже когда-то видел по телевизору. Или в кинотеатре... Что-то вроде «Блэйда».

— В каком смысле? — Павел Григорьевич явно понимал, о чем говорит Михаил, но хотел услышать его трактовку.

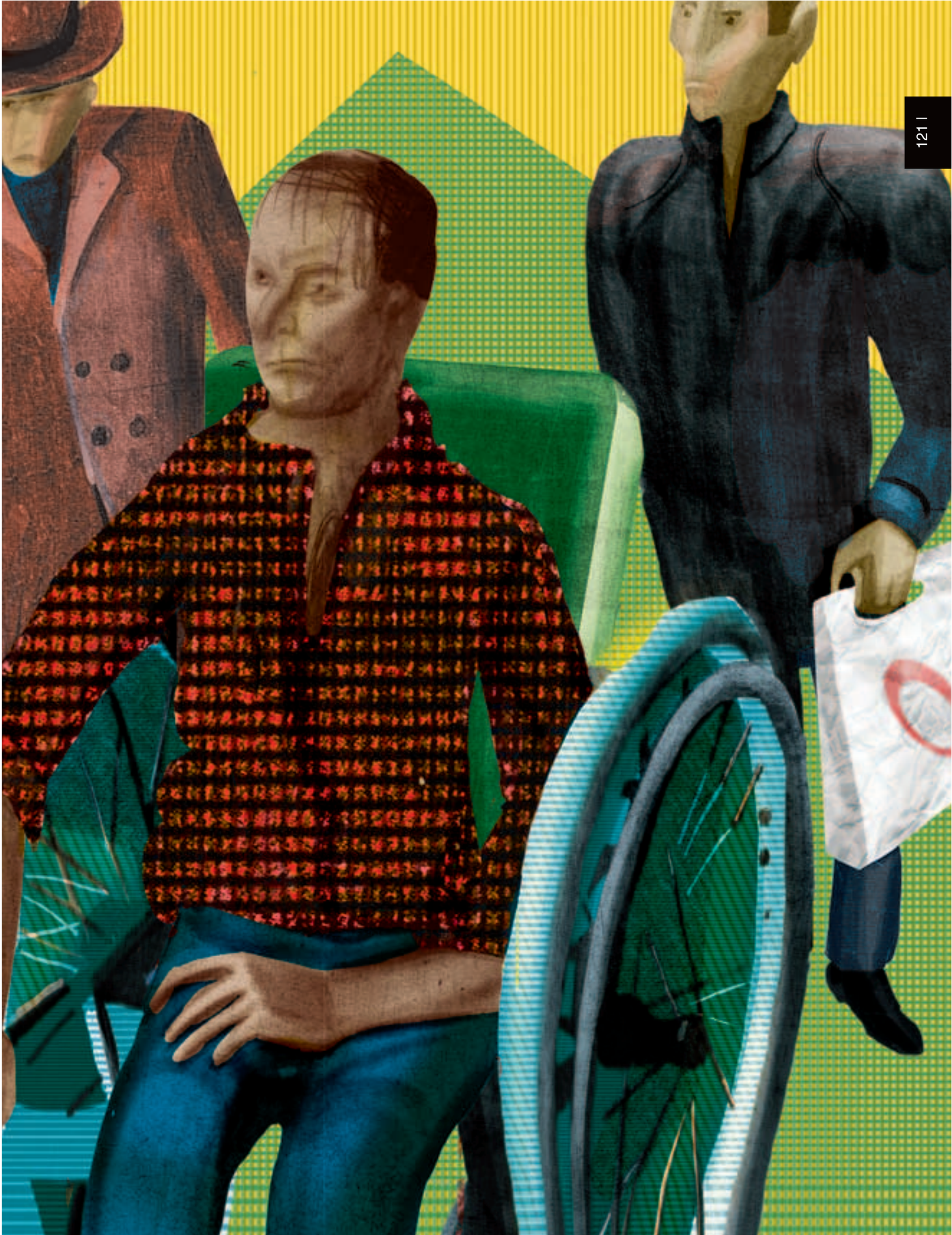
— Ну, как сказать... Даже слов не хватает. Как вроде одни нехорошие люди пришли к таким же нехорошим, но из этого всего вырастает некий намек на героизм, всесильность, вседозволенность. А корни-то одни — что у тех, что у других.

— То есть он ничем не отличается от того, против кого мы попросим его играть?

— Да. Блэйд. Точно. Плюс Блэйд был идейным, а мы несем деньги, поэтому наш вариант более гнилой, — Михаил нервничал все больше и больше. — Давайте уже побыстрее все закончим и уйдем отсюда.

Павел Григорьевич поправил галстук, посмотрел по сторонам и шагнул в комнату. Михаил сделал пару шагов следом за ним. Следующую минуту они молчали, так как не в силах были произнести ни слова.

Комната произвела на них жутковатое впечатление. Сквозь давно не мытое окно, не прикрытое ничем — никакого намека на занавески не было — пробивалось настолько мало света, что даже серо-голубой отблеск экрана телевизора казался здесь ярким. Возле окна в инвалидной коляске сидел че-



ловек, который был целью их визита. Его окутывало призрачное облако того самого сладковатого дыма, который Павел Григорьевич безошибочно опознал как конопляный.

Миша осмотрелся как будто скрывая свое действие, одними глазами. Человек в кресле внезапно почувствовал это любопытство, проследил за его взглядом, хрипло хмыкнул, прикоснулся губами к папиросе и аккуратно, едва-едва затянулся.

— Что вы ищете? — спросил он, выпустив струйку дыма. — Вы и так проторчали в коридоре почти десять минут — я уж решил, что вы ушли.

— Я ищу то, ради чего пришел, — ответил Михаил, выходя из-за спины Павла Григорьевича. — Я ищу компьютер.

— Вы скупаете старую оргтехнику? — скривился человек. Взявшись руками за колеса, он катнул коляску вперед примерно на метр и выехал из конопляного тумана. Гости сумели наконец-то разглядеть хозяина квартиры, и Михаил вновь убедился в том, что его мысль о скорейшем отбытии отсюда была единственно правильной.

«МИХАИЛ ОГЛЯНУЛСЯ В ПОИСКАХ МЕСТА, КУДА МОЖНО БЫЛО БЫ ПРИСЕСТЬ. ЕГО ПОВЕРГЛО В ШОК ТО, ЧТО ОН УСЛЫШАЛ!»

Человек был явно молодым — не больше тридцати лет, однако эта молодость скорее угадывалась на каком-то подсознательном уровне. Михаил просто понимал, что такие, как этот человек, не могут быть старыми. Но внешне хозяин квартиры тянул лет на пятьдесят, не меньше. Морщинистые руки, лежащие на колесах, были похожи на руки деревенского труженика, который всю жизнь не расставался с лопатой, но никак не на руки хакера — человека, не державшего в руках ничего тяжелее мышки.

Какой-то потрепанный байковый халат с несколькими пятнами неопределенного цвета на нем, спортивное трико серо-синего цвета с эмблемой Adidas на бедре. Совершенно неожиданно картину довершали новые тапочки непонятной фактуры с какими-то розовыми звездочками. Михаил увидел это, хмыкнул и отвел глаза — абсолютно очевидно, что у человека, сидящего в инвалидном кресле, обувь должна все время оставаться новой.

— Ни о какой скупке техники и речи быть не может, — Павел Григорьевич явно ожидал подобного разговора и был к нему готов. — Я слышал от очень и очень авторитетных людей, не бросающих слов на ветер, что здесь, в этих стенах я смогу найти человека, который когда-то столкнулся нос к носу с Сетевым Вампиром. Столкнулся и остался жив. Единственный, кто сумел уцелеть и продолжать жить после этого, не окружив себя охраной.

Человек в кресле скривился от упоминания о том, что и Михаил услышал впервые.

— Жить... — прошептал он, но Павел Григорьевич его услышал.

— Вот именно — жить, — он сделал пару шагов навстречу хозяину. — И мне нужен этот человек и все его знания, умения и навыки. Но, к великому моему сожалению, я не вижу в этом доме того оружия, которым этот человек может сразиться, — я не вижу компьютера.

Михаил оглянулся в поисках места, куда можно было бы присесть, так как его повергло в шок то, что он услышал. Увидев рядом с собой табуретку, края которой были словно погрызены неизвестным животным, он опустился на нее и жадно стал ловить каждое слово. Еще вчера Павел Григорьевич сказал ему, что они нанесут визит именно хакеру, способному решить некоторые проблемы их бизнеса. И все! Ни о каких вампирах, ни о каком оружии для сражений не было и речи!

— Неужели я ошибся? — продолжил Павел Григорьевич, выждав несколько секунд. — Неужели вы расстались с мыслью о том, что можно продолжать вашу работу?

— Я выпотрошен... — прохрипел хозяин. — Я пуст. Внутри меня чернота. Ни единой искорки света... Я видел Сетевого Вампира. Я говорил с ним. Он оказался сильнее меня. Намного сильнее...

— Я вижу это, — ответил Павел Григорьевич. — Я вижу человека, который растоптан, унижен, обезножен... Но я также вижу человека, который может выполнить работу несмотря на всю свою немощь. И у меня есть чем отблагодарить этого человека.

— Меня зовут Аристарх, — хозяин немного выпрямился в кресле. — Немного напыщенно, но моей вины тут нет: мама постаралась. Не знаю, как вам, а мне нравится.

— Павел Григорьевич, — представился гость. — А это мой помощник Михаил — человек, которому я в недалеком будущем передам свое дело. Поэтому хочу, чтобы он присутствовал здесь от начала и до конца.

Михаил едва не проглотил собственный язык от изумления, услышав последние слова своего босса. «Так я здесь не просто так, а для того, чтобы перенимать опыт?!»

— Что вы хотите от меня? — Аристарх облизнул губы и протянул руку к наполовину пустой трехлитровой банке с водой. Сделав несколько больших глотков, он шумно отдышался, после чего со словами «Извините, у меня диабет... Пью много...» вернул банку на место.

— Тоже наследство Вампира? — поинтересовался Павел Григорьевич.

— Нет, — отрицательно покачал головой хозяин. — Это у меня, как и имя, от мамы. Я на инсулине. С детства. Ведь вот и тогда, когда эта сволочь... Я думаю, он был в курсе. Меня ведь надо было только отключить на пару часов, чтобы я пропустил инъекцию. И все. Кома. А из такой комы не выходят.

— Кто-то успел? — внезапно спросил Михаил. — Кто-то был рядом?

— Нет, — Аристарх махнул рукой. — Сам как-то. Выбрался. Очнулся, дополз до коробки со шприцами, ткнул в бедро. Даже сразу не понял, какой именно вытащил из коробки, они ведь разной силы бывают, эти инсулины. Знаете, кто чем болеет, тот о том и говорит, — он попытался усмехнуться, но получилось вяло и очень неправдиво. — Я вам про диабет могу много рассказать...

— Охотно верю, — Павел Григорьевич оглянулся на Михаила, щелкнул пальцами. Тот послушно встал и отошел в сторону. Босс опустился на его место, покрутил шеей, пробурчал «Хондроз, хондроз... Каждому свое». Аристарх наблюдал за ним с малозаметной долей интереса. Было заметно, что его ноздри раздувались, ловя запах конопли. Казалось, что Аристарх не успел сделать несколько самых главных затяжек до прихода гостей и поэтому был достаточно сильно раздражен.

— Вы хотите поговорить о деле? — Павел Григорьевич оглянулся на Михаила и подмигнул ему. — Думаю, вас как минимум разбирает любопытство: «Что же эти двое могут предложить?» Почему-то мне кажется, что вас нельзя удивить ни одной произнесенной вслух денежной суммой и вообще какими бы то ни было материальными ценностями.

— Поговорить? Хочу, — ответил Аристарх. — Ну, и в отношении всего остального вы совершенно правы. Деньгами меня не удивишь, смертью не испугаешь. Попробуйте какой-нибудь другой ход, если вы действительно хотите нанять меня, такого, как есть, на работу.

— «Такого» — какого? — поднял бровь Михаил. — Вы намекаете на свое здоровье?

Павел Григорьевич опустил глаза в пол. Аристарх посмотрел на Михаила и наконец-то смог улыбнуться по-настоящему:

— На что? На здоровье? С тех пор как я перестал представлять себе, что такое здоровье, прошло много лет... В двенадцать лет у меня выявили диабет, в шестнадцать с половиной я упал с четвертого этажа по собственной ребячьей глупости... Нет, в кресле я не поэтому, молодой человек... В двадцать четыре я перенес одну очень нехорошую инфекцию — менингит, слышали про такую? Это когда или умираешь, или дураком остаешься. Неплохой из этого сделали анекдот, в смысле что я не умер. Ну а в тридцать шесть я стал тем, кого вы сейчас видите.

— А сколько вам сейчас? — Михаил не мог сдержаться, вопрос сам просился наружу.

— Сорок два. И вы с трудом верите в это.

— Точно, — тихо сказал Михаил и прислонился к стене.

Аристарх отъехал к окну, выглянул на улицу, чему-то кивнул — то ли солнцу, то ли птицам... Потом повернулся к Павлу Григорьевичу и спросил:

— Будем говорить?

— Да, — тот поднял голову и встал, чтобы подойти ближе. — Но сначала я назову вам цену, которую готов заплатить за вашу работу, еще не состоявшуюся.

Он вытащил из внутреннего кармана пиджака маленький листок бумаги и протянул его Аристарху. Тот взглянул на несколько написанных там слов и цифр, и его губы мелко затряслись.

— Это правда? Вы действительно в ответе за каждый знак на этом листке?

— Без всякого сомнения, уважаемый. Но тут, как вы можете видеть, не хватает нескольких цифр — для полноты картины. (Аристарх снова жадно стал всматриваться в листок, а увидев то, что искал, даже застонал от злости.) Я сообщу их вам по окончании работы. Вижу по вашим глазам, что вы готовы выполнить сейчас для меня любое задание, лишь бы узнать недостающую информацию.

Аристарх поднял полные боли и страдания глаза на Павла Григорьевича и прохрипел:

— Что я должен делать? Умоляю, давайте быстрее!

— Вот это уже совсем другой разговор, — Павел Григорьевич кивнул. — Другая тональность, другие желания. Наконец-то! Я вижу здесь человека, способного решить мою проблему. Но где же ваш компьютер? Я слышал... Точнее сказать, у меня есть информация, что с некоторых пор вы не подходите к нему и за ненадобностью уничтожили. Так ли это? Если у вас нет компьютера, я предоставлю вам любой, какой вы только захотите, в течение двух часов. Его соберут по вашему заказу и привезут сюда быстро, насколько я знаю своих работников.

Аристарх окинул взглядом комнату и посмотрел на Павла Григорьевича снизу вверх:

— Не все так плохо, как кажется.

Он подъехал к старой полированной стенке, откинул крышку, сразу ставшую подобием письменного стола, и гости увидели в глубине шкафа аккуратный сияющий Macintosh.

— Это, конечно, не самое совершенное сетевое оружие в этом мире, но он мне нравится своей безотказностью, — прокомментировал Аристарх. — Ни разу не подвел меня... Кроме одного раза, но там я сам был виноват. Не учел некоторые факторы, о существовании которых просто обязан был догадываться. Знаете, Сеть полна таких загадок, что никакой фантазии не хватит, но задним числом понимаю, что мог бы и подготовиться к бою, мог бы просчитать чужие ходы, мог ответить так, что не я, а он сидел бы сейчас в инвалидном кресле! А лучше — лежал в гробу!..

— Кто «он»? — шепнул Михаил Павлу Григорьевичу. — Почему я не в курсе?

— Тихо, — резко ответил тот. — Так вы готовы слушать?

— Готов.

— Значит, так, — Павел Григорьевич немного прошелся по комнате, потирая руки. — Я и мой друг представляем одну довольно влиятельную корпорацию. Фирму, напрямую связанную с высокими технологиями. Не буду приводить здесь ее название, но, поверьте, не ради тайны. Просто мне так привычнее: лучше скрыть очевидное, чем потом разгребать последствия. Тем более что вы сами будете в состоянии догадаться, не утрачена ли ваша способность к анализу...

Аристарх рассмеялся громко, в его смехе было презрение.

— Утрачена? Способность к анализу? А как, по-вашему, я остался в живых тогда? Только лишь потому, что сумел быстро просчитать все на пальцах и предсказать развитие события на полхода вперед!

«Как-то не очень верится насчет “быстро”, — подумал Михаил, продолжая рассматривать инвалидное кресло Аристарха. — Чего же он тогда вот так? В таком виде? Значит, все-таки не просчитал». А вслух сказал:

— Дай бог, чтобы эта способность действительно сохранилась в полном объеме.

— В Сети нет Бога, — внезапно резко ответил Аристарх. — Нет и не может быть. А я дитя Сети — кровь и плоть от нее, поэтому мне виднее...

— Да я, собственно, не об этом, — ответил Михаил, не понимая реакции Аристарха, уж совсем она была сейчас не к месту. — Я-то все о том, что нас интересует. О ваших способностях.

— Они никуда не делись, — снова металлическим тоном ответил хозяин. — Если сомневаетесь в чем-то, ищите другого. Я как-нибудь и без вас обойдусь.

— Давайте не будем идти по противоположным курсам, — Павел Григорьевич попытался развести двух людей, не понимающих друг друга. — У нас с Михаилом здесь и сейчас вполне определенная цель, и я не хотел бы, чтобы наше сотрудничество закончилось не начавшись. Я могу продолжать?

— Давайте, не тяните. Но! Никаких упоминаний о сверхъестественных силах. Ведь даже Сетевой Вампир — человек вполне реальный.

— Вы уверены, что это не просто феномен интернета, а человек? — удивился Михаил, познания которого о Сетевом Вампире ограничивались какими-то обрывочными сведениями. — Настоящий человек из плоти и крови?

— Вы сейчас говорите как Маугли, — хмыкнул Аристарх. — Да, это человек. Следовательно, он смертен. А значит, я его накажу. Как только сделаю вашу работу, так сразу возьмусь за свою.

«Маугли... — подумал Михаил и решил больше в беседу не вмешиваться. — Пусть сами договариваются».

— Итак, продолжу, если вы не против. Интересы моей корпорации на рынке компьютерных технологий пересекаются с интересами многих других фирм. Одна из них составляет нам очень серьезную конкуренцию. Еще немного усилий, и она вытеснит нас с рынка...

— Элементарные вещи, — прокомментировал Аристарх. — Волчьи законы.

—... А я не хотел бы видеть гибель моей организации, у истоков которой я стоял четырнадцать лет назад, — закончил свою мысль Павел Григорьевич.

— Следовательно, я должен предпринять какие-то шаги, и, как вы понимаете, они должны быть максимально действенны, а следовательно, незаконны.

Аристарх понимающе кивнул так, как будто только и занимался всю жизнь чем-то незаконным, но очень нужным людям.

— Чтобы осуществить свои планы, мне нужен такой человек, как вы — личность, способная проникать в Сеть всей своей сущностью, — Павел Григорьевич остановился прямо напротив Аристарха и пристально посмотрел ему в глаза. — Вы действительно именно такой, каким я вас себе представлял?

Аристарх молчал, глядя в одну точку и тщательно избегая встречаться взглядом с гостем.

— Молчите? Ну да ладно... Далее. В чем суть проблемы? В том, что, как и всегда и везде в истории, есть человек, который двигает конкурирующую фирму вперед, к успеху так же, как это делаю для своей корпорации я. И чтобы остановить продвижение конкурентов на рынке, этого человека надо убрать.

— Надолго ли это остановит их? — внезапно спросил Аристарх. — На смену одному придет другой... На этом в свое время обожглись цареубийцы. Наверняка у вашего врага есть масса заместителей, которых он себе подбирал не по размеру груди, если только его контора не занимается распространением в Сети порнографических фильмов.

— Нет, до порно им далеко, — отмахнулся Павел Григорьевич. — Дело в основном касается интернет-индустрии, рынка браузеров и вообще софта для работы с Сетью, в Сети и еще много чего нужного и полезного.

— Я, наверное, давно не читал газет, — склонил голову хозяин. — Неужели русские чего-то добились в этой области?

— Вы пессимист, я это понял сразу, — Павел Григорьевич покачал головой. — Да, мы пока еще не в первых рядах...

— Станешь пессимистом поневоле, если не сможешь стоять на своих собственных ногах... — глаза Аристарха зло сверкнули. — Мы это уже проходили. Помните, когда хотели поднять автомобилестроение и пытались воздействовать на ввоз иномарок, облагали дополнительными налогами, запрещали правый руль, пристрелили кое-кого, некоторых заставили покинуть страну, строили заводы, делили награбленное... И где они все? Неужели вы думаете, что вас не постигнет их судьба? Чем вы отличаетесь от них?

— Проповеди ни к чему, — за босса ответил Михаил. — Давайте лучше работать в указанном направлении.

— Да, мой коллега прав, — подхватил мысль Павел Григорьевич. Очень не хотелось признаваться самому себе в том, что Аристарх прав практически на сто процентов. — Необходимо убрать главного программиста конкурентов. Выключить из жизни. Максимально выгодный вариант — его смерть...

— Я сейчас не в силах, — тут же отрицательно замотал головой Аристарх. — Это совершенно точно, можете меня не убеждать. Не потяну.

— Хорошо. Ну, а если вот так... — и Павел Григорьевич легонько махнул рукой в сторону самого хакера.

— Болезнь? Инвалидное кресло? А лучше реанимация, — покачал головой тот. — Понимаю. Давайте координаты.

Аристарх с силой крутанул колеса и прокатился пару метров по комнате. Михаил едва успел убрать ногу и проводил хакера взглядом.

— Работа, работа... — пропел Аристарх, развернулся на месте — похоже, креслом он владел в совершенстве, как Паганини скрипкой. «Если бы и на компьютере так же... — подумалось Михаилу. — Только бы мы не ошиблись выбором».

Колеса скрипели, пока хозяин выделывал па посредине комнаты. Внезапно он остановился и сказал:

— Я отлучусь на несколько минут. Просьба ничего не трогать.

И выехал в другую комнату, откуда спустя некоторое время раздался какой-то шорох и бормотание.

— Ну, как? — одними губами спросил Павел Григорьевич.

Михаил покачал головой, не зная, что сказать.

— Вот так-то... — сам себе ответил босс и развел руками. Похоже, он тоже был не очень доволен, что-то в разговоре с Аристархом ему явно не удалось. — Неужели я ошибся в нем? Он совсем не такой, каким я его представлял себе. Я шел на встречу с одержимым, с фанатиком Сети, задавленным на время своим недугом, надеялся разбудить в нем жажду жизни, но он не такой. Он явно не болен. Такое впечатление, что он притаился на время и просто ждал того, кто принесет ему адрес Сетевого Вампира на блюдечке с голубой каемочкой.

— А по-моему, насчет его болезни вы не ошиблись, — прошептал Михаил, подойдя поближе. — Наркоман... Ведет себя так нервно, резко, переменчиво, и потом эта фраза про Бога. К чему он решил закрыть мне рот, когда я произнес «Слава богу...»?

— А насчет менингита? Как тебе? — Павел Григорьевич подмигнул Михаилу. — И ведь не умер. Если не соврал, конечно...

— Не соврал, не бойтесь! — донеслось из соседней комнаты. Гости от неожиданности вздрогнули.

— Вот так слух... — не таясь, прокомментировал Павел Григорьевич. — Никаких секретов в пределах квартиры.

— Точно, — Аристарх, поскрипывая колесами, показался в дверях. Гости его не узнали.

Он преобразился до неузнаваемости. Лицо было закрыто какой-то поляризованной маской, которая в зависимости от угла зрения меняла свой цвет от розового к голубому. В области глаз она была выпуклой, и поэтому голова Аристарха казалась похожей на голову стрекозы. В районе правого уха — некое подобие гарнитуры, которая загибала свои обводы в ушную раковину и выдвигала ко рту тоненький усик микрофона. На руках — нечто, отдаленно напоминающее перчатки рокеров, с укороченными пальцами. Вдоль кистей, уходя куда-то под рукава халата, тянулись, пропадая, серебристые провода. Он подъехал к своим гостям, совершил вокруг них нечто вроде круга почета и, наконец, спросил:

— Мне удалось вас удивить?

— Вот именно при помощи... этого... вы совершаете свои виртуальные подвиги? — с трудом произнес Павел Григорьевич.

— Да, — кивнул Аристарх. Маска несколько раз сменила цвет. — Осталось

только включить компьютер и вычислить вашего врага. Проживет он недолго... Надеюсь, ваша корпорация уделит мне небольшой процент своих акций? Потом, потом — когда все будет так, как вы планируете?

Павел Григорьевич развел руками, соглашаясь. Собственно, он и не надеялся, что сумеет отделаться от хакера одной только маленькой бумажкой с очень важными для него цифрами.

— Разумеется. Все, что пожелаете.

— А в каком-то старом советском фильме про войну говорили: «Все, что могу. Все, что могу...» — Аристарх подкатил к тому шкафу, который скрывал внутри себя Mac, снова скрипнул дверцей, включил компьютер и, пока тот загружался, начал священнодействовать с какими-то маленькими девайсами рядом. Включил одну черненькую коробочку, другую. На них поочередно загорались и гасли контрольные светодиоды. Нечто подобное он включал и на себе, бормоча под нос то ли молитву, то ли заученный раз и навсегда список или последовательность действий.

Михаил смотрел на все это и понимал, что имеет дело с кибершаманом, именно с таким существом: какое-то виртуальное колдовство царило сейчас в квартире Аристарха. Хакер на некоторое время прекратил всякую активность, стал рассматривать таблицы, появившиеся на экране, затем что-то прошептал в микрофон — компьютер откликнулся разноцветной диаграммой, построил график, выдал несколько вопросов, на которые Аристарх снова ответил тихим голосом. Разобрать, что он говорит, было невозможно.

Неожиданно Павел Григорьевич понял, что Macintosh сейчас распознает хозяина по голосу: все эти графики были контрольной аудиограммой, которая сверялась с хранимой в памяти. Похоже, Аристарх давно не занимался своей работой, потому что компьютер с трудом, с четвертой попытки, сумел принять голосовой пароль.

— Вы слишком много курите, — сказал Павел Григорьевич, давая понять, что он разбирается в происходящем.

— Спасибо, что напомнили, — отозвался Аристарх, втащил из кармана халата папиросу, закурил и аккуратно загладил горящий край послюнявленным пальцем — чтобы разгоралось не сильно. По комнате снова поплыл сладковатый запах конопли.

— Это необходимо? — спросил Михаил, который был уверен на сто два процента, что они имеют дело с законченным наркоманом, которому осталось совсем чуть-чуть до полного провала их мероприятия.

— Не то слово... — буркнул в ответ Аристарх, продолжая общаться с компьютером. — Где искать вашу мишень? Не стесняйтесь, заказывайте...

Павел Григорьевич приблизился к Аристарху вплотную и шепнул ему на ухо несколько слов.

— Вы уверены, что он сейчас там? Человек должен быть сейчас в Сети — я ведь не волшебник, не умею работать через холодильник или микроволновку. Только через компьютер!

Павел Григорьевич кивнул.

— Не сомневайтесь. Сейчас у нас около шести вечера? Самое время... Привычки изучены досконально. Рабочий день до пяти, человек он одинокий, расслабляется подобным образом ежедневно в течение полутора часов по окончании рабочего дня.

— Разведка?

— Приходится быть Геббельсом и собирать порочащую информацию, иначе не уцелеешь в этом мире.

— Идеальный объект, — выражение глаз Аристарха было скрыто маской, но губы демонстрировали хорошее настроение. Хакер улыбнулся. — Полное слияние с Сетью, раскрепощение, расслабление... Лучше бы живую девочку купил.

— Покупал, — согласился Павел Григорьевич. — Не те ощущения.

— Вы-то откуда знаете? — вырвалось у Михаила.

— Я не про себя. Я про... Объект, как его назвал Аристарх, — ответил Павел Григорьевич, но чувствовалось, что он смущен двусмысленностью ситуации.

— Ну-ну, — хохотнул хакер. — Значит, так. Я работаю. Вы можете идти. Оставьте телефон, сообщу о результате. Если он человек известный, узнаете в новостях, если не очень публичный, то... Про киберсекс же узнали? Вот и о здоровье осведомитесь.

— Я хочу остаться, — вдруг сказал Павел Григорьевич. — Никогда не видел ничего подобного.

— Я на публику не работаю. Не Дэвид Копперфильд. Здесь не шоу, — не согласился Аристарх.

— Назовите цену. За какую сумму вы сможете превратить это все в шоу?

— Я же сказал, деньги меня интересуют не в первую очередь, никогда не чувствовал их власти над собой.

— Миша, прикажи принести пакеты из машины.

— Что за пакеты? — напрягся Аристарх.

— Продукты. Много хорошей еды. Фрукты. Соки. Водка, — Павел Григорьевич произносил слова с оттяжкой, выдерживая паузы.

— После травы всегда так есть хочется, — понимающе протянул хозяин. — Знаешь, чем купить... Ладно, неси, Мишаня. А я пока начну.

Он вплотную подъехал к столу, прижав к нему колеса и колени. Михаил увидел, как Аристарх наклонился к экрану, и порадовался тому, что сейчас можно уйти. Он совершенно не хотел присутствовать при всяких криминальных сетевых разборках с участием хакеров-убийц и прочей уголовной гадости. В коридоре, протянув руку к двери, он вдруг услышал что-то вроде «Ух ты...», вырвавшееся у Павла Григорьевича. Потом раздались его же слова: «Точно, это он... Значит, вот как оно бывает». Михаил не выдержал, выскочил на лестницу и, не оглядываясь, рванул вниз, прикидывая, сколько времени ему лучше оставаться в машине.

Тем временем в квартире Аристарха происходило нечто удивительное. Павел Григорьевич не понимал того, что происходит на экране: буйство каких-то бешено бегущих снизу вверх строк с загадочными знаками и недоступной для восприятия информацией. Но главное происходило не там...

Павел Григорьевич не мог оторвать глаз от маски Аристарха. Она перестала просто менять цвета, на ее поверхности что-то происходило. Несколько радужных бликов неопределенных очертаний постепенно превращались в цветную картинку, словно отраженную от чего-то, — несколько домов, лиц, надписи в зеркальном отражении... Павел Григорьевич смотрел на все это как на кино, которое Аристарх крутит внутри маски, при этом его глаза казались кинопроектором.

Сквозь непроницаемый материал Павлу Григорьевичу транслировалось то, как Аристарх путешествует в Сети в поисках объекта. Хакер пробивался сквозь череду серверов, сетевых экранов, легким

движением руки стирал информацию, которая могла бы привести тех, кому он навредит, к нему самому. Изредка на маске вспыхивали какие-то круги, проносились зеленые и синие штрихи, словно трассирующие пули, — Павел Григорьевич пытался объяснить себе их происхождение какой-то сетевой активностью Аристарха, но все наблюдаемое было настолько загадочно и таинственно, что спустя несколько минут он перестал вообще задумываться над тем, что происходит.

Руки хакера жили своей жизнью, пальцы на клавиатуре легонько подрагивали, в нужный момент находя нужную клавишу. Чаще это было одно нажатие, реже — быстрая пулеметная очередь, которая заканчивалась громким нажатием клавиши ввода. Компьютер, несомненно, каким-то образом реагировал на все это, но понять, что же происходит, было все-таки невозможно.

Неожиданно на маске Аристарха отобразилось лицо человека, о котором сегодня шла речь, — лицо человека, который сейчас стал объектом атаки хакера. Лицо, сначала нечеткое, затем изображенное ясно, всплыло откуда-то из глубины и прорисовалось прямо по контуру маски. Казалось, что у Аристарха новое лицо — настолько оно вписывалось в контуры маски.

«Человек с маски» улыбнулся Павлу Григорьевичу и шевельнул губами, спрашивая что-то. В этот момент Аристарх впервые произнес громко и отчетливо:
— Здравствуйте...

Гость поразился перемене его голоса. Откуда в нем только взялись высокие женские нотки? Но потом он увидел некое подобие модулятора, пристегнутое в проекции гортани, и понял, что хакер был готов и к этому.

Лицо на маске снова что-то произнесло и игриво подмигнуло.
— Как обычно, Лайонелл? — спросил Аристарх. — Я прекрасно помню ваши пристрастия... В каком сегодня белье?

Лайонелл посмотрел куда-то вниз, потом снова перед собой...

«In black... — прочитал по губам Павел Григорьевич. — В черном... Но он говорит по-английски, а Аристарх по-русски... Какая разница, в конце концов! Наверняка там переводчик на лету транслирует!»
— Апартаменты? Природа? Берег моря? — Аристарх разговаривал с Лайонеллом как дорогая проститутка, которая знает цену себе. И неудивительно, они оба сейчас были на сайте по предоставлению услуг киберсекса, только, похоже, Аристарх подменил собой виртуальную модель.

Было невозможно сразу понять, что говорил клиент, его тирада на английском была длинной, Аристарх пару раз кивнул и нажал что-то на клавиатуре. Павел Григорьевич вдруг понял, что ждет Михаила: было очень неуютно одному в квартире с человеком, который ушел в кибернетический астрал; Аристарх перестал для него быть живым человеком, он превратился в робота.

— И все-таки апартаменты... — голос был по-прежнему теплым, вкрадчивым, как раз как голос гейши. — Я одобряю ваш выбор: на море сегодня пасмурно, в лесу, думаю, сыро, и еще… Я так не люблю всякую мошкару! Ну что, милый, вперед — в сказку?

Странно было слышать такое из уст человека в старом байковом халате, спортивном трико, в то время как на столе рядом с компьютером дымился непогашенный «косяк» с марихуаной. Странно и как-то даже неприятно для человека с обычной сексуальной ориентацией, каким всю жизнь Павел Григорьевич себя считал, отдавая пальму первенства в сексе женщинам.

Лицо Лайонелла на время исчезло, в маске отразились какие-то просторные коридоры, множество дверей, они шли куда-то, шли, Аристарх что-то шептал, нежно и совсем не по-мужски...

Наконец, перед ними распахнулся дверной проем, несколько неярких ламп проплыли перед глазами Павла Григорьевича. Потом на фоне большой кровати под кружевным одеялом снова появилось лицо Лайонелла.
— Ты не против, если я включу музыку? Что-то легкое... Какой-нибудь старый блюз. Хорошо?

Похоже, Лайонелл согласился. Музыка, конечно же, заиграла, но где-то далеко — Павел Григорьевич был уверен, что Аристарх ее слышит. Да и вообще Аристарх был сейчас в полном контакте с Лайонеллом, погрузившимся в интернет. Звуки, видения, ощущения — все проходило сквозь его тело и Macintosh.
— Присядь на край кровати, дорогой... Я сниму твой пиджак. Зачем так сильно завязывать галстук?! — удивленный возглас гейши. — Ну так же нельзя... Распустим узел. У тебя так напряжены мышцы... Я помассирую тебе плечи. Здесь... и здесь.

Павел Григорьевич заметил, что пальцы Аристарха шевелятся легко, практически незаметно, но где-то там, в астрале, он массировал сейчас тело Лайонелла, также подключенное к сенсорному костюму.
— Вот, уже лучше... Тебе тепло, уютно... Поворачивай головой. Медленно, медленно... А теперь ложись, я разомну тебе спину. С каким маслом это сделать сегодня?

И вдруг Павел Григорьевич заметил, что на экране компьютера что-то изменилось: картинка стала более упорядоченной, появились какие-то таблицы с время от времени меняющимися показателями. Он присмотрелся к данным таблиц и вдруг понял, что видит перед собой какой-то очень серьезный медицинский сканер, который удаленно вычисляет параметры здоровья Лайонелла. Иногда на экране выстраивались графики, отмечались какие-то цифры. Пару раз из невидимых колонок слышались тихие гудки.

«Он отслеживает какие-то параметры здоровья, которые могут ему сейчас понадобиться, — понял Павел Григорьевич, с трудом вникнув в то, что происходит на экране. — Неужели он сможет повлиять на него физически?»
— Откинь покрывало... Ложись... Ты любишь смотреть, как я раздеваюсь, я знаю... Какая приятная музыка...

Все-таки жутковато было слышать все это из уст мужчины — Павел Григорьевич покачал головой в такт своим мыслям и посмотрел на маску. Сложно было что-то разобрать в пятнах света, но было ясно, что кибернетическая гейша близка к тому, что заняться сексом со своим клиентом.
— Ну, не будь таким зажатым, милый... В прошлый раз ты набросился на меня как вихрь... — Аристарх, продолжая говорить, быстро прошелся по клавиатуре. — Не хочешь? Устал? Хорошо, я все сделаю сама...

Павел Григорьевич понял, что сейчас начнется все самое интересное: Аристарх нанесет удар во время секса. Он подошел вплотную к хакеру и принялся рассматривать показания медсканера. Постепенно он разобрался с тем, что здесь к чему, и заметил, что у Лайонелла постепенно растет пульс и давление: сенсорный костюм делал свое дело, выполняя все свои возбуждающие функции. Так продолжалось две, три минуты...

Внезапно Аристарх шепнул, на этот раз своим голосом:
— Усилитель...

И протянул руку к какому-то устройству рядом с компьютером. Мягко нажатая кнопка поначалу не произвела никакого эффекта, и вдруг Павел Григорьевич понял, что начинает проваливаться куда-то...
—... Шеф!.. — кто-то тряс его за плечо. — Павел Григорьевич! Что с вами? Что здесь произошло?
— Ничего страшного, — раздался где-то за спиной голос Аристарха. — Захотел посмотреть и посмотрел. Я же не обязан предупреждать обо всех побочных эффектах своей работы.

Павел Григорьевич понял, что лежит на полу возле инвалидного кресла, ткнувшись лицом в то, что когда-то было ковровым покрытием, а теперь стало простой грязной тряпкой. Он закашлялся от обилия пыли, схватился за сердце, почему-то вдруг обнаружил, что оно болит, но… все было в порядке. Рядом с ним на коленях стоял Михаил. В глазах молодого парня был страх и недоверие, он подумал, что случилось что-то ужасное. Сзади него по ком-

нате были рассыпаны яблоки. Еще три пакета, брошенные у двери, являли наружу яркие этикетки.

Руки Михаила помогли ему подняться. Голова кружилась, во рту было сухо. Он покачнулся и резко опустился на табуретку. Кашель так и рвался изнутри...

— Что это было? — сумел произнести он спустя несколько минут, когда в горле все успокоилось. — Я помню слово «Усилитель», а потом — все, провал...

Аристарх в это время медленно потягивал папиросу у подоконника, выпуская дым вертикально вверх. Маски на нем уже не было, провода тоже куда-то делись. Он не смотрел на своих гостей, думая о чем-то далеком...

— Я спрашиваю! Что со мной случилось?! — повысил голос Павел Григорьевич. — Я думаю, я вправе знать и об этом, и о том, что там с нашим заказом!

— Все просто, — выдохнул в очередной раз пару колец Аристарх. — Во время включения усилителя эмоций поле зацепило и вас... Уж не знаю, что там у вас было на душе, но оно выросло во много раз и повергло вас в обморочное состояние. Сразу отвечу на следующий невысказанный вопрос: да, я знал о том, что так будет. Знал и не сказал. Я не хотел, чтобы вы видели все целиком, поэтому сам момент воздействия на человека, которого вы мне заказали, оказался вам недоступен.

— Что вы с ним сделали? — спросил Михаил, который не видел вообще ничего.

— Пока он развлекался с виртуальной проституткой, медицинский сканер изучал все его слабые стороны. У него в его сорок с небольшим лет уже имелись умеренные признаки стенокардии. Проще говоря, его уже преследовали болевые приступы в области сердца...

— Значит, я был прав, — тихо сказал Павел Григорьевич. — Сканер... Послушайте, а как вы умудрялись держать одновременно два канала связи: и через сервер киберсекса, и со своим медицинским оборудованием? Ведь вы же должны были оценивать всю информацию, которая поступала вам с двух направлений. Я бы понял, если бы за вас все делал компьютер, но это не так и, к тому же, значительную часть процесса вы взяли на себя!

— Техническая сторона работы — это мой секрет, — Аристарх причмокнул губами, затягиваясь. Чувствовалось, что он расслабляется, глаза его были прикрыты. Казалось, что и его голос становится тише. — Хотя… все это можно объяснить талантом. Даже больше — гением. Но мне не нужны все эти эксцентрические звания. Вы яблочки не соберете? — внезапно спросил он у Михаила. Тот огляделся по сторонам и, не отрывая глаз от Аристарха, наклонился, принялся шарить пальцами по полу, складывая яблоки в пакет.

— Что сейчас с Лайонеллом? — спросил Павел Григорьевич. — Он вне игры?

— Да, — кивнул Аристарх, затушив папиросу. — У него обширный инфаркт. Не скажу вам точно, какой локализации, но он сейчас наверняка в реанимации. Вы же не просили его убивать, ведь так? — хакер спрашивал не ожидая ответа. — Вот я и сделал то, что сделал. Тяжелая болезнь, которая на продолжительное время выведет его из работы над конкурирующим проектом и не позволит в дальнейшем использовать возможности в полную силу. Теперь он будет бояться... Бояться компьютеров, виртуальности... Наверное, даже женщин и секса. Представляете, — он хохотнул, — вы трахаете девочку, а вас настигает инфаркт! Вот это супер. Просто, по Павлову, выработка рефлекса. Теперь, когда он вдруг увидит красивое черное белье на девочке с обложки глянцевого журнала, его скрутит приступ стенокардии!

Михаил тем временем собрал все, подобрал с пола остальные пакеты и поставил их к стенке.

— Так дело сделано? — спросил он, не обращаясь конкретно ни к кому. — Главный программист компании... Их компании... выведен из строя?

— Да, — кивнул Аристарх. — Выведен-выведен. Надеюсь, навсегда. Хотя лично я в этом нисколько не заинтересован. А вот как теперь вы будете с этим жить? Знать, что вы ЛУЧШИЕ только потому, что САМЫХ ЛУЧШИХ вывели из игры?

— Как-нибудь проживем, — тяжело сказал Павел Григорьевич. — Сколько уже можно терпеть то, что наши компании на мировом рынке ничего не стоят? Сколько вы навскидку вспомните русского софта, более или менее конкурентоспособного?

— Даже вспоминать не хочу, — крутанулся на месте Аристарх. — Как представлю, сколько этого самого конкурентоспособного софта является таковым только потому, что исходники у кого-то украдены, потому что программистов покупают, убивают, компрометируют. Потому, что кругом воровство, пиратство. Всюду такие, как вы, делают бизнес. Я никоим образом не оправдываю акул западного бизнеса, но уж больно все у нас грязно...

— К черту лирику, — отмахнулся Павел Григорьевич. — Я знаю про вас все. Вы — такой же, каким считаете и меня. Когда Сетевой Вампир нанес удар по вашей нервной системе, вы приняли решение бороться с ним до конца. Вот только никто тогда не поверил вашим бредням, вас упрятали в психиатрическую лечебницу, отлучили от компьютера... Не знаю, сколько всякой химии было пропущено через вас, но вы были признаны выздоравливающим и выпущены через полтора года — настолько серьезным было ваше состояние.

— Представьте себе состояние самих психиатров, когда существование сетевого Вампира было официально подтверждено, — огрызнулся Аристарх. — Но я не виню их. Разговоры об этом привидении, об этом монстре Сети всегда напоминали «Секретные материалы». Похоже, у вас чудесные осведомители.

— Стараемся, — продолжил Павел Григорьевич. — Поэтому не надо обвинять меня в смертных грехах. Выйдя из больницы, вы посвятили дальнейшую жизнь сетевой деятельности незаконного характера. Вы готовили себя к повторной встрече с Сетевым Вампиром, чтобы на этот раз выйти из нее победителем, отомстив за свою инвалидное кресло и тягостное существование. И на этот раз в процессе подготовки вы не гнушались ничем. Список ваших заданий и побед — это, конечно, не достояние выпуска новостей. Но, тем не менее, узнать кое-что было можно. Как вы живете после всех ваших гнусностей? Вы утопили в дерьме, в крови и безумии десятки публичных людей, вы изучали на них свои высокие технологии, свои «ноу-хау» вроде этого костюма, аналога которому я пока не знаю!

— Десятки? — усмехнулся Аристарх. — Да на одной этой маске тридцать шесть побед в виртуале, включая сегодняшний случай! Трое умерли после встречи со мной, восемь человек получили инсульты и инфаркты — вот как Лайонелл сегодня! У остальных те или иные виды поражений нервной системы — такое же, что Вампир сделал со мной! Параличи, немота и много всякой другой гадости! И все они, все они хотели жить и приносить пользу! Но такие, как вы, считали, что от них только вред, и приходили ко мне делать заказы!

— Тридцать шесть... — прошептал Михаил и зачем-то оглянулся на дверь. — А всего-то сколько?

— Не считал, — зло ответил Аристарх, не глядя на того, кто задал вопрос. — У меня нет никакой бухгалтерии, если вы думаете, что я могу заглянуть в приходно-расходную книгу и узнать, сколько человек заплатили мне гонорары за мою работу! Когда у вас нет ног, вам становится совершенно все равно, сколько их и какие деньги они несут! Я, как видите, рад пакету яблок и килограмму пельменей просто потому, что не могу выйти на улицу! В этой стране все наперекосяк: лифта нет, по лестнице не скатишься! Я не видел улицы уже много лет, и все благодаря сетевым войнам!

— Не пытайтесь вызвать во мне жалость, — Павел Григорьевич встал и подошел к Аристарху вплотную. — Между мной и вами нет разницы, мы сейчас вместе довели человека до инфаркта. Я — потому что я так хотел, вы — потому что это умели. Я знаю, что таких, как вы, в мире много — тех, кто получил удар от Сетевого Вампира и до сих пор не может прийти в себя. Жизнь их растоптана и смята, они больны, несчастны, брошены. Но они пытаются бороться. Некоторые из них погибают, снова и снова сражаясь с этим монстром, а ведь это всего лишь человек... Такой же, как вы, но только с гипертрофированным чувством виртуала. Он атакует вас и питается вашим здоровьем, вашими эмоциями, вашими непрожитыми годами. Боритесь — и пусть даже мой пакет яблок поможет вам в этой борьбе. Вы хотели увидеть недостающие цифры на том листочке? Получите.

Павел Григорьевич вынул из кармана «Паркер» и дописал их по памяти. Положив листок возле компьютера, он сказал на прощание:

— Мы в расчете? Желаю вам... Искренне желаю вам победы в поединке. Думаю, что в суровом мире бизнеса я еще не раз о вас вспомню. Вы нужны мне живым. С огромной неохотой даю вам сейчас этот адрес — плод длительных поисков и серьезных финансовых вложений. Найти его было трудно, но все-таки реально. Понимаю, что за другую цену вы не согласились бы, но очень хочу встретиться с вами вновь. Если останетесь в живых, позвоните мне. Прощайте.

И они ушли.

Аристарх сидел в кресле и смотрел в окно неподвижным взглядом.

— Кругом сплошное лицемерие, — произнес он тихо, когда его гости уже сели в черную «Волгу» и отъехали от дома. — Прийти, заказать человека, а потом распинаться в любви и человеколюбии. Я ему нужен явно не для философских бесед. Такие люди — им стоит только начать устранять конкурентов уголовными методами... Потом уже не остановишь.

Он немного отъехал от окна в сторону стола и посмотрел на листок с сетевым адресом.

— Но ведь тут есть и другая сторона, — сказал он сам себе. — Чем меньше таких, как я, тем выше вероятность, что в следующий раз придут ко мне. Принцип Маклауда соблюдается везде в этой жизни, в том числе в виртуале. ДОЛЖЕН ОСТАТЬСЯ ТОЛЬКО ОДИН.

Он погладил руками колеса коляски, встал, сделал несколько упражнений, прошел на кухню и сделал себе кофе. Из пакетов, принесенных гостями, он достал продукты, забил ими холодильник, вернулся в комнату и поставил к столу вместо коляски табуретку.

— Ну, какой еще идиот рискнул назваться моим именем? — спросил он у компьютера, надевая маску. — Людям никогда не давала покоя чужая слава. И Сетевой Вампир, глядя в листок с адресом, начал очередную атаку

заполни анкету — сделай журнал лучше

ЕСЛИ ТЫ ХОЧЕШЬ ПОМОЧЬ НАМ ДЕЛАТЬ ЖУРНАЛ, ВСТУПАЙ В ФОКУС-ГРУППУ СПЕЦА! УЧАСТНИКИ ФОКУС-ГРУППЫ СМОГУТ ПЕРВЫМИ ОЦЕНИТЬ ПРЕДСТОЯЩИЕ НОВОВВЕДЕНИЯ, ВЫСКАЗЫВАТЬ СВОЕ МНЕНИЕ О КАЖДОМ НОМЕРЕ НАПРЯМУЮ РЕДАКЦИИ. ОТ ТЕБЯ ТРЕБУЕТСЯ НЕМНОГО: БЫТЬ В ОНЛАЙНЕ, ПЕРИОДИЧЕСКИ ОТВЕЧАТЬ НА ВОПРОСЫ РЕДАКЦИИ И, САМОЕ ГЛАВНОЕ, ЖЕЛАНИЕ. ЧТОБЫ ПОПАСТЬ В ФОКУС-ГРУППУ, НУЖНО ВСЕГО ЛИШЬ ЗАПОЛНИТЬ ЭТУ АНКЕТУ И ПРИСЛАТЬ ЕЕ НАМ. ЕСЛИ ТЫ НЕ ХОЧЕШЬ БЫТЬ В ТЕСТ-ГРУППЕ, ВСЕ РАВНО ПРИШЛИ АНКЕТУ — НАМ ЭТО ОЧЕНЬ ВАЖНО!

Заполненную анкету присылай по адресу:
101000, Москва, Главпочтампт, а/я 654, Хакер Спец,
с пометкой «Анкета» или на vote@real.xakep.ru.

_03.06(64)

Давно ли ты читаешь «Хакер Спец»?
- ○ с первых номеров
- ○ около года
- ○ несколько последних номеров
- ○ первый раз

Как ты считаешь, изменился ли «Хакер Спец» за последнее время?
- ○ да, улучшился
- ○ да, ухудшился
- ○ нет, по-моему, не изменился

Какой из последних номеров тебе понравился больше всего?
- ○ 12.05(61) — Электронные деньги
- ○ 01.06(62) — Backup
- ○ 02.06(63) — Tuning
- ○ 03.06(64) — Game coding

Понравился ли тебе новый дизайн СПЕЦа?
- ○ да
- ○ нет
- ○ не обращаю внимания на дизайн

Хотелось бы тебе новых рубрик в ОФФТОПИКе?
- ○ да
- ○ нет

Достаточно ли объемна ТЕМА НОМЕРА?
- ○ вполне
- ○ ее нужно увеличить
- ○ слишком большая

Какие журналы ты читаешь, кроме СПЕЦа?
- ☐ Хакер
- ☐ CHIP
- ☐ CHIP Special
- ☐ Компьютерра
- ☐ Upgrade
- ☐ Мир ПК
- ☐ Upgrade Special
- ☐ другой(ие)

_____

Какой оптический привод в твоем компьютере?
- ○ CD-ROM/CD-RW
- ○ Combo CD-RW/DVD-ROM
- ○ DVD-ROM/DVD-RW

Часто ли ты бываешь на xakep.ru?
- ○ постоянно
- ○ иногда
- ○ очень редко
- ○ никогда не был

Предложи тему для очередного номера:

_____

о себе

ФИО _____
Где ты живешь _____
E-mail _____
Сколько тебе лет _____
- ○ меньше 17
- ○ 18-20
- ○ 21-23
- ○ 24-27
- ○ 28-30
- ○ 30-33
- ○ больше 33

Твое семейное положение
- ○ холост
- ○ женат

В каком вузе ты учишься (учился)
- ○ техническом
- ○ гуманитарном
- ○ я не учусь в вузе

Связана ли твоя работа с ИТ?
- ○ да
- ○ да — планирую работать в ИТ
- ○ нет
- ○ я не работаю

Твой средний месячный доход?
- ○ меньше $100
- ○ $100-300
- ○ $300-700
- ○ больше $700

Сможешь ли ты сам собрать компьютер?
- ○ с закрытыми глазами
- ○ по книжке
- ○ сомневаюсь

Какой у тебя канал в интернет?
- ○ выделенка
- ○ dial-up
- ○ нет интернета

Чем ты пользуешься для общения в Сети?
- ☐ e-mail
- ☐ чаты
- ☐ ICQ и другие мессенджеры
- ☐ другое _____

На каком языке ты пишешь?
- ☐ Assembler
- ☐ C/C++
- ☐ Pascal/Delphi
- ☐ Basic/VB
- ☐ Perl
- ☐ другое _____
- ☐ я не программер

С какими платформами у тебя есть опыт работы?
- ☐ PC (Windows)
- ☐ *nix (Unix, Linux, BSD)
- ☐ Macintosh
- ☐ Palm OS
- ☐ Pocket PC (Windows CE)
- ☐ EPOC/Symbian
- ☐ другое _____

Какие из перечисленных вещей у тебя есть?
- ☐ DVD-плеер
- ☐ DVD-ROM
- ☐ MP3-плеер
- ☐ ноутбук
- ☐ домашний кинотеатр
- ☐ мобильный телефон
- ☐ КПК (коммуникатор)
- ☐ цифровой фотоаппарат
- ☐ цифровая видеокамера
- ☐ GPS-навигатор

☐ **Да, я хочу в фокус-группу!**

Попробуйте подписаться в редакции, позвоните нам.

(это удобнее, чем принято думать :)

(game)land


SYNC


Лучшие цифровые камеры


Хакер


Хакер Спец


Железо


Страна Игр


PC Игры


CyberSport


Мобильные компьютеры


Total DVD


DVD Эксперт


Total Football


Onboard


Mountain Bike Action


Хулиган


Свой бизнес

★ Для подписчиков в Москве курьерская доставка в день выхода журнала
★ Дешевле, чем в розницу
★ Гарантия доставки и замены в случае потери
★ Специальные предложения для подписчиков
★ Первый номер подписки высылается по звонку вместе с заполненной квитанцией для оплаты

780-88-29 (для Москвы)
8-800-200-3-999 (для России)
ВСЕ ЗВОНКИ БЕСПЛАТНЫЕ
Мы работаем с 9 до 18 по рабочим дням

УЖЕ В ПРОДАЖЕ

Тестирование новейших моделей КПК, ноутбуков и сотовых телефонов

Яблочная религия
Тест ноутбуков на Mac OS

Перечитывая Symbian
Обзор возможностей Symbian S60 (2nd edition)

Софт для КПК на Linux
И от бабушки ушел, и от дедушки ушел...

Великие топографы
Карта для OziExplorer своими руками

Шаг за шагом
- Устанавливаем в iBook G4 модуль Wi-Fi
- Заменяем жесткий диск ноутбука PowerBook G4
- Делаем Mac из PC
- Собираем новостные рассылки у себя на КПК
- Scumm Virtual Machine
- Организовываем прием почты на КПК
- Vito ButtonMapper
- Spb Weather
- Мобильный Интернет от «МегаФон Москва»
- Архивируем Интернет при помощи iSilo
- Изучаем возможности Battery Pack Pro

Мобильные компьютеры