FPS

26
2013

Tube Open Movie
Интервью с Бассамом Курдали

GIMP: Ломо-эффект как в Instagram

Физический движок своими руками. Часть III

Генерируем случайные уровни
Осваиваемся в SDL2

Корпорация зла
Почему у Microsoft нет будущего

...и многое другое!

Независимый электронно-познавательный журнал. Издается с 2008 г. Доступен по CC-BY-NC-SA.

FPS №26

FPS – бесплатный, свободно распространяемый электронный журнал, посвященный разработке компьютерных игр и сопутствующей тематике.

FPS охватывает широкий круг тем: на страницах журнала рассматриваются вопросы программирования игр с использованием разнообразных движков и графических библиотек, публикуются материалы по двумерной и трехмерной компьютерной графике, включая уроки по популярным графическим пакетам и редакторам, а также различные статьи по теоретическим вопросам, дизайну и философии компьютерных игр.

Журнал издается с января 2008 г. и на данный момент выходит раз в два-три месяца.

Главный редактор: **Тимур Гафаров**
Дизайн и верстка: **Тимур Гафаров**
Обложка: **Алия Тятигачева**

По вопросам сотрудничества обращайтесь по адресу:
gecko0307@gmail.com

Blender Новости

25-27 октября в Амстердаме состоится XXII международная Конференция Blender. Уже в пятый раз подряд она будет проходить в здании старинного театра Де Балье в самом центре города. Программа конференции состоит из презентаций и мастер-классов от ведущих специалистов по кинематографу, мультипликации, 2D/3D-графике и разработке игр. Свои разработки представят ученые из крупных университетов, одиночные программисты и различные организации со всего мира.

До недавних пор Blender для создания игр использовали, главным образом, OpenSource-разработчики и инди-команды. Однако сейчас в сторону этого пакета смотрит все больше крупных студий, в числе которых – сама Valve. Представитель компании обратился к разработчикам Blender с вопросом о возможности распространения программы через сервис Steam: в долгосрочной перспективе Valve рассматривает Blender как предлагаемую для своих пользователей платформу для создания модов.

Steam уже предоставляет средства для быстрого поиска, обмена и установки модов – аналогично Blender может использоваться для подготовки элементов модов. Valve желает обеспечить в Blender возможность быстрого внесения изменений и их публикации для пользователей Steam нажатием одной кнопки.

Для начала планируется обеспечить бесплатную поставку через Steam штатной сборки Blender – той же, что распространяется с официального сайта. Этот шаг позволит привлечь пользователей и разработчиков к созданию всех необходимых плагинов и инструментов бесшовного взаимодействия с сервисами Steam.

Тем временем разработчики Blender приняли важное и для многих ставшее неожиданным решение: перевести рендер-движок Cycles с копилефт-лицензии GPL на пермиссивную Apache 2.0. Это было мотивировано желанием обеспечить лицензионную совместимость со сторонними библиотеками и продуктами.

Cycles изначально развивается как независимый и готовый для использования в других продуктах движок рендеринга. В отличие от GPL, лицензия Apache позволяет использовать Cycles с коммерческими и закрытыми проектами, а также использовать наработки и модифицированные варианты движка в своих системах, без необходимости открытия кода внесенных изменений. При этом код под лицензией Apache остается совместим с кодом GPL и может использоваться в составе GPL-проектов.

Кроме того, перевод Cycles на пермиссивную лицензию можно рассматривать как ответный шаг навстречу сторонним открытым разработкам, которые уже используются в Blender или которые планируется задействовать. Например, под открытыми пермиссивными лицензиями доступны такие библиотеки, как OpenEXR, Open Shading Language, OpenSubdiv, PTex и OpenVDB. Cycles поддерживает интеграцию с большинством из указанных библиотек – использовать данные системы и в ответ предъявлять более жесткие требования по использованию своего кода выгладит не совсем корректно, поэтому решено применить аналогичные условия распространения и для Cycles.

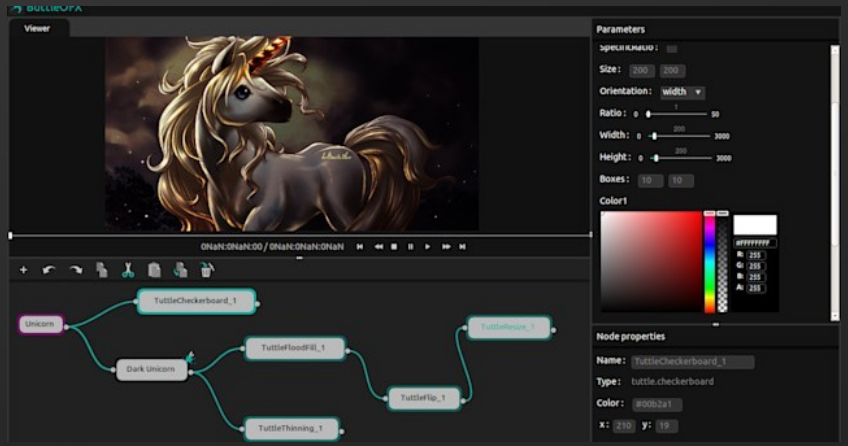
Изменения затронули только Cycles – остальной собственный код Blender, как и прежде, поставляется под лицензией GPL, и никаких изменений в этом плане не предвидится.

А что же нового в мире сторонних рендеров для Blender? Не так давно стало известно, что авторы OctaneRender – коммерческого рендер-движка, над которым, кстати, работал создатель Cycles Брехт ван Ломмель – выпустили бета-версию своего плагина для Blender. Плагин также представляет собой коммерческий продукт – стоимостью €99. Доступны версии для Windows 64-bit и Mac OS X (о поддержке Linux, увы, пока ни слова).

OctaneRender – признанный многими профессионалами CG-индустрии быстрый физически корректный рендер, имеющий поддержку GPU и пользовательский интерфейс на основе узлов. Представленный плагин обеспечивает высокую степень интеграции в Blender и тот же уровень инерактивности при работе над сценой, материалами и освещением, что и в случае со встроенным GPU-движком Cycles.

Мощный и гибкий интерфейс узлов (нодов) – одна из ключевых особенностей Blender, за которую программа получила широкую популярность как удобный инструмент композитинга. До сих пор Blender'у в этом отношении не было равных в мире СПО: создание специализированного открытого софта для композитинга до сих пор было сообществу не по зубам – предыдущие попытки (Ramen и Synapse) успешными назвать никак нельзя. Однако не все смирились с такой ситуацией: французские программисты из инженерного института IMAC работают над проектом **ButtleOFX**.

У программы достаточно серьезная основа – открытый фреймворк **TuttleOFX**, основанный на стандарте OFX. TuttleOFX активно разрабатывается студией спецэффектов Mikros Image; в его создании на раннем этапе также поучаствовали студии Duran Duboi и HD3D SAS. Главное преимущество использования этого фреймворка – автоматическая поддержка плагинов OFX. Например, команда успешно протестировала в ButtleOFX работу плагинов Sapphire.



Впрочем, минимальный полезный набор эффектов есть в самой программе, поскольку TuttleOFX поставляется с собственным набором модулей OFX. Это различные инструменты цветокоррекции на основе Color Transform Language, OpenColorIO LUT и т.д., всякие полезные фильтры вроде подавления шума и размывания, геометрические преобразования, текст.

Для ввода и вывода ButtleOFX использует третьесторонние библиотеки, подключаемые тоже через плагины: OpenImageIO, ImageMagick, LibRaw, OpenJPEG, OpenEXR, TurboJPEG. Для поддержки DPX в программе с нуля написан собственный код, а для поддержки видео применяется Libav. Пользовательский интерфейс написан на QML – декларативном языке Qt.

И все-таки: что насчет поддержки OFX в Blender? Вот что говорит по этому поводу Йерун Баккер: «В ходе переписывания композитора на тайлах мы изучили OFX. Как платформа для композитинга этот стандарт показался нам неплохим, но мы наткнулись не некоторые проблемы с API. Впрочем, дело было в 2010-2011 годах – с тех пор все могло измениться... На тот момент OFX работал только на CPU, и планировщик у него работал на нодах, в то время как мы выбрали планировщик на тайлах и одновременное использование CPU и GPU. Так что пока OFX не «подтянется» к этим требованиям, его поддержка в Blender особого смысла иметь не будет...»

Хорошая новость для тех, кто соскучился по анимационным короткометражкам в Blender: Вакас Маджид и международная команда художников и аниматоров выпустили тизер мультфильма **«Luke's Escape»**. Одновременно с этим запущена кампания по финансированию фильма.

Судя по описанию сюжета и тизеру, нас ждет триллер – идея сюжета, как говорит Вакас, родилась из написанной им поэмы. Люк, обычный мальчишка, видит один и тот же ночной кошмар. Во сне он оказывается на странной фабрике, где каждый выход охраняется вооруженными охранниками. Наконец он ухитряется сбежать – но что, если во тьме окружающего фабрику мертвого леса водятся существа пострашнее? Сможет ли Люк совладать со своими страхами?..

Проект, помимо прочего, интересен своей командой. Арт-директором в нем работает Рейнант Мартинес, который последние полгода не вылезает с первой страницы BlenderNation. Экспертом широкого профиля по CG в команде является Оливер Виллар Дис. Концепт-арт рисует Брюс Машбат, моделированием занимается Даниель Кройтер, анимацию взял на себя Клаудио Эспинар Руис.

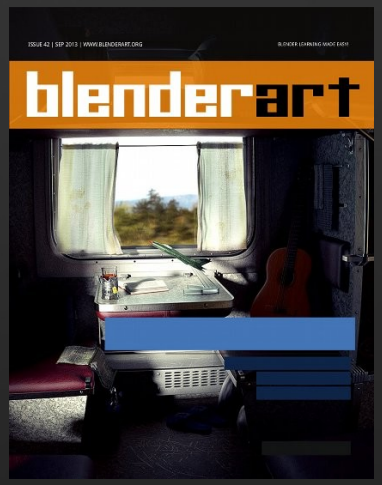
В работе художники активно пользуются двумя инструментами: GIMP и Blender. 95% кадров рендерится в Cycles. В течение последнего года авторы уже выкладывали несколько учебных видеороликов, построенных на материалах тизера. Сейчас команда рассматривает возможность публикации исходных материалов проекта под свободной лицензией после выпуска мультфильма.

К 11 сентября команда надеется собрать $17000 – эти средства пойдут на оплату работы художников и аниматоров. В случае успешного краудфандинга, согласно текущим оценкам, команда сможет завершить работу над фильмом к середине следующего года. Поддержать проект деньгами все желающие могут **на IndieGoGo**.

Кстати, если вы используете Blender, и вам есть что показать миру, добро пожаловать на сайт **Blender-Projects.com** Это развивающийся коммьюнити-портал для обмена конентом и презентации работ – таких, как рендеры, видеоролики, модели, дополнения, учебные материалы и т.д. Можно добавлять контент с Youtube, Vimeo и Sketchfab, создавать команды и вести дневники разработок. На данный момент сайт находится на стадии открытого бета-тестирования.

Завершается работа над книгой «Art of Blender» – альбомом с лучшими работами, созданными при помощи этого пакета, который планирует издать сайт **CG Cookie**. В книге будут представлены кадры из блендеровских фильмов: Big Buck Bunny, Sintel, Tears of Steel, Project London, Caminades, Tube Project (ныне уже известный как Wires for Empathy) и множества других, наряду с галереей 50 известных художников, среди которых Эндрю Прайс, Лукас Фалькао, Рейнант Мартинес, Джош Мауль, Нита Равалджи, Джонатан Лампел и другие.

Проект отчасти благотворительный – 5% дохода от продаж книги через магазин Blender пойдут в Blender Foundation. Поступление альбома в продажу ожидается в ноябре этого года. Предзаказ обойдется вам с 10-процентной скидкой – в €37.80 (обычная цена составляет 42 евро).



А в сентябре вышел 42 номер замечательного журнала **«BlenderArt»** – самого популярного анлгоязычного интернет-издания, посвященного Blender.

В этом выпуске читателей ждет много полезной информации: уроки, статьи жара «making-of», галерея работ и другие материалы. Номер вышел небольшим – зато в некоторой степени сменил дизайн.

Журнал «FPS» отслеживает все самые свежие новости из мира Blender, моделирования, анимации и рендеринга! В следующем номере ждите очередную подборку новостей. Оставайтесь с нами и держите руку на пульсе последних событий!

TUBE Open Movie

Интервью с режиссером Бассамом Курдали

Не так давно на сайте BlenderNetwork.com было опубликовано интервью с Бассамом Курдали – анимационным режиссером, который известен публике по фильму «Elephants Dream» («Мечта слонов») 2006 года.

Эта короткометражка стала первым полностью открытым фильмом, созданным целиком при помощи СПО. Бассам родился в Дамаске, получил образование в США и в данный момент проживает в Массачусетсе. Сейчас он со своей командой URCHN работает над новым шедевром: фильмом «Wires for Empathy», ранее известным под кодовым названием «Tube Open Movie». Этим летом на YouTube появился свежий трейлер проекта – по всей видимости, следует ожидать релиза в самом ближайшем будущем!

Бассам, каково твое образование и профессиональный стаж?

Я обладаю тем, что называется междисциплинарной подготовкой – технической в области разработки программного обеспечения и творческой в качестве режиссера анимационных фильмов. Сейчас я занимаюсь режиссурой и всем, что связано с компьютерной графикой. Я режиссер, сценарист и концептер в «Wires for Empathy», также занимаюсь подготовкой программных инструментов проекта. Я, в какой-то мере, пробовал себя везде – от моделирования до анимации и проработки освещения.

Каков масштаб производства у проекта?

Эпический. Для нашего небольшого бюджета фильм имеет огромный размах во всех отношениях: множество сцен, несколько моделей героев, толпы, фоновая анимация, VFX. Работа над фильмом ведется с 2008 года. Единовременно над ним работают от 5 до 10 человек – кто-то локально, кто-то удаленно. Помимо основной команды, у нас около 60 сторонних участников.

Как влияет на производство использование свободного ПО?

Динамика производства в значительной степени напоминает СПО-проект и мало похожа на традиционную работу по созданию мультфильма – можно рассматривать нашу команду как контрибьюторов, каждый из которых работает над своей частью проекта и делает коммиты в наш SVN-репозиторий. Факт использования нами СПО – это одновременно и источник вдохновения, и средство обеспечения работ такого типа: нет барьеров, которые бы помешали художникам делать свой вклад – таких, как лицензионные ограничения, особенности платформ и т.д. При этом весь ход производства открыт для публики и не является какой-то корпоративной тайной.

Какая часть проекта создается в Blender?

Практически вся трехмерная графика. Однако мы не настаиваем, чтобы моделлеры использовали именно Blender (хотя большинство используют именно его), так как импорт моделей – тривиальная задача. Анимация, освещение, композитинг и видеомонтаж делается в Blender. Для 2D-графики мы используем Krita, MyPaint, GIMP, Inkscape. Кроме того, мы используем СПО и для управления проектом - Helga, SVN, Attract и др.

Есть ли проблемы, с которыми вам приходится сталкиваться?

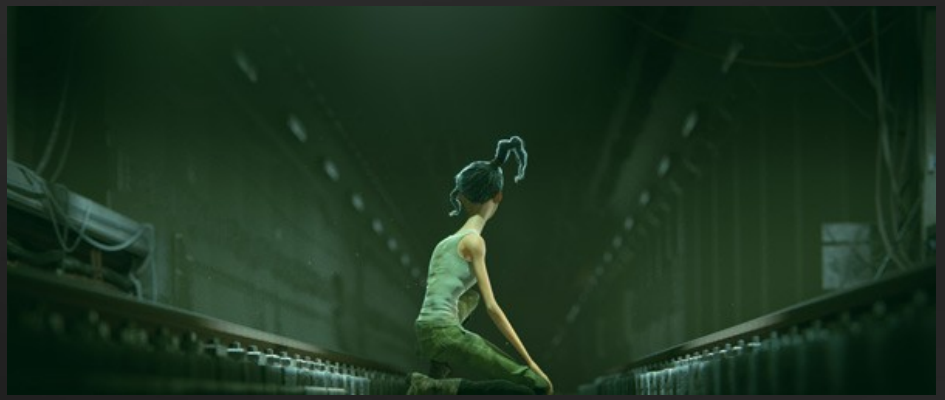
Бывают время от времени. На любом производстве случаются аварии, мелкие или крупные. И потом, наш проект длится уже довольно долго, и файлы, созданные в первой примененной нами версии Blender, должны быть совместимы и с новыми версиями – вплоть до финального рендеринга. Это распространяется на все: на скрипты, модели, анимации и т.д. Со временем это перестало быть крупной проблемой, так как мы используем инструменты автоматизации. Другой важной задачей является моральная поддержка команды – очень важно показать людям, сколько этапов уже пройдено, чтобы работа не была столь изнурительной. Выпуск трейлера стал для всех нас важным моментом – команда, наконец, увидела хоть что-то законченное.

У вас есть релиз-план?

В принципе, да – мы постепенно идем к цели. Если удастся сохранить достигнутый темп работы, можно ожидать релиза в начале 2014 года.

Какова будет приблизительная себестоимость производства?

Трудно, конечно, оценить в долларах то, за что мы не платили – но, я думаю, сумма где-то в районе от 100 до 200 тысяч. Я учитываю стоимость технической стороны проекта, которую нам обеспечил Хэпмширский колледж (студия, рабочие станции, рендер-ферма, IT-поддержка, электропитание и т.д.), средства от фандрейзинга, время и труд команды, различные накладные расходы и прочие мелкие затраты.



Скажите, пожалуйста, читателям что-нибудь напоследок!

Могу добавить одно: часто кажется, что работа над компьютерной мультипликацией сильно изолирует – вы с головой зарываетесь в монитор, полностью отрываясь от окружающего мира. Однако этот проект также породил и усилил множество связей между художниками и программистами, так что нельзя отрицать его коммуникативную составляющую. Так что спасибо всем, кто работает над «Wires for Empathy»!

Пожелаем Бассаму удачи в завершении его проекта! Вы можете следить за развитием этого эпического открытого фильма на сайте **http://urchn.org**, а также в Twitter: **https://twitter.com/tubeproject.**

Оригинал интервью:
BlenderNetwork.com

Перевод: ***Тимур Гафаров***
gecko0307@gmail.com

Обзор дополнений Blender

Выпуск 5

Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. Наш журнал отслеживает выход новых полезных дополнений для Blender 2.6x, которые могут заинтересовать пользователей, использующих программу в качестве инструмента для разработки игр или создания игрового контента.

Если вы разрабатываете собственное дополнение или просто нашли в Интернете чей-то интересный проект, будем очень рады, если вы напишете нам об этом и поделитесь ссылкой: ***gecko0307@gmail.com****, либо пишите в наше сообщество:* ***http://gplus.to/fps_community****.*

Text Scrambler

Дополнение реализует эффект скрэмблинга для объектов текста – то есть, подстановку случайных символов в строке фиксированной длины. Что самое интересное – результат можно анимировать и получить эффект «дешифровки»: постепенного подбора правильных букв из постоянно меняющегося хаоса символов. Естественно, необходимо при этом использовать моноширинную гарнитуру. Проект основан на известном скрипте Бассама Курдали Typewriter.

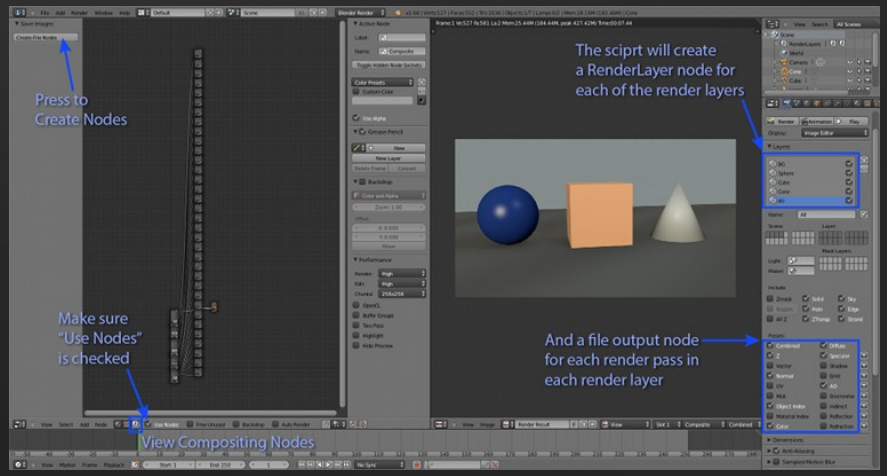
Автор: Martin Wacker

Ссылка на скачивание

Save Render Layers

Этот скрипт дает возможность сохранять слои рендеринга и данные проходов в отдельные файлы – штатный интерфейс Blender, как известно, позволяет единовременно сохранять в файл только одно изображение (собственно, результат рендеринга – текущий слой). Однако часто бывает нужно иметь такие данные, как альфа-канал, буфер глубины, карту нормалей отдельно от собственно данных о цвете пикселей – например, для постобработки и композитинга в GIMP, Photoshop или Nuke.

Дополнение позволяет существенно сэкономить время, так как с ним не нужно много раз повторять одну и ту же операцию по переключению режимов просмотровщика изображений или связей в редакторе узлов.



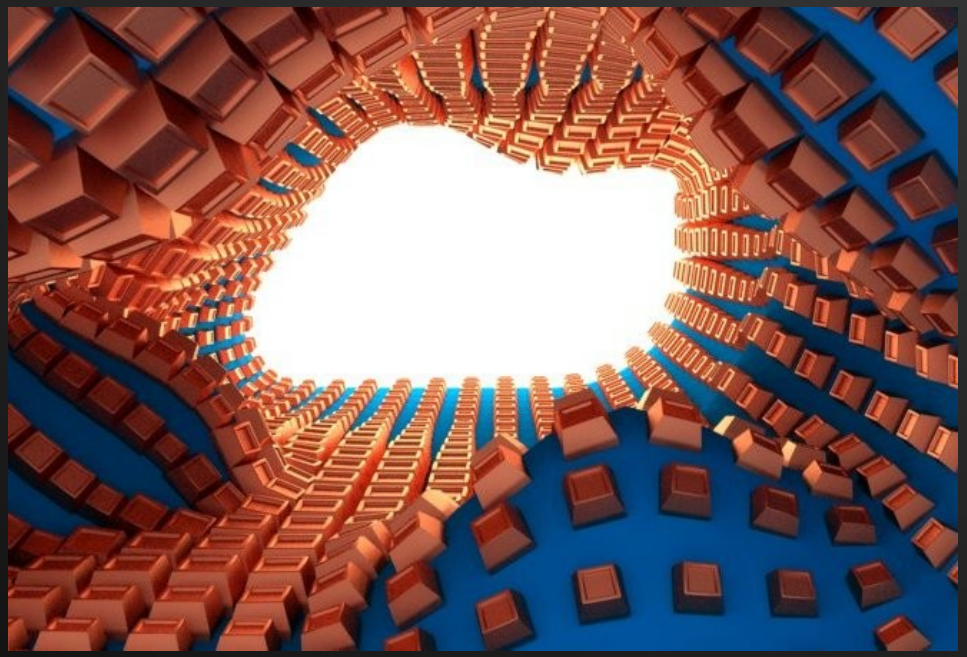
Автор: Tamir Lousky

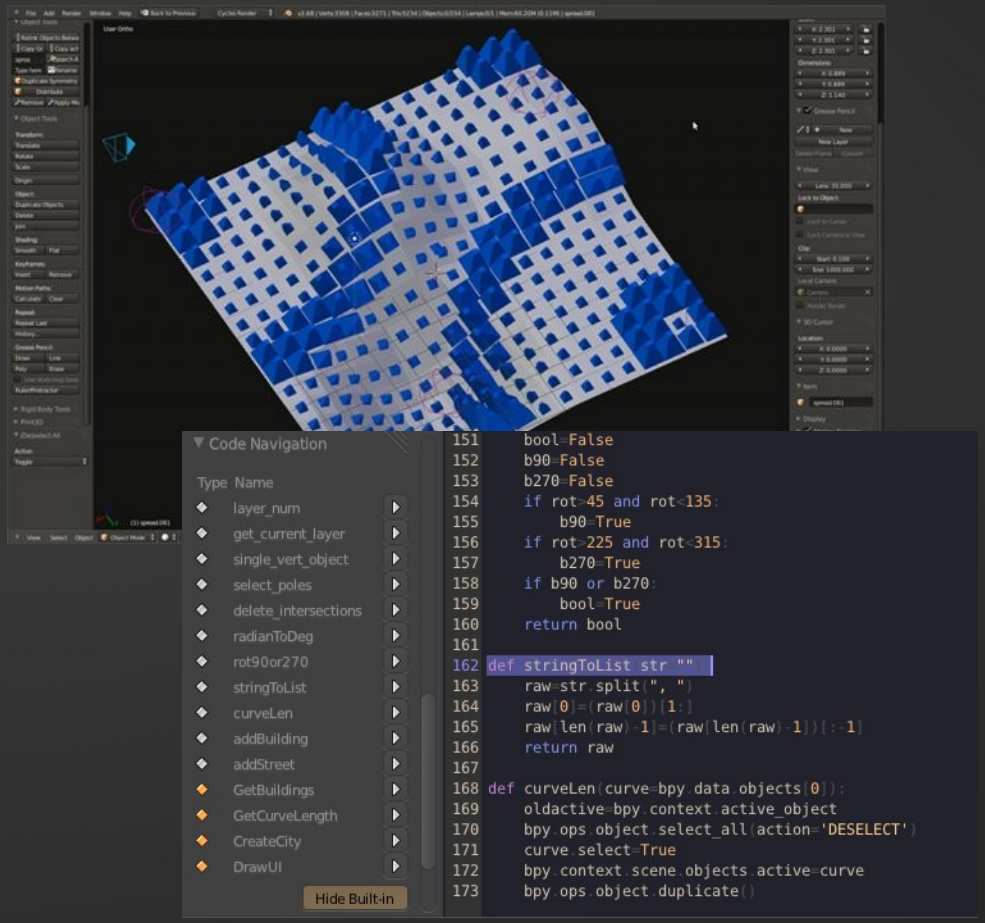
Ссылка на скачивание

Sverchok

«Сверчок» – разработка отечественных программистов, что вызывает определенное чувство гордости. Это скрипт для генерации параметрических массивов объектов – аналог инструмента Grasshopper для пакета Rhino. Название «Сверчок» как раз возникло по аналогии с «Кузнечиком». Он позволяет архитекторам и дизайнерам создавать сложные органические формы. Принцип работы «Сверчка» сводится к «распылению» копий объекта-донора на поверхности объекта-реципиента с учетом позиций объектов-аттракторов для управления масштабом копий. Аддон пока находится на ранней стадии развития – в будущем планируется добавить больше возможностей.

Авторы:
Никита Городецкий и Александр Недовзин
Сайт проекта:
http://nikitron.cc.ua/blend_scripts.html





Code Navigation

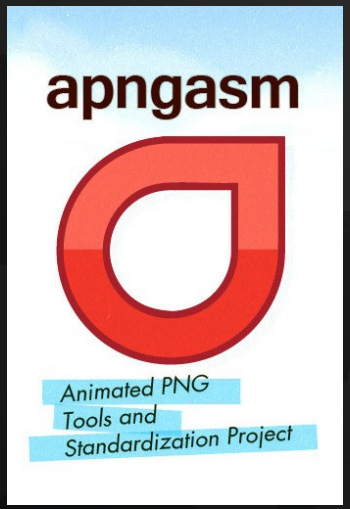
Скрипт навигации по Python-коду в текстовом редакторе Blender. Выводит список всех классов и функций в текущем исходнике, позволяет перемещаться к ним путем нажатия кнопки.

Автор: Greg Zaal
Ссылка на скачивание

GIMP: Новости

APNG в GIMP и Inkscape

Хорошая новость для мультипликаторов: в GIMP планируется поддержка формата APNG (анимированный PNG). Разработчик Рей Кагецуки включил Inkscape и GIMP в состав инструментов, которые могут получить экспорт в APNG при финансировании через Kickstarter. В проекте планируется использовать наработки из apngasm – консольного ассемблера APNG, разработанного Максом Степиным. Кстати, Макс собственной персоной участвует в этом проекте.



Планируется также подготовить пакеты для apngasm для Debian и Ubuntu Linux, инсталляторы для OSX и Windows, gem для Ruby (rapngasm) с нативным интерфейсом, а также создать кроссплатформенный графический инструмент авторинга APNG на основе apngasm с предпросмотром в режиме реального времени. Долгосрочных планов, как говорится, громадье: создание платного плагина поддержки APNG для Photoshop, Java-порт библиотеки с поддержкой Android, порт на JavaScript/CoffeeScript, расширения для Python и PHP и многое другое.

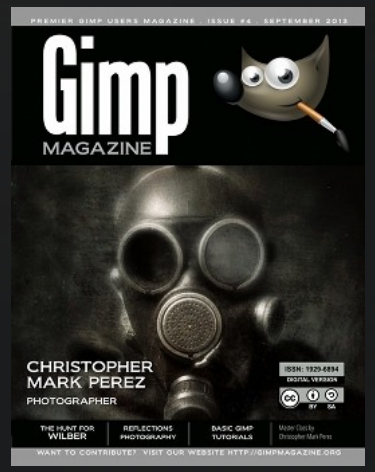
https://github.com/apngasm/apngasm

GIMP Magazine #4

В начале сентября вышел четвертый номер замечательного англоязычного журнала GIMP Magazine. Журнал, как всегда, радует интересным материалом, красочными иллюстрациями и качественным дизайном. В данном выпуске вы найдете интервью с фотографом-портретистом Кристофером Марком Перезом, уроки по виньетированию, созданию триптихов и созданию винтажного эффекта, галерею художественных работ и многое другое.

Журнал можно бесплатно скачать в формате PDF, прочитать онлайн на сервисе Issuu, купить в бумажном виде ($26) или в специальной версии для iPad ($5). PDF-версия журнала распространяется по лицензии CC-BY-SA.

Подробности на сайте проекта:
http://gimpmagazine.org



GIMP Фотолаборатория

LOMO-эффект как в Instagram

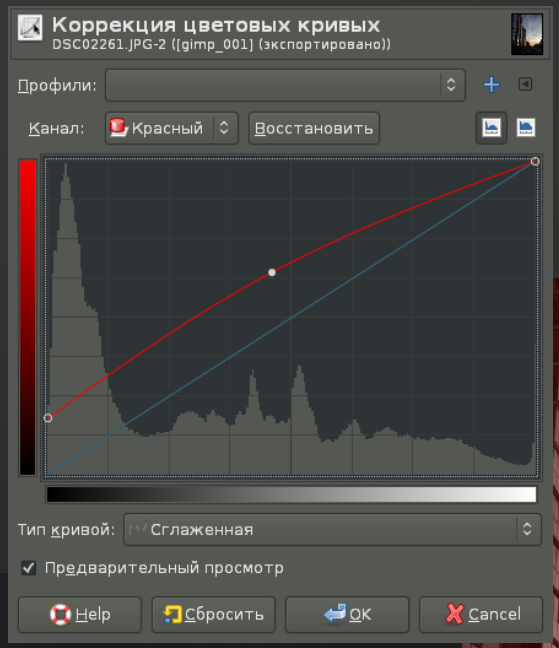
Здравствуйте, уважаемые читатели журнала и пользователи графического редактора GIMP! Добро пожаловать снова в рубрику «Фотолаборатория». Сегодня я поделюсь с вами очень простым приемом для воссоздания винтажного ломографического фильтра Nashville из коллекции Instagram. Кто-то скривится: «Попса!», а я скажу: интересный, приятный глазу эффект – если, конечно, им не злоупотреблять и использовать с умом.

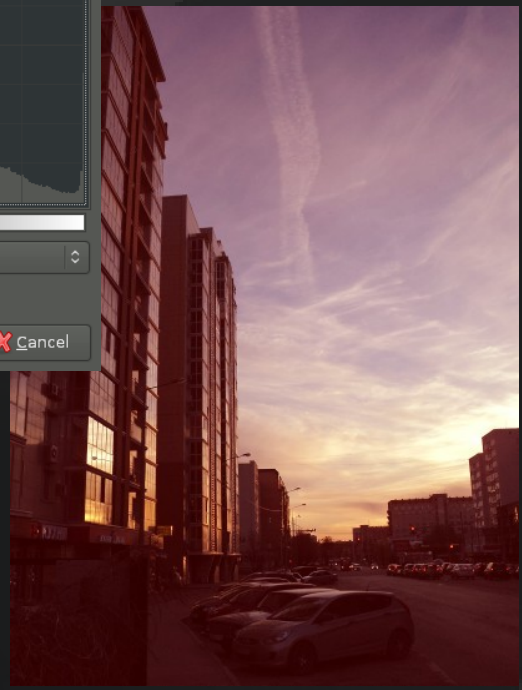
Ломография – это вид современного фотоискусства, который ставит своей целью запечатлеть на снимках жизнь во всех ее проявлениях, такой, какая она есть. Ломографы зачастую используют простые и даже примитивные фотоаппараты: здесь не обязательна точная цветопередача и аккуратная резкость, главное – поймать момент, найти необычный ракурс!

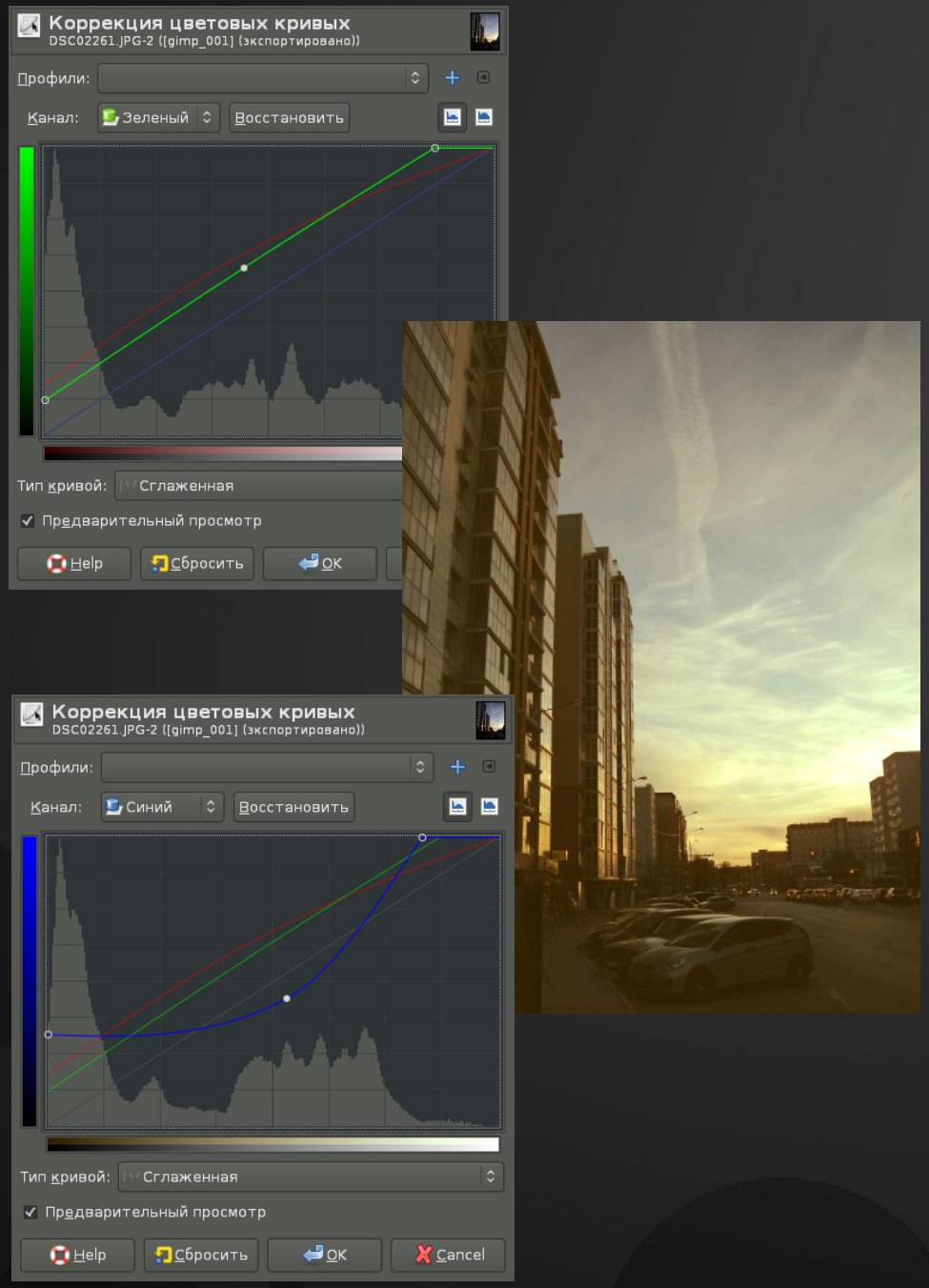
Впрочем, для этого урока я взял снимок, сделанный обычным фотоаппаратом – ломо-эффекты сейчас очень часто используются не только фотографами-экспериментаторами, но и рядовыми дизайнерами.

1. Все, что нам понадобится – это инструмент коррекции цветовых кривых (Цвет > Кривые...). Эффект делается путем настройки всех трех каналов – красного, зеленого и синего.

2. Выставьте кривые так, как показано на скриншотах. Конфигурация может слегка меняться – в зависимости от вкусовых предпочтений и характеристик фотографии.

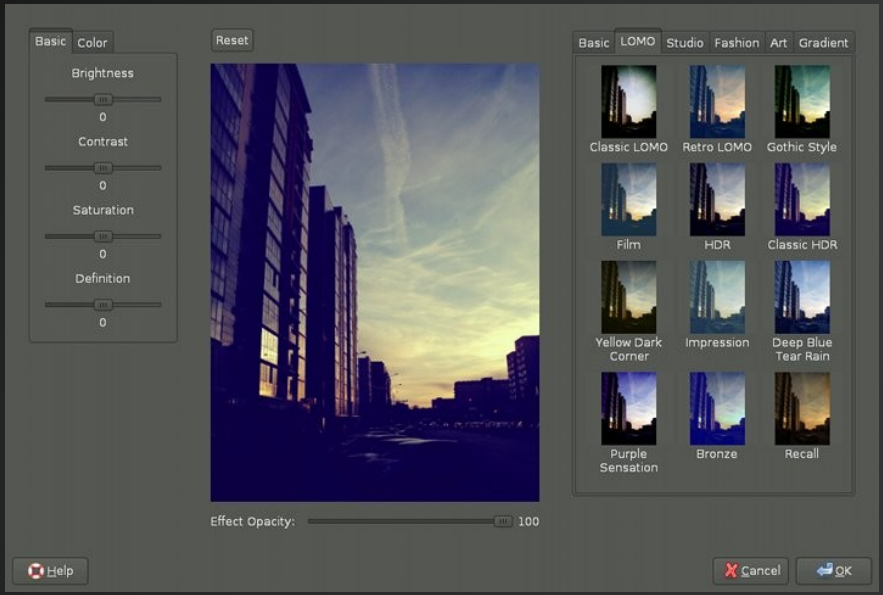






3. Собственно, вот и результат.

К слову, нечто похожее делает плагин Beautify (фильтр LOMO > Classic HDR), но, на мой взгляд, у нас получилось лучше.



4. Теперь можно добавить затемнение по краям – еще один распространенный фотоэффект. Оставляю эту задачу читателю для самостоятельного решения.

Леон Завальский

Blender

настольная книга

«Blender. Настольная книга» – это проект от журнала «FPS» по созданию полноценного русскоязычного электронного руководства по основам работы в Blender 2.6. Целевая аудитория – начинающие пользователи программы (как перешедшие со старых версий, так и начинающие знакомство с Blender «с нуля»). Книга будет представлять собой сборник статей, охватывающих различные аспекты использования Blender, скомпонованных по принципу «от простого к сложному».

Издание будет распространятся бесплатно, по лицензии Creative Commons BY SA. На данный момент активно ведется подготовка текста книги.

К работе над книгой приглашаются все желающие! На почтовый ящик редакции (**gecko0307@gmail.com**) принимаются статьи и уроки, а также общие советы и предложения. Кроме того, книге нужны графические материалы: авторские художественные работы, интересные скриншоты, демонстрационные рендеры, схемы, диаграммы и т.д. Весь Ваш вклад в книгу обязательно будет учтен, и Ваше имя будет указано в списке авторов.

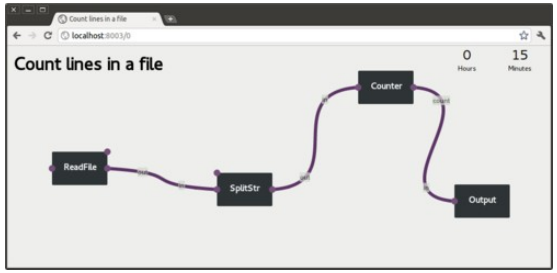
Язык D

Новости «с Марса»

свежие релизы и обновления

Dendrite

Dendrite – это реализация парадигмы потоково-ориентированного программирования (flow-based programming, FBP) на D. Основная идея FBP заключается в полном разделении логики программы на отдельные независимые «черные ящики» с портами ввода/вывода – таким образом, что граф исполнения программ может быть задан вне самих алгоритмов. Это значительно облегчает распараллеливание, а также обновление и сопровождение кодовой базы. «Низкоуровневые» программисты сосредотачиваются каждый на своем «черном ящике», в то время как системные архитекторы занимаются объединением этих компонентов в общую сеть.



Концепция потоков в Dendrite во многом напоминает конвейеры в UNIX, но есть и отличия. Сообщения, которыми обмениваются компоненты Dendrite, не обязательно представляют собой октеты, это могут быть данные любого типа: например, целочисленные значения или кадры видео. Кроме того, обмен сообщениями происходит путем передачи не самих данных, а ссылки на них. Наконец, приложение не обязано быть цепочкой-конвейером и может иметь любую желаемую топологию.

https://bitbucket.org/eris0xaa/dendrite
http://gplus.to/flowbased

Arrow – логическая игра на D

Мини-игра наподобие «Тетриса» – фактически, клон игры GuruGuru, втроенной в телефоны Pantech. Она отличается от классического «Тетриса» оригинальной механикой: вместо простых блоков здесь блоки-стрелки. Вы должны выстраивать линии из стрелок, указывающих в одном направлении. Есть четыре однонаправленные стрелки (вверх, вниз, вправо, влево), две двунаправленные (горзонтальная и вертикальная) и одна четырехнаправленная. Как только выстроена линия в 3 и больше стрелок, она исчезает, а все соседние с ней стрелки поворачиваются на 90 градусов. Есть также «бетонный» блок без направления, который исчезает только вследствие исчезновения его соседа.

https://github.com/gecko0307/arrow

dmech – физика на D

dmech – трехмерный физический движок для D. Проект находится на ранней стадии разработки и пока не вполне пригоден для коммерческого использования, но в данный момент это, по сути, единственная нативная реализация трехмерной динамики твердых тел на D. dmech поддерживает базовые геометрические тела (бокс, сфера, цилиндр, конус, эллипсоид) – этот список может быть расширен любыми конвексными объектами, заданными support-функцией. Есть поддержка сочленений типа Ball-Socket и Slider. Движок использует dlib в качестве математической библиотеки.

https://github.com/gecko0307/dmech

dlib 0.1.2

Коллекция библиотек dlib обновилась до версии 0.1.2. Нововведения касаются, в основном, пакета dlib.image: добавлена поддержка свертки изображений (dlib.image.filters.convolution), а также цветового пространства HSV, на основе которого реализованы эффекты Chroma Key («зеленый фон») и Color Pass (выборочное обесцвечивание). Кроме того, не так давно репозиторий dlib переехал на GitHub и обзавелся поддержкой пакетного менеджера DUB.

https://github.com/gecko0307/dlib
https://github.com/gecko0307/dlib/releases/tag/v0.1.2
http://code.dlang.org/packages/dlib

Anchovy

Anchovy («Анчоус») – набор мультимедийных библиотек для создания игр и графических приложений на основе dlib и Derelict3. Включает средства отрисовки GUI поверх OpenGL. Планируется также поддержка аудио, клиент-серверный сетевой фреймворк и инструменты локализации. Находится на стадии активной разработки и не рекомендуется автором для использования в крупных проектах – API может меняться от версии к версии. Anchovy работает под управлением Windows, поддержка Linux и Mac OS X не заявлена, но библиотека должна собираться и под эти платформы.

https://github.com/MrSmith33/anchovy

FewDee – 2D-библиотека

FewDee – это экспериментальная система разработки 2D-игр и игровых прототипов. Использует небезызвестную библиотеку Allegro. Проект находится на ранней стадии разработки, но может заинтересовать пользователей Allegro.

https://bitbucket.org/lmb/fewdee
https://github.com/lmbarros/FewDee

glad – еще один биндинг OpenGL

Больше биндингов OpenGL, хороших и разных! По существу, glad – это генератор биндингов; то есть, он позволяет вам выбрать любую версию API и выдает загрузчик только для нее – ничего лишнего. Это особенно полезно сейчас, когда одновременно существуют несколько версий OpenGL (1.x, 2.x, 3.x и 4.x). Заявлена поддержка всех версий, включая новейшую 4.4, и всех без исключения расширений, что выгодно отличает эту библиотеку на фоне того же Derelict3, где оставлена только «ванильная» часть стандарта.

https://github.com/Dav1dde/glad

Биндинги под Derelict

Позитивная новость: в публичном доступе появилось несколько неофициальных Derelict-биндингов к C-библиотекам: в частности, OpenCL, BASS, FANN и libsndfile. Для FANN (реализации нейронной сети) и libsndfile (библиотеки чтения и записи различных аудиоформатов) доступны также объектно-ориентированные врапперы. Биндинги доступны в репозитории пакетов DUB: **http://code.dlang.org**

Specd – unit-тестирование на человеческом языке

Библиотека Specd позволяет писать unit-тесты не с помощью выражений assert, как обычно, а с помощью спецификаций, описанных особыми семантическими конструкциями, которые напоминают литературный английский – например, "foo".length.must.equal(3). Библиотека чем-то похожа на фреймворки Spec2 и ScalaTest для языка Scala.

https://github.com/jostly/specd

DScanner

Анализатор исходного кода на D. Умеет выводить AST в формате XML, проверять синтаксис и выводить сообщения об ошибках, форматировать исходники в HTML с подсветкой синтаксиса, подсчитывать количество строк кода и т. д.

https://github.com/Hackerpilot/Dscanner

Gumbo-d

Не успела Google выложить в публичный доступ свой HTML-парсер Gumbo, как уже появился биндинг к ней для языка D! Как отмечается, библиотека может быть использована совместно с веб-фреймворком (например, vibe.d) в качестве движка шаблонов.

https://github.com/bakkdoor/gumbo-d

cassandra-d

Анонсирован cassandra-d – клиент для распределенной СУБД Apache Cassandra с поддержой языка запросов CQL (Cassandra Query Language). Реализована спецификация протокола CQL 1.0. Для справки: Cassandra была разработана в недрах Facebook и впоследствии передана в фонд Apache Software Foundation для дальнейшего развития проекта. Промышленные решения на базе Cassandra используют такие крупные компании, как Cisco, IBM, Reddit, Digg, Rackspace и Twitter.

https://github.com/rjmcguire/cassandra-d

DSWS – веб-сервер на D

Тех, кому vibe.d кажется чересчур громоздким, может заинтересовать DSWS – маленький простой веб-сервер на D.

https://github.com/gedaiu/DSWS

XML-RPC на D

Наш соотечественник Павел Кириенко реализовал на D клиент и сервер, работающие по протоколу XML-RPC (XML Remote Procedure Call). Это стандарт вызова удаленных процедур, использующий XML для кодирования сообщений и HTTP в качестве транспортного механизма.

https://github.com/pavel-kirienko/xmlrpc-d

QtE

Минималистичный биндинг Qt4. Есть версия для D2 и Forth.

http://qte.ucoz.ru

DQuick – GUI на D

Реализация пользовательского интерфейса на основе SDL и FreeType с Lua в качестве скриптового языка.

https://github.com/D-Quick/DQuick

DDT 0.7.0

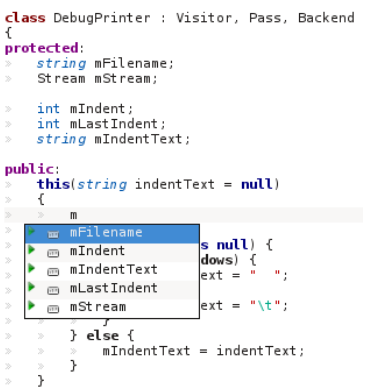
Вышла новая версия DDT 0.7.0 под кодовым названием «Midnight Riders». В данном релизе дебютирует новый парсер, значительно превосходящий старый по производительности. Напомним, DDT (D Development Tools) – это плагин поддержки D для среды разработки Eclipse. Проект распространяется по лицензии Eclipse Public License.

http://code.google.com/a/eclipselabs.org/p/ddt
http://www.eclipse.org/

Lumen

Одноименный с известной российской альтернатив-группой проект Lumen – это плагин автозаполнения для компонента KTextEditor, работающий в KDE-шных IDE и текстовых редакторах (таких, как Kate и KDevelop). Плагин основан на сервере автозаполнения DCD, обеспечивает поддержку автодополнения кода, семантического анализа и отладки (есть интеграция с GDB). Плагодаря этой связке становится возможным использовать KDevelop для разработки приложений на D.

https://github.com/Dav1dde/lumen
https://github.com/Hackerpilot/DCD



DCD – сервер автодополнения кода

Название говорит само за себя: DCD – это демон автодополнения, совместимый практически со всеми IDE и текстовыми редакторами, где есть поддержка плагинов или скриптов. На данный момент DCD работает с Textadept, Kate/KDevelop, Vim и Emacs. DCD состоит из серверной и клиентской частей. Клиент (dcd-client) работает на стороне текстового редактора или из командной строки. Сервер (dcd-server) ответственен за кэширование импортируемых файлов, обработку информации и возврат ее клиенту.

https://github.com/Hackerpilot/DCD

Наши проекты

Cook

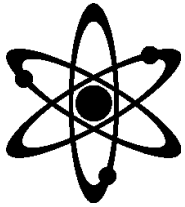
Программа автоматизации сборки проектов на языке D. В отличие от аналогичных инструментов (Make, CMake, Scons, Jam, DSSS и др.), Cook не требует конфигурационного файла: всю информацию о проекте она получает самостоятельно, сканируя модули (файлы *.d). При этом программа отслеживает прямые и обратные зависимости между модулями: если модуль был изменен, необходимо скомпилировать заново не только его, но и все модули, которые от него зависят (это важно, если был изменен внешний интерфейс модуля: объявления классов, семантика шаблонов и т.д.). Для этого Cook производит лексический анализ модулей - но не всех, а только тех, которые были изменены со времени последнего анализа. Данные анализа кэшируются в файл для повторного использования (кэш автоматически обновляется при пересборке). Cook работает в Windows и Linux.

http://code.google.com/p/cook-build-automation-tool/

dlib

Коллекция библиотек «на все случаи жизни» для D, которая может быть использована в игровых движках и других мультимедийных приложениях. Написана на D2 с использованием Phobos, не имеет никаких других внешних зависимостей. Разработка dlib пока находится на ранней стадии - API нестабилен и может измениться в любой момент, если появится возможность улучшить общую архитектуру.

http://code.google.com/p/dlib/



Физический движок своими руками

Часть III. Солвер контактов

В предыдущем номере журнала (FPS №25 '13, «Физический движок своими руками. Часть 2. Обнаружение столкновений») мы рассмотрели базовые принципы обнаружения пересечений между геометрическими телами и научились вычислять информацию, описывающую контакт двух тел – точка, нормаль и глубина взаимопроникновения. В этой части статьи мы рассмотрим варианты реакции на столкновение (Collision Response) – тела должны каким-то образом разойтись, чтобы столкновение между ними больше не обнаруживалось, и они перестали погружаться друг в друга.

В кинематике обычно используется нефизичная реакция: объекты просто «выталкиваются» вдоль вектора нормали на величину, равную половине глубины взаимопроникновения:

$$\vec{x} = \vec{x}_0 + \frac{1}{2} d \vec{n}$$

Первый объект выталкивается вдоль *+n*, а второй – вдоль *-n* (нормаль контакта всегда указывает от второго тела к первому). Эту операцию называют также корректировкой позиций (Position Correction). Она очень хорошо себя показывает в играх без физики – если вы добрались до этого места, у вас уже должно быть достаточно знаний для создания собственного нефизичного движка обработки столкновений, подходящего для игр жанра экшн (различного рода шутеры, платформеры и т.д.). Но в физичном Collision Response так делать нельзя, это приведет к нестабильности. Кроме того, необходимо учитывать массы тел. Поэтому мы будем использовать импульсы.

Импульс – это векторная величина, мера механического движения тела. Импульс тела равен произведению массы тела на его скорость:

$$\vec{p} = m \vec{v}$$

Единицей измерения импульса в системе СИ является килограмм-метр в секунду (кг·м/с). Аналогично существует его вращательный аналог – момент импульса:

$$\vec{L} = \vec{r} \times \vec{p}$$

где *p* – линейный импульс, *r* – радиус-вектор (точка, в которой приложен линейный импульс). Момент импульса измеряется по системе СИ в $м^2 \cdot кг \cdot с^{-1}$.

Два тела при столкновении обмениваются одним и тем же импульсом, но приобретают разные скорости из-за того, что имеют разные массы. Наша задача заключается в нахождении этого самого импульса – собственно, это и есть та самая «физическая магия», которую делает солвер контактов.

Строго говоря, «solver» переводится на русский язык как «решатель» – это, по сути, решатель систем линейных уравнений (СЛАУ). Но в сообществе разработчиков физических движков традиционно используется термин «солвер».

Одно уравнение в нашей системе имеет вид

$$\vec{J} \cdot \vec{v} = 0$$

где *J* – строка матрицы коэффициентов (ее также называют матрицей Якоби), *v* – вектор обобщенных скоростей двух тел: $(v_1, \omega_1, v_2, \omega_2)$, где v_1 и v_2 – линейные скорости тел, w_1 и w_2 – угловые.

Строка матрицы Якоби описывает ограничение свободы, вводимое данным уравнением в систему. Оно заключается в том, что тела не должны сближаться – то есть, проекция относительной скорости двух тел на нормаль контакта должна быть нулевой.

Соответственно, строка матрицы имеет вид (n_1, w_1, n_2, w_2), где

$$\vec{n}_1 = \vec{n}$$
$$\vec{w}_1 = \vec{n} \times \vec{r}_1$$
$$\vec{n}_2 = -\vec{n}$$
$$\vec{w}_2 = -\vec{n} \times \vec{r}_2$$

где n – нормаль контакта, r_1 и r_2 – точка контакта относительно первого и второго тела.

Смысл тут в том, что первое тело может двигаться только в направлении n_2 и вращаться вокруг оси w_1, а второе – двигаться в направлении n_2 и вращаться вокруг оси w_2. Иными словами,

$$\vec{n}_1 \cdot \vec{v}_1 + \vec{w}_1 \cdot \vec{\omega}_1 + \vec{n}_2 \cdot \vec{v}_2 + \vec{w}_2 \cdot \vec{\omega}_2 = 0$$

Сколько же таких уравнений в нашей системе? Столько же, сколько было обнаружено контактов (пар столкнувшихся тел). Но давайте на время забудем, что мы решаем систему, и представим, что нужно решить всего одно уравнение – для столкновения одной пары тел. Нам нужно каким-то образом мгновенно изменить скорости тел так, чтобы удовлетворить вышеописанному условию – то есть, найти импульс, которым обмениваются тела при столкновении. Назовем амплитуду этого импульса λ. Тогда

$$\vec{v} = \vec{v}_0 + \frac{\vec{n}\,\lambda}{m}$$

где v – скорость тела после столкновения, v_0 – до столкновения, n – нормаль контакта, m – масса тела.

Аналогично – угловая составляющая:

$$\vec{\omega} = \vec{\omega}_0 + \frac{\vec{n}\,\lambda \times \vec{r}}{I}$$

где ω – угловая скорость тела после столкновения, ω_0 – до столкновения, r – точка контакта относительно тела, I – момент инерции тела.

Теперь уравнение контакта можно выразить как

$$(\vec{n}_1, \vec{w}_1, \vec{n}_2, \vec{w}_2) \cdot \left((\vec{v}_1, \vec{\omega}_1, \vec{v}_2, \vec{\omega}_2) + \left(\frac{\vec{n}_1 \lambda}{m_1}, \frac{\vec{w}_1 \lambda}{I_1}, \frac{\vec{n}_2 \lambda}{m_2}, \frac{\vec{w}_2 \lambda}{I_2} \right) \right) = 0$$

Если вынести λ за скобки, получаем

$$(\vec{n}_1, \vec{w}_1, \vec{n}_2, \vec{w}_2) \cdot \left((\vec{v}_1, \vec{\omega}_1, \vec{v}_2, \vec{\omega}_2) + \lambda \left(\frac{\vec{n}_1}{m_1}, \frac{\vec{w}_1}{I_1}, \frac{\vec{n}_2}{m_2}, \frac{\vec{w}_2}{I_2} \right) \right) = 0$$

или

$$a + \lambda b = 0$$

где

$$a = (\vec{n}_1, \vec{w}_1, \vec{n}_2, \vec{w}_2) \cdot (\vec{v}_1, \vec{\omega}_1, \vec{v}_2, \vec{\omega}_2)$$
$$b = (\vec{n}_1, \vec{w}_1, \vec{n}_2, \vec{w}_2) \cdot \left(\frac{\vec{n}_1}{m_1}, \frac{\vec{w}_1}{I_1}, \frac{\vec{n}_2}{m_2}, \frac{\vec{w}_2}{I_2} \right)$$

Следовательно,

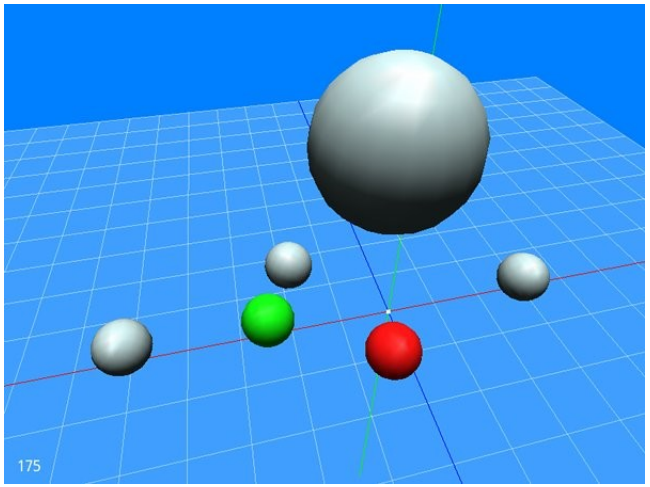
$$\lambda = \frac{-a}{b}$$

Применив полученный импульс к телам ($n\lambda$ к телу 1 и $-n\lambda$ к телу 2), находим скорректированные скорости после столкновения. Таким образом, мы удовлетворили одному ограничению свободы. Но в системе, как мы помним, их много – по одному на контакт, и, причем, необходимо удовлетворить им всем одновременно. Большинство прямых методов решения СЛАУ слишком медленные, поэтому, как правило, используются итеративные методы, постепенно делающие решение все более и более точным – к примеру, метод Якоби или Гаусса-Зейделя. При этом, если просто последовательно решать контакты один за другим, как описано выше, то полученный результат будет сходиться к такому же результату, как если бы их решали в системе.

Реализация

```
void solveContact(ref Contact c)
{
  RigidBody body1 = c.body1;
  RigidBody body2 = c.body2;

  Vector3f r1 = c.point - body1.position;
  Vector3f r2 = c.point - body2.position;

  Vector3f relVelocity = Vector3f(0.0f, 0.0f, 0.0f);
  relVelocity += body1.velocity
              + cross(body1.angVelocity, r1);
  relVelocity -= body2.velocity
              + cross(body2.angVelocity, r2);

  float velProj = dot(relVelocity, c.normal);

  // Если тела уже разлетаются, завершаем работу
  if (velProj > 0.0f)
      return;

  // Строка матрицы Якоби
  Vector3f n1 = c.normal;
  Vector3f w1 = cross(c.normal, r1);
  Vector3f n2 = -c.normal;
  Vector3f w2 = cross(-c.normal, r2);

  // Вычисляем импульс
  float a =
      dot(n1, body1.velocity)
    + dot(w1, body1.angVelocity)
    + dot(n2, body2.velocity)
    + dot(w2, body2.angVelocity);

  float b =
      dot(n1, n1 / body1.mass)
    + dot(w1, w1 / body1.inertia)
    + dot(n2, n2 / body2.mass)
    + dot(w2, w2 / body2.inertia);

  float lambda = -a / b;

  if (lambda < 0.0f)
      lambda = 0.0f;

  // Корректируем скорости
  body1.velocity += n1 * lambda / body1.mass;
  body1.angVelocity += w1 * lambda / body1.inertia;
  body2.velocity += n2 * lambda / body2.mass;
  body2.angVelocity += w2 * lambda / body2.inertia;
}
```

Продолжение следует...



Making-of

Логическая мини-игра **Arrow**

Логические игры в наши дни постепенно отошли на второй (и даже на третий) план, уступив на рынке 2D-игр место динамичным аркадам и головоломкам с реалистичной физикой. После знаменитого Bejeweled в этом жанре давно не появлялось ничего принципиально нового. И профессиональных разработчиков можно понять – какой смысл создавать очередной клон «Тетриса»? Кто его купит?..

Но я акцентирую ваше внимание на слове «профессиональные». Многие начинающие игроделы – абсолютное большинство, если быть честным – равняются на крупные студии и большие коммерческие проекты, стараются «сделать, как у них, только круче». Это серьезная ошибка. Как писал Линус Торвальдс, никогда нельзя браться за проект и думать, что он станет большим. В одиночку просто физически невозможно охватить все тонкости студийных пайплайнов, которые применяются в профессиональной среде, даже если вы одновременно художник, программист, левелдизайнер, композитор, человек и пароход. Вы должны, в первую очередь, иметь цельное видение проекта, и начинать учиться этому на «убийце Крайзиса» – верх самонадеянности.

Самый, на мой взгляд, подходящий полигон для этого – маленькие и простые логические игры. Игровая механика в них, как правило, описывается простым набором правил. Умение придумывать и выделять эти правила – ключ к пониманию того, как, собственно, создаются все игры в целом. Логическая игра – хороший пример абстрагирования идеи (набора правил) от реализации (алгоритма), что является хорошим тоном в разработке любого программного продукта. Например, правила игры в шахматы существуют, принципиально не меняясь, уже сотни лет - а алгоритмы, позволяющие компьютеру играть в шахматы с человеком, были изобретены сравнительно недавно; их очень много, и они беспрерывно совершенствуются.

Давайте попробуем выделить правила для несложной игры наподобие «Тетриса», которую я недавно написал. Это, фактически, клон игры GuruGuru, предустановленной на мой телефон (Pantech PG-1500). Она отличается от классического «Тетриса» новой механикой: вместо простых блоков здесь блоки-стрелки. Игрок должен выстраивать линии из стрелок, указывающих в одном направлении.

1. Игровое поле представляет собой матрицу 8x8.

2. В каждый момент времени ячейка матрицы может быть либо пустой, либо занятой одним из следующих типов стрелок:

- Стрелка влево
- Вправо
- Вверх
- Вниз
- В обе стороны по горизонтали
- В обе стороны по вертикали
- Во все четыре стороны

Есть также «камень», не представляющий собой никакой тип стрелки – он введен для усложнения игрового процесса.

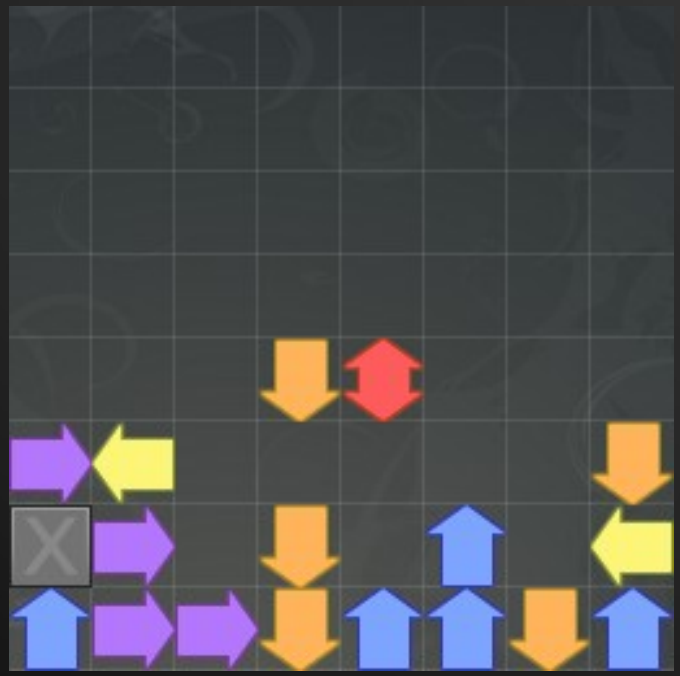
3. Горизонтальная или вертикальная линия из 3 и более стрелок, указывающих в одном направлении, исчезает. При этом допустимы комбинации, при которых двунаправленные или четырехнаправленные стрелки задают два разных направления по обе стороны от себя.

4. Если какая-либо стрелка исчезает, то стрелки в соседних с ней ячейках матрицы поворачиваются по часовой стрелке. Соседними они могут быть только по горизонтали и вертикали – диагонали не учитываются. Если сосед – «камень», то он тоже исчезает.

5. Остальное – в точности как в традиционном «Тетрисе». Сверху падают случайные блоки – пока они падают, игрок может их перемещать и поворачивать. Я предусмотрел только два типа блоков – состоящие из одной или из двух ячеек. Поворот возможен только в одном направлении – по часовой стрелке. Если повороту мешает «стена» из непустых ячеек или граница матрицы, поворот невозможен.

Игра, формально, длится до тех пор, пока матрица не заполнится доверху. Но можно ввести состояние «выигрыша» – например, если игрок набирает определенное количество очков.

Я, естественно, не буду приводить в этой статье полную реализацию вышеописанных правил – это займет слишком много места. По большей части, это рутинный код – есть только несколько хитрых моментов, которые я хочу рассмотреть подробнее. Фрагменты кода, которые я буду использовать в качестве примеров, написаны на языке D, как и сама игра.



Игровое поле

У некоторых начинающих (в особенности, у бывших пользователей игровых конструкторов типа Game Maker) сразу возникнет соблазн представить стрелки отдельными объектами класса Arrow с полями x, y и direction. Почему так делать нельзя? Все дело в том, что логика игры предусматривает много операций с рядами и колонками ячеек. Если даже вам вначале это не кажется очевидным, вы впоследствии это поймете. Разумнее будет представить матрицу в виде двумерного массива состояний (Empty, Left, Right, Up, Down и т.д.). Конечно, кроме состояния, у ячейки может быть различная внутренняя служебная информация, но это уже детали.

Игровое время

Не лишним будет сразу ввести игровой таймер. Например, состояния матрицы автоматически обновляются каждую секунду:

```
float timer = 0.0f;
float procStep = 1.0f;

void step(float delta)
{
    timer += delta;
    if (timer >= procStep)
    {
        timer = 0.0f;
        processMatrix();
    }
}
```

Функция processMatrix обновляет падение активного блока, обнаруживает ряды одинаковых стрелок, поворачивает их соседей и т.д. В то же время, возможность игрока перемещать и поворачивать активный блок не должна быть ограничена этим таймером.

Обнаружение одинаковых рядов

Это, фактически, сердце реализации. По правде говоря, я не претендую на то, что мой алгоритм – самый-самый, но он работает, и работает достаточно быстро. Я сразу разделил обнаружение на две части – проход по рядам и колонкам. Работают они одинаково. Читаем состояния слева направо (или сверху вниз). Запоминаем, с какой ячейки начался ряд одинаковых стрелок, а также направление ряда. Если нашли три и больше – помечаем их как удаленные. Если нашли при этом дву- или четырехнаправленные стрелки, запоминаем последнюю найденную – с нее начнется еще один потенциальный ряд, который может иметь другое направление.

Я не случайно пишу «помечаем как удаленные», а не «удаляем». Удалить мы их всегда успеем, а вот повернуть их соседей после удаления уже не получится. Так что сначала нам нужна эта «карта» удаляемых ячеек – найти их соседей не составит особого труда.

Случайные блоки

Элемент случайности – важная составляющая игровой механики. Задача сводится к тому, что требуется выбрать случайное направление стрелки с учетом того, что некоторые из них имеют большую вероятность выбора, чем другие – то есть, будут появляться чаще. Говорят, что они имеют больший вес. Я написал особую реализацию алгоритма такой выборки – она работает не с обычными массивами, а с перечислениями (enum). В шаблон функции передаются два перечисления – самих элементов и их весов.

```
import std.traits;
import std.random;
import std.algorithm;

E weightedRandomEnum(E, W)()
    if (isNumeric!W &&
        is(E == enum) && is(W == enum) &&
        EnumMembers!T.length == EnumMembers!W.length)
{
    enum members = [EnumMembers!E];
    auto randomNumber = uniform(0, weightsSum);

    foreach(i, weight; weights)
    {
        if (randomNumber < weight)
            return members[i];
        else
            randomNumber -= weight;
    }

    assert (0, "Should never get here");
}
```

Как видно из перечисления Weights, чаще всех будут появляться Left, Right, Up и Down. Реже – Horizontal и Vertical. Затем – Omni и, наконец, Stone.

Графика

Графическая часть в данном случае мало связана с игровой логикой. Все, что требуется от графического движка – уметь рисовать на экране затекстуренные квадратики. С этим справится практически любая библиотека, будь то OpenGL, SDL, DirectX, Cairo или что-либо еще. Хорошая практика – максимально отделить логику от графики, создать абстрактный интерфейс к графическому бэкенду, который состоит из высокоуровневых операций типа «загрузить картинку», «нарисовать прямоугольник». В этом случае будет очень просто портировать игру на любую платформу – потребуется лишь написать соответствующую реализацию бэкенда.

Скачать Arrow можно **здесь**

Ознакомиться с исходниками игры можно на GitHub:
https://github.com/gecko0307/arrow

Тимур Гафаров

Процедурная генерация игровых карт

Многим кажется, что процедурная генерация игрового контента (PCG, procedural content generation) – это недосягаемая «магия», доступная только гуру. Однако на самом деле PCG может быть изучена начинающим разработчиком. В этом уроке я расскажу, как процедурно сгенерировать систему подземелий.

Генерирование проходит в несколько шагов:

1. Случайно поместить созданный контент в игровом мире;

2. Убедиться, что контент размещен в нужном месте;

3. Убедиться, что контент достижим игроком;

4. Повторять эти шаги до тех пор, пока уровень не соберется правильно.

Первое, что мы собираемся сделать – это случайно разместить комнаты подземелья. Прежде чем начать, мы должны заполнить карту уровня тайлами стены – пройтись циклом по каждой ячейке карты (по 2D-массиву) и поместить тайл.

Не забывайте преобразовать пиксельные координаты тайла в координаты ячейки. Если вы хотите перейти от пикселей к сеточному расположению, то следует разделить пиксельные координаты на ширину тайла. Чтобы перейти от сетки к пикселям, нужно умножить координаты ячейки на ширину тайла.

Например, если мы хотим поместить верхний левый угол нашей комнаты в координаты (5, 8) на нашей сетке, и у нас есть тайл шириной 8 пикселей, нам нужно поместить этот угол в (5*8, 8*8) или (40, 64) в пиксельных координатах.

Создадим класс для комнаты (код на языке Haxe):

```
class Room extends Sprite
{
    // сеточные координаты углов комнаты
    public var x1:Int;
    public var x2:Int;
    public var y1:Int;
    public var y2:Int;

    // ширина и высота комнаты
    public var w:Int;
    public var h:Int;

    // центр комнаты
    public var center:Point;

    // конструктор
    public function new(x:Int, y:Int, w:Int, h:Int)
    {
        super();
        x1 = x;
        x2 = x + w;
        y1 = y;
        y2 = y + h;

        this.x = x * Main.TILE_WIDTH;
        this.y = y * Main.TILE_HEIGHT;
        this.w = w;
        this.h = h;

        center = new Point(Math.floor((x1 + x2) / 2),
        Math.floor((y1 + y2) / 2));
    }

    public function intersects(room:Room):Bool
    {
        return (x1 <= room.x2 &&
                x2 >= room.x1 &&
                y1 <= room.y2 &&
                room.y2 >= room.y1);
    }
}
```

У нас есть переменные, содержащие ширину и длину каждой комнаты, положение точки центра комнаты и позиция четырех углов. Есть функция, сообщающая, пересекает ли эта комната другую. Также отметим, что все координаты, за исключением переменных x и y, находятся в сеточной системе.

Теперь у нас есть основа для построения комнаты. Как же процедурно сгенерировать и разместить ее на уровне? Благодаря генератору случайных чисел, это не так сложно. Все, что нам нужно сделать – это задать случайные значения переменным x и y нашей комнаты в пределах карты и дать случайные значения ширины и длины комнаты в предела заданного диапазона.

Так как мы используем случайные позицию и размеры для нашей комнаты, мы обязаны проверить ее пересечение с ранее созданными комнатами, когда мы заполняем наше подземелье. Мы уже объявили метод intersects, который позаботится об этом.

Каждый раз, когда мы собираемся разместить новую комнату, мы вызываем метод intersects для пары комнат из нашего списка. Этот метод возвращает логическое значение true, если комнаты пересекаются и false – в противоположном случае. Мы можем использовать это значение, чтобы решить, что делать с комнатой, которую мы пытаемся разместить.

```
private function placeRooms()
{
    // создаем массив для хранения комнат
    rooms = new Array();

    // генерируем комнаты
    for (r in 0...maxRooms)
    {
        var w = minRoomSize +
            Std.random(maxRoomSize - minRoomSize + 1);

        var h = minRoomSize +
            Std.random(maxRoomSize - minRoomSize + 1);

        var x = Std.random(MAP_WIDTH - w - 1) + 1;
        var y = Std.random(MAP_HEIGHT - h - 1) + 1;

        var newRoom = new Room(x, y, w, h);

        var failed = false;

        for (otherRoom in rooms)
        {
            if (newRoom.intersects(otherRoom))
            {
                failed = true;
                break;
            }
        }

        if (!failed)
        {
            createRoom(newRoom);
            rooms.push(newRoom);
        }
    }
}
```

Но что, если у вас есть комнаты, до которых игроку никак не добраться? Большинство игр, использующих процедурно сгенеированный контент, стараются такого не допускать - но это не всегда является лучшим проектным решением. Подобные недосягаемые комнаты могут добавить интересной динамики к вашему подземелью.

Однако, вне зависимости от того, как вы к этому относитесь, необходимо проверять, может ли игрок перемещаться по такому уровню: будет неприятно, если, например, выход из подземелья окажется полностью заблокирован.

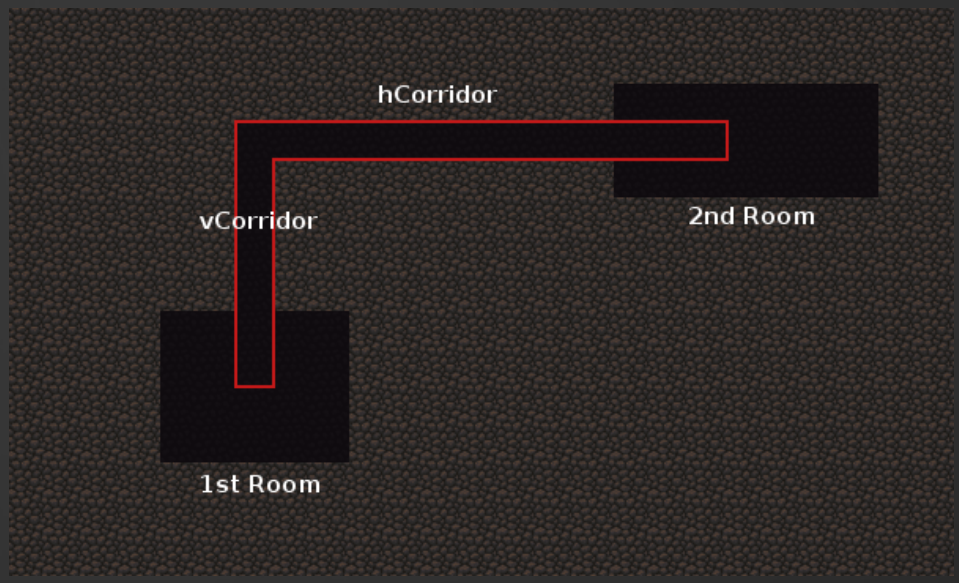
Давайте попробуем справиться с этой досягаемостью. Наша цель заключается к соединении комнат так, чтобы мы могли пройти через подземелье и, в конечном итоге, достичь выхода и перейти к следующему уровню. Мы можем добиться этого, добавив коридоры между комнатами.

Нам необходимо добавить переменную point для отслеживания центра каждой созданной комнаты. Всякий раз, когда мы создаем и размещаем комнату, мы определяем ее центр и «подключаем» ее к центру предыдущей комнаты.

Создаем коридоры:

```
private function hCorridor(x1:Int, x2:Int, y)
{
    for (x in Std.int(Math.min(x1, x2))...
            Std.int(Math.max(x1, x2)) + 1)
    {
        map[x][y].parent.removeChild(map[x][y]);
        map[x][y] = new Tile(Tile.DARK_GROUND, false,false);
        addChild(map[x][y]);
        map[x][y].setLoc(x, y);
    }
}

private function vCorridor(y1:Int, y2:Int, x)
{

    for (y in Std.int(Math.min(y1, y2))...
            Std.int(Math.max(y1, y2)) + 1)
    {
        map[x][y].parent.removeChild(map[x][y]);
        map[x][y] = new Tile(Tile.DARK_GROUND, false,false);
        addChild(map[x][y]);
        map[x][y].setLoc(x, y);
    }
}
```

Эти функции действуют почти одинаково, но одна создает коридоры горизонтально, а другая вертикально.



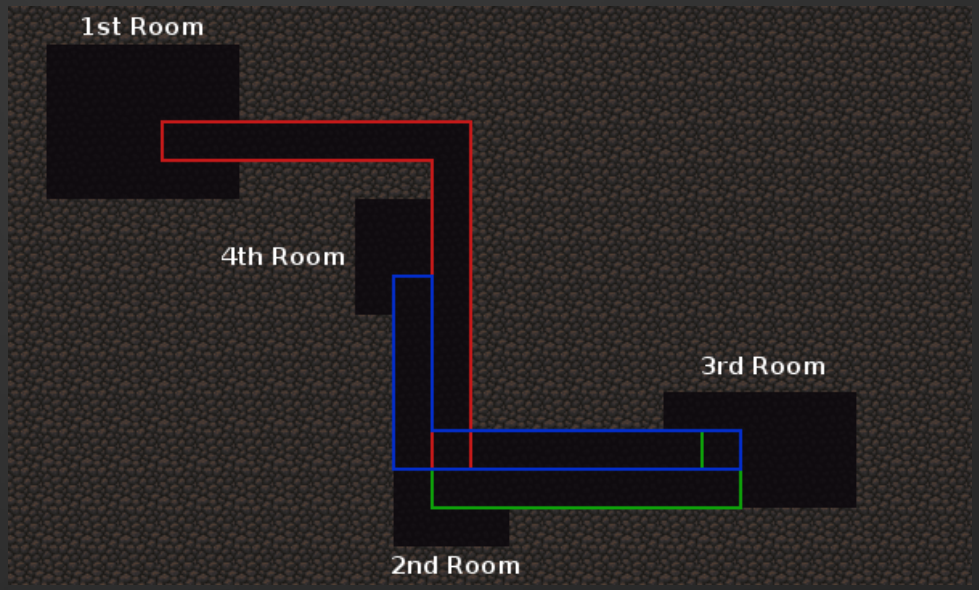
Нам нужны три переменные для этого. Для горизонтальных коридоров нам нужны начальные значения x, конечные значения x и текущее значение y. Для вертикальных коридоров нам нужны начальные и конечные значения y и текущие x.

Поскольку мы движемся слева направо, нам нужны два соответствующих значений x, но только одно значение y, так как мы не будет двигаться вверх или вниз.

Когда мы движемся вертикально, нам понадобятся значения y. В цикле for в начале каждой функции мы перебираем от начального значения (x или y) до конечного значения, пока мы не вырезаем весь коридор.

Теперь у нас есть коридор, и мы можем изменить функцию placeRooms(), добавив новые функции для генерации коридоров:

```
private function placeRooms()
{
  rooms = new Array();
  var newCenter = null;
  for (r in 0...maxRooms)
  {
    var w = minRoomSize +
            Std.random(maxRoomSize - minRoomSize + 1);
    var h = minRoomSize +
            Std.random(maxRoomSize - minRoomSize + 1);
    var x = Std.random(MAP_WIDTH - w - 1) + 1;
    var y = Std.random(MAP_HEIGHT - h - 1) + 1;

    var newRoom = new Room(x, y, w, h);

    var failed = false;
    for (otherRoom in rooms)
    {
      if (newRoom.intersects(otherRoom))
      {
        failed = true;
        break;
      }
    }

    if (!failed)
    {
      createRoom(newRoom);
      newCenter = newRoom.center;
      if (rooms.length != 0)
      {
        var prevCenter = rooms[rooms.length - 1].center;
        if (Std.random(2) == 1)
        {
          hCorridor(Std.int(prevCenter.x),
                    Std.int(newCenter.x),
                    Std.int(prevCenter.y));
          vCorridor(Std.int(prevCenter.y),
                    Std.int(newCenter.y),
                    Std.int(newCenter.x));
        }
        else
        {
          vCorridor(Std.int(prevCenter.y),
                    Std.int(newCenter.y),
                    Std.int(prevCenter.x));
          hCorridor(Std.int(prevCenter.x),
                    Std.int(newCenter.x),
                    Std.int(newCenter.y));
        }
      }
    }

    if (!failed) rooms.push(newRoom);
  }
}
```



На картинке вы можете проследить за созданием коридоров от первой к четвертой комнате: красного, зеленого, а потом синего. Вы можете получить интересные результаты в зависимости от размещения комнат – например, два коридора рядом друг с другом создадут коридор двойной ширины.

Теперь у вас есть от чего отталкиваться при создании процедурной игры. Последний шаг – генерация случайных предметов, врагов, NPC и других интерактивных объектов.

Оригинал: ***Иеремия Гоердт***
http://gamedev.tutsplus.com
Перевод: ***Наталия Чумакова***

Осваиваемся в SDL2

В середине августа, наконец, состоялся официальный релиз библиотеки SDL 2.0, упрощающей написание игр и мультимедийных приложений. В числе наиболее интересных новшеств можно полный переход на аппаратное ускорение при 2D-рендеринге, поддержку ForceFeedback (вибрации) для джойстиков, поддержку захвата звука, аудио в формате 7.1, многооконный режим, доступ к буферу обмена, поддержку одновременного использования нескольких устройств ввода, multi-touch, поддержку многомониторных конфигураций, Unicode-символов, drag'n'drop и многое другое.

Напомним, что SDL предоставляет кроссплатформенный доступ к таким средствам, как быстрый вывод 2D-графики, обработка ввода, проигрывание звука, вывод 3D-графики с использованием OpenGL и множеству иных сопутствующих операций, независимо от используемой ОС, что значительно упрощает создание мультимедийных приложений и игр. SDL можно назвать «кроссплатформенным DirectX». Новая версия библиотеки доступна под пермиссивной лицензией Zlib, которая позволяет использовать ее в коммерческих продуктах. SDL 2.0 уже вовсю используется компанией Valve в платформе Steam и многих играх, поставляемых через нее.

К сожалению, SDL 2.0 мало совместима с 1.x – вам придется так или иначе переписывать свои проекты. С другой стороны, с этим можно пока не торопиться: старые проекты могут использовать SDL 1.x, ее поддержка никуда не денется. А вот для новых нет никаких причин не использовать 2.0. Особенно радостно программистам на D: ведь поддержка SDL 2.0 в Derelict3 есть уже достаточно давно.

Итак, приступим. Для начала нужно где-то достать саму SDL2. Для Windows и Mac OS на официальном сайте есть готовая бинарная сборка. Для других платформ придется компилировать самостоятельно (или, в случае с Linux, установить из репозиториев).

В этом примере я покажу, как загрузить через SDL изображение из файла и вывести его в окошко на экране.

Импортируем нужные модули:

```
module main;
import std.conv;
import derelict.sdl2.sdl;
```

Загружаем библиотеку:

```
void main()
{
    DerelictSDL2.load();
    /* остальной код - сюда */
}
```

Инициируем SDL:

```
if (SDL_Init(SDL_INIT_EVERYTHING) == -1)
    throw new Exception("Failed to init SDL: "
    ~ to!string(SDL_GetError()));
```

Создаем окно 640x480:

```
SDL_Window* win = null;
win = SDL_CreateWindow("Hello World!",
    100, 100, 640, 480, SDL_WINDOW_SHOWN);
if (win is null)
    throw new Exception("Failed to create window: "
     ~ to!string(SDL_GetError()));
```

Создаем контекст 2D-рендеринга:

```
SDL_Renderer* ren = null;
ren = SDL_CreateRenderer(win, -1,
    SDL_RENDERER_ACCELERATED |
    SDL_RENDERER_PRESENTVSYNC);
if (ren is null)
    throw new Exception("Failed to create renderer: "
    ~ to!string(SDL_GetError()));
```

Загружаем картинку:

```
SDL_Surface* bmp = null;
bmp = SDL_LoadBMP("hello.bmp");
if (bmp is null)
    throw new Exception("Failed to load BMP: "
    ~ to!string(SDL_GetError()));
```

Использование формата BMP, конечно, не самая лучшая идея – куда разумнее будет воспользоваться библиотекой SDL_Image (или же вовсе создать свой собственный формат), но это уже тема для отдельной статьи.

Создаем из изображения текстуру (рендеринг теперь основан на текстурах):

```
SDL_Texture* tex = null;
tex = SDL_CreateTextureFromSurface(ren, bmp);
```

Изображение теперь можно удалить из ОЗУ – текстура уже скопирована в видеопамять, и дублировать данные нам ни к чему.

```
SDL_FreeSurface(bmp);
```

Очищаем экран и рисуем текстуру:

```
SDL_RenderClear(ren);
SDL_RenderCopy(ren, tex, null, null);
SDL_RenderPresent(ren);
```

По идее, эта операция должна находиться в главном цикле программы. Но наш пример преследует максимальную простоту, поэтому мы ограничимся простой двухсекундной паузой: окно появляется, показывает картинку и само исчезает.

```
SDL_Delay(2000);
```

Теперь нужно освободить все выделенные ресурсы:

```
SDL_DestroyTexture(tex);
SDL_DestroyRenderer(ren);
SDL_DestroyWindow(win);
```

Последнее, что мы делаем перед завершением работы программы - это вызываем SDL_Quit:

```
SDL_Quit();
```

В последующих номерах журнала мы продолжим изучение SDL2 и рассмотрим все новые функции библиотеки.

Тимур Гафаров
gecko0307@gmail.com

*Программист Евгений Степанищев создал реализацию игры в шахматы, примечательную тем, что она написана только с использованием языка линуксовой утилиты **sed** (это консольный редактор потоков, применяющий различные предопределенные текстовые преобразования к последовательному потоку текстовых данных).*

Поддерживается игра человека с компьютером, который производит позиционную оценку на один ход.

*Репозиторий проекта: **https://github.com/bolknote/SedChess**.*

```
770         # младшие разряды для вычитания остались?
771         /:11*B/ {
772             # переносим разряды вычитаемого на младший
773             :sub-array::cy
774             # если переносить нечего, то всё, получается число меньше нуля,
775             # возвращаем ноль, выходим
776             /1.*M/ ! {
777                 s/.*/:::B/
778                 b sub-array::end
779             }
780
781             s/\(.*\)1:\(.*M\)/\1:1111111111\2/
782
783             /1M/ ! b sub-array::cy
784
785             # добавляем к вычитаемому
786             s/:\(1*\)\(M.*\) \(:.*\)#S/:\2 \3\1#S/
787
788             b sub-array::minus
789         }
```

Плагин для DeleD на D

DeleD (http://www.delgine.com) – популярная программа для дизайна уровней. Она располагает широким набором инструментов низкополигонального моделирования (включая сглаживание и булевы операции), создания UV-разверток, генерации карт освещения, анимации, рендеринга при помощи встроенного трассировщика лучей, а также экспорта/импорта различных 3D-форматов (Collada, X, OBJ, 3DS, B3D, DIF). Собственный формат сцен DeleD (DXS) размечен в XML, что позволяет легко и интуитивно задействовать его в игровых движках или конвертировать в другие форматы. В 2010 году DeleD стал бесплатным и открытым – с этого момента программа развивается на базе сообщества. Коммьюнити-версия называется DeleD CE (Community Edition).

Программистам DeleD предлагает очень простой API для создания плагинов. В отличие от Blender, плагины здесь пишутся не на встроенном языке, а на стороннем – сгодится любой язык, который может компилировать DLL-библиотеки (C/C++, Delphi, C# и др.) В данной статье я хочу поделиться опытом создания плагина к DeleD на языке D.

Сначала – рутинный код, которым сопровождается любая DLL, написанная на D (я поместил его в модуль dll.d):

```
module dll;

import std.c.windows.windows;
import core.sys.windows.dll;

__gshared HINSTANCE g_hInst;

extern (Windows)
BOOL DllMain(HINSTANCE hInstance,
             ULONG ulReason,
             LPVOID pvReserved)
{
    switch (ulReason)
    {
        case DLL_PROCESS_ATTACH:
            g_hInst = hInstance;
            dll_process_attach(hInstance, true);
            break;

        case DLL_PROCESS_DETACH:
            dll_process_detach(hInstance, true);
            break;

        case DLL_THREAD_ATTACH:
            dll_thread_attach(true, true);
            break;

        case DLL_THREAD_DETACH:
            dll_thread_detach(true, true);
            break;

       default:
           break;
    }
    return true;
}
```

Теперь основной модуль:

```
module main;

import std.conv;
import std.string;
import dll;
```

Структура CallBackRecord хранит данные для обмена с программой:

```
struct CallBackRecord
{
    uint RequestID;
    char* RequestXML;
    char* ResponseXML;
    uint ResponseSize;
}
```

```
alias extern(Windows)
    uint function(TCallBackRecord*) TCallBackProc;

TCallBackProc g_Callback;
```

Объявляем функции, которые обязательно должны быть в плагине. Они предоставляют программе информацию о плагине (название, описание, версия и т. д.)

```
export extern(Windows)
{
    char* PluginName()
    {
        return cast(char*)"My Cool Plugin";
    }

    char* PluginDescription()
    {
        return cast(char*)"DeleD plugin written in D";
    }

    char* PluginDeleDVersion()
    {
        return cast(char*)"1.6";
    }

    char* PluginVersion()
    {
        return cast(char*)"1.0";
    }

    char* PluginAuthor()
    {
        return cast(char*)"Timur Gafarov aka Gecko";
    }

    char* PluginEmail()
    {
        return cast(char*)"gecko0307@gmail.com";
    }

    void PluginSetCallback(TCallBackProc cbProc)
    {
        g_Callback = cbProc;
    }

    void PluginExecute()
    {
        deledPluginMain();
    }
}
```

Функция PluginExecute вызывается в тот момент, когда пользователь захочет запустить ваш плагин. В моем случае она, в свою очередь, вызывает функцию deledPluginMain:

```
void deledPluginMain()
{
    string data = deledGetData("
      <request>
        <primitives subset=\"all\" />
      </request>
    ");

    /* делаем что-нибудь с data */
}

string deledGetData(string RequestXML)
{
    TCallBackRecord Rec;
    Rec.RequestID = PR_GETMEM;
    Rec.RequestXML = cast(char*)toStringz(RequestXML);
    g_Callback(&Rec);

    Rec.RequestID = PR_GETDATA;
    Rec.ResponseXML =
        cast(char*)malloc(Rec.ResponseSize);
    g_Callback(&Rec);

    string res = to!string(Rec.ResponseXML);
    free(Rec.ResponseXML);

    return res;
}
```

Все данные, которыми плагин обменивается с DeleD, размечены в XML, так что вам понадобится какая-нибудь библиотека для разбора и генерации XML. В стандартной библиотеке Phobos есть модуль std.xml – но использовать его обычно не советуют, так как активно разрабатывается более мощная и быстрая замена ему.

А пока она в разработке, можно поискать альтернативу на просторах Сети – например, простейший XML-парсер есть в моей коллекции библиотек **dlib**.

Теперь самое важное – компиляция. Здесь есть хитрость: имена файлов экспортируемых stdcall-функций не должны быть декорированы (а extern(Windows) по умолчанию декорирует их в стиле MS VS). Чтобы добиться этого, напишем скрипт для компоновщика (DEF-файл):

```
LIBRARY MYPLUGIN
DESCRIPTION 'DeleD plugin written in D'
EXETYPE NT
CODE PRELOAD DISCARDABLE
DATA WRITE
EXPORTS
  PluginName @2
  PluginDescription @3
  PluginDeleDVersion @4
  PluginVersion @5
  PluginAuthor @6
  PluginEmail @7
  PluginSetCallback @8
  PluginExecute @9
```

Собираем библиотеку так:

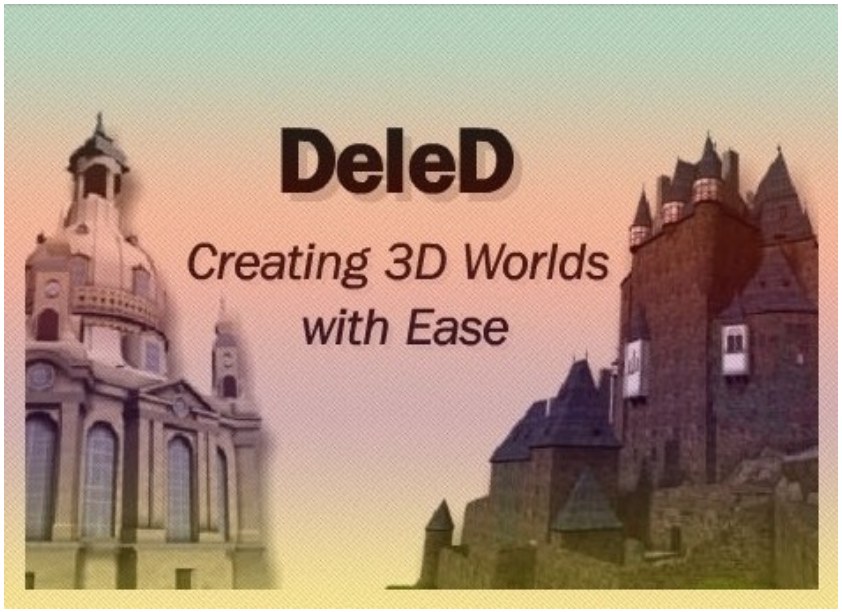
```
dmd -ofMyPlugin.dll -L/IMPLIB main.d dll.d main.def
```

Полученную библиотеку (MyPlugin.dll) необходимо скопировать в каталог Plugins в папке с установленным DeleD.

Полезные ссылки:

Официальный репозиторий плагинов
Документация на DeleD Wiki

Тимур Гафаров
gecko0307@gmail.com



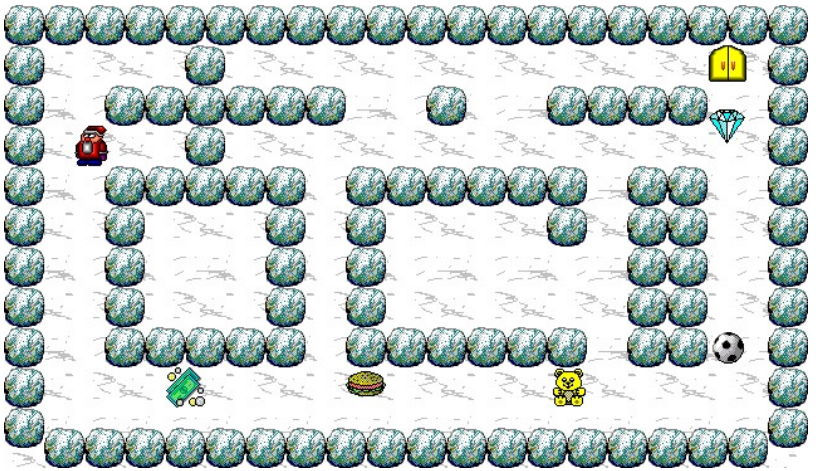
Как я стал D-шником

С тех пор, как у меня появился компьютер, меня всегда интересовало то, как он работает. Надо сказать, что, хотя по профессии я художник, я всю жизнь интересуюсь механизмами и электроникой. Больше всего, конечно, меня привлекает создание и обработка изображений при помощи вычислительной техники. Но знакомства с одними только графическими редакторами мне оказалось мало – гораздо интереснее изучить и понять, как, собственно, все эти фотошопы и 3ds-max'ы устроены. Ну и, конечно, игры – куда же без них? Мне кажется, что компьютерные игры можно считать «восьмым искусством» после кино. Эволюция четко прослеживается: сначала были статические картины, затем появились «движущиеся» – кинематограф и мультипликация; логично допустить, что следующее поколение произведений искусства должно быть интерактивным: зритель сам должен принимать участие в раскрывающемся перед ним действии. Художники XXI века – это создатели игр...

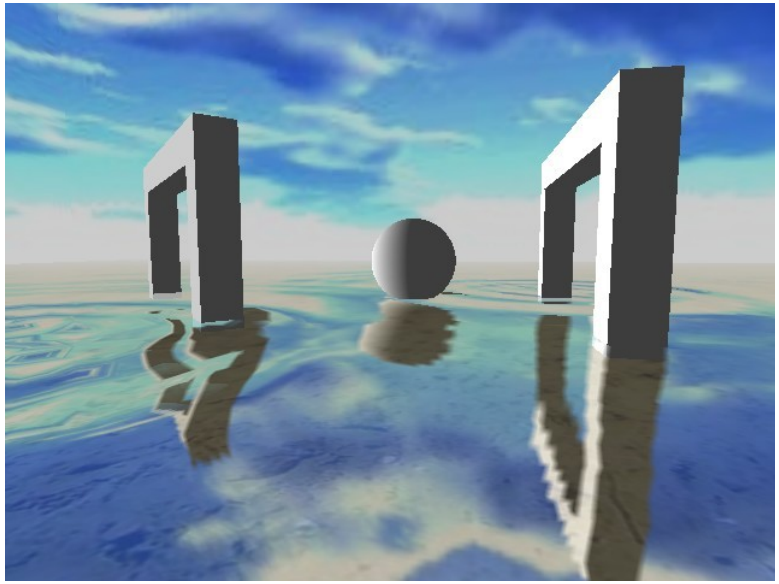
Моим первым языком программирования был BASIC. Если быть точным, его надмножество Blitz BASIC – это, в сущности, была целая среда для создания простых приложений и игр. Тогда меня поразил сам факт того, что программа для компьютера – это не что иное, как набор инструкций, человекочитаемый текст на специальном языке! Впрочем, человекочитаемость иных текстов на BASIC можно поставить под сомнение – особенно помня слова Эдсгера Дейкстры: «Студентов, ранее изучавших Бейсик, практически невозможно обучить хорошему программированию. Как потенциальные программисты они подверглись необратимой умственной деградации...» И, тем не менее, эта ни с чем не сравнимая магия переменных, циклов, функций – равно как и желтых букв на синем фоне – сделала свое дело. Не говоря уже о трехмерной графике на основе DirectX 7, которая была встроена в язык – стоит ли говорить, насколько поражала воображение возможность вот так, запросто, безо всяких специальных знаний нарисовать на экране вращающийся кубик?..

До кубиков, правда, дело дошло нескоро – ведь необходимо было изучить азы. В этом большую помощь оказали школьные уроки информатики: помимо непосредственно азов (устройство компьютера, двоичная система, теория алгоритмов и т.д.), на них изучался всеми любимый и тепло вспоминаемый Pascal. Конечно, работать в DOS-режиме Windows 95 было еще тем удовольствием, но именно тогда началось мое знакомство со всем семейством паскалеподобных языков, что не могло не сказаться на будущих предпочтениях.

Однако Паскаль, при всех его достоинствах, не слишком хорошо подходил для моей главной цели – создания 2D-игр (трехмерная графика Blitz BASIC мне тогда была еще не по зубам). И тут мне посчастливилось наткнуться на Game Maker – специализированную среду для быстрой разработки игр с собственным редактором уровней, объектной системой, графическим редактором и встроенным скриптовым языком GML. Это была некая смесь Pascal и C++: можно было либо использовать фигурные скобки, либо begin/end.



GML завоевал мое внимание на долгие пять лет. За это время я изучил Game Maker, что называется, вдоль и поперек, написал множество разнообразных 2D-демок (большинство из которых, увы, не сохранилось) и несколько полноценных игр – вы, кстати, можете ознакомиться с некоторыми тогдашними творениями в **моем блоге**. Меня разочаровывало только одно: отсутствие в программе приличных инструментов для рисования трехмерной графики. Был простенький 3D-режим на основе Direct3D – очень медленный, без доступа к программируемому конвейеру и другим современным «наворотам». Хотя, справедливости ради стоит отметить, что и на нем можно было делать неплохие вещи.



К этому времени появились специализированные 3D-движки, специально написанные для GM. В их числе был враппер популярной Delphi-библиотеки GLScene – Xtreme3D, поддерживавший практически все возможности оной: большое количество поддерживаемых форматов моделей и текстур, анимация, различные спецэффекты вроде динамической воды и системы частиц, менеджер управления ресурсами, встроенная проверка столкновений и даже физика на основе движка ODE. Я заинтересовался, изучил этот движок и даже открыл по нему сайт – **http://xtreme3d.narod.ru** (впрочем, после переезда на Ucoz он переживает не лучшие времена).

Так совпало, что именно в это время я познакомился с движением СПО (свободного программного обеспечения), и это навсегда изменило мое отношение к компьютерам. В какой-то момент моими привычными инструментами стали исключительно свободные программы – GIMP, Blender, OpenOffice.org и т.д. Я понял, что свобода изучения, изменения и распространения программ важнее их качества – хотя это, на первый взгляд, кажется абсурдным. Так утверждает Ричард Столлман – великий человек, основатель проекта GNU, автор лицензии GPL и создатель таких программ, как компилятор GCC и текстовый редактор Emacs. Над его словами часто иронизируют – но в итоге он всегда оказывается прав. Закрытость и несвободность ПО – это зло, которое в равной степени вредит и его пользователям, и разработчикам. Пользователям – потому что они никогда не могут быть уверены, что закрытая программа не следит за ними, не ворует их информацию. А разработчикам – потому что закрытую программу труднее поддерживать, улучшать и исправлять: ведь этим занимаются только работники фирмы, владеющей программой. В результате ошибки в ПО исправляются годами – а то и вовсе не исправляются (и это не шутка). А если бы исходный код был публично доступен, любой квалифицированный программист сразу исправил бы найденный баг, отправил бы разработчикам патч, предложил бы рекомендации к улучшению тех или иных конструкций. И таких помощников были бы сотни – взгляните на крупные открытые проекты, развиваемые сообществом. Наконец, если фирма, занимающаяся разработкой программы, обанкротится, то сообщество ее пользователей, не имея на руках исходного кода, оказывается асболютно беспомощным – но зачастую люди продолжают использовать такие «мертвые» программы годами и десятилетиями, если им нет альтернативы. Такая ситуация, к примеру, наблюдается в нашей промышленности и оборонной технике. Вот такие абсурды порождает собственническое отношение к коду.

Осознав все это, я пришел к выводу, что проприетарная Xtreme3D (к тому же, по всей видимости, заброшенная автором) – это тупиковый путь. В разработке ПО нужно пользоваться только свободными инструментами – и я начал изучать C++, выбрав этот язык как наиболее доступный из всех компилируемых.

Моей первой IDE под Windows стала Dev-C++ - среда, основанная на инструментарии GCC/MinGW. Язык сразу поразил меня своей красотой и выразительностью – до этого я не имел никакого понятия о ООП и составных типах данных. Впечатлившись этим богатством и относительной простотой его использования (Dev-C++ позволяла устанавливать пакеты расширения – можно было найти пакеты для работы с OpenGL, DirectX, Irrlicht и т.д.), я загорелся идеей написать свой собственный 3D-движок для Game Maker.

Дело шло неплохо – пока я в один прекрасный день не установил Linux. Это прямого отношения к моему программерскому хобби не имело, но в силу обстоятельств оказало на него сильное влияние. Несколько лет я большую часть времени проводил в Linux – и, естественно, писал под него программки, изучал линуксовые инструменты для разработчиков. К Game Maker уже практически не прикасался. В какой-то момент я понял, что нет больше смысла привязывать себя к Windows – и с тех пор мой движок существует как отдельный самодостаточный проект. Правда, он до сих пор не принял устоявшейся формы. Нет ни релизов, ни версий, ни постоянного репозитория, ни даже конкретного названия – это просто движок. Периодически я выпускаю какие-то демки и мини-игры, основанные на нем. Но окончательной версии, готовой к использованию сторонними лицами, пока нет.

Дело в том, что я неизлечимо болен перфекционизмом. Я постоянно улучшаю уже написанное и многократно переписываю все с нуля. Очень редко я бываю доволен сделанным, поэтому у меня не наберется и тысячи стабильных строк кода, которые бы я никогда не менял.

Другая причина – переход на язык D. Поскольку я любитель-одиночка, я могу позволить себе роскошь выбирать язык по своему вкусу – и я знаю, что все профессионалы завидуют таким, как я, хотя и тщательно скрывают это под маской снобизма и показной крутизны. К счастью, к тому моменту, когда я познакомился с D, у меня еще не была накоплена слишком большая кодовая база, и не было таких инструментов, от которых я не мог бы отказаться.

Мое представление о D изначально было весьма смутным: мне казалось, что это какое-то расширение C++ – наподобие того, как сам C++ является расширением C. Ознакомившись со статьей на Википедии, я почувствовал, что это именно то, что мне нужно: компиляция в машинный код, как у C++, и автоматическое управление памятью, как в Java (но надо сказать, что с Java мне на тот момент еще не приходилось работать, поэтому сравнивал я, естественно, с C++).

И это не считая многочисленных полезных мелочей, которые D унаследовал от своих многочисленных предшественников – C#, Python, Haskell и др. Обрадовало то, что компилятор D компактен и легок в установке: распаковал архив в любой каталог и работай. В одном архиве – версии для Windows, Linux и FreeBSD, плюс подробная документация по языку и стандартной библиотеке.

Я пишу на D уже несколько лет и очень доволен языком. С каждым релизом он становится все лучше. Не берусь рекомендовать его тем, кто вынужден тянуть вагон legacy-кода на C++ (все-таки, полной бинарной совместимости с C++ у D нет), а также тем, кто такого багажа не имеет, но планирует на полном серьезе трудоустраиваться куда-нибудь в качестве программиста (там тоже будет legacy). Но тот, кто свободен от этого балласта – хакер, любитель, инди-разработчик – обязательно оценит язык по достоинству. Даже если у вас есть некоторое количество кода на «плюсах», без которого вы не можете жить, ничто не мешает потратить пару вечеров и портировать его на D. Сам я так и сделал – некоторые компоненты моей библиотеки dlib были портированы с C++.

dlib начиналась как библиотека линейной алгебры (манипуляции над векторами, матрицами, кватернионами и т.д.) – такие пишет для себя каждый игровой программист. Алгебра сейчас составляет пакет dlib.math. Потом были добавлены средства вычислительной геометрии (dlib.geometry) – игровому движку требуются функции обнаружения пересечений фигур, пространственные измерения и т.д. Вслед за ней – пакет для хранения и обработки изображений (dlib.image), который родился в результате изучения мной различных алгоритмов фильтрации. Эта часть dlib еще далека от завершения – к примеру, полностью поддерживается только формат PNG, не реализованы многие другие функции. Однако dlib.image уже вполне годится для использования в играх. Чем будет dlib в будущем? Вероятнее всего, универсальной вычислительной библиотекой: в перспективе я планирую добавить поддержку обработки и кодирования аудио и видео. dlib можно будет применять как backend для построения игровых движков, симуляторов, аудио- и видеопроигрывателей, программ-конвертеров, различных графических редакторов, инструментов монтажа и композитинга.

Тимур Гафаров

Обновления Mesa

В конце августа состоялся релиз Mesa 9.2. Этот выпуск имеет экспериментальный статус – после проведения окончательной стабилизации кода, будет выпущена стабильная версия 9.2.1. В числе основных нововведений Mesa 9.2 – поддержка новых расширений OpenGL, определенных в стандартах OpenGL 3.2/3.3 и 4.x, поддержка аппаратного декодирования видео на видеокартах AMD (для доступа к этой функциональности используется интерфейс VDPAU), новый Gallium-драйвер для GPU семейства Adreno a220.

В настоящее время в Mesa обеспечена полноценная поддержка OpenGL 3.1 для видеокарт Intel и Radeon. Поддержка OpenGL 3.2/3.3 пока не является полной, но уже очень близка к данному состоянию – остается довести до конца реализацию GLSL 1.50 и поддержку геометрических шейдеров. Кстати, недавно разработчики из Intel добавили в Mesa большую серию патчей, реализующих поддержку геометрических шейдеров в драйвере Intel. Геометрические шейдеры позволяют генерировать новые графические примитивы (точки, линии, треугольники) после стадии вершинных шейдеров, используя GLSL. Отмечается, что в планы Intel входит полная реализация поддержки OpenGL 3.3 в своем драйвере до конца календарного года – и следующая Mesa может выйти с номером версии 10.0, отражающим факт достижения поддержки OpenGL 3.3.

Тем временм компания CodeWeavers представила результаты работы по улучшению поддержки Direct3D в Wine. В отличие от ранее доступной штатной прослойки для трансляции вызовов Direct3D в OpenGL, которая вызывала много нареканий с позиции производительности, новый код позволяет увеличить производительность игр на 50-100% и обеспечить скорость их запуска в Wine на том же уровне, что и в Windows – или даже быстрее! В частности, значительный рост производительности отмечается для игр на базе движка Source, StarCraft 2 и в бенчмарке 3DMark 2001.

Напомним, Mesa – это свободная реализация API OpenGL. Официально она не сертифицирована, но на практике вполне соответствует стандарту. Mesa ориентирована на обеспечение высокой производительности, в том числе за счет использования аппаратного ускорения работы с графикой, поддерживаемого видеоадаптерами. Mesa3D лежит в основе графической подсистемы операционных систем с открытым исходным кодом, так что этот проект имеет большое значение для всех пользователей, не имеющих возможности или не желающих использовать закрытые драйверы и библиотеки OpenGL от производителей оборудования. В данный момент Mesa является одной из самых популярных реализаций OpenGL для Unix-подобных ОС.

Новые игры под Linux

Metro: Last Light - шутер от первого лица, сиквел игры Metro 2033. Игра добилась высоких оценок со стороны критиков.

Half-Life: Source была создана в результате переноса классической Half-Life на более современный движок Source.

Один из разработчиков 3D-шутера от первого лица Metro: Last Light, созданного компанией 4A Games по мотивам популярного фантастического романа Дмитрия Глуховского «Метро 2033», подтвердил информацию о подготовке версии игры для Linux. Указано, что портирование пока находится на начальном этапе, и дата завершения работы пока неизвестна. Ранее в сети появились слухи о скором выходе Metro: Last Light для Linux, основанные на упоминании Linux в журнале изменений на странице игры в SteamDB.

Компания Valve представила Linux-версии игр Half-Life: Source и Half-Life Deathmatch: Source. Игры доступны в бета-режиме через клиент Steam. Кстати, общее число доступных в каталоге Steam игр для Linux на момент написания статьи достигло 172.

На конференции разработчиков игр GDC компании Valve и Nvidia выступили с докладом, посвященным портированию игр под Linux. В качестве мотивов почему следует портировать игры на OpenGL и Linux, упоминаются:

- Открытость операционной системы;
- Производительность и безопасность;
- Быстрый рост популярности Linux в качестве игровой платформы: портирование игр на Linux – логичный промежуточный шаг при портировании игр на мобильные платформы, где также доминируют стандарты семейства OpenGL. При использовании OpenGL, портирование игр на любые отличные от Windows платформы становится нетрудным делом;
- Возможность использовать все возможности оборудования – благодаря механизму расширений, OpenGL не ограничивает разработчиков приложений, как это происходит в случае с Direct3D. К примеру, Direct3D 10/11 недоступен для Windows XP – использование OpenGL позволяет охватить и эту ОС, которая до сих пор используется достаточно широко. К тому же спецификации OpenGL развиваются комитетом, в котором может принять участие любая заинтересованная сторона, для этого не требуется огромные суммы денег.

● Наконец, под Linux официально доступен клиент Steam;

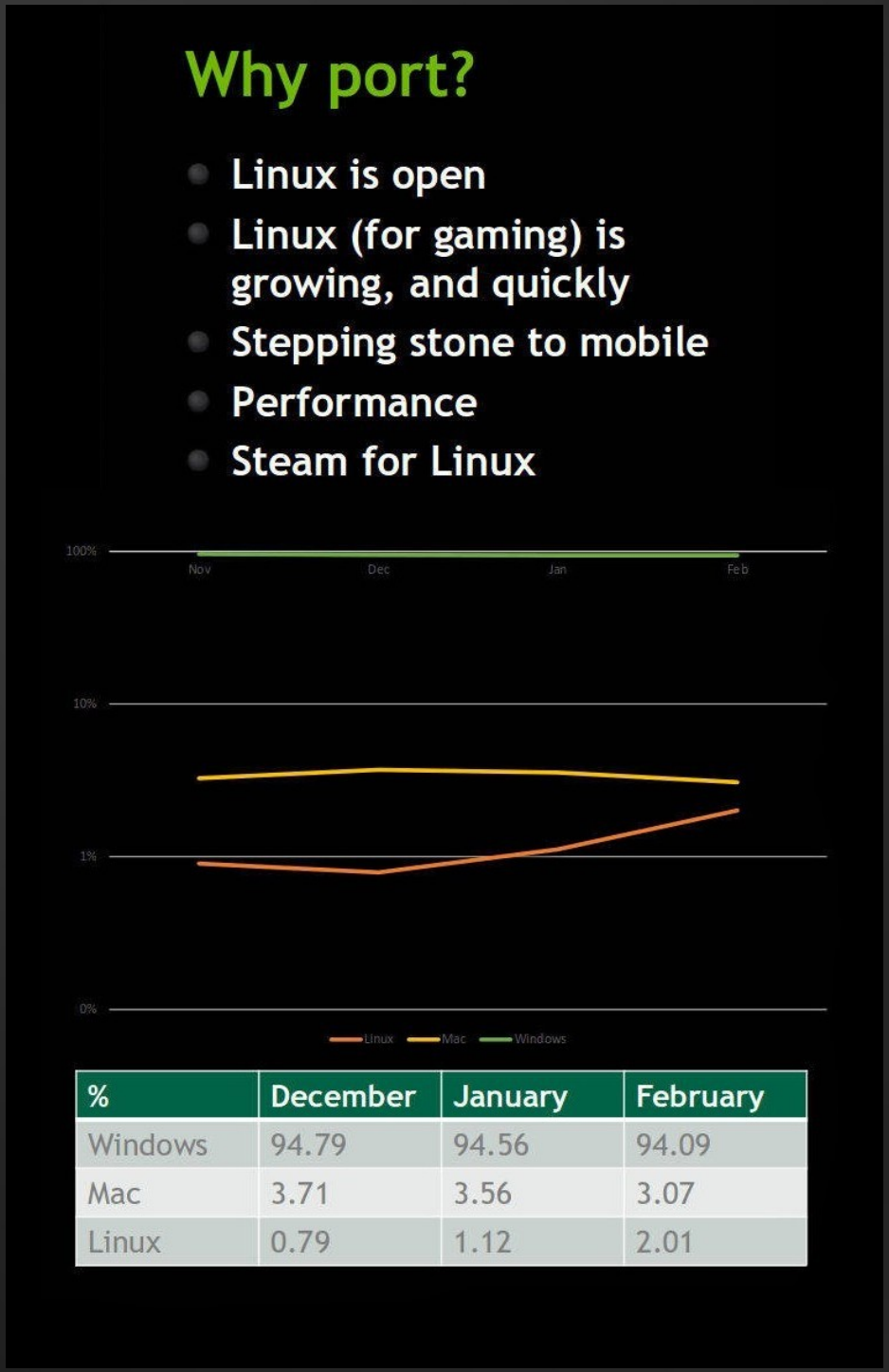
Для управления окнами и пользовательским вводом настоя-тельно рекомендуют использовать мультимедийную библиотеку SDL – Valve пользуется данной библиотекой при портировании своих проектов, что доказывает ее пригодность для достаточно требовательных применений. Кроме того, основной разработчик SDL – Сэм Лантинга – в настоящее время работает в Valve.

Valve также упомянула и проблемы, с которыми столкнулись программисты:

● Файловые системы в UNIX-подобных ОС чувствительны к регистру имен файлов, тогда как в Windows файловые системы по умолчанию игнорируют регистр. Для игр это, как правило, не является проблемой, так как игровые ресурсы обычно поставляются в платформо-нейтральных контейнерах. Тем не менее, в процессе разработки это может вызвать трудности. Наиболее простым решением является перевод имен всех ресурсов в нижний регистр.

● Вместо презиционного таймера RDTSC, специфичного для x86, рекомендуется использовать вызов clock_gettime, не зависящий от архитектуры процессора.

● Многие игры также используют raw mouse input, когда весь ввод мыши монопольно отправляется одной программе. Это хорошо работает для игр, однако некоторые оконные менеджеры при этом также перенаправляют и весь клавиатурный ввод – это может отключить работу Alt+Tab и других горячих клавиш, что способно вызвать неудовольствие пользователей в некоторых ситуациях.



%	December	January	February
Windows	94.79	94.56	94.09
Mac	3.71	3.56	3.07
Linux	0.79	1.12	2.01

«Корпорация зла»

Почему у Microsoft нет будущего

Хочу предупредить с самого начала: эту статью не стоит расценивать как попытку сыграть в Нострадамуса. Я лишь попробовал сделать выводы из того, что сложилось на сегодняшний день. Не поймите меня неправильно: я сам отчасти являюсь пользователем Windows, как и все. Но использовать Windows и исповедовать виндуизм – это не одно и то же. Я не желаю никому зла, я просто трезво оцениваю ситуацию.

Не так давно сотрудник Microsoft, работающий над ядром Windows NT, опубликовал через Tor анонимное публичное сообщение, в котором рассказал о причинах упадка компании. Развитие системы идет значительно медленнее, чем у конкурентов – но причины тому не технические, а, скорее, социальные. Над Windows никто не работает ради собственного блага – никто из программистов компании не заинтересован в том, чтобы сделать ее лучше. Время от времени в компанию приходят люди, которые пытаются что-то сделать, но они почти всегда терпят неудачу. Даже о безопасности своих продуктов Microsoft начала заботиться только после того, как она стала представлять серьезную угрозу для бизнеса. А о повышении производительности и речи не идет, они даже не ставят таких целей.

Проблему усугубляет структура подразделений внутри компании. Разработчики Windows поделены на отдельные команды, которые слабо взаимодействуют между собой. Обмена патчами между командами практически нет. Программисты не выполняют работы сверх плана, занимаются только рутинными задачами – полностью отсутствует творческий стимул. Энтузиазм здесь не поощряется. Сравните это с сообществом разработчиков ядра Linux: если кому-то удастся, например, повысить производительность обхода дерева каталогов хотя бы на 5%, то он герой и всеобщий любимец!

Microsoft страдает от утечки кадров. Она уже неспособна удержать у себя талантливых и опытных специалистов – все они постепенно уходят в Google и другие крупные компании из Сиэтла. «Корпорации зла» приходится нанимать студентов буквально с университетской скамьи. А молодежь, как правило, не имеет полного понимания того, как работают сложные системы – и, что самое главное, не хочет ничего менять в том, что уже работает. Молодые сотрудники предпочитают заниматься реализацией новых функций вместо улучшения старых.

Это хорошо видно по последним версиям Windows: они пестрят новинками, но старые проблемы, по сути, не исправлены. Доходит до смешного: даже сами программисты Microsoft понимают, что cmd.exe – это жуткий, просто вопиющий архаизм. Многие пытались его улучшить, но не смогли. Под Windows до сих пор нет приличной текстовой оболочки – ничего, что хотя бы немного приближалось по возможностям к Bash и UNIX Shell.

На каком-то выступлении десятилетней давности Стив Баллмер кричал «Developers, developers, developers!» Сейчас это можно расценивать как отчаянный вопль о помощи. Недавно компания начала предлагать «девелоперам» по 100 долларов за каждое приложение для Windows 8 и Windows Phone – не от хорошей жизни, надо думать...

При этом объемы откровенного бреда, которые выдает маркетинговый и правовой отделы компании, превышает все допустимые нормы. Вспомните хотя бы рекламную кампанию «Get the facts» и нашумевшее «дело Поносова». Недавно MS в очередной раз насмешила весь Интернет, направив в Google запрос об исключении из результатов поиска веб-страниц, предлагающих скачать халявный Офис – и, как оказалось, чуть ли не половина из этих «злостных пиратов» на самом деле распространяла бесплатный и открытый OpenOffice, который к продукции Microsoft никакого отношения, ясное дело, не имеет. А в другой раз в «списке пиратских сайтов» (видимо, по ошибке) оказалась одна из страниц сайта... microsoft.com. Ирония видна невооруженным глазом – вот уж кого действительно нужно заблокировать!

310. http://thepiratebay.se/torrent/5676574/Open_OFFICE_2010_3.2.0_FULL_
311. http://extrafr.net/torrent/cf80aefb2b70e418203fd14b9d63373d079d4922
312. https://piratereverse.info/torrent/5676574/Open_OFFICE_2010_3.2.0_FULL_(The_Complete_Office_Suite)
313. http://filesborn.com/free-download-office-7-microsoft_8.html
314. http://www.torrenthound.com/hash/cf80aefb2b70e418203fd14b9d63373d079d4922/comments
315. http://www.torrenthound.com/hash/cf80aefb2b70e418203fd14b9d63373d079d4922/files
316. http://www.torrzilla.com/torrent/open%2Boffice%2B2010%2B3%2B3%2B2%2Bfull%2Bthe%2Bperfect%2
317. http://bittrend.com/3m0nu-open-office-2010-3-3-2-full-the-perfect-open-office-suite-torrent.html
318. http://www.vitorrent.org/torrent/3ba9d6ec72178ddf31467d1aa12e0262e4223300
319. http://www.torrenthound.com/hash/3ba9d6ec72178ddf31467d1aa12e0262e4223300/comments
320. http://www.torrenthound.com/hash/3ba9d6ec72178ddf31467d1aa12e0262e4223300/files
321. http://www.torrenthound.com/hash/3ba9d6ec72178ddf31467d1aa12e0262e4223300/torrent-info/amcharts_ke
322. http://www.torrenthound.com/hash/3ba9d6ec72178ddf31467d1aa12e0262e4223300/torrent-info/Open-OFFI
323. http://www.frtorr.net/torrent/3ba9d6ec72178ddf31467d1aa12e0262e4223300

А вот еще один «перл»: на сайте официальной документации по Windows для разработчиков **MSDN** международный стандарт компьютерной графики OpenGL помечен как... «технология, не рекомендуемая к использованию». Мало того, что OpenGL использует, мягко говоря, немалая часть Windows-софта, весь рынок мобильных игр сейчас полностью сидит на OpenGL. DirectX сейчас есть только на Windows и Xbox, а OpenGL – буквально везде.

Что самое смешное: при всем своем странном понимании нужд разработчиков, MS вряд ли когда-нибудь откажется от поддержки OpenGL (как и от всех остальных своих исторических API) – потеря обратной совместимости с тоннами существующих Windows-программ будет последним гвоздем в крышку ее гроба.

Уже сейчас NVIDIA и Valve агитируют разработчиков игр переходить на Linux и OpenGL – Windows перестает удовлетворять даже геймеров. Это стало особенно ясно после выхода Windows 8.

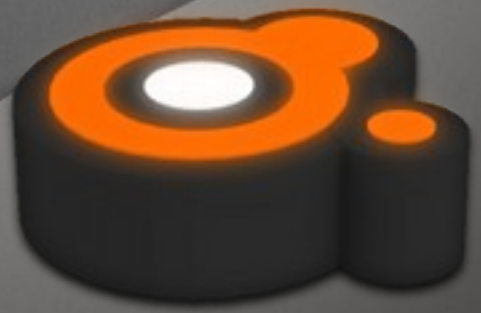
Но даже это не последний анекдот. Не успели HTC сотоварищи оправиться от патентных преследований MS в сторону мобильной ОС Android, как корпорация... сама направила в Еврокомиссию жалобу о применении Google нечестных методов конкуренции. Абсурдна сама формулировка претензии: в качестве одного из методов «нечестной» конкуренции называется бесплатное распространение Android в качестве свободного ПО, так как в данном случае имеет место «поставка продукта по цене ниже себестоимости, что не позволяет компаниям, развивающим проприетарные системы, развивать конкурирующие решения». Комментарии, я полагаю, излишни...

Все это говорит лишь об одном: способы ведения бизнеса, практикуемые Microsoft, перестают себя оправдывать в современном мире. Нынешние пользователи компьютеров уже не так наивны, как прежде: люди просто используют то, что им нравится, а не то, что навязано корпорациями. Те компании, которые это понимают, достигли успеха. А тех, что считает, что все еще можно делать деньги на, извините за выражение, экскрементах мамонта, будут постигать нудачи – это вполне закономерно.

Тимур Гафаров

Это все!

Надеемся, номер вышел интересным. Если Вам нравится наш журнал, и Вы хотели бы его поддержать – участвуйте в его создании! Отправляйте статьи, обзоры, интервью и прочее на любые темы, касающиеся игр, графики, звука, программирования и т.д. на **gecko0307@gmail.com**.



http://gplus.to/fpsmag