Независимый электронный журнал о разработке компьютерных игр
Издается с 2008 г. Доступен по CC-BY-NC-SA

№22
2013

# FPS

## «FPS» 5 лет спустя...

## Быстрая проверка столкновений на D

## OSL в Blender

## Закрытое железо:
что находится в «черном ящике»?

*+ многое другое!*


**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**

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

*8 февраля 2013 года журналу «FPS» исполнилось ровно 5 лет. За этот срок издание претерпело немало изменений: позади эксперименты с форматом и дизайном, поиски своей тематической ниши, вечные проблемы с местом размещения журнала в Интернете и множество других «приключений», через которые прошла редакция во главе с вашим покорным слугой. В связи с этим, хотелось бы оглянуться назад и подвести итоги первой «пятилетки», если можно так выразиться...*

# «FPS» 5 лет спустя...

История журнала «FPS» (название которого некоторые дотошные читатели расшифровывают как «АЗЫ» – по соответствию букв на клавиатуре) берет свое начало в 2008 г.

На тот момент в русскоязычной части Интернета практически не было электронных изданий, посвященных именно разработке компьютерных игр – существовавшие тогда игровые и компьютерные журналы, хоть и публиковали иногда материалы по этому направлению, но, как правило, не брали геймдев своей основной темой. Совсем иная ситуация наблюдалась за рубежом: огромной популярностью пользовались издания, адресованные, главным образом, инди-разработчикам. Это, в первую очередь, «Game Maker's Data Magazine», «GMTech», «MarkUp» и другая электронная периодика, корни которой уходят в сообщество пользователей конструктора игр с мировой популярностью Game Maker.

Многие из них, к сожалению, давно прекратили свое существование – но именно такие прекрасные PDF-издания, как «MarkUp», и послужили образцами и источниками вдохновения при работе над ранними выпусками «FPS».

Мы задумывали его как небольшой журнал для аудитории пользователей Game Maker, охватывающий, в основном, вопросы разработки трехмерных игр при помощи данного инструмента. Начинали буквально «с нуля»: у команды из двух человек не было ни опыта в компьютерной верстке, ни толковой издательской платформы, ни даже собственного сайта или какого-либо другого информационного ресурса – первый номер журнала, вышедший в свет 8 февраля 2008 г., был размещен буквально на файло-обменнике...

Зато был энтузиазм и бесконечное желание сделать как можно более качественный медиа-продукт, не уступающий западным аналогам. Удалось ли это нам – судить читателю, но широкий резонанс в русскоязычном геймдев-коммьюнити, тысячи прочтений на сервисе Issuu.com, десятки благодарных писем в редакцию – это что-нибудь да значит?..


Тогда, в 2008-м, нам удалось совершить неплохой старт: номер был достаточно тепло встречен читателями и авторами знаменитого эмуляционно-игрового журнала **«TOF»** – ныне, увы, закрытого.

На форуме команды «TOF» и был анонсирован «FPS» №1. Успешное становление журнала вряд ли было бы возможным без портала gamecreating.3dn.ru (ныне **gcup.ru**), который по интересному совпадению открылся примерно в одно время с «FPS».

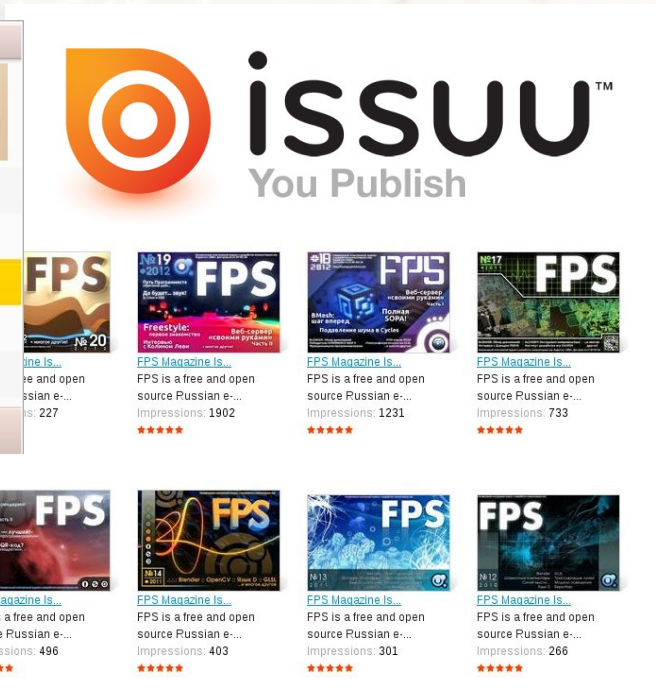
Администрация любезно разместила журнал и новость о его выходе на страницах своего сайта, который в короткие сроки стал одним из крупнейших и наиболее посещаемых геймдев-ресурсов Рунета. На новостном сервисе gcup.ru до сих пор размещаются анонсы новых номеров журнала.

Менее чем через месяц был выпущен второй номер. Журнал планировался как ежемесячный - первое время так и было, впоследствии «FPS» стал выходить реже. 30 марта 2008 г. начал работу сайт xtreme3d.hut2.ru (ныне **xtreme3d.narod.ru**), посвященный 3D-движку для Game Maker – Xtreme3D. Журнал долгое время хостился на нем. Последующие выпуски журнала в 2008 г. содержали много материала по Xtreme3D и другим аналогичным движкам. К сожалению, автор Xtreme3D забросил работу над проектом, и, в связи с отсутствием в открытом доступе исходников, сейчас этот движок уже не так актуален, как прежде. Поэтому мы впоследствии решили расширить тематические границы и вещать со страниц журнала о «особенностях национальной разработки игровых движков» на C++, с использованием «чистого» OpenGL, SDL и других библиотек.

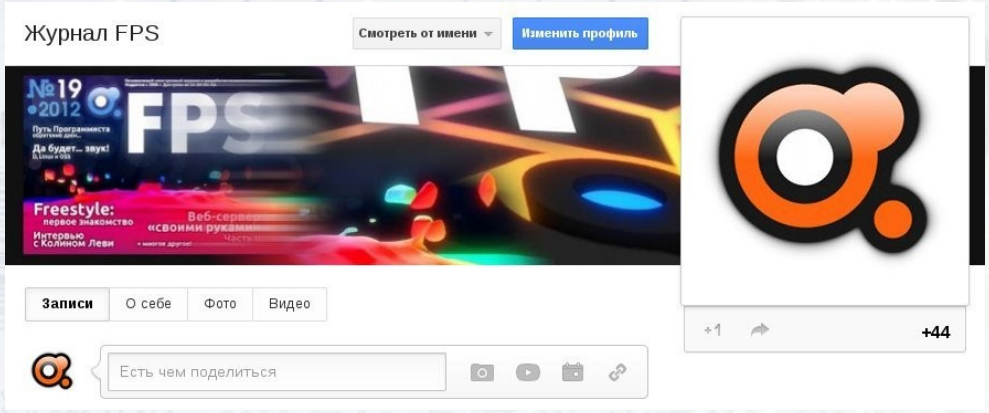

Шестой номер «FPS», выпущенный в ноябре 2008 г., в отличие от предыдущих, вышел в горизонтальном формате. Такое решение было принято по одной простой причине – горизонтальная полоса удобнее читается на горизонтальном дисплее. Идея была позаимствована у журнала **«BlenderArt»**. Кстати, материалы, посвященные свободному пакету трехмерного моделирования **Blender**, постепенно сформировали одно из центральных направлений нашего издания.

Журнал помнит и тяжелые времена. В течение 2009 г. был выпущен всего один номер. Это был критический период: одно время даже стоял вопрос о передаче проекта сторонним лицам. Помогли многочисленные письма и положительные отзывы от читателей. К счастью, кризисы миновали, и теперь мы делаем стабильные 4-5 номеров в год. Стабильность стала одним из основополагающих принципов нашей работы: в этом изменчивом мире, где ежегодно создаются и рушатся десятки стартапов, мы решили твердо обнадежить читателя своей незыблемостью.

Был этап сотрудничества со сторонними организациями: совместно с проектом **«Мобильный киоск»** была выпущена специальная Java-версия журнала, адаптированная для мобильных телефонов. Сейчас, впрочем, мы считаем, что век J2ME подходит к концу: наступила эра смартфонов и планшетов на базе Android, мобильные приложения реинкарнировались в другом формате, и чтение PDF на мобильном устройстве уже отнюдь не выглядит фантастикой. Куда более привлекательными оказались сервисы онлайн-просмотра документов – Issuu.com и GoogleDocs – которые мы взяли «на вооружение», когда Сеть окончательно изменила свое лицо в сторону Web 2.0.

Говоря о эпохе Web 2.0, нельзя не упомянуть бывший официальный сайт журнала (fpsmag.zymichost.com), который был открыт в октябре 2010 года и просуществовал всего пару лет. В связи со сменой политики нашего хостинг-провайдера, который летом 2012 года неожиданно заблокировал доступ к своим сервисам для России и ряда стран СНГ по причине спам-аттак, мы приняли решение вовсе не иметь собственного сайта и функционировать полностью на базе социальных сетей. Все-таки проект некоммерческий, оплату полноценного хостинга и доменного имени мы позволить себе пока не можем – а возможностей, предлагаемых бесплатными социальными сервисами, как оказалось, хватило с лихвой. Осенью 2012 года нашим новым «координационным центром» стала публичная страница в Google+ – **gplus.to/fpsmag**.




Совсем недавно количество выпущенных номеров журнала уже перевалило за двадцать – а когда-то мы сомневались, что оно дойдет и до десяти... Сегодня «FPS» – это журнал для программистов, художников, моделлеров, линуксоидов, энтузиастов движения СПО, хакеров и просто творческих людей, ищущих и умеющих находить в этом мире качественную «пищу для ума». Мы ратуем за свободу слова и свободный обмен информацией – журнал распространяется на условиях лицензии Creative Commons (CC-BY-NC-SA). Мы рады любому сотрудничеству, приглашаем в наш авторский коллектив всех желающих, приветствуем любые новые идеи и предлагаем возможность совершенно бесплатно размещать на страницах журнала анонсы и рекламу интересных проектов (не обязательно игровых, главное – интересных!)


Хотелось бы поблагодарить всех авторов, принявших участие в развитии «FPS». Благодарим и язвительных критиков, чьи высказывания порой приходится читать – поверьте, без вас не было бы того упорства, с которым мы из года в год делаем свою работу. Наконец, примите особую благодарность и вы, уважаемые Читатели, за вашу преданность и поддержку словом!

# Blender Новости

21 февраля 2013 года вышла новая версия Blender 2.66. Это один из самых значительных релизов за последние месяцы. В списке нововведений – множество новых функций:

● Наконец-то – дебют технологии, которая нужна как воздух всем пользователям пакета, небезразличным к режиму лепки: динамическая топология! Она «на лету» изменяет геометрию модели, прямо во время работы инструментов лепки. Это позволяет избежать деформации, связанной с наращиванием объема на низкополигональных частях мешей. Данное нововведение вплотную приближает Blender к специализированным программам трехмерной лепки, таким как ZBrush и Sculptris (в частности, оно является аналогом DynaMesh в ZBrush). Подробности по динамической топологии можно найти на **wiki.blender.org**.

● Вторая важная особенность: более тесная интеграция физического движка Bullet: поддержка симуляции твердых тел без необходимости запуска игрового движка. Это позволит реализовать максимально правдоподобную физику для анимации и кино. Как и динамическая топология, это «долгострой» – один из старых проектов GSoC, завершение которого пользователи ждали уже несколько лет...

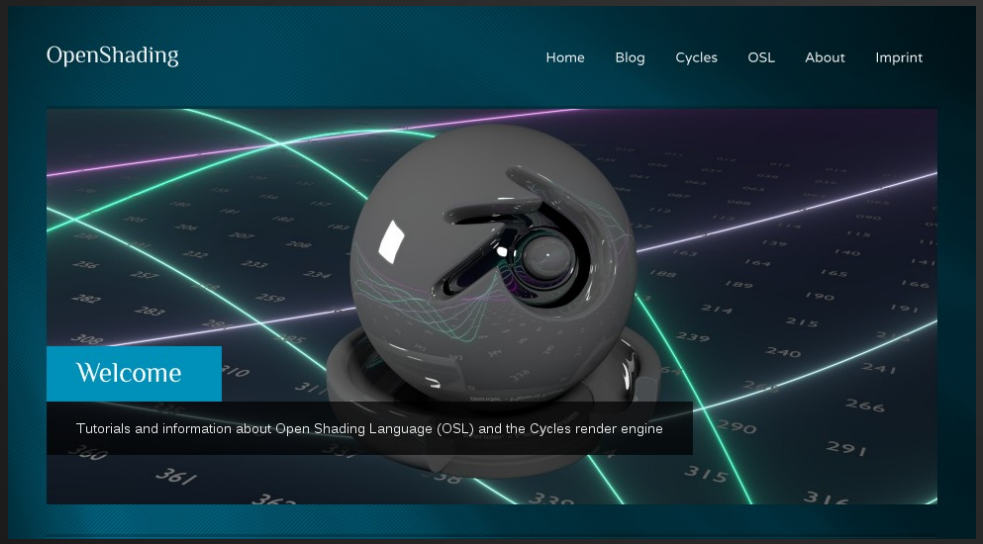
● Реализована поддержка рендеринга волос и меха в рендер-движке Cycles. Пока она находится на ранней стадии и имеет достаточно много ограничений, но уже показывает неплохие результаты на практике: 1000 нитей + 50 дочерних нитей у каждой, при 50 сэмплах на пиксель – за 2 минуты на CPU. Возможно использование узлов для управления цветом волос.


● Долгожданная возможность: инструмент «Фаска» (Bevel) теперь поддерживает отдельные вершины, а не только ребра. Кроме того, улучшены модификаторы для мешей, симулятор жидкостей, прозрачность для изображений, игровой движок, поддержка формата Collada и многие другие компоненты пакета.

Скачать сборку Blender 2.66 для Windows, Linux и Mac OS X можно на странице http://download.blender.org/release/Blender2.66/ Пользователям Linux следует обратить внимание, что разработчики окончательно отказались от поддержки сборок для старых версий glibc (ниже 2.11).

В конце 2012 года состоялся релиз Blender 2.65, который мы волею судеб не успели отследить вовремя. В данную версию разработчики пакета также внесли значительное число серьезных нововведений:

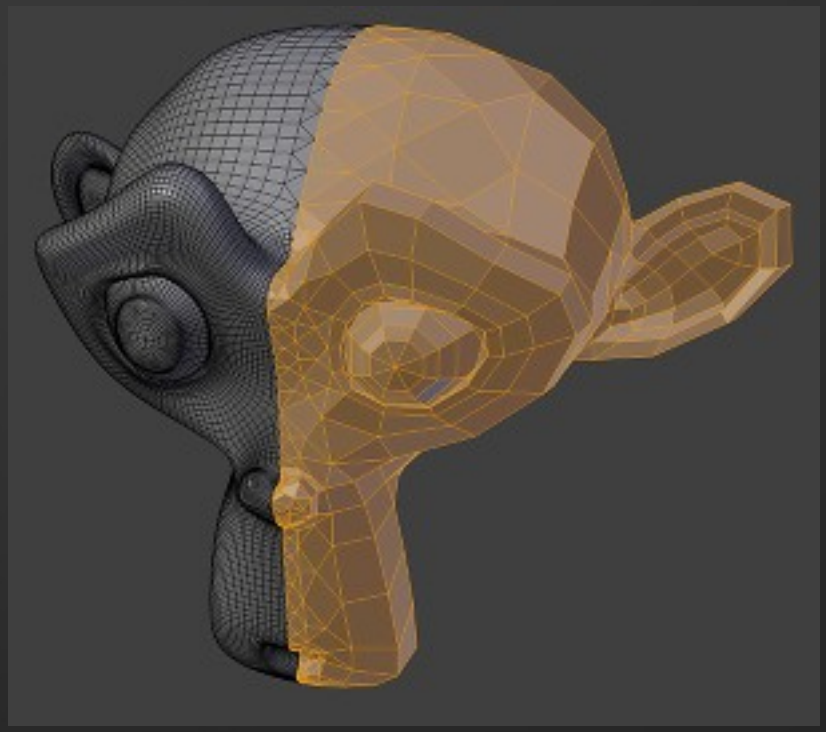
● Наконец-то появилась долгожданная поддержка Open Shading Language (OSL) – нового языка программирования шейдеров от Sony Pictures. Теперь рендер-движок Cycles является полностью программируемым в прямом смысле этого слова: материалы можно описывать не только в виде графа узлов, но и шейдерных программ в духе традиционных GLSL или RSL. К сожалению, OSL-шейдеры пока не компилируются для GPU и работают только в CPU-режиме – однако скорость исполнения байт-кода OSL и без того поражает воображение... Подробнее о языке и особенностях его взаимодействия с Cycles читайте несколькими страницами ниже.




● Для Cycles также добавлена поддержка анизотропных материалов и размытия при движении (Motion Blur).

● Подвергся серьезному обновлению симулятор дыма: теперь он «из коробки» поддерживает симуляцию огня. Объект Domain работает на порядок быстрее. Дым теперь можно выпускать с поверхности мешей без использования системы частиц.

● Новинки в сфере моделирования: улучшенный инструмент «Фаска» (Bevel) и невероятно полезный новый инструмент «Симметризация» (Symmetrize), позволяющий, как ясно из названия, симметризировать топологию относительно координатных осей. Кроме того, был улучшен модификатор Decimate, позволяющий уменьшить количество полигонов в меше - появились новые алгоритмы редуцирования. Также был добавлен новый модификатор Triangulate, «на лету» упрощающий N-гоны до треугольников.



Недавно были озвучены планы Брехта ван Ломмеля относительно стратегии развития Cycles в ближайшем будущем. В число первостепенных целей Blender входит улучшение эффекта смещения (Displacement), реализация «убер-шейдеров», сочетающих одновременно диффузные, бликовые и прозрачностные свойства, дальнейшая оптимизация OSL, а также множественная выборка по значимости (Importance Sampling) для источников света.

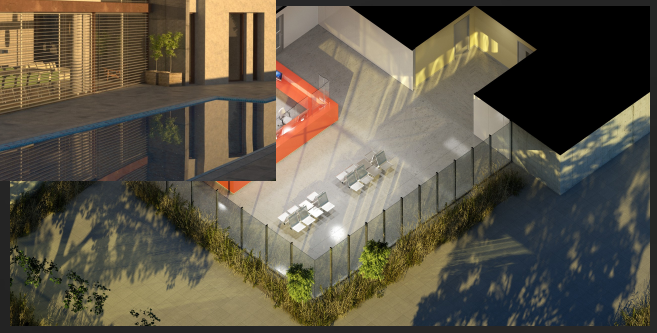
В долгосрочной перспективе – разносторонний рефакторинг BVH для улучшения ситуации с рендерингом волос и добавление поддержки воксельных текстур (отсюда, кстати, полшага до рендеринга в Cycles дыма и огня!)

Для энтузиастов, заинтересованных в развитии OSL в Cycles, открылся специализированный сайт **http://openshading.org**. На нем сосредоточена актуальная информация по OSL, уроки по языку, примеры шейдеров и многое другое.

Не так давно мы писали о проекте по портированию Blender на Java – не успели мы выразить предположение, что когда-нибудь пользователи увидят Blender под Android, как стало известно о начале работы по реализации поддержки этой мобильной ОС, по меньшей мере, для игрового движка BGE. Это один из проектов Google Summer of Code этого года – занимается им, кстати, наш соотечественник Александр Кузнецов. Он проводит работу по адаптации кода Blender Game Engine для работы с OpenGL ES 2.0 и выполняет частичное портирование Blender для Android. Конечной целью разработки является полное портирование всех компонентов Blender на платформу Android и формирование пригодной для конечных пользователей сборки. Вчерашняя фантастика на глазах становится реальностью!

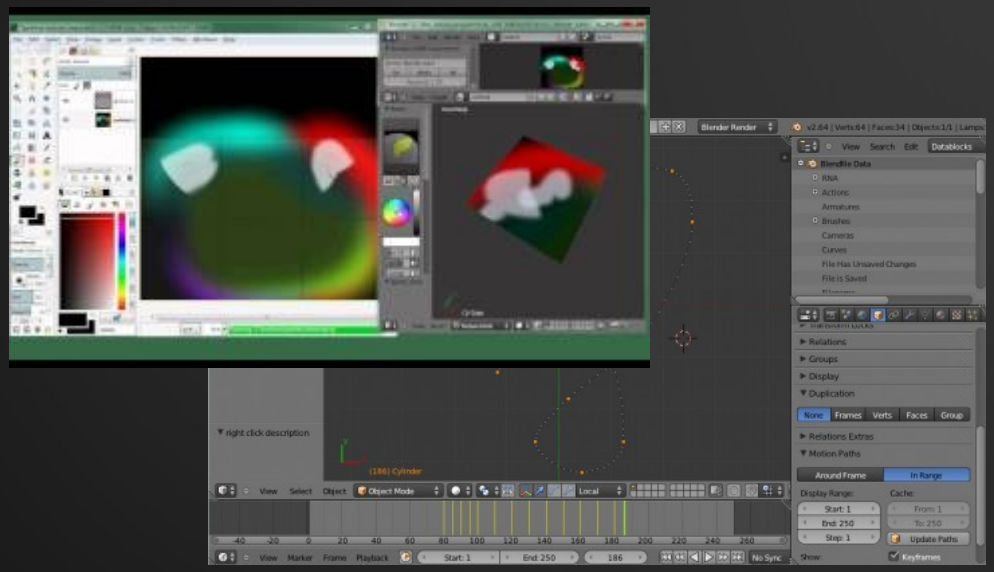


Вышла новая версия популярного коммерческого рендер-движка **Indigo**, предназначенного, в первую очередь, для архитектурной визуализации. Основными новинками Indigo 3.4 стали поддержка ортогонального рендеринга, секционных плоскостей (позволяющих, например, сделать невидимой часть стены здания, чтобы снаружи показать интерьер), карт нормалей, а также многочисленные улучшения по части производительности ядра движка. Indigo интегрируется в Blender посредством дополнения Blendigo.



Как говорит профессор Фарнсворт из «Футурамы» – хорошие новости! Дополнение External Paint Autorefresh, которое вошло в 3 выпуск нашего обзора плагинов для Blender, стало полностью бесплатным и открытым – исходники отныне распространяются по лицензии MIT. Напомним, EAP – это средство синхронизации изображений, одновременно открытых в Blender и графическом редакторе (Photoshop или GIMP). Ранее дополнение стоило $5.00, бесплатно распространялась только trial-версия. Теперь разработчики «делают ставку» на свой новый платный продукт: Real Time Animation – аддон, позволяющий записывать анимацию движения объектов в реальном времени, перетаскивая или поворачивая их мышью.

**https://sites.google.com/site/pointatstuffweb/external-paint-autorefresh**
**https://sites.google.com/site/pointatstuffweb/real-time-animation**



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!

**Вы разрабатываете перспективный проект? Открыли интересный сайт? Хотите «раскрутить» свою команду или студию? Мы Вам поможем!**

# Спецпредложение!

«FPS» предлагает уникальную возможность: совершенно БЕСПЛАТНО разместить на страницах журнала рекламу Вашего проекта!! При этом от Вас требуется минимум:

● **Соответствие рекламируемого общей тематике журнала**. Это может быть игра, программное обеспечение для разработчиков, какой-либо движок и/или SDK, а также любой другой ресурс в рамках игростроя (включая сайты по программированию, графике, звуку и т.д.). Заявки, не отвечающие этому требованию, рассматриваться не будут.

● **Готовый баннер или рекламный лист**. Для баннеров приемлемое разрешение: 800x200 (формат JPG, сжатие 100%). Для рекламных листов: 1000x700 (формат JPG, сжатие 90%). Содержание — произвольное, но не выходящее за рамки общепринятого и соответствующее грамматическим нормам. Совет: к созданию рекламного листа рекомендуем отнестись ответственно. Если не можете сами качественно оформить рекламу, найдите подходящего художника.«Голый» текст без графики и оформления не принимается.

● Краткое описание Вашего проекта и — обязательно — **ссылка на соответствующий сайт** (рекламу без ссылки не публикуем).

● Заявки со включенными **дополнительными материалами для журнала** (статьи, обзоры и т.д.) не только приветствуются, но даже более приоритетны.

**Заявки на рекламу принимаются на почтовый ящик редакции**: **gecko0307@gmail.com** (просьба в качестве темы указывать «Сотрудничество с FPS», а не просто «Реклама», так как письмо может отсеять спам-фильтр).

Прикрепленные материалы (рекламный лист, информация и пр.) могут быть как прикреплены к письму, так и загружены на какой-либо надежный сервер (убедительная просьба не использовать RapidShare, DepositFiles, Letitbit и другие подобные файлообменники — загружайте файлы на свой сайт, блог или ftp-сервер и присылайте статические ссылки). Все материалы желательно архивировать в формате zip, rar, 7z, tar.gz, tar.bz2 или tar.lzma.

# Open Shading Language

Open Shading Language (или, сокращенно, OSL) - это новый язык описания шейдеров, открытый проект, инициированный Ларри Гритцом в Sony Pictures Imageworks. OSL уже несколько лет используется в рамках собственного рендер-движка компании и успешно применяется в кинопроизводстве. Среди последних фильмов, где применялся этот язык – «Люди в черном 3», «Человек-паук», «Отель Трансильвания» и др. В начале 2010 года компания выпустила OSL как свободное ПО. Спецификация языка и исходный код референтной реализации доступны на GitHub:
**https://github.com/imageworks/OpenShadingLanguage**.

Совсем недавно поддержка языка появилась и в Blender/Cycles. Как и другие языки программирования шейдеров, OSL предоставляет удобный и гибкий процедурный способ описания свойств поверхностей материалов для получения достаточно сложных эффектов. Например, не так давно для Cycles был написан полноценный шейдер **Ambient Occlusion** всего в 66 строк кода. Язык компилируется в байткод, эффективность его исполнения на виртуальной машине очень высока.

В отличие от GLSL, HLSL, Cg, RSL и других уже существующих шейдерных языков, OSL максимально абстрагируется от конкретного графического конвейера – это позволяет интегрировать его практически в любой физически корректный рендер. У этого подхода есть свои неоднозначные особенности: к примеру, в OSL-шейдерах нет прямого доступа к данным о источнике света. Более того, шейдер не дает на выходе цвет пикселя в явном виде – вместо этого возвращается функция-замыкание, которая описывает уравнение отражаемого поверхностью света. Поэтому запрограммировать какие-либо новые модели освещения, как это делается в GLSL, так просто не получится – вы вынуждены оперировать только встроенными в рендер BRDF (BSDF).

Кто-то, возможно, разочаруется, но это оправданное ограничение. Отсутствие доступа к источникам света продиктовано парадигмой глобального освещения, от которой, как ни крути, никуда в Cycles не денешься – свет падает на поверхности не только напрямую от ламп, но и от других поверхностей. Это и заставляет полностью пересмотреть традиционный подход к программированию шейдеров...

Точно так же OSL ничего не знает и о пикселях на экране. Язык оперирует исключительно математикой BSDF. Замыкания после выполнения, опять-таки, выдают не пиксели, а значения энергетической яркости (radiance), выражаемые по системе СИ в Вт/(м²·ср) (ватт на кв. метр на стерадиан). По сути, все, что делает шейдер на OSL – это возвращает взвешенную сумму замыканий.

Замыкание, в контексте компьютерного «языкостроения» – это особая функция, которую можно хранить, передавать и принимать как обычный объект. Именно это и делается в OSL: вместо того, чтобы выбирать сэмплы и выполнять BSDF для определенного направления луча, возвращая из шейдерной программы результирующий цвет (по сути, статус кво), из нее возвращается специальный объект, описывающий саму процедуру выполнения BSDF.

Как только рендер-движок получает этот объект, он может использовать его для вычисления значений сэмплов по своему усмотрению – с учетом глобального освещения или без такового.

Предвещая скептицизм консервативных программистов, отметим: опыт Sony Pictures показывает, что такой инновационный метод описания шейдеров вполне оправдывает себя в «большом продакшене».

Несмотря на все эти тонкости, создание простейшего шейдера на OSL – дело нехитрое. Синтаксис языка близок к C и GLSL, и, если вам когда-либо приходилось писать шейдеры, освоиться будет нетрудно. Список стандартных типов данных OSL включает int (целочисленные значения), float (числа с плавающей запятой), color (значения цвета), vector (векторы), point (точки Евклидового пространства), matrix (матрицы 4x4) и string (строки):

```
int x = 10;
float Kd = 0.5;
color woodColor = color(0.7, 0.5, 0.3);
vector dir = vector(1.0, 0.0, 0.0);
point p = point();
string textureName = "wood.tx";
matrix m = matrix(
    m00, m01, m02, m03,
    m10, m11, m12, m13,
    m20, m21, m22, m23,
    m30, m31, m32, m33
);
```

Есть поддержка массивов:

```
color paintcolors[3] =
{
    color(0, 0.25, 0.7),
    color(1, 1, 1),
    Color(0.75, 0.5, 0.2)
};
```

Вы можете объявлять любые функции, возвращать из них значения, как и во всех языках с C-подобным синтаксисом. Функции могут объявляться локально внутри других функций. Основная шейдерная функция имеет спецификатор shader (либо surface, displacement, light или volume – в зависимости от специфики шейдера) и ничего не возвращает – ее вывод определяется в списке параметров.

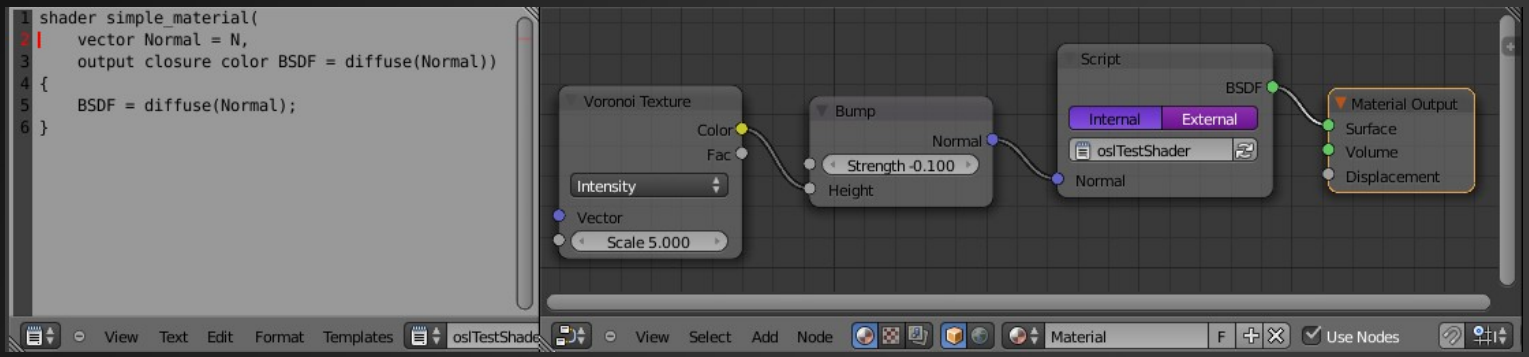
Чтобы использовать OSL, необходимо создать в текстовом редакторе Blender новый объект текста и ввести код шейдера. Затем в редакторе узлов нужно добавить узел «Скрипт» (Add -> Script -> Script) и указать текст с нашим кодом. Можно загрузить шейдер из внешнего файла – для этого нужно нажать кнопку External.

Вот, пожалуй, простейший шейдер, реализующий обычное диффузное рассеивание:

```
shader simple_material(
    vector Normal = N,
    output closure color BSDF = diffuse(Normal))
{
    BSDF = diffuse(Normal);
}
```

Все параметры шейдерной функции должны быть инициализированы значениями по умолчанию – этого требует спецификация языка, и, собственно, так и должно быть в случае с Blender: если пользователь не передаст никакое значение во входной разъем, узел будет использовать известное ему дефолтное.

В данном случае параметр только один – Normal – нормаль, которую использует диффузная BSDF. По умолчанию это обычная нормаль поверхности, но в нее можно передать, например, вектор нормали, полученный из карты высот – этим занимается узел Bump. На приведенном скриншоте показано, как именно это сделать.




В языке также предусмотрена поддержка метаданных – для передачи в хост-приложение таких вещей, как подсказки в графическом интерфейсе. Эти подсказки прекрасно работают и в Blender. Например:

Вот список BSDF, которые можно использовать в шейдерах:

```
diffuse(N)
oren_nayar(N, roughness)
reflection(N)
refraction(N, ior)
microfacet_beckmann(N, roughness)
microfacet_beckmann_refraction(N, roughness, ior)
microfacet_ggx(N, roughness)
microfacet_ggx_refraction(N, roughness, ior)
phong_ramp(N, exponent, colors[8])
diffuse_ramp(N, colors[8])
translucent(N)
transparent()
ashikhmin_velvet(N, roughness)
ward(N, T, roughness_u, roughness_v)
diffuse_toon(N, size, smooth)
specular_toon(N, size, smooth)
```

Подобно C, OSL поддерживает заголовочные файлы и макроопределения:

```
#include "stdosl.h"
#define PI 3.14
```

```
color DiffuseColor = 1
[[
    string help = "Diffuse color",
    float min = 0,
    float max = 1
]]
```

Это выражение объявляет параметр типа color и сразу же определяет минимальный и максимальный пороговые значения для него, а также указывает текст подсказки, который выводится на экран при наведении курсора мыши на элемент интерфейса, визуально представляющий этот параметр. Есть так называемые атрибуты – данные, получаемые из текущего контекста хост-программы. Например, вы можете получить случайное число через атрибут «object:random»:

```
float Random = 0;
getattribute("object:random", Random);
```

На это пока все, в дальнейшем мы будем более углубленно знакомить читателя с языком и на страницах журнала раскроем все тонкости разработки OSL-шейдеров «на все случаи жизни»...

Кстати, разработчик Томас Дингс, выполнивший львиную долю работы по интеграции OSL в Blender, недавно запустил новый сайт, посвященный языку – **http://www.openshading.com**. Там публикуется актуальная информация по OSL, видеоуроки, примеры различных шейдеров и многое другое.

# Python-скриптиг в Blender

## Добавляем новый тип объектов

В предыдущих номерах журнала мы уже затрагивали основы создания скриптов-расширений для Blender (цикл «Python и интерфейс Blender», FPS №12 '10, №13 '11). Сегодня мы рассмотрим еще один важный аспект скриптинга, который, несомненно, не обойдет стороной ни один заинтересованный игровой разработчик. Это принципы расширения списка объектов, которые пользователь может добавить на сцену. Предполагается, что с такими объектами будет работать плагин-экспортер контента для вашего движка – это может быть класс power-up'ов, врагов и других интерактивных сущностей, которые расставляются на карте.

Разумеется, вы не сможете создать класс объектов с абсолютно новыми опциями и стилем отрисовки – для этого придется внедряться в исходники Blender. Но расширить какой-нибудь уже существующий класс (например, Empty) пользовательскими свойствами (Custom properties) вполне реально.

Для реализации этой задумки нужно объявить специальный класс-оператор, наследующий от bpy.types.Operator:

```
class AddCustomObject(bpy.types.Operator):
  bl_idname = ("scene.add_custom_object")
  bl_label = ("Object :: add custom object")

  name = bpy.props.StringProperty()

  value = bpy.props.EnumProperty(
    attr = "values",
    name = "values",
    default = "collectible",
    items =
    [
      ("collectible", "Collectible", "Collectible"),
      ("spawnpos", "Spawn Position", "Spawn Position")
    ]
 )
```

Поле bl_idname – это имя оператора, items – список объектов, которые умеет создавать этот оператор. В данном случае это два типа игровых объекта – позиция появления игрока при старте уровня (spawnpos) и некий подбираемый приз (collectible).

Объявляем метод execute, который будет выполнен при вызове оператора. Добавив новый объект, указываем его тип (spawnpos или collectible) и соответственным образом добавляем нужные пользовательские свойства. Для collectible я добавил свойства CollectibleType и CollectibleCost – тип приза (по умолчанию – монета) и его ценность (по умолчанию 10).

```
  def execute(self, context):

    bpy.ops.object.add(
        type = 'EMPTY',
        location = bpy.data.scenes[0].cursor_location)

    for curr in bpy.data.objects:

      if curr.type == 'EMPTY' and curr.select:

        curr['ObjectType'] = self.value

        if self.value == "collectible":
          curr.empty_draw_type = 'SPHERE'
          curr.empty_draw_size = 0.5
          curr['CollectibleCost'] = 10
          curr['CollectibleType'] = 'coin'

        elif self.value == "spawnpos":
          curr.empty_draw_type = 'ARROWS'
          curr.empty_draw_size = 1.0

    return {'FINISHED'}
```

Осталось только зарегистрировать оператор. Текст «Custom Object» будет виден в меню добавления нового объекта (Shift + A), а 'LOGIC' – это идентификатор иконки, которая отображается рядом с этим текстом.

```
def menu_func_add_custom_object(self, context):
  self.layout.operator_menu_enum(
    PhotonAddObject.bl_idname,
    property = "value",
    text = "Custom Object",
    icon = 'LOGIC')

def register():
  bpy.utils.register_module(__name__)
    bpy.types.INFO_MT_add.append(menu_func_add_custom_object)

def unregister():
    bpy.types.INFO_MT_add.remove(menu_func_add_custom_object)
    bpy.utils.unregister_module(__name__)

if __name__ == "__main__":
    register()
```




# 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.

# Обзор дополнений Blender

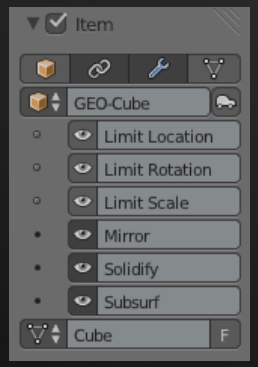
## Выпуск 3

*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...*

## Item Panel

Панель Item Panel отображает все блоки данных выделенного объекта, позволяет переименовывать и менять их, показывает все подключенные к объекту модификаторы и ограничения свободы, предоставляет возможность быстро переключать их видимость – и все прямо в окне трехмерной проекции, на правой панели инструментов. Идеальное вспомогательное средство для управления сложными сценами!

В будущем автор дополнения планирует также добавить на панель отображение материалов объекта, а также дать возможность пакетного переименования блоков данных для нескольких выделенных объектов сразу.





**https://docs.google.com/open?id=0BzRaJiKjEGRpMllpQjkzdXBXTlk**

## NI Mate

NI Mate – это программный комплекс для реализации захвата движения (Motion Capture) при помощи сенсора Microsoft Kinect. В связке с Blender, он позволяет записывать сложную анимацию телодвижений в реальном времени, без использования специальных костюмов и специализированного оборудования – вам понадобится только Kinect, подключенный к ПК. Полученная анимация может быть использована в том числе и в играх.


Напомним, Kinect – это популярный игровой «контроллер без контроллера», изначально разработанный для Xbox 360, «напичканный» датчиками для распознавания движений, голоса, жестов и мимики. С момента его выхода в 2011 году, Kinect стал излюбленным объектом внимания различных умельцев, хакеров, «стартапщиков», которые уже не в первый раз демонстрируют оригинальные способы применения этой технологии.

NI Mate доступен для Windows (включая Windows 8), Linux и Mac OS X. Пакет платный, лицензия на одного пользователя стоит € 198.40. Можно получить бесплатную trial-версию, если зарегистрироваться на официальном сайте. Она не имеет функциональных ограничений, но работает только в течение часа за один запуск и выводит nagscreen'ы.

**http://www.ni-mate.com**

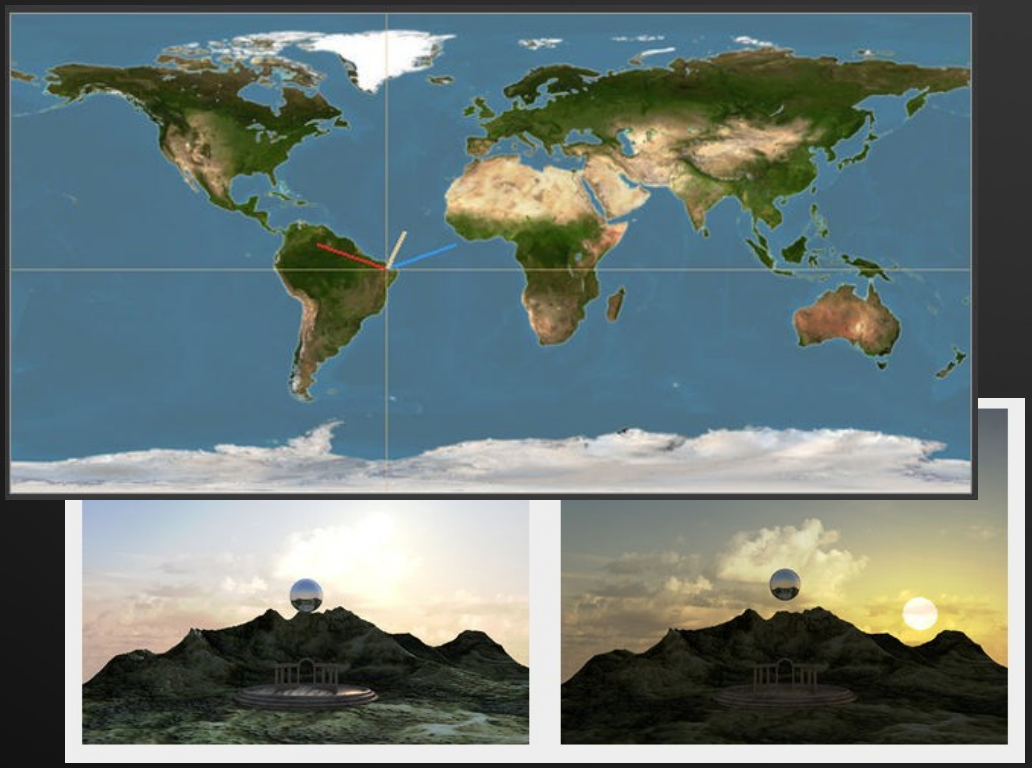
# Sun Position

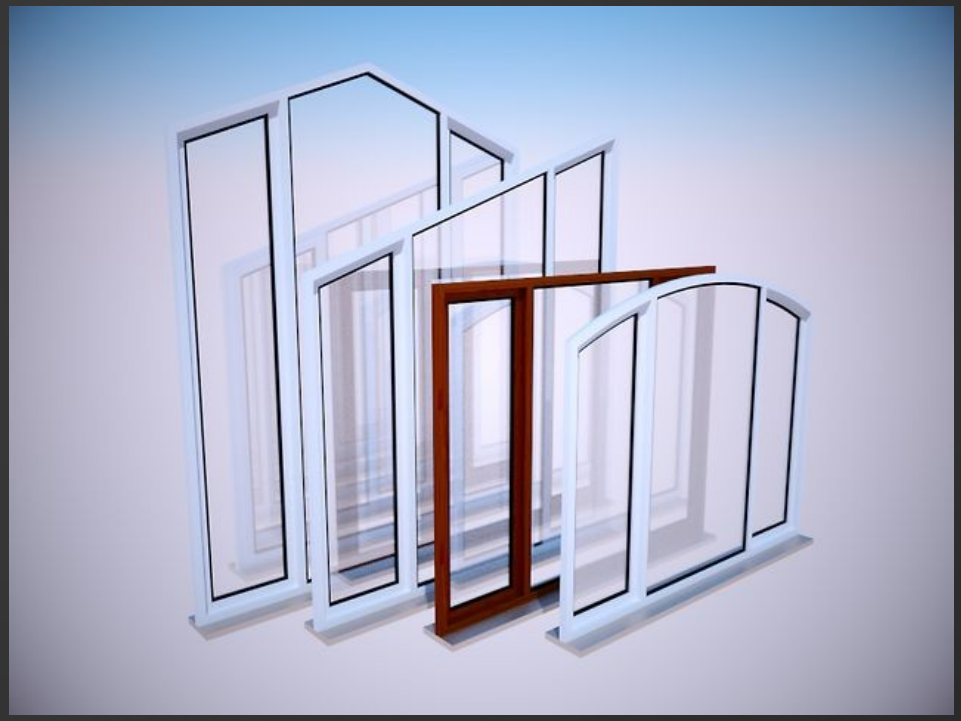
А это дополнение пригодится тем, кто занимается фотореалистичным рендерингом. Оно позволяет согласовать направление солнечных лучей на сцене с определенным географическим местоположением, временем и датой – для этого даже предусмотрен виджет в форме карты мира. Можно указывать часовые пояса.

Кроме того, дополнение поддерживает Cycles и предоставляет инструменты для вычисления правильного положения Солнца исходя из сферической HDR-карты окружения. Но и это еще не все: поскольку любое численное значение в Blender анимируемо, при помощи Sun Position вы даже можете реализовать анимацию плавной смены дня и ночи.

Дополнение распространяется по лицензии GPL.

**Страница проекта**





# Window Generator

Архитекторы и дизайнеры интерьеров наверняка оценят этот скрипт, расширяющий список примитивов Blender настраиваемым объектом окон. Дополнение вызвало у меня легкое «дежа вю»: нечто подобное я когда-то видел в 3ds Max. Здесь, правда, настройка может быть осуществлена только в момент создания объекта (после этого все изменения будут сохранены в виде статической геометрии), но зато дизайн окошек поддается почти неограниченной подгонке под нужный архитектурный стиль.

Можно создавать арочные или фронтонные завершения, контролировать пропорции оконного профиля, количество проемов по вертикали и горизонтали, толщину рам и т.д.

**http://dl.dropbox.com/u/23340323/add_Window.py**

*Анимационная студия Dreamworks выпустила в виде открытого продукта часть специализированного программного обеспечения, разработан-ного в процессе создания фильма «Хранители снов». Примечательно, что данная картина является одной из самых дорогих проектов студии, на ее создание было потрачено 145 млн долларов, а разработка необходимых для производства программных технологий заняла несколько лет...*

# OpenVDB

В итоге появился проект OpenVDB, в рамках которого подготовлен C++ тулкит для использования в производстве анимационных эффектов, создающих иллюзию объема – таких как дым или аморфные материалы. По сравнению с проприетарными продуктами (такими, как RenderMan и Maya), OpenVDB позволяет выполнять операции с аморфными материалами быстрее и используя меньший объем данных. В качестве причин открытия кода называется желание более повсеместного распространения формата VDB и превращения его в стандарт. Отмечается, что конкурентное преимущество от эксклюзивного использования созданных технологий сулит меньшую выгоду, чем появление штатной поддержки нового формата в анимационных пакетах различных производителей!

OpenVDB включает в себя библиотеку на C++ с реализацией иерархических структур данных и инструментов для эффективного хранения и обработки данных рассеянных и меняющихся во времени объемных объектов с дискретизацией по трехмерной сетке. В состав также включена реализация формата VDB, инструменты для преобразования мешей, а также набор алгоритмов для использования представленных структур данных для выполнения таких задач, как фильтрация, CSG, композитинг, сэмплинг, вокселизация и т. д.

Код тулкита поставляется под свободной лицензией Mozilla Public License. Компания Side Effects Software уже объявила о намерении включить в следующий релиз пакета Houdini средства для обработки объемных данных на базе OpenVDB. Примечательно, что модуль для интеграции с Houdini уже доступен для загрузки в исходных текстах. Кроме того, опубликован набор примеров моделей в формате VDB, для которого в состав OpenVDB включен специальный просмотрщик, использующий OpenGL.

# Интерьвю: Фабио Руссо

Фабио Руссо – разработчик нового интерфейса для слоев изображений в Blender. Он также работает над расширенным модификатором Array. Сайт **BlenderDiplom** не так давно взял у него интервью, перевод которого мы приводим на страницах журнала.

Если вам хочется видеть в Blender слои, оформленные как в Photoshop или GIMP, вы можете принять посильное участие в кампании по сбору средств, которую запустил Фабио на своем сайте:

**http://ruesp83.com**

Речь в интервью как раз идет о тонкостях фандрейзинга...

**Как ты попал в мир Blender?**

Посмотрев фильм «Тачки», я начал искать бесплатные программы трехмерного моделирования. Ничто не вызвало у меня такой восторг, как сцена, в которой герои едут через горы. Каждый раз, когда я ее вижу, я поражаюсь...

С этого и начался мой путь в 3D-графике. Я перепробовал несколько пакетов, но все они были слегка сложноватыми. Затем я натолкнулся на Blender. Хотя он тоже поначалу казался сложным, в нем была логическая структура.

Тот факт, что я программист и могу «покопаться» в исходном коде, заставил меня «влюбиться» в эту программу навсегда...

**Когда ты начал работу по улучшеню модификатора Array?**

Как-то раз я работал с Array для небольшой визуализации. Мне показалось, что он довольно «убогий», в нем даже нет возможности добавить элемент случайности. С тех пор я начал изучать исходный код Blender. Я начал с простых функциональных возможностей – таких, как упомянутая случайность.

Потом я ознакомился с тем, что предлагают в этом плане другие программы, и подумал, что все это неплохо бы реализовать и в Blender.

Конечно, впереди еще много работы. После интеграции BMesh я решил на время приостановить проект и вернуться к нему, когда я буду лучше знаком с этой новой системой.

**Твой второй проект – слои для изображений, именно для него ты и начал фандрайзинг...**

Меня вдохновили другие проекты по Blender, которые также занимаются сбором средств на разработку – и, учитывая, что у меня нет постоянной работы, это выглядело очень привлекательно. Но главное – это, конечно, мотивация: я хочу завершить проект, чтобы не разочаровать людей, которые в меня верят.

**Как ты определяешь, сколько «стоит» каждая новая реализованная функция/этап работы?**

Это было непросто... У меня ушло около недели на формирование плана работы. Определяя  бюджет проекта, я на каждом этапе сравнивал его с другими фандрейзерскими проектами и выбирал, что нужно сделать. На основании пользовательских потребностей я установил, на мой взгляд, достаточно разумные суммы для каждого этапа.

**Насколько хорошо проект финансируется – и, вообще, развивается – сейчас?**

Я думаю, что извлек пользу из спонсорских средств. Я могу посвящать проекту 5-6 часов ежедневно. Сейчас я работаю над интеграцией множественных слоев в Blender – с первого взгляда мне показалось, что это легко, но я столкнулся с определенными проблемами. Кроме того, остаются проблемы с интеграцией слоев в текстуру.

**Почему ты выбрал прямую оплату, а не одну из таких платформ, как IndieGoGo или ChipIn?**

Я брал пример с других проектов, а они все используют PayPal. Я мало что знал об этой системе платежей и начал изучать ее. И пришел к выводу, что она достаточно надежна для меня и для спонсоров – в том смысле, что я могу вернуть им деньги, если потребуется.

**Расширенный модификатор Array так и не был внесен в основную ветку Blender, когда ты приступил к работе над слоями. Программисты, работающие над другими проектами, уже «имеют за душой» по нескольку завершенных проектов, прежде чем берутся за сбор средств – не думаешь ли ты, что этот факт, в какой-то мере, сдерживает спонсоров от помощи тебе?**

Думаю, нет. Расширенный модификатор Array был просто моим способом изучить устройство Blender. Он далек от завершения, но я вернусь к нему после экзаменов в университете. Мне он очень нравится.

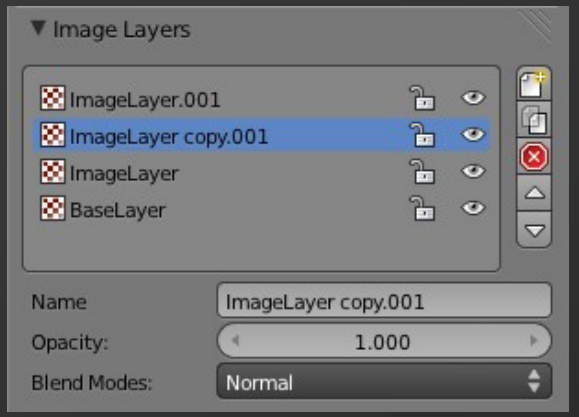
**Что ты делаешь для того, чтобы поддерживать интерес общественности к твоему проекту?**

Я надеюсь завершить все этапы как можно быстрее – считаю, что моя работа серьезно поможет художникам, работающим в Blender. Пользователи поймут, что это именно то, чего им всегда хотелось иметь в програме.

**Каковы, по-твоему, шансы того, что твой проект войдет в основную ветку?**

Когда он будет завершен, я вступлю в переговоры с соответствующими инстанциями, чтобы мы были уверены в необходимости внесения этого кода в trunk.

**Спасибо за интервью! Желаем тебе удачи с твоими проектами!**




Сборки Blender со включенными новыми слоями для изображений можно достать на GraphicAll:

Win32: **http://www.graphicall.org/909**
Win64: **http://www.graphicall.org/908**
Linux x86: **http://www.graphicall.org/1003**
Linux x86_64: **http://www.graphicall.org/1004**
Mac OS X 32-bit: **http://www.graphicall.org/1017**

*Сайт проекта:*
***http://ruesp83.com/***

*Оригинал интервью:*
***http://www.blenderdiplom.com***

*Перевод:*
***Gecko***

# GIMP: Новости

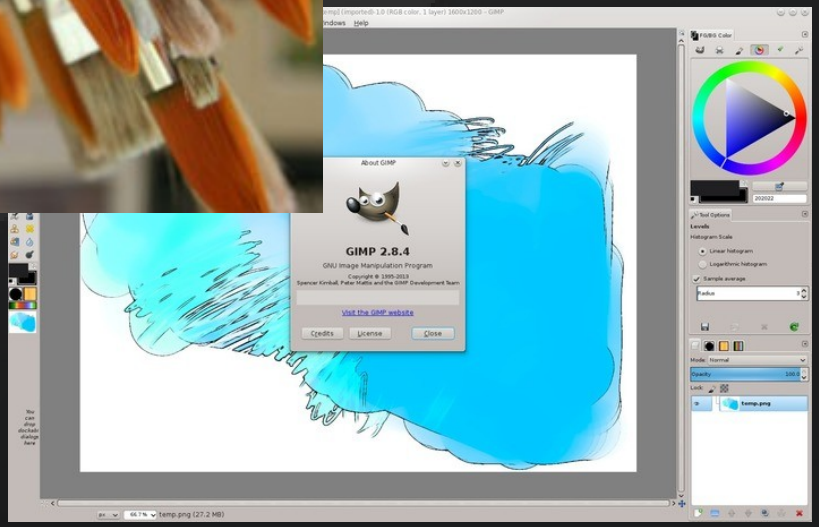
## Вышел GIMP 2.8.4

На днях тихо и незаметно произошло обновление стабильной версии GIMP. Это, в основном, исправляющий релиз – было выловлено множество багов, проведены улучшения интерфейса.

Самые важные изменения в этой версии:

- названия фильтров в диалогах сохранения и экспорта стали более внятными;
- контур кисти теперь перерисовывается намного быстрее;
- состояние окна программы (распахнутое на весь экран) стало запоминаться между сеансами;
- инструмент ввода текста теперь работает даже с изображением без слоев и умеет сохранять форматирование, заданное из диалога параметров инструмента;
- по умолчанию обратному концу стилуса (планшетного пера) автоматически назначается инструмент «Ластик» – как на настоящем канцелярском карандаше;
- была обновлена и подкорректирована русская локализация программы;
- улучшена поддержка платформы Mac OS X;
- улучшена работа плагина BMP.

http://gimp.org

# GIMP + Samsung Galaxy

Зарубежные энтузиасты предложили остроумное решение для того, чтобы нивелировать набившую оскомину путаницу между понятиями «графический планшет» и «планшетный компьютер»: приложение для Android 4.0+, превращающее любой планшет или телефон в графический планшет, который можно использовать вместе с GIMP. Что самое интересное, оно работает полностью по сети, безо всякой проводной связи между устройством и хост-машиной: после запуска приложения достаточно указать IP-адрес компьютера, где работает GIMP и приступать к рисованию: устройство будет передавать данные о позиционировании стилуса и силе нажатия.


Работа приложения успешно протестирована на Samsung Galaxy Note 10.1 и Samsung Galaxy S2. Разумеется, разрешение у таких гаджетов меньше, чем у настоящего графического планшета – но задачи превратить устройство на базе Android в аналог Wacom у разработчиков и не было. Из серьезных минусов стоит отметить отсутствие поддержки не-UNIX систем: драйвер написан специально для X11, поэтому работает эта «машинерия» только в Linux. Поддержки Windows и Mac в планах пока нет.

Все подробности – в **блоге разработчика**.

# GIMP Magazine: выпуск №2

Авторская команда англоязычного электронного журнала «GIMP Magazine» работают не покаладая рук: в декабре прошлого года вышел второй номер издания, в котором были опубликованы эксклюзивное интервью с Давидом Ревуа – известным художником-иллюстратором, уроки для начинающих пользователей GIMP, различные мастер-классы от профессионалов, галерея компьютерной графики и многое другое. Кроме того, журнал объявил конкурс «Wilber Art Contest» – на лучшее изображение знакомого всем волчонка Уилбера, талисмана программы.

На подходе уже третий номер «GIMP Magazine» – читателей ждет знакомство с творчеством талантливого фотографа Андреи Зановелло, а также основательное сравнение «по пунктам» двух гигантов обработки изображений: GIMP 2.8.0 и Adobe Photoshop CS 5.5 Extended. Выход номера намечен на 6 марта 2013 года.

«GIMP Magazine», кстати, как и наш журнал, доступен для онлайн-чтения на сервисе Issuu.com.

**http://gimpmagazine.org**

# Секреты мастерства Python-Fu

## Скрипт для создания миниатюр

Не знаю, как другие, но я терпеть не могу рутинную, механическую работу! Недавно вот пришлось вручную изготовлять миниатюры изображений произвольного разрешения, центрируя и уменьшая их до квадрата размером 128x128. К счастью, в GIMP есть возможность автоматизировать данное действие при помощи языка Python.

Создаем новый скрипт, импортируем gimpfu:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from gimpfu import *
```

Функция, которая непосредственно и выполняет работу по созданию миниатюры, принимает активное изображение, слой, ширину и высоту миниатюры:

```
def python_fu_thumbnailer(image, layer, width, height):
```

Сохраняем первоначальные размеры изображения:

```
  origWidth = pdb.gimp_image_width(image)
  origHeight = pdb.gimp_image_height(image)
```

Вычисляем новые размеры и позицию миниатюры относительно оригинала (она должна быть отцентрирована по вертикали или горизонтали – в зависимости от ориентации изображения):

```
  newWidth = origWidth
  newHeight = origHeight

  newX = 0
  newY = 0

  if (origWidth > origHeight):
    newWidth = origHeight
    newX = -(origWidth/2 - newWidth/2)
  elif (origWidth < origHeight):
    newHeight = origWidth
    newY = -(origHeight/2 - newHeight/2)
```

Масштабируем слой и приводим изображение к его размеру:

```
pdb.gimp_layer_resize(layer,newWidth, newHeight,newX,newY)
pdb.gimp_image_resize_to_layers(image)
pdb.gimp_context_set_interpolation(INTERPOLATION_LANCZOS)
pdb.gimp_image_scale(image, width, height)
```

Регистрируем скрипт:

```
register(
    "python-fu-thumbnailer",
    "Thumbnailer",
    "Thumbnailer 0.1",
    "Timur Gafarov",
    "(c) Copyright 2013 Timur Gafarov",
    "09-02-2013",
    "Make a thumbnail...",
    "RGB*, GRAY*",
    [
        (PF_IMAGE, "image", "Target image", None),
        (PF_DRAWABLE, "drawable", "Target layer", None),
        (PF_SPINNER, "width","Width:",128,(1,262144,1)),
        (PF_SPINNER, "height","Height:",128,(1,262144,1))
    ],
    [],
    python_fu_thumbnailer,
    menu = "<img>/Python-Fu/Transform")

main()
```

# Язык D

## Новости «с Марса»

### свежие релизы и обновления

## DMD 1.076 и 2.061

Состоялся релиз очередных минорных версий референсного компилятора DMD – 1.076 и 2.061 для D1 и D2 соответственно. Эта версия включает базовую поддержку 64-битных процессоров под Windows. Спецификация D2 пополнилась пользовательскими атрибутами времени компиляции для любых объявлений (User-Defined Attributes, UDA). Кроме того, было исправлено множество багов в самом компиляторе и стандартной библиотеке Phobos. Как обычно, доступны бинарные пакеты для всех платформ, включая Windows, Linux, Mac OS X и FreeBSD.

**http://dlang.org/download.html**

## LDC 0.10.0

Вышла новая версия LDC 0.10.0. Напомним, LDC – это компилятор D, использующий LLVM для генерации машинного кода. LDC соответствует спецификациям D1 1.075 и D2 2.060. Релиз приурочен к выходу LLVM 3.2. В будущем разработчики планируют координировать выпуск новых версий LDC с расписанием релизов DMD. Исходный код предоставляется под лицензией BSD. Доступны готовые бинарные сборки LDC2 для Linux (x86 и x86_64) и Mac OS X (x86_64). Идет работа над поддержкой Win64. Кроме того, как уверяют разработчики, в тестовой стадии работает генерация кода для остальных платформ инфраструктуры LLVM – таких, как ARM и Sparc. Кстати, на официальном Usenet-форуме D появилась отдельная группа, посвященная LDC: Digitalmars.D.ldc. А через какое-то время для Gentoo появился официальный оверлей LDC. Компилятор постепенно становится зрелым и привлекает все большее количество пользователей.

**https://github.com/ldc-developers/ldc**

## D Wiki

На официальном сайте D появился свой собственный википроект. Теперь любой пользователь может внести свою лепту в создание исчерпывающей базы знаний по языку, стандартной библиотеке, компиляторам и различным другим проектам, так или иначе относящимся к D.

**http://wiki.dlang.org**

## Сообщество D на Google+

Хорошая новость для активных пользователей сервисов «Корпорации добра»: Брэд Андерсон организовал группу программистов D в социальной сети Google+. В нее может вступить любой желающий – высказывайтесь, делитесь мнениями, анонсируйте свои проекты на D!

**https://plus.google.com/u/0/communities/100033468228217743303**

## DPaste переезжает

Основатель DPaste, популярного сервиса обмена исходниками и онлайн-выполнения кода на D, столкнулся с финансовыми проблемами – в связи с этим, DPaste начинает постепенный переезд на другой домен и сервер. Сервис временно доступен по адресу **http://dpaste.1azy.net**. Следите за новостями.

# Amber

Хоть D1 и подошел к концу своего жизненного цикла, еще существуют приверженцы первой версии языка, не желающие мириться с таким положением дел. Ларс Ивар Игесунд, один из создателей Tango, организовал проект Amber – новый язык программирования, надмножество D1, использующий LLVM в качестве бекэнда и Tango в качестве стандартной библиотеки. Amber написан на D и, в перспективе, нацелен на способность bootstrapping'а – компиляции самого себя. Проект находится на стадии альфа.

Что интересно, Ларс и его товарищи по Tango уже не в первый раз пытаются сформировать альтернативное сообщество программистов D1, независимое от «официальной» политики Уолтера Брайта. Сама Tango возникла именно с такой идеологией. Затем группа «тангистов» положила начало проекту LDC по разработке альтернативного компилятора D1. Но прошло время – и библиотека, и компилятор усилиями сторонних энтузиастов сфокусировались на D2, и процесс, похоже, уже не остановить: D1 постепенно становится историей. Но кто знает, возможно, Amber возродит D1 и сосредоточит вокруг себя еще одно крупное коммьюнити «дишников»?..

**https://bitbucket.org/larsivi/amber**

# ICE

Двумерный скролл-шутер с видом сверху, написанный на D. Геймплей игры напоминает Tyrian и Raptor: Call of the Shadows. Специальная компонентная система позволяет добавлять в игру новые типы объектов без перекомпиляции кода. Проект находится на ранней стадии разработки, но уже выглядит многообещающе.

**http://icegame.nfshost.com**

# YASTACS

За этой нетривиальной аббревиатурой скрывается длинная расшифровка: «Yet Another Space Trading And Combat Simulator». Это проект по созданию полноценного трехмерного космосима на D – с использованием Ogre в качестве графического движка. Исходники доступны по лицензии GNU GPL v2.

**http://yastacs.blogspot.hu**
**https://code.google.com/p/yastacs**

# Dgame

Dgame – молодой, но развивающийся движок для создания 2D-игр на D. Основан на OpenGL и SDL. Дизайн движка был вдохновлен такой библиотекой, как Pygame для Python, а также фреймворком SFML для C++.

**http://dgame-dev.de**

# Новый форум Derelict

Основатель и бессменный лидер проекта Derelict – Майк Паркер, более известный под ником Aldacron – объявил о переезде официального форума проекта с DSource на другой сервер.

**http://dblog.aldacron.net/forum**

# Higgs

D уверенно проникает в массы через веб-программирование! Максим Шевалье из Университета Монреаля (Канада) недавно представила Higgs – новый JIT-компилятор и быструю виртуальную машину для JavaScript, написанные на D. Поддержка распространяется на архитектуру x86_64 и POSIX-совместимые ОС (Linux, UNIX, Mac OS X и т.д.) Исходники доступны по лицензии BSD. Проектом, кстати, тут же заинтересовались в Mozilla.

**https://github.com/maximecb/Higgs**
**https://air.mozilla.org/higgs-jit**

## Abstract Database Interface

ADBI – это аналог ActiveRecord для Ruby, ООП-абстракция для доступа к реляционным базам данных. Она устроена так, что записи таблиц в БД соответствуют отдельным экземплярам класса. ADBI еще далека от завершения, но уже поддерживает SQLite.

**https://github.com/blm768/adbi**

## TinyRedis 1.2.1

Обновился D-драйвер к открытой документо-ориентированной СУБД Redis. Данный релиз облегчает взаимодействие D и Redis на уровне типов данных. Исправлены баги, улучшена документация.

Для справки: Redis хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения. Разрабатывается компанией VMware, крупнейшим производителем решений для виртуализации. Работает на большинстве POSIX-систем; в настоящее время Microsoft активно работает над переносом Redis на Windows.

**http://adilbaig.github.com/Tiny-Redis**
**http://redis.io**

## Враппер для Awesomium

Awesomium – это проприетарный фреймворк для разработки веб-интерфейсов в нативных приложениях. Например, его можно встраивать в игры – это позволит пользователям получить доступ к сетевому контенту, совершать покупки и участвовать в жизни игрового сообщества непосредственно из игры, не выходя из нее. Awesomium бесплатен для некоммерческого использования. Изначально фреймворк ориентирован на работу в .NET-приложениях, но теперь есть возможность использовать его и в D – благодаря проекту aws-wrap.

**http://code.google.com/p/aws-wrap**
**http://awesomium.com**

## Привязка к libldap

Для D появился биндинг к OpenLDAP – свободной реализации протокола LDAP (Lightweight Directory Access Protocol). Это протокол прикладного уровня для доступа к сетевым каталогам. Он был разработан консорциумом OSI и изначально ориентировался на локальные корпоративные сети и большие мэйнфреймы. Специалисты Мичиганского университета выполнили адаптацию этого сложного и редко используемого «простыми смертными» протокола к нуждам современного Интернета и персональным компьютерам.

**http://my.opera.com/run3/blog/2012/12/02/libldap-d-bindings**
**http://www.openldap.org**

## GtkD 2.1

GtkD – биндинг и объектно-ориентированная обертка над функциями кроссплатформенного тулкита Gtk+ – обновился до версии 2.1. Поддержка теперь распространяется на последнюю версию Gtk 3.6, исправлены баги, касающиеся работы под 64-битной Windows. Кроме того, GtkD собирается последней версией компилятора DMD 2.061. Для тех, кто еще не готов отказаться от Gtk 2, официально сохраняется поддержка ветки GtkD 1.x. Исходники проекта доступны по лицензии LGPL – она позволяет проприетарным программам статически линковаться с GtkD, не требуя при этом открытия кода. Это позволяет использовать биндинг для создания коммерческих продуктов.

**https://github.com/gtkd-developers/GtkD**

## Visual D 0.3.35

Вышла новая версия Visual D – проекта по интеграции D в среду разработки Microsoft Visual Studio. В новой версии – стабильный семантический анализатор, улучшения системы сборки и множество багфиксов.

**http://www.dsource.org/projects/visuald**
**http://www.dsource.org/projects/visuald/browser/trunk**
**https://github.com/rainers/visuald**

## Mono-D 0.4.9.5

Mono-D – плагин для среды MonoDevelop, предоставляющий поддержку языка D в этой IDE – обновился до версии 0.4.9.5. Этот и несколько предыдущиз минорных релизов включают статический анализ подмешиваний (mixins), условий времени компиляции (static if), ограничений (constraints), парсинг пользовательских атрибутов (UDA), многочисленные улучшения автозавершения кода и исправление различных багов.

**http://mono-d.alexanderbothe.com**

## CGDB 0.6.7

Состоялся релиз CGDB – легковесного фронтэнда для отладчика GNU с интерфейсом в стиле Vim. Ключевым нововведением данной версии является включение подсветки синтаксиса для D.

**http://cgdb.github.com**

## DVM 0.4.1

DVM (D Version Manager) – система для автоматической установки и управления различными версиями компиляторов D – обновилась до 0.4.1. Данный релиз носит, в целом, исправляющий характер.

**https://github.com/jacob-carlborg/dvm**

## Dlang-workspace

Это проект по созданию удобной инфраструктуры для работы над компилятором DMD и библиотекой Phobos, создания форков и веток, которые бы не конфликтовали со стабильным DMD, установленным на вашей системе по умолчанию. Dlang-workspace работает на Linux и Mac OS X, поддержка Windows – на подходе.

**https://github.com/carlor/dlang-workspace**

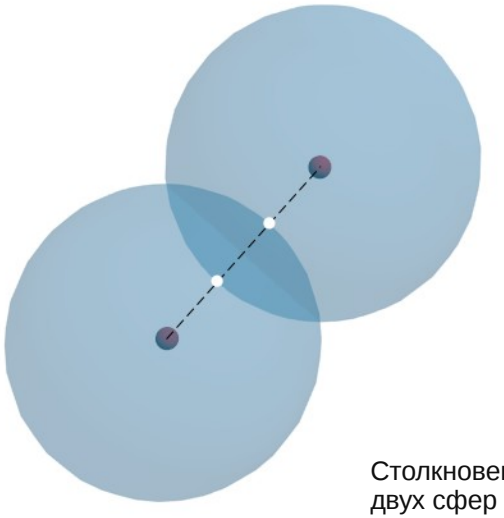


# Оптимизация проверки столкновений

Оглядываясь на многолетний опыт использования готовых игровых и графических движков (Game Maker, 3D Game Studio, Xtreme3D, GLScene, Irrlicht, BGE и др.), я не могу не отметить, что главное и первоочередное, о чем необходимо позаботиться при создании собственного 3D-движка – это быстрая система обнаружения столкновений. К сожалению, в Сети не так уж и много подробного материала на эту тему, как хотелось бы. Попробую исправить эту ситуацию!

В первую очередь нужно сказать, что существует далеко не один метод обнаружения и решения столкновений между игровыми объектами. В несложных 3D-играх они могут быть упрощенно представлены обычными ограничивающими сферами (Bounding Sphere), а пересечение двух сфер обнаружить крайне просто: если расстояние между центрами сфер меньше суммы их радиусов, то они пересекаются. Далее, при помощи нехитрых векторных выкладок, можно найти глубину их взаимного проникновения и нормаль контакта – и уже на основе этой информации просто-напросто «вытолкнуть» сферы друг из друга на нужное расстояние, на котором пересечение уже не обнаруживается. Такой подход будем считать базовым: он подойдет для столкновений между интерактивными объектами (игрок, враги, призы и различные другие сущности). Иногда вместо сфер используют другие геометрические тела – например, ориентированные боксы.

Гораздо сложнее обстоит дело с обнаружением столкновений игрока и персонажей с геометрией уровня – землей, зданиями, различными неподвижными объектами. Здесь, ясное дело, простыми телами не обойдешься – придется «честно» обнаруживать пересечения сферы с полигонами. Допустим, что уровень состоит только из треугольных полигонов. Функцию пересечения сферы и треугольника написать не слишком трудно, весь вопрос в вычислительной стоимости всей этой затеи: ведь, по идее, нужно проверить на пересечение все (!) треугольники, составляющие уровень – а их может быть десятки и даже сотни тысяч...

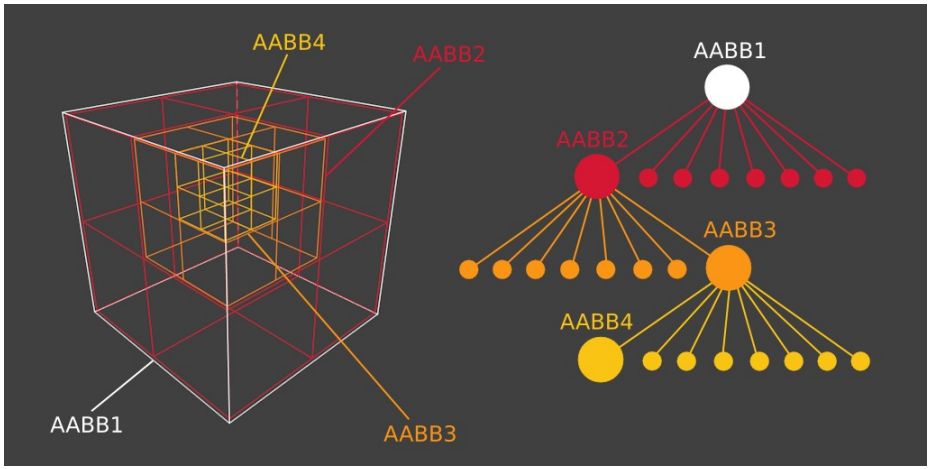


Столкновение двух сфер

Для того, чтобы сократить количество этих проверок, программисты неустанно придумывают все новые приемы и хитрости. Большинство из них основаны на представлении набора треугольников, составляющих сцену, в виде дерева. Узлы дерева – это геометрические тела, которые так или иначе отделяют одни группы треугольников от других, а листья – сами треугольники или их группы.

Когда мы хотим проверить весь уровень на пересечение со сферой, представляющей, например, игрока, то мы не перебираем все треугольники, а предварительно «спускаемся» вниз вдоль дерева, постепенно отсекая ненужные сектора пространства, и проверяем только 5-10 ближайших к игроку треугольников. Это и дает столь желанный выигрыш в быстродействии.

Существует несколько типов таких деревьев. В октарном дереве (Octree) в качестве узлов используются боксы, выровненные по координатным осям (AABB – Axis Aligned Bounding Box), рекурсивно разбивающиеся на восемь равных частей. За корень дерева берется большой AABB, который заключает в себе всю геометрию уровня. Он разбивается на 8 боксов-потомков, каждый из которых «забирает» свою часть геометрии. Те снова разбиваются, и так далее – пока не закончатся треугольники или не будет достигнута определенная глубина рекурсии.




В бинарном дереве (BSP) нет боксов – вместо них используются разделяющие плоскости. Каждый узел BSP-дерева имеет двух потомков, содержащих, соответственно, правую и левую группу объектов по две стороны от некой заданной плоскости в трехмерном пространстве.

Построить BSP сложнее, чем Octree – нужно использовать особые эвристики для нахождения наиболее выгодной разделяющей плоскости. Но выгоды от использования такого дерева оправдывают это: ведь проверить, с какой стороны от плоскости находится та или иная точка (например, центр ограничивающей сферы) до смешного просто, и весь процесс спуска по дереву занимает ничтожно мало времени.

Более того – наличие ограничивающих плоскостей позволяет с легкостью реализовать сортировку объектов в зависимости от удаленности: нужно просто рендерить всякий раз первыми те узлы, которые по отношению к виртуальной камере находятся на противоположной стороне плоскости. Поэтому неудивительно, что BSP-деревья широко используются ведущими разработчиками игровых движков – в частности, их постоянно применяют в своих продуктах id Software.

Мы же рассмотрим подробно построение третьего типа деревьев – BVH (Bounding Volume Hierarchy – иерархия ограничивающих объемов). Фактически, это обобщенный вариант идеи Octree. Здесь нет строгих правил: ограничивающими объемами могут служить не только AABB, но и любые другие тела. Количество потомков у узла может быть произвольным, а их размеры необязательно пропорциональны родительским.

Для простоты мы возьмем частный случай BVH: с AABB в качестве ограничивающих объемов и двумя потомками на узел – условно «правым» и «левым». Такой вариант, кстати, часто используется трассировщиками лучей для ускорения рендеринга: лучи, попадающие в «пустые» боксы, просто игнорируются и не проверяются на пересечение с объектами сцены.

Нечто похожее делается и при обнаружении столкновений. Сначала мы проверяем, находится ли ограничивающая сфера игрока в AABB корневого узла, содержащем весь уровень целиком. Если нет – то и нет смысла проверять дальше: игрок по определению не столкнулся с геометрией уровня. В противном случае, берем по очереди оба узла-потомка и проверяем, не находится ли игрок в их AABB. Продолжаем эту проверку рекурсивно для всей цепочки потомков до конечного, который потомков не имеет – в нем и содержится группа из нескольких треугольников. И только теперь проверяем, пересекает ли игрок эти треугольники. В итоге вся процедура сокращается до нескольких десятков проверок на пересечение сферы с AABB (очень «дешевая» с вычислительной точки зрения процедура) плюс 5-10 проверок на пересечение сферы с треугольником. Почти «бесплатный» алгоритм!

Для реализации алгоритмов в статье я использовал язык D, причем, в самых простых и ясных конструкциях, которые наиболее близки к псевдокоду. Из-за ограничений формата я не буду приводить здесь реализацию AABB, функций пересечения и различных других общеупотребительных компонентов.

```
class BVHNode
{
 Triangle[] tris;
 AABB aabb;
 BVHNode[2] child;
}

void checkBVHNodeVsPlayer(
 BVHNode node,
 Player player)
{
 if (node.aabb.intersectsSphere(player.sphere))
 {
  if (node.child[0] !is null)
   checkBVHNodeVsPlayer(node.child[0], player);
  if (node.child[1] !is null)
   checkBVHNodeVsPlayer(node.child[1], player);
  foreach(tri; node.tris)
  {
   Contact c = checkTriVsSphere(tri, player.sphere);
   if (c.intersectionFact)
    player.position += c.normal*c.penetrationDepth;
  }
 }
}
```

Таким образом, использовать уже готовое BVH-дерево очень просто. Куда сложнее его построить – и основная сложность заключается в том, как разделить треугольники в AABB на две группы для двух узлов-потомков.

Самый простой способ – выбрать разделяющую плоскость аккурат по центру бокса: сначала выбираем ось (x, y или z), вдоль которой бокс имеет макимальный размер, затем разбиваем его плоскостью, перпендикулярной этой оси и проходящей через центр бокса.

Таким образом, использовать уже готовое BVH-дерево очень просто. Куда сложнее его построить – и основная сложность заключается в том, как разделить треугольники в AABB на две группы для двух узлов-потомков.

Самый простой способ – выбрать разделяющую плоскость аккурат по центру бокса: сначала выбираем ось (x, y или z), вдоль которой бокс имеет макимальный размер, затем разбиваем его плоскостью, перпендикулярной этой оси и проходящей через центр бокса.

```
struct SplitPlane
{
  float split;
  Axis axis;
}

SplitPlane getHalfMainAxisSplitPlane(AABB box)
{
  Axis axis = boxGetMainAxis(box);
  return SplitPlane(box.position[axis], axis);
}
```

Однако этот способ имеет плохую адаптивность. Классическое Octree имеет тот же недостаток: в каждом узле получается достаточно много пустого пространства. Существует распространенная эвристика, позволяющая разбить бокс более «умно» – SAH (Surface Area Heuristic). Введем функцию стоимости обхода (traversal cost), которая будет показывать, насколько дорого по вычислительным ресурсам будет обработать данный узел. Стоимость вычисляется по следующей формуле:

$$C_M = E + \frac{S_{\text{Left}} N_{\text{Left}}}{S_M} + \frac{S_{\text{Right}} N_{\text{Right}}}{S_M}$$

Где C – результирующая стоимость, M – узел, Left и Right – левый и правый потомки, E – стоимость прослеживания пустого узла (некоторая константа, для простоты ее можно принять равной нулю), S – площадь поверхности узла, N – число примитивов (треугольников) в узле. Нужно выбрать разделяющую плоскость так, чтобы минимизировать эту функцию.

```
float SAHCost(AABB leftBox, uint numLeft,
              AABB rightBox, uint numRight,
              AABB parentBox)
{
  return surfaceArea(leftBox)
       / surfaceArea(parentBox) * numLeft
       + surfaceArea(rightBox)
       / surfaceArea(parentBox) * numRight;
}
```

Площадь поверхности для бокса вычислить нетрудно:

```
float surfaceArea(AABB box)
{
  float width = box.pmax.x - box.pmin.x;
  float height = box.pmax.y - box.pmin.y;
  float depth = box.pmax.z - box.pmin.z;
  return 2.0f *
    (width * height
   + width * depth
   + height * depth);
}
```

Алгоритм посторения BVH с учетом SAH выглядит следующим образом: мы перебираем несколько возможных вариантов разбиения, вычисляем для каждого из них стоимость. Та разделяющая плоскость, которая дает минимальную стоимость, и используется для разделения бокса надвое.

Вопрос в том, какие варианты рассматривать, и в каком количестве. В моей реализации BVH я сначала нахожу главную ось, как описано выше, затем прохожу вдоль нее на некотрое заданное количество шагов – например, 12. Иными словами, при нахождении минимальной стоимости обхода, я рассматриваю 12 вариантов разделения бокса по главной оси:

```
SplitPlane getSAHSplitPlane(
  Triangle[] tris, AABB box)
{
  Axis axis = boxGetMainAxis(box);

  float minAlongSplitPlane = box.pmin[axis];
  float maxAlongSplitPlane = box.pmax[axis];
  float bestCost = float.nan;
  float bestSplitPoint = float.nan;

  const int iterations = 12;
  foreach (i; 0..iterations)
  {
    float valueOfSplit =
      minAlongSplitPlane +
      (
        (maxAlongSplitPlane - minAlongSplitPlane) /
        (iterations + 1.0f) * (i + 1.0f)
      );

    SplitPlane SAHSplitPlane =
      SplitPlane(valueOfSplit, axis);
    AABB[2] boxes =
      boxSplitWithPlane(box, SAHSplitPlane);

    uint leftTrisNum = 0;
    uint rightTrisNum = 0;

    foreach(tri; tris)
    {
      if (boxes[0].intersectsAABB(tri.boundingBox))
        leftTrisNum++;
      else if (boxes[1].intersectsAABB(tri.boundingBox))
        rightTrisNum++;
    }

    if (leftObjectsLength > 0 && rightObjectsLength > 0)
    {
      float cost = SAHCost(boxes[0], leftTrisNum,
                           boxes[1], rightTrisNum, box);
      if (bestCost.isNaN || cost < bestCost)
      {
        bestCost = cost;
        bestSplitPoint = valueOfSplit;
      }
    }
  }

  return SplitPlane(bestSplitPoint, axis);
}
```

Теперь построение BVH-дерева выглядит тривиально:

```
BVHNode buildBVH(Triangle[] tris,
  uint maxTrisPerNode)
{
  AABB box = boxFromTriangles(tris);
  SplitPlane sp = getSAHSplitPlane(tris, box);

  AABB[2] boxes = boxSplitWithPlane(box, sp);

  Triangle[] leftTris;
  Triangle[] rightTris;

  foreach(tri; tris)
  {
    if (boxes[0].intersectsAABB(tri.boundingBox))
      leftTris ~= obj;
    else
    if (boxes[1].intersectsAABB(tri.boundingBox))
      rightTris ~= obj;
  }

  BVHNode node = new BVHNode();
  node.tris = tris;
  node.aabb = box;

  if (leftTris.length > 0 || rightTris.length > 0)
    node.tris = [];

  if (leftTris.length > 0)
    node.child[0] =
      buildBVH(leftTris, maxTrisPerNode);
  else
    node.child[0] = null;

  if (rightTris.length > 0)
    node.child[1] =
      buildBVH(rightTris, maxTrisPerNode);
  else
    node.child[1] = null;

  return node;
}
```

# 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/**

# Дао программиста...

*Некоторые из принципов, которых я придерживаюсь в своей работе, можно выразить в виде нижеследующих афоризмов. Они частично основаны на философии UNIX-way, подробнее о которой вы можете прочитать в «FPS» №19 ('12).*

1. Лучшая программа – это та, которая не была написана.

2. Простота – мать надежности.

3. Сложное должно быть простым.

4. Обычный программист думает: «Как сделать так, чтобы оно работало?»
   Хакер думает: «Как сделать так, чтобы оно не сломалось?»

5. Оптимизация – это зло.

6. Если программа падает, в ней есть ошибка.
   Если программа не падает, в ней все равно есть ошибка.
   Просто она еще себя не проявила.

7. Защита от дурака важнее защиты от недоброжелателя.

8. Хороший код не нуждается в комментариях.

9. Красивое решение – правильное решение.

10. Улучшить порой сложнее, чем переписать заново.

# Закрытое железо

## Что находится в «черном ящике»?

**В «FPS» №20 (2012) мы уже задавались вопросом – что ждет ПК в ближайшем будущем, если сохранятся нездоровые тенденции к превращению персонального компьютера из мощного, гибкого инструмента для работы, творчества и самообразования в некий урезанный донельзя по функцио-нальности «терминал» для фейсбуков и инстаграмов, при помощи которого люди будут просто убивать без пользы свое свободное время в социальных сетях, а корпорации тем временем – заколачивать деньги? Предлагаем задуматься еще раз: что такое ПК – модная игрушка или все-таки вычис-лительная машина?..**

Лично мне очень досадно, что слово «планшет» окончательно приобрело априорный смысл «планшетный компьютер». Всего каких-то пять лет назад под ним понимался дигитайзер – графический планшет для рисования, а еще раньше – деревянная доска для закрепления бумаги, которой пользуются чертежники. Ныне же планшет – это то, что нам пытаются впихнуть вместо полноценного компьютера. Не поймите меня неправильно – у меня тоже есть планшетный компьютер, но я не расцениваю его как замену ПК предыдущих поколений. Я использую его для чтения книг и играю на нем в Angry Birds – и не более того! Но беда в том, что массовому потребителю, как оказалось, от компьютера большего и не нужно. Он воспринимается людьми как «черный ящик», подобный телевизору, микроволновке или любому другому бытовому прибору – телевизор показывает каналы, микроволновка греет еду, а компьютер открывает любимый «вконтактик»...

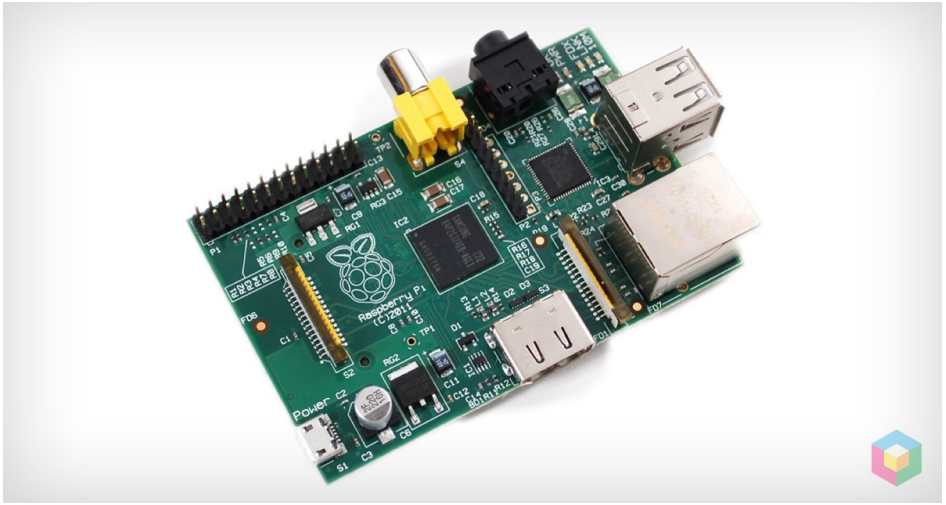
Однако прежний, «классический» ПК никогда не был "черным ящиком"! Он всегда имел разборное устройство, позволял подбирать индивидуальные конфигурации комплектующих, выжимать из этого «железа» максимум производительности, творить разнообразные «чудеса», экспериментировать, изобретать и придумывать. И многие этим пользовались – этих людей и прозвали хакерами. Думаю, что не погрешу против истины, предположив, что нынешним корпорациям такой класс пользователей невыгоден, неприемлем и вообще противен. Кстати, само слово «хакер» в последние годы приобрело подчеркнуто негативный оттенок – случайно ли?..

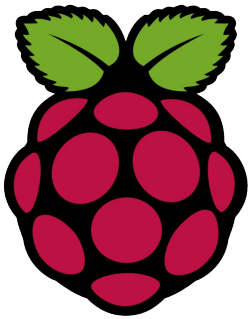
Один из кошмарных симптомов всей этой чумы, поразившей рынок персональных компьютеров – пресловутая «безопасная загрузка» в UEFI. Ведущие поставщики Linux-систем, вместо того, чтобы бороться с этой заразой, организовывать петиции, разговаривать с производителями оборудования (делать хоть что-нибудь!), «прогнулись» под Microsoft – иначе не скажешь. Ubuntu, SUSE, Fedora – все как один подписывают первоначальные загрузчики ключом MS. Этим уже занимается даже Linux Foundation с их собственным загрузчиком loader.efi. То есть, «Да здравствует SecureBoot!» – так, выходит?..

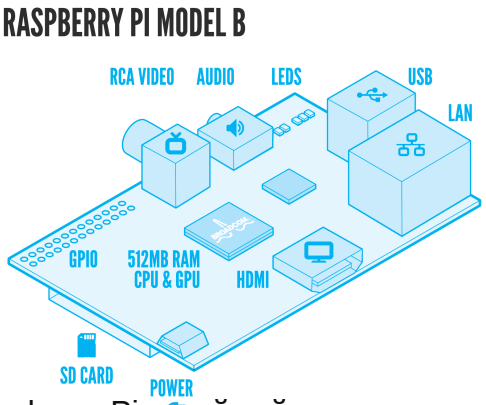
Дело ведь не в том, сможем ли мы установить Linux на свой новый компьютер или не сможем – это слишком примтивный взгляд на вещи. Та же Canonical, надо думать, свою армию пользователей в обиду не даст, об этом можно не беспокоиться. На самом деле, это вопрос личной свободы для каждого.

Что толку для хакера, если список корпораций с взаимной договоренностью, с их продуктами, услугами и готовыми решениями, пусть и основанными на Linux, пополнится еще одной маркой? Если вы не можете запустить на вашей системе произвольный код, а только «нотариально заверенный» и подписанный чьими-то ключами – это же, фактически, равносильно тому, что «большие дяди» будут решать за вас, какие книги вам читать и какой информацией обмениваться с друзьями! В этом случае нет особой разницы, установите вы на свою машину Linux или Windows – она все равно останется «черным ящиком». Простите, но мы не для того переходили на СПО, чтобы из нас делали такое же стадо потребителей, только другой породы!..

Так какие же, вы спросите, альтернативы? Ведь мы все зависим от аппаратных вендоров – клепать микропроцессоры в гараже еще никто не научился. Но, к счастью, не все производители оборудования придерживаются идеологии «большого бизнеса». Бизнес, как известно, бывает и малым, а компьютеры, как оказалось – открытыми и дешевыми.







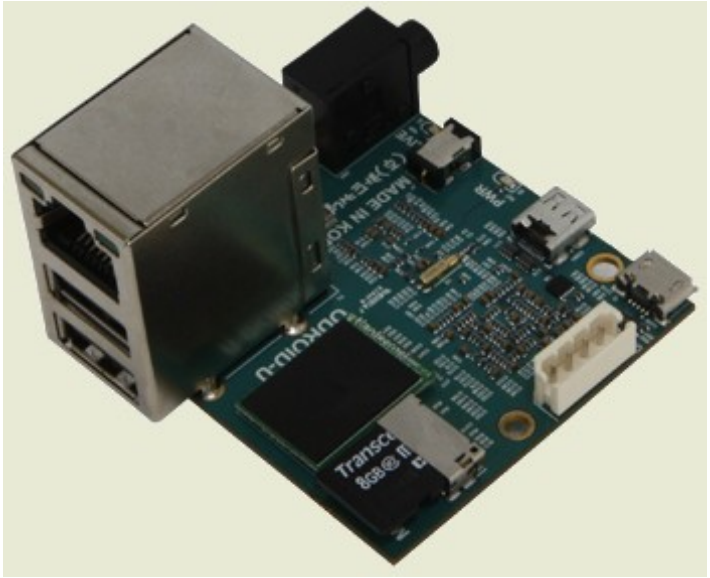

Один из таких – Raspberry Pi, идейный наследник старого доброго ZX Spectrum (и, по любопытному совпадению, тоже из Великобритании – разрабатывается в лабораториях Кембриджа). Это одноплатный компьютер размером с кредитную карту, на борту которого 512 Мб памяти, ARM-процессор с тактовой частотой 700 МГц, MMC-накопитель, HDMI, USB, Ethernet и интегрированное видеоядро с поддержкой OpenGL ES 2.0. Как среднестатистический планшет, но открытый и свободный для экспериментов - подключай монитор, клавиатуру и используй как хочешь: в качестве медиацентра, интернет-ТВ, игровой приставки или образовательной платформы. Никаких ограничений – и все это за смешную цену в $35!

Работает «Малиновый Пи» под управлением целого ассортимента свободных ОС: под ARM можно собрать практически любой «хакерский» дистрибутив, будь то Debian, Fedora, Gentoo или FreeBSD. Для Raspberry даже разработали специализированную ОС Raspbian. Мощность компьютера позволяет запускать довольно «тяжелые» программы, включая современные браузеры и офисные пакеты.

Raspberry Pi стал настоящим «открытием года» – на ARM TechCon 2011 он победил в номинации «Hardware Design», энтузиасты всего мира используют его в робототехнике, в качестве бортового компьютера для автомобилей, для создания «умных домов», сетевых маршрутизаторов, серверов, вычислительных кластеров, систем наблюдения. Школьники мечтают получить его на день рождения, в Интернете для него появляется все больше новых программ, возникают целые сообщества «распберристов»...

Другой проект, близкий к Raspberry – плата Odroid U2. При размере 48x52 мм, она укомплектована четырехъядерным процессором ARM Cortex-A9 1.7 ГГц, видеоускорителем Mali-400 с аппаратной поддержкой декодирования видео 1080p и 2 Гб памяти. Из разъемов – Ethernet, 2 порта USB 2.0, один порт microUSB и microHDMI. В качестве программной начинки для установки на платы подготовлена серия прошивок на базе Android 4.0 и 4.1. Также поддерживается установка Ubuntu. Полный набор схем распайки элементов на плате опубликован в открытом доступе. Стоимость устройства – $89.

Не правда ли, напоминает эпоху 8-разрядных домашних компьютеров? Разработанный с целью создать максимально дружественное для начинающих программистов устройство, знаменитый ZX Spectrum стал для многих первым компьютером вообще, завоевал сердца всех геймеров своего времени, стал настоящей легендой во всем мире. Возможно, настало время вернуть ценности «Спектрума» – ведь психологическая тяга к такого рода устройствам есть до сих пор. Современные хакеры не дают старым любительским компьютерам пылиться в чуланах, создают для них новое ПО, модификации, эмуляторы... Простота, дешевизна и, что называется, «ламповость» таких устройств делают их куда более привлекательными для любителя электроники, нежели все разрекламированные гламурные гаджеты, вместе взятые...



Так возможна ли «цифровая утопия» – где все устройства открыты, снабжаются подробными спецификациями, дружественны к хакерам и движению СПО? Можно ли делать бизнес, не запирая свои разработки «за семью печатями», не боясь, что их украдут конкуренты? Ведь в области программирования эта модель уже давно опробована и успешно работает – в мире, где информация не продается, украсть ее попросту невозможно.

Практика показывает, что самоорганизующиеся системы более эффективны, нежели империи и монополии – достаточно вспомнить знаменитое эссе Эрика Реймонда «Собор и Базар».

Помимо «базарной модели» разработки ПО, на пути к столь желанной хакерской утопии важную роль играют распределенные сети – P2P, I2P, TOR и др., которые никем не контролируются, не могут быть «отключены» и будут существовать, пока есть хотя бы два компьютера, соединенных между собой.

Сейчас уже начали говорить о полностью автономной всемирной беспроводной сети, независимой от системы DNS и международных шлюзов. Криптоанархисты даже создали децентрализованную платежную систему – небезызвестную Bitcoin, которая в нашей умозрительной «цифровой утопии», теоретически, может стать единой всемирной валютой...

Последние технические новинки, создаваемые в соответствии с принципами Open Hardware, подтверждают – очередная цифровая революция не за горами. Недавно стартовала разработка полностью свободного ноутбука – Novena. Используемая в нем материнская плата будет поставляться с полным набором спецификаций. Воспользовавшись опубликованными открытыми схемами любой производитель при желании может наладить производство компонентов и принять участие в их доработке и развитии. Ноутбук будет укомплектован только открытым программным обеспечением, включая все драйвера и прошивки.

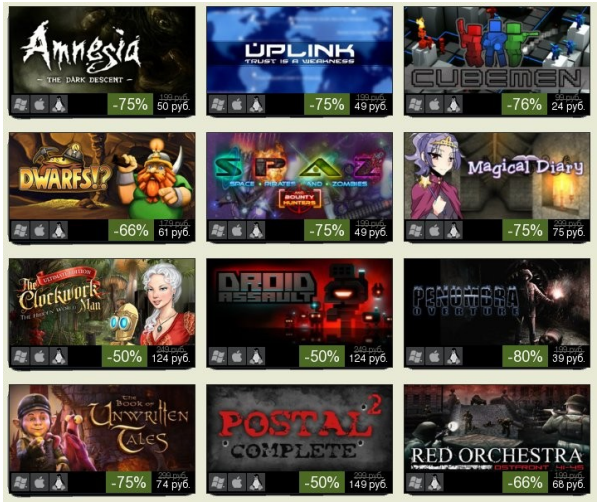
А что же игровые устройства? Консоли всегда стояли особняком и представляли собой образцовое воплощение «черного ящика» – взять хотя бы ту же Sony, которая уже почти 20 лет упрямо делает из своей PlayStation «неприступный бастион». Чего стоят только недавние страсти по поводу взлома прошивки PS3. Но, оказывается, полным ходом идет развитие открытых игровых консолей – недавно на Kickstarter начался сбор средств для производства портативной консоли GCW-Zero, которая будет работать под управлением Linux. Приставка, в первую очередь, ориентирована на запуск ретро-игр: для этого в комплект будет включена большая подборка эмуляторов Atari 2600, NES, SNES, MegaDrive, PlayStation I и Nintendo 64. Пользователю будет предоставлена возможность полностью контролировать устройство, в том числе заменять и модифицировать прошивку, а также удаленно подключаться к устройству по SSH или SFTP.

Другая открытая консоль – Pandora – уже вовсю продается. Это, в сущности, ориентированый на игры миниатюрный субноутбук – самая мощная портативная консоль на рынке, обладающая рядом уникальных возможностей, которые вам и не снились. Этот «Ящик пандоры» также использует СПО (специальную версию дистрибутива Linux Ångström) и эмулирует множество ретро-приставок: N64, PlayStation, Atari Jaguar, Amiga, GameBoy Advance, Sega Mega-CD, NeoGeo, Sega Genesis, Sega Master System, Game Gear, Commodore 64, NES, PC-engine/TurboGrafx 16 и другие. Разработчики устройства предоставляют пользователям полную свободу для экспериментов и разработки своего ПО и игр: в системе используется SDL и OpenGL. В Интернете уже функционируют целые репозитории нативных программ для «Пандоры»...

Между тем, свою консоль на базе Linux планирует выпустить сама Valve. Гэйб Ньювелл в недавнем интервью изданию Kotaku подтвердил информацию о планах компании по производству собственной игровой консоли, которая будет конкурировать с устройствами нового поколения от Sony и Microsoft. А это уже что-нибудь да значит! В связи с разочарованием в Windows 8, все больше разработчиков видят в Linux достойную кандидатуру в игровые платформы будущего.

Та же Valve подтвердила свое слово делом: совсем недавно на Linux был портирован клиент Steam. За время тестирования была проведена большая работа по оптимизации графического стека, увеличению производительности драйверов и адаптации Ubuntu как платформы для запуска игр. К инициативе Valve по превращению Linux в первоклассную игровую платформу присоединились такие компании, как Canonical, Intel, AMD и NVIDIA.

В Steam были незамедлительно опубликованы Linux-порты классических игр Valve. Вслед за Team Fortress 2, Half-Life и Counter-Strike 1.6 началось распространение Counter-Strike Source, разработанного в 2004 году. Одновременно было объявлено о проведении большой распродажи игр в Steam для Linux. Линуксовые версии игр продавались со скидкой в 50-80% – и это был уже второй шаг в рекламной кампании Valve по продвижению Linux. На первом этапе пользователям Windows-клиента Steam была показана реклама с предложением установить Ubuntu.




Кроме того, недавно стало известно, что компания Blizzard также собирается выпустить Linux-версию одного из своих бестселлеров. Скорее всего, это будет Warcraft, информация о портировании которого периодически всплывает еще с 2011 года. Если эксперимент будет успешен, в будущем для Linux будут адаптированы и другие игры компании.

Если это не показатель, тогда что? Очевидно, вот-вот будет достигнута некая «критическая масса», и состоится всеобщий переход на Linux – а это значит, что мир должен измениться. Может быть, обрисованная нами «цифровая утопия» – дело сравнительно отдаленного будущего, но «смутное время» корпоративной диктатуры, время проприетарщины и DRM явно подходит к концу. «Железо», как и софт, должно быть открытым.

# Linux: полезные команды

И снова – набор разнообразных полезных консольных команд для пользователей ОС Linux.

Групповое изменение размера изображений:

```
$ mogrify -resize 1459x1094! *.jpg
```

То же самое, но с сохранением соотношения сторон:

```
$ mogrify -resize 1459x1094 *.jpg
```

Сжать видео с камеры для веба:

```
$ ffmpeg -i INPUT.MPG -s 640x360 -b 640k -ab 192±k -vcodec
mpeg2video -acodec mp2 OUTPUT.MPG
```

Собрать последовательность кадров в единый видеоролик:

```
$ ffmpeg -i frame%04d.png -vcodec huffyuv test.avi
```

или

```
$ ffmpeg -i frame%04d.png -vcodec mjpeg -sameq test.avi
```

Смонтировать/размонтировать флешку с файловой системой NTFS (предварительно узнав ее обозначение при помощи dmesg):

```
$ dmesg
$ sudo mount -t ntfs-3g /dev/sdc1 /media/NTFSDISK -o force
$ sudo umount /media/NTFSDISK
```

Узнать информацию о CD/DVD-приводе:

```
$ dmesg | egrep -i --color 'cdrom|dvd|cd/rw|writer'
```

...и о том, какие диски он умеет читать и записывать:

```
$ more /proc/sys/dev/cdrom/info
```

Сконверировать в PDF сразу множество документов при помощи LibreOffice:

```
$ libreoffice3.4 --headless -convert-to
pdf:writer_pdf_Export -outdir ~/outdir
~/inputdir/*.odt
```

Shell-скрипт, компилирующий все исходники на C в текущем каталоге без Make-файла (сборка инкрементальная – т. е., каждый раз компилируются только те исходники, которые были изменены с момента предыдущего цикла сборки):

```
#!/bin/sh
CC="gcc"
CFLAGS="-I../include"
LFLAGS="-shared"
TARGET="libjewelscript.so"
for file in ./*.c; do
    local src=$(basename $file)
    local obj=${file%%.*}.o
    if test $src -nt $obj; then
        echo "CC $src > $obj";
        $CC $CFLAGS -c -o $obj $src || { exit 1; }
    fi
done
objects=$(ls -dm *.o | tr ',' ' ')
echo "CC *.o > $TARGET";
$CC $LFLAGS -o $TARGET $objects || { exit 1; }
```

Shell-скрипт, разыменовывающий все символьные ссылки в текущем каталоге:

```
#!/bin/sh
for file in ./*.h; do
    if [ -h "$file" ]; then
        echo "unlinking $file...";
        cp --remove-destination `readlink $file` $file
    fi
done
```

Узнать максимальное количество аргументов, которое можно передать программе:

```
$ getconf ARG_MAX
```

Работаем с Git и репозиторием на GitHub:

1. Создаем ключ аутентификации SSH:

```
$ ssh-keygen -t rsa -C "your_email@youremail.com"
```

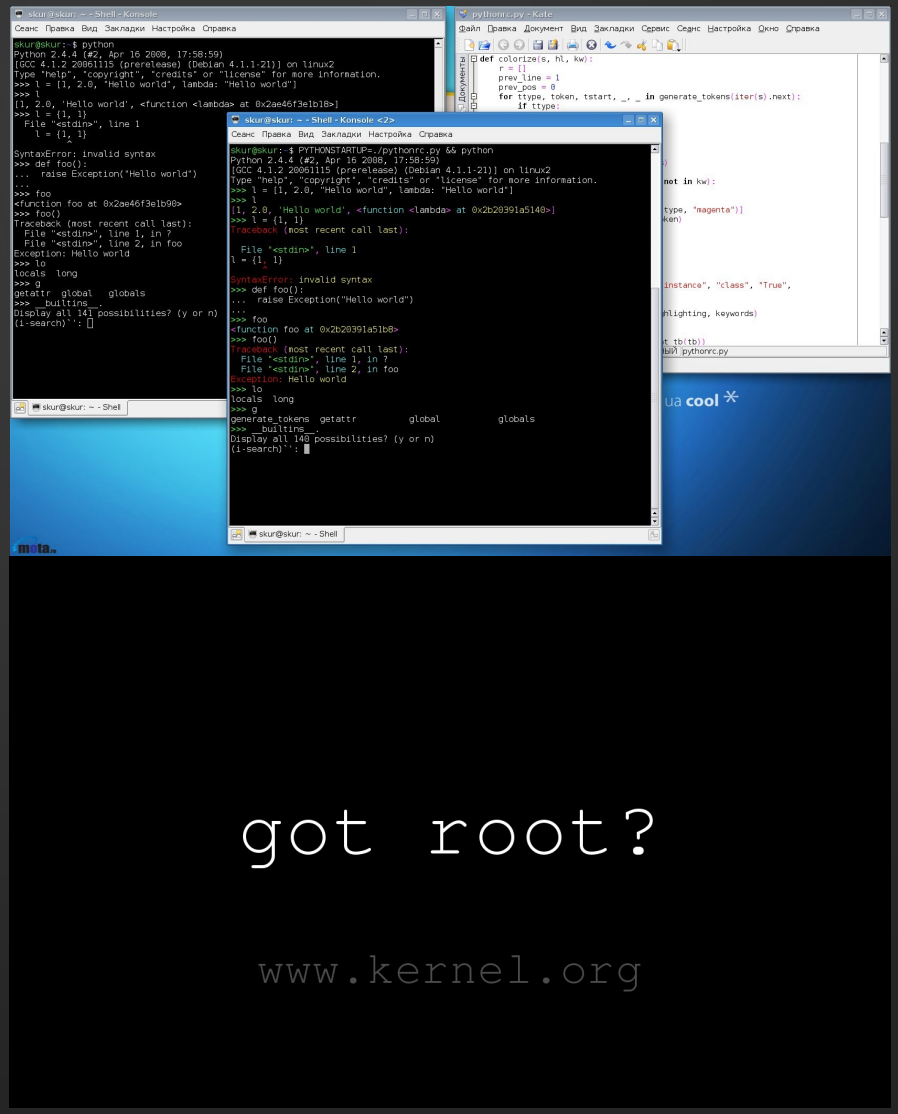
2. Добавляем содержимое файла ~/.ssh/id_rsa.pub в список ключей на вашем аккаунте на GitHub (Settings > SSH Keys > Add SSH key) .

3. Загружаем локальную рабочую копию репозитория по зашифрованному соединению:

```
$ git clone git@github.com:username/repository.git
```

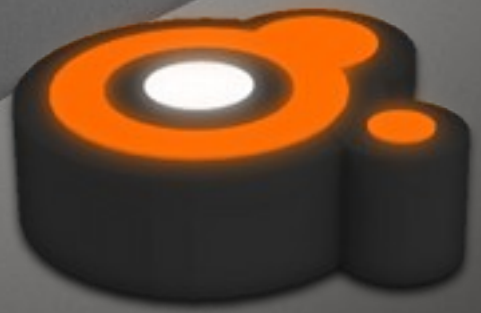
4. Создаем/удаляем файлы, вносим изменения, а затем:

```
$ git rm old.file
$ git add new.file
$ git commit -m "Made some changes"
$ git push origin master
```



# 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