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

FPS №45

FPS *– бесплатный, свободно распространяемый электронный журнал, посвященный компьютерному творчеству.*

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 редакторам, игровые обзоры, а также статьи по игровой теории, геймдизайну и современным мультимедийным формам искусства.

Журнал издается с января 2008 г.
Периодичность выхода: раз в два месяца.

Главный редактор: **Тимур Гафаров**
Дизайн и верстка: **Наталия Чумакова**
Обложка: **Тимур Гафаров**

Наш сайт: **http://fps-magazine.cf**

По вопросам сотрудничества обращайтесь по адресу:

gecko0307@gmail.com

Blender Новости

Начнем, как всегда, с анонса новой версии Blender. Недавно вышло обновление 2.78a – это, в основном, исправляющий релиз: всего было исправлено 69 багов.

Скачать Blender 2.78a для всех платформ можно здесь: **http://blender.org/download**

На прошедшей в конце октября Blender Conference 2016 были объявлены лауреаты ежегодной премии **Suzanne Award**, вручаемой анимационным проектам, которые снимаются при помощи Blender. В этом году было всего три номинации – «Лучший анимационный фильм», «Лучший дизайн» и «Лучший короткометражный фильм». Примечательно, что в финал попали работы, о которых мы ранее писали на страницах журнала.

В номинации «Лучший анимационный фильм» победил **«Katsu Cats»** – NPR-анимация о котиках и восточных единоборствах. Режиссер – Диллон Гу, США.

В номинации «Лучший дизайн» победа досталась **«Sub-Surface»**, потрясающему оффлайн-демо от финской группы 1/2-bit Cheese. Тема работы – подводный мир.

Ну а приз «Лучший короткометражный фильм» взял **«The Collinwood Fire»**, 6-минутный фильм, посвященный трагедии 1908 года, когда 172 ребенка погибли при пожаре в школе Коллинвуда (Огайо). Режиссер – Дэниел Хоутон, США.

Этой осенью состоялась большая премьера: фильм **«Ozzy»** от канадской студии Tangent Animation, вышедший в 14 октября (в России 3 ноября под названием «Большой собачий побег»), стал, по всей видимости, первым коммерческим анимационным фильмом, полностью снятым при помощи Blender!

Жизнь домашнего пса по кличке Оззи переворачивается, когда, отправляясь в отпуск, хозяева отдают его в элитный отель для собак – но оказывается, что на самом деле это настоящая собачья тюрьма. Чтобы прорваться через систему охраны и злющих сторожевых псов, Оззи вместе с такими же бедолагами, как он, должен будет спланировать самый хитроумный и дерзкий побег в истории...

https://www.youtube.com/watch?v=i3EwkrG_4ZE

Появление подобного проекта можно считать поворотным событием в мире Blender – фактически, это означает, что профессиональный анимационный продакшн может быть полностью переведен на СПО. Кстати, Джефф Белл и Кен Зорниак из Tangent Animation выступили на Blender Conference, рассказав о том, как Blender был использован при создании «Ozzy»:

https://www.youtube.com/watch?v=CqSEtTF8hPE

Авторы серии анимационных короткометражек «Caminandes» выпустили 360-градусную версию отрывка из **«Caminandes 3: Llamigos»** – посмотреть ее можно прямо на YouTube, который с недавних пор поддерживает VR-видео:

https://www.youtube.com/watch?v=uvy--ElpfF8

Из недавних анимационных премьер отметим **«Advanture of Tanu»** – короткометражку о мальчике по имени Тану от индийской студии Weybec, известной по фильму «Monkaa».

https://www.youtube.com/watch?v=K-flWuvRVhA

Очень интересный 4-минутный фильм сняли немецкие аниматоры: **«Totentanz»** – это стильный готический макабр под знаменитую «Пляску смерти» Камиля Сен-Санса, отрендеренный в NPR-графике.

https://vimeo.com/184574722

Кроме того, вышли тизеры сразу нескольких анимационных проектов, находящихся в производстве. Это, в первую очередь, **«Elulu»** от чилийской студии Belofilms – полнометражный фильм с очень необычным творческим дизайном, премьера которого назначена на 2018 год.

https://www.youtube.com/watch?v=RwPS3TSEQgE

«Hope» от венесуэльской студии Uplop – это красивая короткометражка о приключениях маленького осьминогоподобного инопланетянина.

https://www.youtube.com/watch?v=rmHDa4m8Gtc

«Natura» от мексиканской студии Chaman Animacion Studio – не менее интересный проект, фильм о растительных существах наподобие фей.

https://vimeo.com/190655649

Не обойдем стороной и VFX-проекты. Режиссер Михаил Хижняков и белгородская студия Nexus снимают фильм **«Энтропия: завет цивилизации»** – судя по тизеру, работа очень серьезная, спецэффекты ничуть не хуже голливудских!

https://www.youtube.com/watch?v=ZJjlRD3LshY

От киноанонсов – к игровым. Полным ходом идет разработка **RoonSehv: Experimentaliis**, сиквела игры RoonSehv: NeTerra, о которой мы писали в прошлом номере журнала. Напомним, это один из лучших фанатских игровых проектов по мотивам культовой серии Myst – трехмерный квест, где игрок изучает фэнтезийный мир и разгадывает головоломки, погружаясь в его историю и мифологию. Первая часть стала настоящим хитом в сообществе поклонников Myst, а вторая обещает быть еще интереснее. Уже доступна ранняя тестовая сборка Experimentaliis с одной полностью проработанной головоломкой.

http://www.indiedb.com/games/roonsehv-experimentaliis

Вышел первый превью-релиз **Armory3D** – активно разрабатывающегося 3D-движка для Blender, технологически превосходящего BGE. Разработчики предупреждают, что данная версия движка может содержать баги и предназначена исключительно для тестирования.

На первых порах сборки Armory будут распрстраняться на платной основе (€ 50.00) в целях финансирования проекта – в будущем планируется сделать движок бесплатным. Все единожды купившие Armory получают будущие обновления бесплатно. Исходный код движка полностью открыт под лицензией MPL и доступен на GitHub.

https://armory.itch.io/armory3d

Также были опубликованы браузерные демки, демонстрирующие работу Armory на примере простой сцены с анимацией персонажа. Как подчеркивают авторы, демки предназначены для оценки производительности движка – есть несколько вариантов одной и той же сцены с различными методами рендеринга (прямой, отложенный и гибридный). Для лучшей производительности рекомендуется запускать их в Chrome.

http://armory3d.org/download.html

Blend4Web, открытый фреймворк для создания браузерных 3D-приложений в Blender, обновился до версии 16.10. Из главных нововведений можно отметить систему поиска пути на основе классического алгоритма A*, улучшенные логические узлы, внесено несколько важных оптимизаций производительности и потребления видеопамяти.

https://www.blend4web.com

Кстати, разработчики Blend4Web не так давно запустили новую серию уроков по движку для начинающих от Андря Прахова, разработчика игр и автора нескольких книг по Blender.

https://www.blend4web.com/en/community/article/221

Увидела свет 21 версия знаменитого рендер-движка **RenderMan** от студии Pixar. Релиз примечателен включением эксклюзивных шейдеров, IES-профилей и световых фильтров, применяющихся в Pixar на производстве анимационных фильмов – например, нового убер-шейдера PxrSurface, который активно использовался при создании «В поисках Дори». Также улучшен встроенный инструмент подавления шума и добавлена поддержка рендеринга под устройства виртуальной реальности.

Напомним, с 2015 года RenderMan доступен бесплатно для использования в некоммерческих целях. Бесплатная версия полностью функциональна, не имеет временных ограничений и не накладывает водяные знаки.

Движок интегрируется в Maya, Katana, Houdini и Blender. Стоимость коммерческой лицензии – $495.

https://renderman.pixar.com/view/non-commercial-renderman

В прошлом номере мы уже писали о том, что компания Insydium анонсировала плагин, позволяющий использовать рендер-движок Cycles в Cinema4D, одном из самых популярных коммерческих 3D-пакетов. И недавно состоялся долгожданный релиз **Cycles 4D**. Стоимость одной лицензии – от £60.

http://insydium.uk/cycles4d

В рамках проекта JavaBlend развивается инструментарий для чтения/записи файлов формата *.blend на языке Java.

http://homac.cakelab.org/projects/JavaBlend/spec.html

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!

Памятка читателю

В Интернете часто можно встретить вопросы о том, где скачать старые номера нашего журнала. Отвечаем. Архив всех номеров «FPS» (с 2008 по 2016 гг.) можно найти сразу на нескольких сервисах:

На файловом хостинге **DropBox**:

https://www.dropbox.com/sh/b7lgxxh6nxbxre9/uVvzqU8_j-

В **Документах Google** (для скачивания файлов нужен аккаунт Google):

https://docs.google.com/folderview?id=0B1BIzRb1uMv-bnpHNDhwZTI4eHc

В электронном издательстве **Issuu.com**:

http://issuu.com/tgafaroff/docs

Для тех, кто предпочитает скачивать с торрентов – некоторые номера журнала есть на **РуТрекере**:

http://rutracker.org/forum/viewtopic.php?t=4403193

Blender для начинающих

Часть 3. Материалы и текстуры

Данный цикл статей основан на материалах, которые были подготовлены для книги «Blender. Настольная книга». Она создается авторами нашего журнала уже несколько лет, но до сих пор находится в состоянии, далеком от готовности. Поэтому мы решили публиковать ее частями, чтобы информация не пропадала даром.

Материал – это набор параметров, определяющих характер поверхности объекта (а в некоторых случаях – и его объема). Они включают цвет (если быть точным, несколько компонентов цвета для разных составляющих освещенности), текстуру, параметры прозрачности, отражения и преломления и многие другие. В общем случае, эти параметры определяют закон, по которому свет должен отражаться от поверхности объекта.

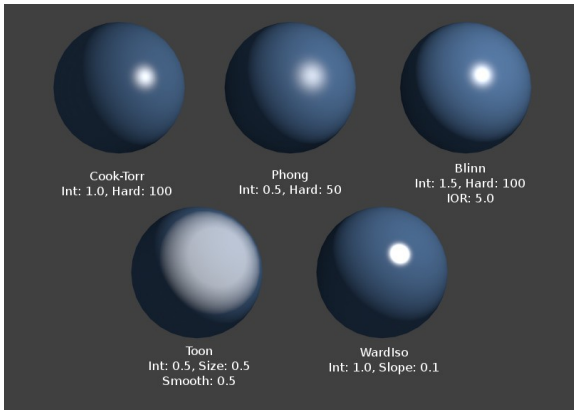
Чтобы создавать на компьютере действительно качественные и реалистичные изображения, необходимо понять, как моделируется освещенность виртуальных объектов. В реальном мире свет состоит из мельчайших частиц, называемых фотонами. Фотон имеет свойства, присущие как волнам, так и элементарным частицам. Фотонов настолько много, что обычно можно пренебречь тем, что световое излучение состоит из отдельных частиц (это важно только в квантовой механике), и рассматривать его как непрерывный поток энергии. В этом случае к свету можно применить статистические законы и смоделировать его на компьютере.

Поток энергии отрывается от источника света и распространяется в пространстве, пока не столкнется с каким-либо объектом. При этом одна часть энергии поглощается веществом объекта, а другая – отражается (поэтому мы видим объекты как темные или светлые). Отраженный поток фотонов меняет свою длину волны в зависимости от свойств вещества, в результате чего мы воспринимаем у разных объектов различные цвета. Также некоторая часть фотонов проходит сквозь материал, и объект выглядит прозрачным. Проходя через вещество, световой поток может преломляться (менять направление) и рассеиваться.

Но если объекты отражают свет, почему же далеко не любая поверхность может служить зеркалом? Все дело в том, что идеальным зеркалом является только идеально гладкая поверхность, в то время как обычные поверхности в той или иной степени шероховаты – то есть, состоят из множества микроскопических граней-отражателей. Свет, попадая на поверхность объекта, многократно отражается от этих микрограней и рассеивается в пространстве, в результате чего мы не видим точных отражений, а только сплошной цвет.

В компьютерных моделях, описывающих материал объекта, этот феномен сведен к простым математическим формулам, по которым можно вычислить степень рассеянной (или, как обычно говорят, диффузной) освещенности в любой точке заданной поверхности.

Самая простая такая формула – закон Ламберта (Lambert), который определяет интенсивность диффузной освещенности в точке как косинус угла между направлением света и нормалью к поверхности в этой точке.



Модель Ламберта хорошо подходит только для сравнительно гладких поверхностей. Для моделирования шероховатой, бархатистой или запыленной поверхности часто используют диффузную модель Орена-Найара (Oren-Nayar), которая основана на предположении, что поверхность состоит из множества бесконечно малых микрограней, освещение каждой из которых описывается моделью Ламберта. Модель Орена-Найара имеет параметр для контроля шероховатости поверхности (Roughness). Этот параметр определяет, сколько света отразится назад в направлении источника света.

В Blender чаще всего используются именно эти две диффузные модели. Помимо диффузной, используется также бликовая составляющая освещенности. Бликовая составляющая (specular term) – это количество света, зеркально отраженного поверхностью. Блик – это прямое отражение источника света на поверхности объекта.

Если учесть, что в компьютерной графике используются идеализированные объекты, возникает закономерный вопрос: почему точечный источник света, не имеющий объема и невидимый сам по себе, отражается как относительно крупный размытый световой блик? Этот феномен также объясняется наличием микрограней: они имеют собственные вектора нормалей, отклонение которых от основной нормали поверхности меняют интенсивность зеркально отраженного света.

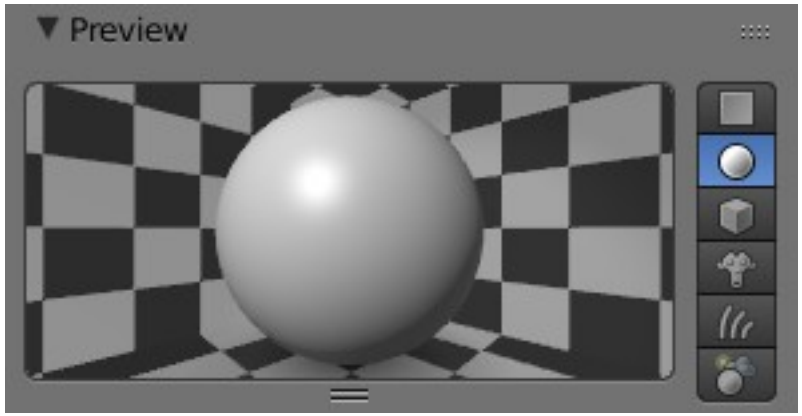
Цвет блика может не совпадать с цветом материала. Это справедливо для некоторых многослойных материалов – например, пластик представляет собой «слоеный пирог» из пигмента и прозрачного полимера: блик дают прозрачные слои, а диффузное рассеивание – цветные. Однородные материалы такого эффекта не производят, и блики на них имеют тот же цвет, что и сами материалы. Яркий пример – металл.

Для сравнительно гладких материалов (таких, как пластик или металл) обычно пользуются эмпирической моделью Фонга (Phong). Она не соответствует точному физическому описанию отражения света, но в большинстве случаев позволяет достичь приемлемых реалистичных результатов. Формула Фонга основана на простом наблюдении: блестящие поверхности дают маленькие и резкие блики, в то время как матовые – большие и размытые. Более согласованная с физикой модель, которая поддерживается в Blender – модель Кука-Торренса (Cook-Torrance). Она основана на допущении, что поверхность состоит из микрограней, каждая из которых является идеальным зеркалом.

Зная, где и как правильно применять эти модели, можно моделировать объекты, по внешнему виду максимально приближенные к реальным.

Так, для матовых поверхностей вроде камня, бетона или бумаги лучше всего подходит модель Ламберта. Блики на пластике, фарфоре, металле, матовом стекле имитируются моделью Фонга. Бархат, вельвет, ковры и некоторые другие виды тканей лучше всего воссоздаются моделями Орена-Найара и Кука-Торренса.

В Blender материал объекту можно добавить в редакторе свойств, который по умолчанию находится в правой части окна программы. Переключите панель со значками на **Material**, добавьте объекту новый материал (если его нет) при помощи кнопки **New**.



Рассмотрим основные параметры материала.

Diffuse. Цвет и модель диффузной (рассеянной) компоненты освещенности. Вы можете указать цвет, нажав по нему левой кнопкой мыши – появится RGB-палитра с возможностью точного подбора каналов цвета.

Specular. Цвет и модель бликовой (зеркальной) компоненты освещенности. Форма блика зависит от выбранной модели и специфичных для нее параметров – это может быть как маленькая резкая точка, так и большое размытое пятно.

Transparency. Если поставить галочку напротив этого параметра, можно сделать объект прозрачным. Степень прозрачности контролируется параметром Alpha. Существует несколько типов прозрачности, в том числе с поддержкой преломления световых лучей, как и в реальных материалах – мы еще рассмотрим их подробнее в следующей главе.

Mirror. Если поставить галочку напротив этого параметра, поверхность объекта будет зеркально отражать окружающие предметы. Степень отражаемости контролируется параметром Reflectivity.

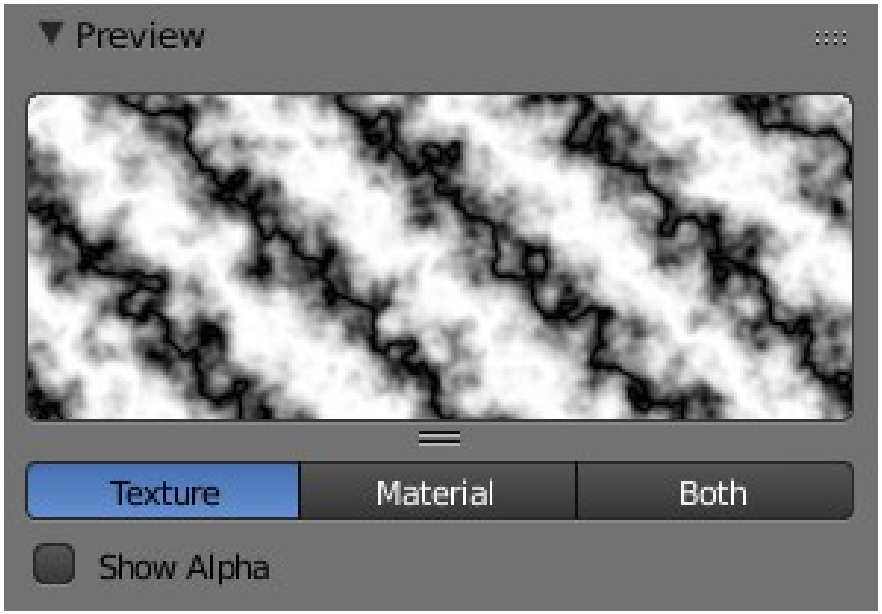
Текстура

Текстура – это изображение, которое определяет цвет (или какую-либо другую характеристику) материала в каждой точке поверхности. Говорят, что текстура накладывается на поверхность: иными словами, создается особая «карта», сопоставляющая каждую точку на поверхности модели с точкой на плоском изображении. Эту карту называют UV-разверткой или просто разверткой. В Blender вы можете создавать развертки вручную, но в случае использования базовых примитивов за вас это автоматически сделает сама программа.

Текстуру можно добавить в материал, переключившись на **Texture** на панели со значками свойств. У материала может быть более чем одна текстура – это часто бывает необходимо для создания сложных эффектов. Добавьте первую текстуру, нажав кнопку **New**. По умолчанию создается процедурная (то есть, автоматически сгенерированная программой) текстура типа «облака» (Clouds). Есть также несколько других типов процедурных текстур: «дерево», «мрамор», диаграмма Вороного, шум и т.д.

Естественно, Blender позволяет в качестве текстуры выбрать произвольное растровое изображение: для этого переключите тип текстуры (**Type**) на **Image or Movie** («Изображение или фильм») и на вкладке **Image** нажмите **Open**.

Если ваш объект – куб (или параллелепипед), то можно указать тип автоматической развертки на вкладке **Mapping**: **Projection -> Cube**.



На вкладке **Influence** («Влияние») можно управлять характеристиками материала, на которые влияет данная текстура. По умолчанию стоит цвет (**Color**) – текстура влияет на диффузный цвет материала. Она также может влиять на бликовую составляющую, прозрачность, отражаемость и геометрию поверхности.

Продолжение следует...

Уважаемые читатели!

Наш журнал регулярно выходит на протяжении почти 9 лет – с февраля 2008 года. Все эти годы он оставался бесплатным изданием, предлагая публике эксклюзивный контент с минимумом рекламы. Мы всегда работали на совесть – не ради денег, а на благо наших читателей. «FPS» был и остается проектом энтузиастов и полностью независимым изданием – мы не защищаем интересы корпораций или политиков, мы пишем о том, что считаем нужным и важным. Мы стоим за свободу слова и творчества, за обмен информацией и знаниями: все материалы журнала можно беспрепятственно копировать, распространять и использовать в любых производных работах.

И мы надеемся, что так будет продолжаться и дальше. Но на создание новых номеров у авторов уходит достаточно много сил и времени, которые никак материально не компенсируются. Поэтому, если вам нравится журнал, и вы хотели бы, чтобы он жил, развивался, становился больше и качественнее, просим поддержать его электронной валютой – при помощи WebMoney, PayPal или Яндекс.Денег, любой суммой на ваше усмотрение. Для нас важен любой, даже маленький вклад!

Наш WMR-кошелек: **R120156543694**

Номер кошелька Яндекс.Денег: **410012052560079**

Адрес PayPal: **gecko0307@gmail.com**

Заранее благодарны!

LevelBuddy. Быстрый левелдизайн в Blender

Blender, будучи мощным медиакомбайном, сегодня особенно активно привлекает внимание разработчиков игр. Blender весьма популярен среди инди-разработчиков и небольших студий. Он обычно используется для моделирования игровых объектов, создания UV-разверток, ретопологии, рисования текстур и анимации.

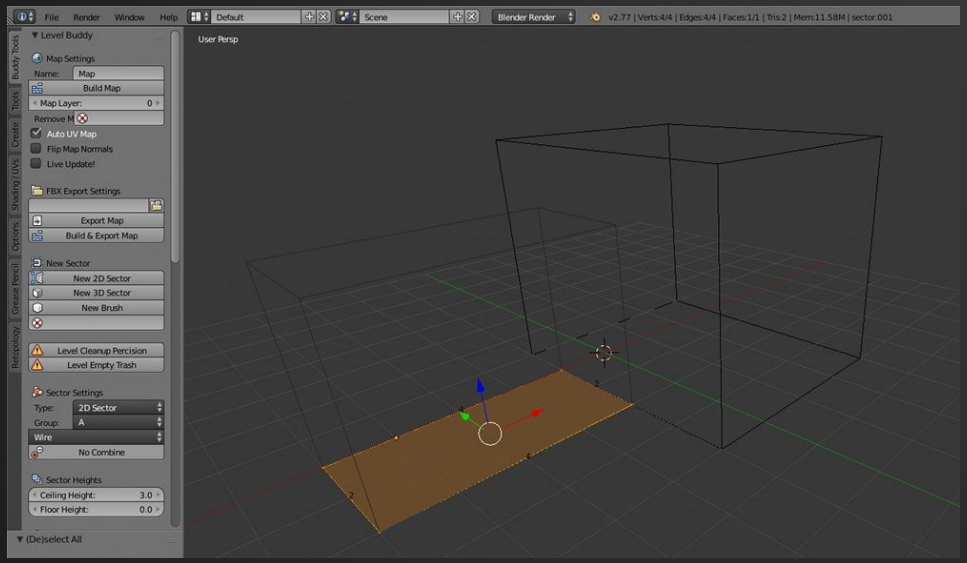
Однако для создания игровых уровней большинство людей до сих пор используют специализированные редакторы – Blender в этой области осваивается не так активно, как хотелось бы. Оно, конечно, понятно – игровые движки обычно используют собственные форматы для хранения карт, и не всегда логика движков и внутреннее представление данных в них могут быть соотнесены с логикой Blender. Но если вы решаете типовую задачу – например, моделируете карту для коридорного шутера или dungeon crawler'a – то Blender вполне удовлетворит ваши нужды. Вам только понадобится установить один маленький, но очень мощный аддон.

Речь идет о **LevelBuddy** от Мэтта Лукаса, автора небезызвестной игры Watch out for Snakes. Это дополнение реализует в Blender полноценный CSG-пайплайн, очень похожий на то, как моделировались карты для Quake и Unreal.

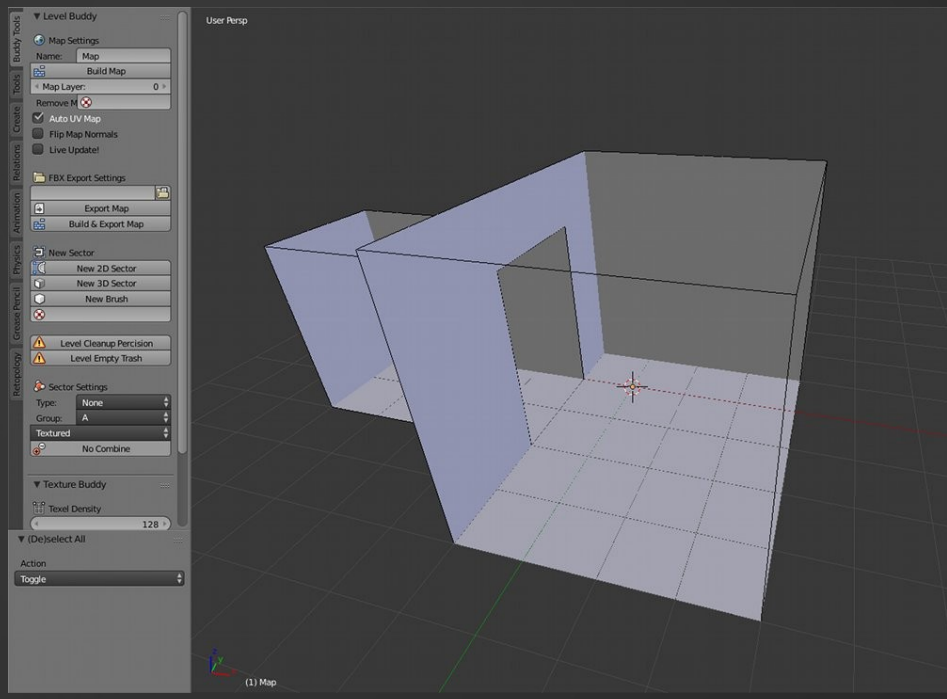
Выглядело это следующим образом. Левелдизайнер составляет уровень из отдельных блоков (так называемых кистей), которые комбинируются булевыми операциями.

К примеру, кисти, представляющие интерьер (комнаты и коридоры) комбинируются друг с другом оператором «ИЛИ» – если состыковать комнату и коридор, в стене автоматически появится проем. При этом CSG-движок автоматически вычисляет текстурные координаты для всех поверхностей. Такой подход позволяет очень быстро моделировать замкнутые помещения. И, хотя впоследствии от CSG-редакторов начали отказываться в пользу обычного полигонального моделирования, даже сегодня этот старый подход во многих случаях оправдывает себя.

Все инструменты аддона расположены на панели инструментов, во вкладке **Buddy Tools**. В LevelBuddy моделирование комнат осуществляется путем редактирования контура полов. Кисть, которая отвечает за комнаты, называется **2D Sector**. Нажмите кнопку **New 2D Sector**, в позиции 3D-курсора появится бокс. Переключитесь в режим редактирования и выставьте нужную вам площадь пола. Высоту пола и потолка можно отрегулировать в полях **Ceiling Height** и **Floor Height**.



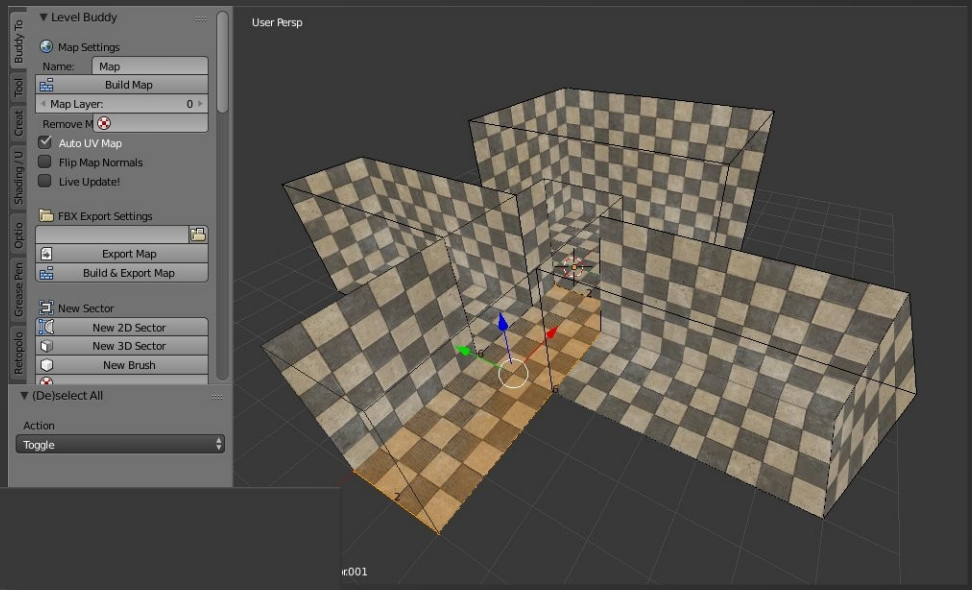
Создайте парочку комнат или коридоров и нажмите **Build Map** – будет сгенерирован меш уровня. После этого можно продолжить работу над уровнем, добавлять новые комнаты и модифицировать старые – LevelBuddy будет обновлять меш при нажатии Build Map. При первом создании меша вы, скорее всего, увидите сообщение об ошибке – не обращайте внимания.

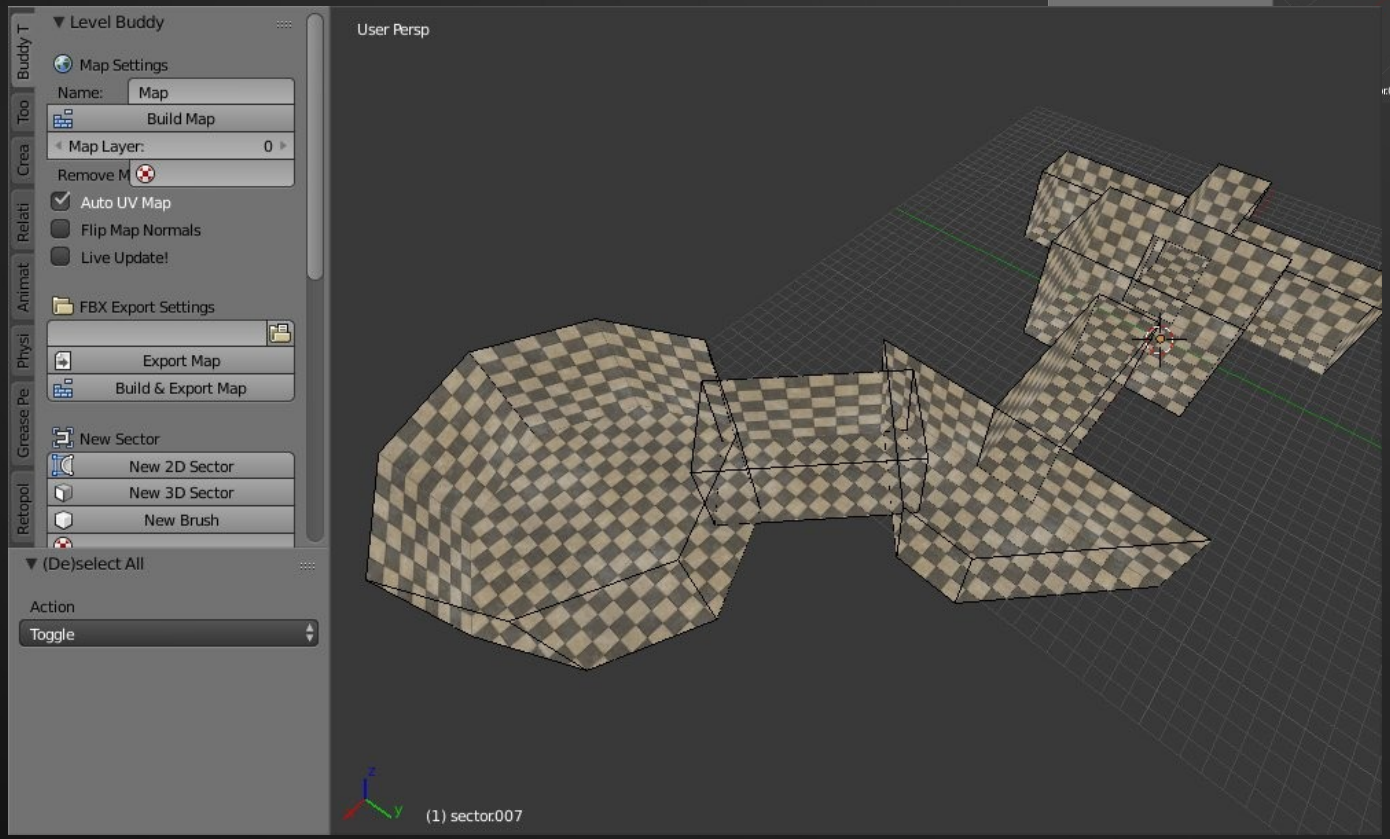


Чтобы увидеть, как на поверхности ложится текстура, сделайте следующее. Добавьте текстуру материалу **Material**, который автоматически создается аддоном. Скорее всего, в нем уже будет созданный текстурный слой – просто выберите тип **Image or Movie** и загрузите текстуру. На данном этапе у всех поверхностей будет одна и та же текстура, но для тестовых целей это сойдет – расставить более разнообразные текстуры можно потом, после получения итогового меша.

Теперь выберите комнату, зайдите в режим редактирования и выделите все вершины. В редакторе UV-разверток выберите вашу текстуру в меню изображения.

Нажмите еще раз Build Map – текстура будет передана мешу, и вы увидите ее во вьюпорте (если, конечно, переключите режим отображения на **Texture**). Имеет смысл также включить отсечение задних граней (**Backface Culling**) и отключить освещенность (**Shadeless**) в настройках **Shading**. Не очень удобно повторять это действие для всех комнат, поэтому рекомендую не создавать их через интерфейс LevelBuddy, а дублировать (**Shift+D**) – это совершенно безопасно, аддон все поймет.





Кстати, комнаты не обязаны быть прямоугольными – вы можете создать любой контур пола, какой вам захочется. Можете поворачивать сектор, подразбивать ребра и т.д.

Колонны и другие неинтерьерные объекты (которые здесь называются «кисти») добавляются кнопкой **New Brush**.

Скачать LevelBuddy можно по следующей ссылке:
https://matt-lucas.itch.io/level-buddy

Обзор дополнений Blender

Выпуск 23

Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. В этом выпуске мы представляем дополнения, связанные с композитингом и постобработкой. Если вы разрабатываете собственное дополнение или просто нашли в Интернете чей-то интересный проект, будем очень рады, если вы сообщите нам об этом и поделитесь ссылкой. Пишите на ***gecko0307@gmail.com****.*

Easy Model Compositing

Аддон автоматически рендерит сцену в несколько проходов – Object, Shadow, Reflection и AO, сразу же сохраняя изображения на диск. Очень полезное средство для последующего композитинга в сторонних программах. Дополнение платное, цена – $9.94. Покупая аддон на CGCookie, вы получаете также бесплатно PBR-материал и модель космического корабля от того же автора.

Разработчик: Blackhart Films
https://cgcookiemarkets.com/all-products/easy-model-compositing-addon

Enrich

Очень любопытное дополнение, реализующее в Blender визуальный композитинг – панель с эффектами для рендеров, которые можно переключать буквально в один клик, без необходимости заходить в редактор узлов. Аддон платный, цена – $12.90.

Разработчик: Akash Hamirwasia
https://gumroad.com/l/enrichbs

Photo Editing Nodes

Аддон от автора Enrich, добавляющий несколько дополнительных узлов композитинга с эффектами постобработки: Film Grain, Bloom, Dirty Lens, Collage, Focus, Photo Frame. Цена – от $5.00.

Разработчик: Akash Hamirwasia
https://gumroad.com/l/ZvIln

Shutter

Коллекция из 22 узлов постобработки: цветокоррекция, виньетирование, Bloom/Glow, подавление шума и множество других фильтров и эффектов. Дополнение платное, цена – $20.00. Один доллар от каждой продажи жертвуется Blender Foundation.

Разработчик: Johnson Martin
https://cgcookiemarkets.com/all-products/shutter-2d-toolset-for-blender

Blender-2-G'MIC

Для Blender появилась интеграция G'MIC, популярного фреймворка обработки изображений, включающего огромную коллекцию качественно реализованных фильтров и эффектов. Благодаря скрипту Blender-2-G'MIC, эти фильтры можно применять во встроенном видеоредакторе Blender (Video Sequence Editor).

Разработчик: Starfall Robles
https://github.com/Starfall-Robles/Blender-2-G-MIC

ColorMaster

Еще один аддон, реализующий автоматический постпроцессинг – в данном случае акцент делается на цветокоррекцию. Дополнение платное, цена – $5.00.

Разработчик: John Roper
https://cgcookiemarkets.com/all-products/colormaster

Бесплатные ресурсы для Blender и не только

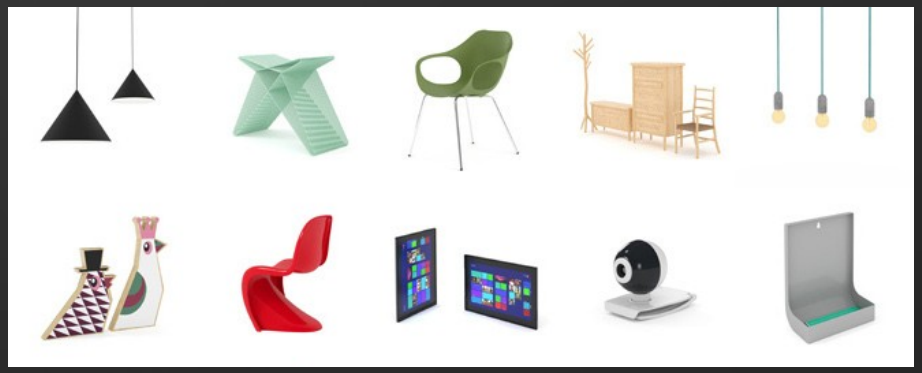
Предлагаем вашему вниманию несколько сайтов, предлагающих бесплатно скачать 3D-модели, текстуры, HDR-карты и т.д.

Сайт **http://www.thatimst3r.com**, посвященный Blender, поделился огромным паком из 98 моделей растений, деревьев, кустов, плодов и т.д. Все модели низкополигональны и пригодны для использования в движке Blender Game Engine.

http://www.thatimst3r.com/download-pk

Блог Addicted to CG поделился бесплатным паком из четырех моделей кустов для Blender. Распространяются по лицензии CC-0 (аналог Public Domain) – можно использовать их в любых целях, в том числе коммерческих.

http://addictedtocg.com/2015/12/27/bush-pack-giveaway



Сайт Chocofur выложил в открытый доступ 25 моделей для Blender (мебель, предметы интерьера, бытовая техника, электроника и т.д.) под лицензией CC-0. Она же распространяется на все текстуры и шейдеры. Для скачивания необходима регистрация.

http://store.chocofur.com/all_free

Сайт PlanetBlender предлагает бесплатно скачать ресурсы для Blender – качественные текстуры камня и коллекцию реалистичных камней с процедурным материалом:

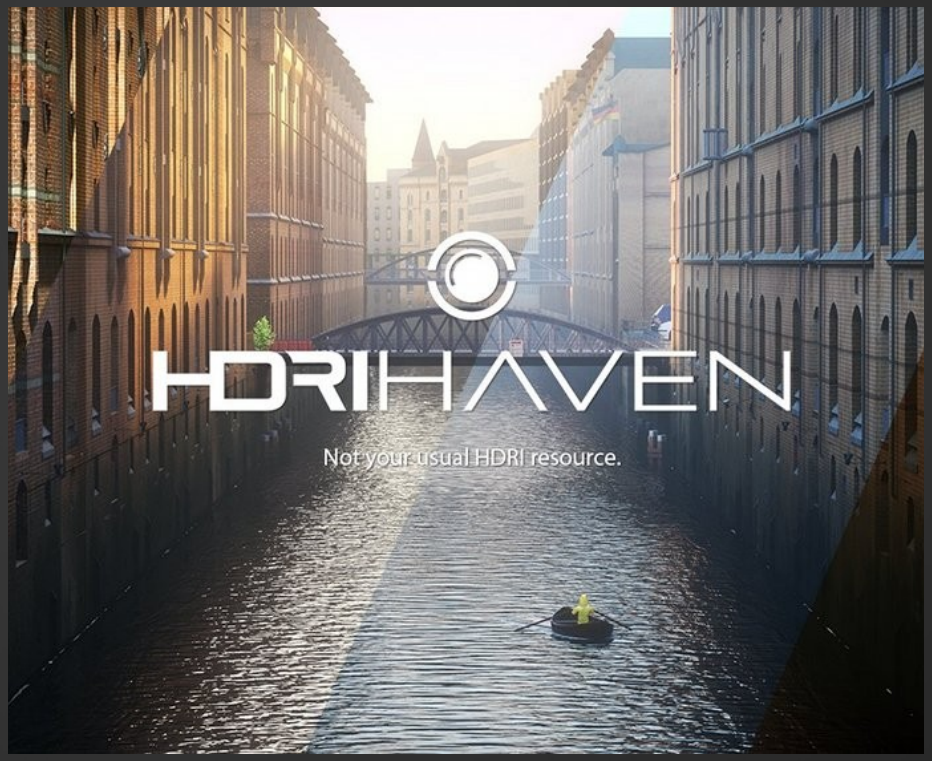
http://planetblender.com/downloads
http://store.chocofur.com/all_free

Скачивайте качественные текстуры каменной кладки и растительности (с картами нормалей, AO, бликов и смещения) от Blender-художника Eisklotz. На сайте есть также парочка моделей и HDR-карта.

http://eisklotz.com/downloads

Сайт DepthFields предлагает бесплатный доступ к десяткам качественных карт высот – в том числе, реально существующих ландшафтов. Загружать карты на сайт может любой желающий.

http://www.depthfields.com



Сайт HDRI Heaven предлагает большой выбор HDRI-карт окружения (то есть, снятых с разной экспозицией и сведенных в формат с широким динамическим диапазоном). Их можно использовать в Blender для рендеринга реалистичного освещения. Особенность данной коллекции – экстраординарно высокое разрешение (16384 x 8192) и диапазон экспозиции в 24 EV.

Загрузка платная, но к каждой карте прилагается также бесплатный вариант с более низким разрешением (1024 x 512).

https://hdrihaven.com

2D-графика: новости

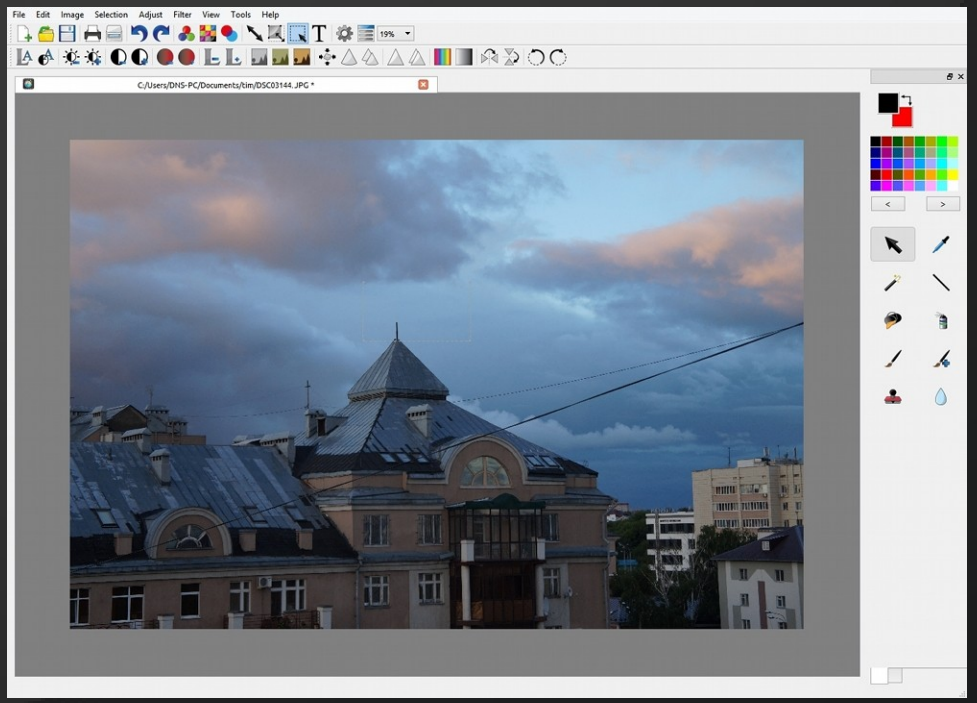
PhotoFiltre LX 1.1

Недавно состоялось обновление проекта PhotoFiltre LX, о котором мы писали в одном из прошлых номеров «FPS» – вышла версия 1.1 под кодовым названием «Raw Style».

В этом релизе появились новые опции в настройках, реализована поддержка RAW, добавлено автоматическое кадрирование, улучшен зум и повышена производительность под Linux.

Напомним, PhotoFiltre LX – это свободный кроссплатформенный клон PhotoFiltre, популярного графического редактора под Windows. Программа предназначена для простейших задач по обработке изображений, как-то: изменение размеров, поворот, кадрирование, автоматическая цветокоррекция и наложение творческих фильтров.

В данный момент PhotoFiltre LX еще не является полной заменой PhotoFiltre, но достаточно быстро приближается к этой цели. В качестве графического тулкита проект использует Qt. Программа работает под управлением Windows, Linux и OSX.



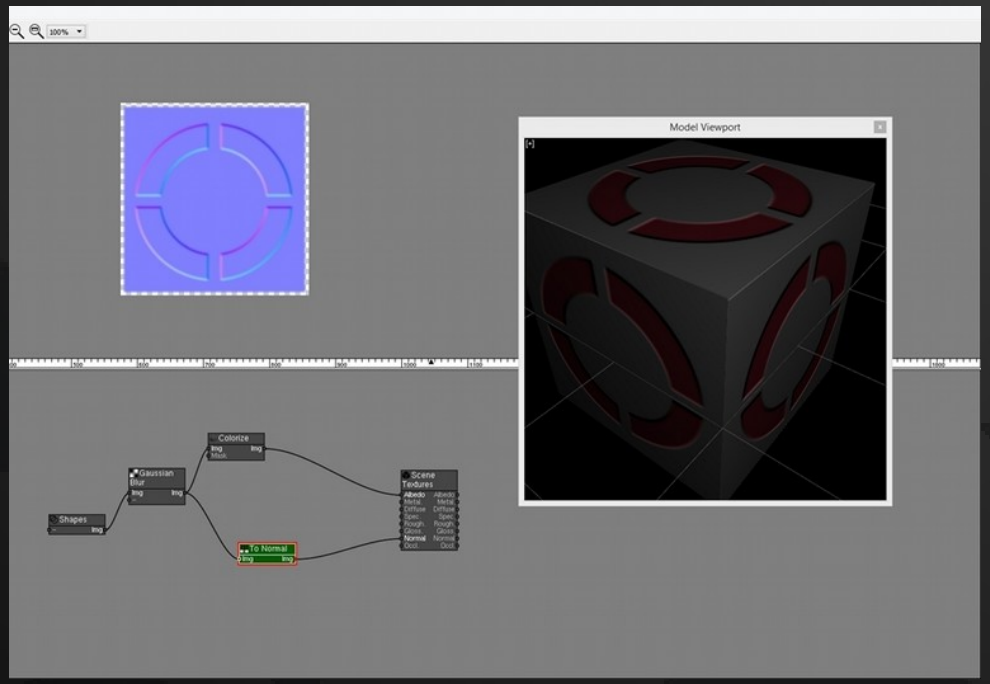
PhotoFiltre LX существует в двух вариантах: бесплатном и коммерческом LX Studio ($15), в котором реализованы некоторые дополнительные возможности. Исходники бесплатной версии доступны по лицензии GPL v3.

http://photofiltre-lx.org

sK1 2.0RC2

Выпущен второй кандидат в релизы векторного графического редактора sK1 2.0, реализующего функциональность, схожую с Corel Draw. Несмотря на статус RC, выпуск фактически является готовым инструментом – нереализованными на данный момент остался только импорт/экспорт сторонних графических форматов. На сайте проекта доступны бинарные сборки для Ubuntu, Linux Mint, Debian, Fedora, openSUSE и Windows.

http://sk1project.net



PixaFlux

PixaFlux – это нечто среднее между векторным редактором и редактором текстур а-ля Substance Painter. При помощи интерфейса на основе узлов пользователь собирает текстуру из простых геометрических форм и сплайнов (а также входных изображений и произвольных штрихов) – все это можно пропустить через различные фильтры, преобразовать в карту нормалей и т.д. Как и в Substance, тут имеется 3D-предпросмотр текстуры на модели с поддержкой PBR.

Программа бесплатна и в настоящее время находится в открытом бета-тестировании. Актуальная версия – PixaFlux 2016.11.11.

http://www.pixaflux.com

Cinelerra 6

Увидела свет новая, шестая по счету версия Cinelerra – одной из старейших свободных программ нелинейного видеомонтажа.

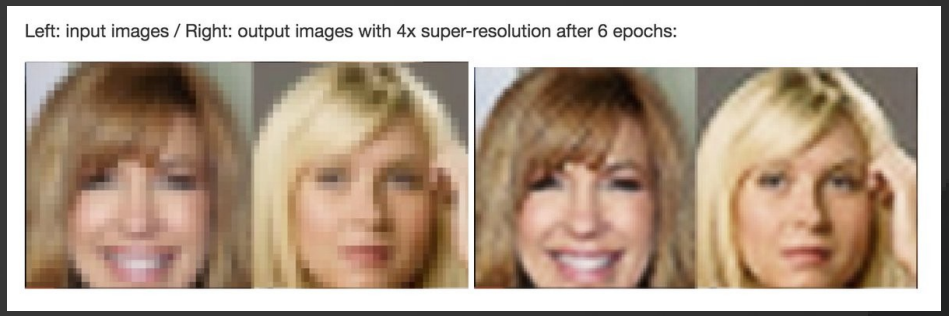
В новой версии обновлен код декодирования h264, оптимизированы компоненты отслеживания движения, добавлен новый интерфейс для передискретизации, в редактор титров добавлена поддержка загрузки файлов с субтитрами.

https://sourceforge.net/projects/heroines

Новые алгоритмы обработки изображений

Лаборатория Tetrachrome открыла исходный код проекта Subpixel, в рамках которого подготовлена реализация сверточной нейронной сети, выполняющей работу по восстановлению качества изображений. Система может использоваться для воссоздания промежуточных пикселей изображения, утерянных в результате масштабирования, или для повышения качества деталей на общих снимках. Код открыт под лицензией MIT.

https://github.com/Tetrachrome/subpixel



Лаборатория искусственного интеллекта Facebook представила открытую реализацию алгоритмов DeepMask и SharpMask для распознавания объектов на фотографиях. Они позволяют на основе машинного анализа классифицировать элементы фотографии, определить, что на них изображено, и с точностью до пикселей выделить объекты из общего фона. DeepMask представляет собой алгоритм выделения сегментов изображения, а SharpMask предоставляет средства для уточнения результата.

Алгоритмы оформлены в виде модулей к библиотеке машинного обучения Torch, написанных на Lua и распространяемых по лицензии BSD.

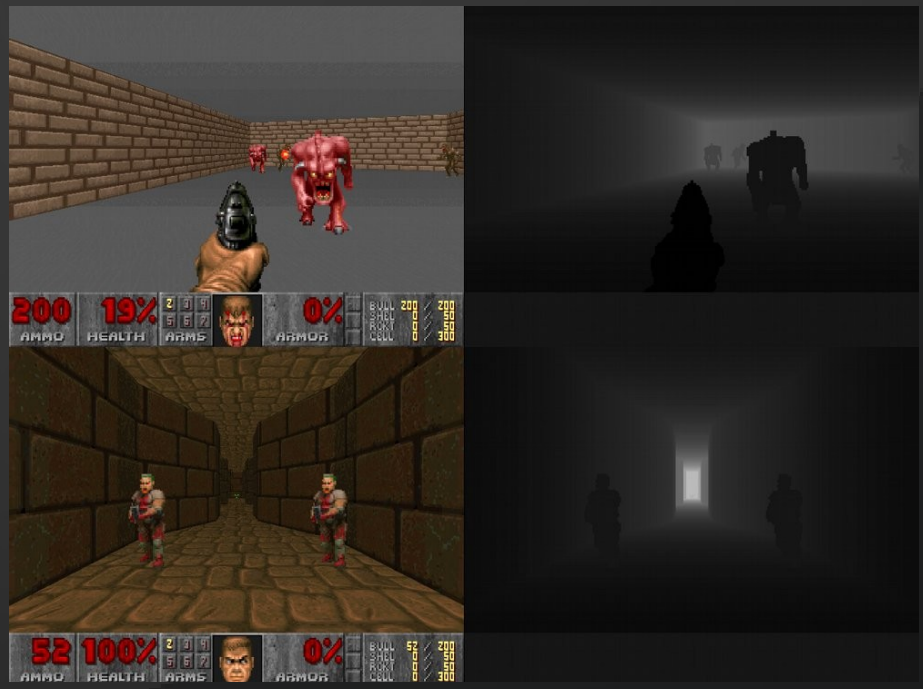
https://github.com/facebookresearch/deepmask

Компания Yahoo представила систему для выявления изображений неприличного содержания с использованием методов глубинного машинного обучения – в текущем виде модель натренирована на выявление порнографии. Готовая модель для классификации подобных материалов опубликована под лицензией BSD и опробована в использовании с открытыми фреймворками машинного обучения Caffe и CaffeOnSpark.

https://github.com/yahoo/open_nsfw

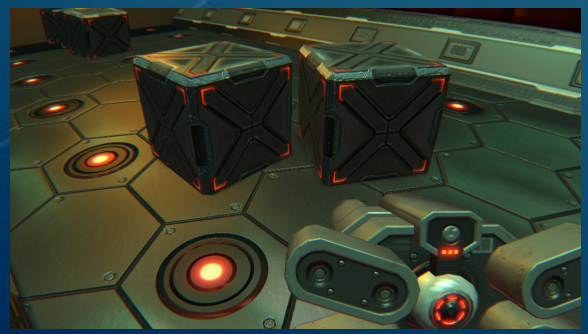
В рамках проекта VizDoom развивается система искусственного интеллекта для игры в Doom. От обычных игровых ботов VizDoom отличается тем, что система работает на основе анализа визуальной информации, отслеживая только изменение содержимого экрана и используя методы машинного обучения. Код проекта распространяется под лицензией MIT. Проект развивается группой исследователей из Технологического университета города Познани (Польша).

Ссылка



Atrium – проект от авторов журнала «FPS» по созданию 3D-игры на языке D, научно-фантастический шутер от первого лица с головоломками, основанными на физике. Проект ставит целью доказать, что на D возможно создание кроссплатформенных игровых приложений AAA-класса. В рамках Atrium разрабатываются собственные графический и физический движки – DGL и dmech, с помощью которых можно будет разрабатывать игры любой жанровой направленности.

Читайте подробнее в блоге Atrium: http://dlanggamedev.blogspot.ru



Язык

Новости «с Марса»

свежие релизы и обновления

Если вы разрабатываете проект, связанный с языком D и хотите рассказать о нем миру, найти новых пользователей, контрибьюторов или тестеров, сообщите об этом нам! Мы готовы регулярно публиковать ваши анонсы со ссылкой на репозиторий и/или страницу проекта. Сообщения принимаем, как обычно, на ящик редакции: ***gecko0307@gmail.com***

• ИНФРАСТРУКТУРА

DMD 2.072.0

Вышла новая версия референсного компилятора D – DMD 2.072.0. Релиз привносит несколько небольших изменений в спецификацию языка, поддержку TLS на 64-битной OSX, а также множество улучшений в стандартной библиотеке.

http://dlang.org/download.html

LDC 1.1.0-beta3

Увидел свет третий бета-выпуск LDC 1.1.0 – новой версии компилятора D с LLVM в качестве бэкенда. Релиз основан на фронтенде и рантайме D 2.071.1, включает поддержку LLVM 3.5-3.9. LDC 1.x полностью поддерживает Linux, OSX, Win32 и Win64, ARM, совместим с Objective-C, включает частичную поддержку Android. В комплекте с компилятором теперь поставляется DUB.

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

Динамическая компиляция

D-программист Иван Бутыгин представил весьма любопытную разработку – форк LDC со встроенным JIT-компилятором. С ним можно пометить любую функцию атрибутом @runtimeCompile, и в исполняемый файл будет встроен соответствующий ей биткод LLVM, который будет компилироваться в машинный код в рантайме. Основное преимущество – динамическая оптимизация кода: можно оптимизировать его под конкретный процессор, на котором он выполняется.

https://github.com/Hardcode84/ldc/tree/runtime_compile

Ocean 2.1.1

Библиотека Ocean обновилась до версии 2.1.1. Это минорный исправляющий релиз, первый после открытия исходников проекта (версия 2.1.0).

Напомним, Ocean – это достаточно старый проект времен D1 и Tango, библиотека стандартной функциональности для высокопроизводительных приложений реального времени. Ocean начинался как расширение Tango, сейчас это полностью самостоятельный проект. Библиотека совместима с D1 и D2, но работает только под Linux. Отличительной особенностью Ocean является минимальное использование сборщика мусора.

https://github.com/sociomantic-tsunami/ocean

• Геймдев и мультимедиа

Derelict-WinTab

Анонсирован Derelict-биндинг к WinTab, интерфейсу к графическим планшетам Wacom под Windows.

https://github.com/buggins/derelict-wintab

SoundTab

На D ведется разработка весьма интересного проекта – программного синтезатора SoundTab, который использует графический планшет для управления звуком. Также поддерживается и управление мышью, но с планшетом есть возможность контролировать громкость силой нажатия пера. Программа основана на Derelict-WinTab для взаимодействия с планшетом и DLangUI для построения интерфейса и пока работает только под управлением Windows. В будущем не исключена поддержка Linux и OSX.

https://github.com/buggins/soundtab

Декодеры аудио

Для D появились нативные декодеры популярных аудиоформатов – MP3, OGG/Vorbis и FLAC. Они являются портами, соответственно, minimp3, stb_vorbis и drflac.

http://repo.or.cz/iv.d.git/blob_plain/HEAD:/stb/vorbis.d
http://repo.or.cz/iv.d.git/blob/HEAD:/drflac.d
http://repo.or.cz/iv.d.git/blob/HEAD:/minimp3.d

• Веб-разработка

libotr

Анонсирован биндинг к библиотеке libotr, реализации протокола для зашифрованного обмена мгновенными сообщениями Off-the-Record. OTR считается одним из наиболее защищенных протоколов связи.

http://repo.or.cz/libotrd.git

D Embedded Database

Анонсирован проект D Embedded Database, направленный на создание нативной файловой СУБД на D, потоково-безопасной и соответствующей требованиям ACID.

Ссылка

mysql-native 0.1.5

mysql-native, клиент для СУБД MySQL и MariaDB, обновился до версии 0.1.5. Релиз носит, в основном, исправляющий характер.

https://github.com/mysql-d/mysql-native

SQLite-D

Вышла бета-версия SQLite-D, библиотеки для чтения баз данных в формате SQLite 3.

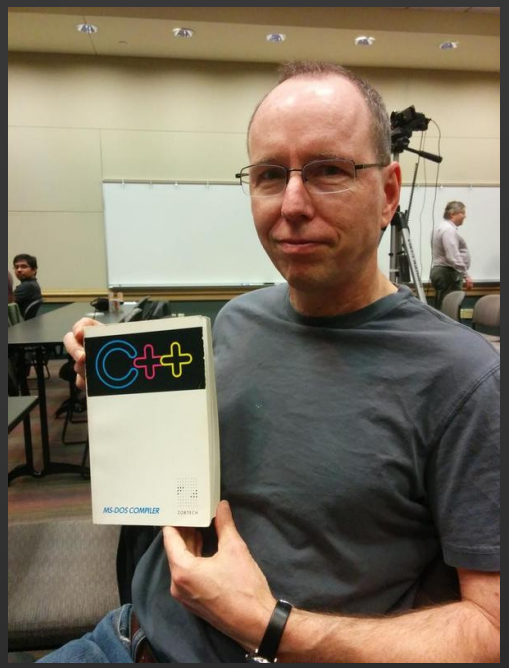
https://github.com/UplinkCoder/sqlite-d

Интервью с Уолтером Брайтом

В «FPS» №31 '14 мы уже публиковали своеобразное интервью с Уолтером Брайтом в форме вопросов пользователей Reddit. Недавно в Блоге D была опубликована довольно интересная беседа с создателем D, в которой он поделился своим нынешним видением своего детища и его перспектив.

– D достаточно быстро набирает популярность в последние годы – особенно после прошедшего недавно DConf и выхода статьи на Wired. Количество скачиваний компилятора DMD в сутки составляет 1200. Что вы думаете о нынешнем состоянии языка, и каковы ваши планы по его продвижению?

– Я не сильно беспокоюсь обо всем этом. Я вложил усилия в то, чтобы сделать D лучшим языком, и не вижу причин заботиться о статистике. Это все равно, что работа исполнительного директора – он не занимается повышением курса акций компании, его больше заботит прибыль. В основном, я занимаюсь работой, которую никто больше не хочет делать – например, регрессионными исправлениями. Конечно, намного интереснее разрабатывать что-то новое, чем поддерживать старое.



– Из улучшений в языке в последнее время на первом месте стоят @nogc и взаимодействие с C++. Почему? Вы ведь всегда говорили, что более тесная совместимость с «плюсами» – непосильно сложная задача...

– Стало ясно, что сборщик мусора нужен далеко не всем, и выделение памяти может быть отделено от логики алгоритмов. Позволить программистам решать, использовать сборщик или нет, является хорошим шагом вперед.

Что касается C++, то я нашел способы обеспечить совместимость с ним, обходя проблемы, которые раньше казались мне непреодолимыми. Например, нашлось решение для исключений – C++ позволяет выбрасывать исключения любого типа, однако мало кто использует при этом что-то помимо экземпляров класса, например, std::exception.

Все, что было нужно – ввести в D атрибут extern(C++) для конструкции throw/catch. Исключения другого типа не поддерживаются и вряд ли будут – но это, тем не менее, хорошее решение, покрывающее потребности большинства.

– Что вы планируете на ближайшее будущее?

– Сейчас мы сосредоточились на улучшении безопасности памяти – защита памяти в настоящее время становится все более важной, особенно в связи с вредоносным ПО, которое эксплуатирует уязвимости, связанные с небезопасным кодом. D всегда поддерживал защищенный доступ к памяти, однако сейчас мы многое пересматриваем и вносим изменения.

– Сколько своего времени вы уделяете D?

– Я работаю над языком полный рабочий день. Половина времени уходит на код, половина – на общение с другими разработчиками, написание статей, интервью и конференции.

– Вам было 42, когда вы начали работать над D – насколько я понимаю, это был первый спроектированный вами язык? Расскажите, почему вы начали работать над ним так поздно – ведь большинство контрибьюторов D сейчас моложе вас.

– Нет, первым языком, который я спроектировал, был ABEL. Сейчас он устарел, так как оборудование, на которое он был рассчитан, уже не используется. Однако он успешно просуществовал свои 10 лет. Вообще-то многие языки были спроектированы пожилыми инженерами. Я сам писал компиляторы в течение всей моей профессиональной карьеры, и за эти годы хорошо усвоил, что отличает хорошие языки от плохих. Наверное, самый главный закон в этой области – чем проще, тем лучше, но, как ни странно, не всегда простые решения даются легко.

– Расскажите об этом подробнее.

– Например, так было с дизайном шаблонов. Мне потребовалось несколько лет, чтобы осознать, что шаблон функции – это, по сути, функция с двумя наборами параметров, времени компиляции и времени выполнения. После этого все встало на свои места.

– Как вы считаете, почему до сих пор нет «приложения-убийцы» на D? Возьмем, например, Ruby – сначала он был никому не известен, но потом появился Rails...

– Я считаю, что для D время для появления «приложения-убийцы» уже давно минуло. Кажется, что все нужные приложения уже написаны, и вряд ли для какого-то языка в будущем появится что-то вроде Rails. Но я, конечно, могу и ошибаться.

– D – это уникальный проект, ведь за ним не стояли спонсоры. У C++ есть Bell Labs, у Go – Google, у Rust – Mozilla, и у всех есть оплачиваемые разработчики, работающие полный день, даже если это Open-Source-проекты. Не думаете ли вы, что в этом и заключается причина того, почему D отстает от них в популярности?

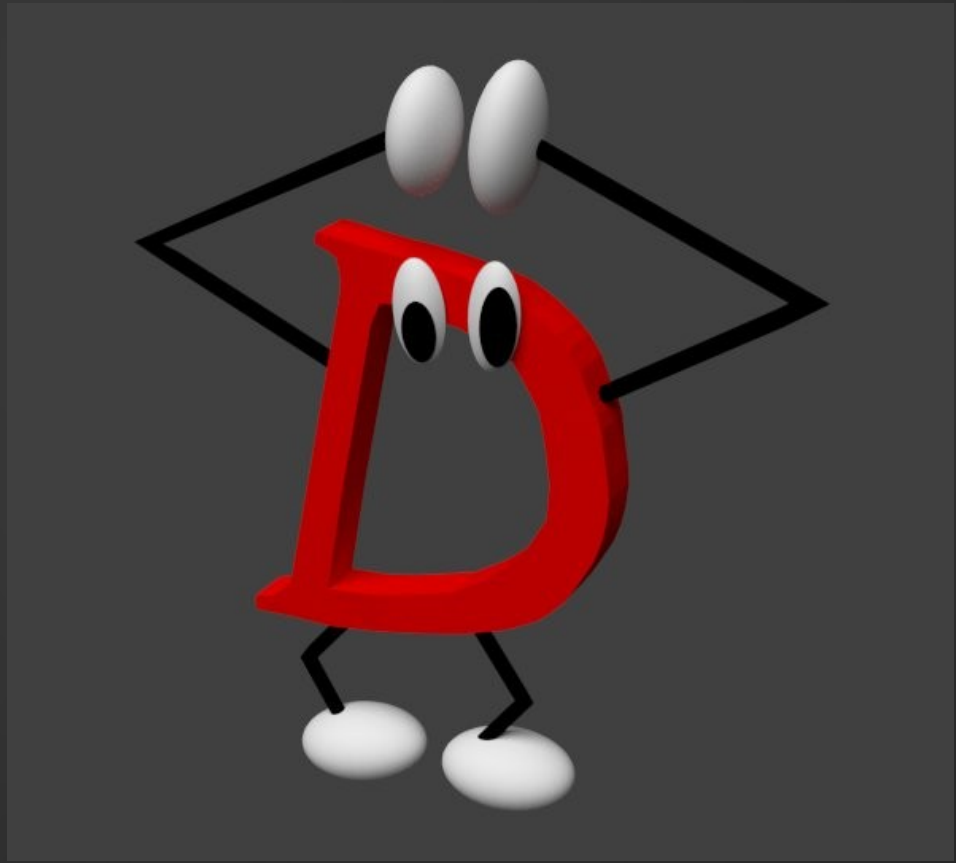
– Мы стараемся работать в этом направлении – недавно организовали некоммерческую организацию D Foundation, с которой компаниям будет проще спонсировать D.

– Приносит ли D какую-то прибыль? Известно, что у вас есть контракт с Facebook на создание линтера для «плюсов», а также препроцессора Warp. К тому же, вы сотрудничаете с Sociomantic и другими компаниями, использующими D.

– Я занимаюсь платными консультациями по D, но стараюсь, чтобы коммерческая деятельность не мешала разработке языка.

– Пишете ли вы код на D, не относящийся к стандартной библиотеке?

– Сам язык отнимает большую часть моего времени, поэтому я редко пишу что-то, помимо небольших утилит. Сейчас, кстати, фронтэнд компилятора D пишется на самом D.



– Стал ли язык тем, чем вы его представляли, когда начинали проект в 1999 году?

– D намного превзошел мои ожидания того времени. В разработке языков программирования с тех пор появилось множество новых идей, и D воплотил в себе лучшие из них.

Оригинал интервью:
https://dlang.org/blog

Kha

Универсальная платформа для разработки игр

Одной из важнейших парадигм в разработке ПО является переносимость – то есть, возможность компилировать программы под разные платформы без необходимости переписывать исходный код. В разные времена эту задачу пытались решать разными способами: началось все со стандартных системных API, но их возможностей вскоре перестало хватать. Появились Java и .Net, однако они оказались пригодны не для всех целей – среди разработчиков реалтаймовых приложений идея выполнять программы на виртуальной машине так и не прижилась.

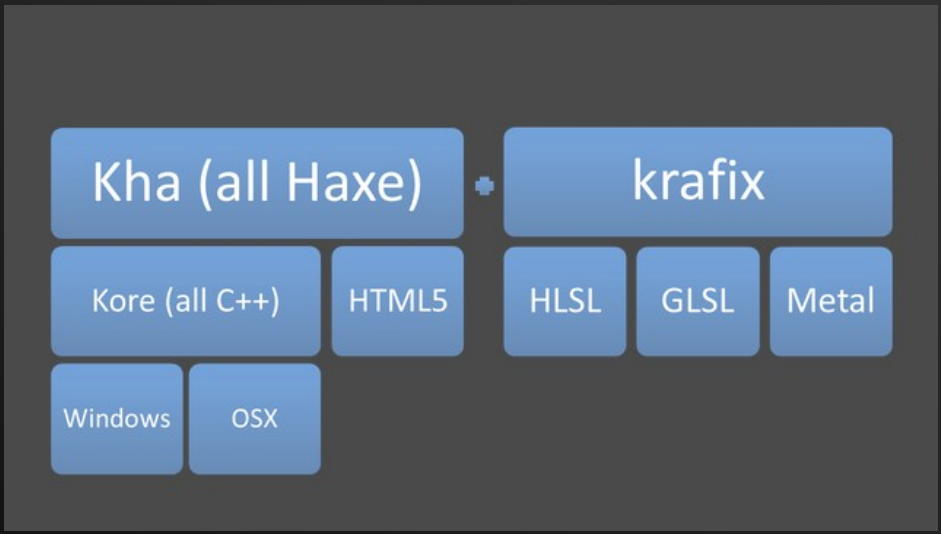
Затем стали появляться новые компилируемые языки с мощными библиотеками, скрывающими низкоуровневые особенности платформ под абстрактыми API – это D, Go, Rust и др. Но и они оказались не в состоянии поспевать за всеми веяниями моды. К примеру, сегодня все большую привлекательность приобретают HTML5 и трансляция в JavaScript – веб можно ругать бесконечно, но нельзя отрицать одного: он по-настоящему кроссплатформенный. Уже сейчас 3D-игры, работающие в браузере с приемлемым FPS – отнюдь не фантастика, а обыденная реальность. К тому же, развился мобильный сектор, а это совершенно отдельный мир, куда все еще трудновато попасть, используя, например, тот же D.

Нетрудно прийти к выводу, что убить двух зайцев – писать на компилируемом языке и получить абсолютную кроссплатформенность – практически невозможно. Отчаялись? Так вот, я вас обрадую: решение этой давней проблемы, похоже, найдено!

В «FPS» №13 '11 мы уже писали о **Haxe** – новом языке программирования, который ориентирован на трансляцию кода в другие языки и под различные платформы, включая C++, Flash, HTML5 и др. Это развитый ООП-язык со статической типизацией и классами, во многом похожий на C++ и C#. Уже тогда, в 2011 году, мы предсказывали Haxe большое будущее – и наступает оно именно сейчас.

Разработчики 2D-игр, наверное, уже давно знают о OpenFL, 2D-движке для Haxe, с помощью которого можно создавать игры под Windows, OSX, Linux, iOS, Android, HTML5 и других платформ. Но как быть с 3D, особенно с современными возможностями видеокарт? Ведь на каждой платформе свои особенности вывода 3D-графики. Можно, конечно, написать абстрактный высокоуровневый 3D-движок, но возможности его будут, так или иначе, ограничены.

Гениальный выход нашли программисты из фирмы KTX Software – они не стали создавать полноценный движок типа XNA, вместо этого предложив низкоуровневый, но при этом полностью переносимый графический API, получивший название **Kha**. Движков такого класса я лично до этого не встречал, поэтому Kha не мог не привлечь мое внимание.



Я узнал про Kha в связи с проектом Armory, новым 3D-движком для Blender, за разработкой которого слежу уже достаточно давно – он пишется именно на Haxe с использованием данного фреймворка. Kha – это, в сущности, универсальная основа для создания специализированных движков и приложений. Его можно рассматривать как некий объектно-ориентированный аналог OpenGL ES. При этом реальным бэкендом, непосредственно выводящим графику, может выступать как OpenGL, так и OpenGL ES, WebGL, DirectX и даже другие игровые движки – например, Unity.

Родная IDE для разработки под Kha – Kode Studio – генерирует проектные файлы для Visual Studio, IDEA и других сред. Как только вы захотите собрать релиз, вы просто запускаете Студию и компилируете. Для отладки в Kode Studio есть встроенный веб-клиент, также среда генерирует HTML5 – вы можете без труда отлаживать проект в любом браузере. При этом никаких изменений в Haxe-коде не требуется.

Список поддерживаемых платформ Kha очень впечатляет: HTML5 (WebGL или Canvas), Flash, Windows (Direct3D 9, Direct3D 11 или OpenGL), OSX, Linux, Android (OpenGL или Metal), Tizen, Unity, PlayStation Vita, Xbox 360 (XNA). Также программы для Kha можно скомпилировать в библиотеки для C# и Java. Сейчас, кстати, идет работа над поддержкой Vulkan. При этом производительность программ на Kha приближается к нативной – запустите демки Armory и убедитесь сами!

Графическая часть Kha довольно минималистична, и это радует. В ней нет лишней функциональности – вы можете рендерить вершинные буферы, задавая свои текстуры, шейдеры и настройки пайплайна, и все. Шейдеры пишутся на GLSL и автоматически преобразуются по необходимости в HLSL и MSL при помощи встроенного транслятора Krafix.

Я решил написать на основе Kha какой-нибудь несложный мультимедийный проект, поэтому в новом году ждите много материала по этой платформе на страницах «FPS»!

http://kha.tech

Тимур Гафаров

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 коммерческие файлообменники — загружайте файлы на свой сайт и присылайте статические ссылки, можно также использовать Dropbbox или Google Drive). Все материалы желательно архивировать в формате zip, rar, 7z, tar.gz или tar.bz2.

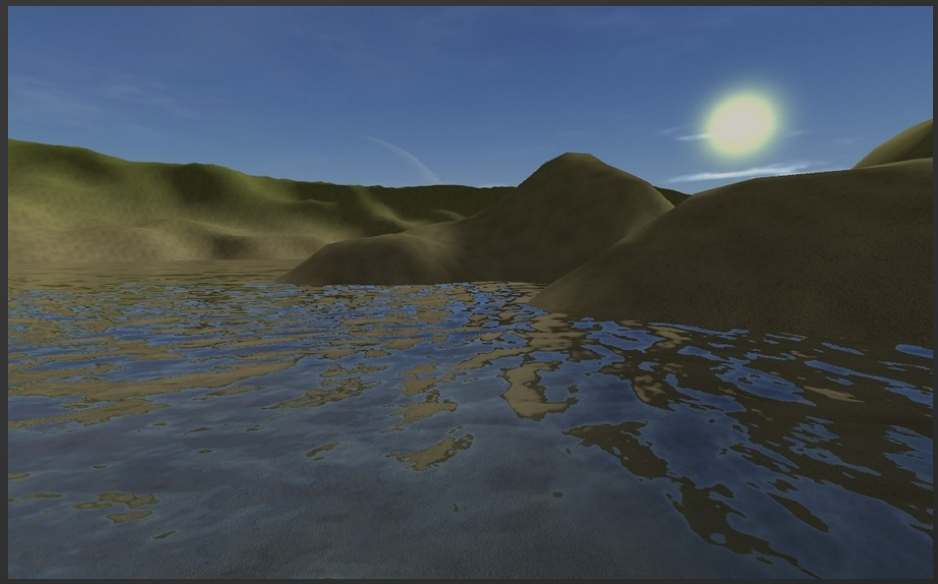
Шейдеры на все случаи жизни

Вода с отражением на GLSL

В «FPS» возвращается рубрика «Шейдеры на все случаи жизни»! В ближайших номерах ждите обзоры спецэффектов и различных современных техник рендеринга с использованием OpenGL и GLSL.

Как мы уже не раз отмечали на страницах нашего журнала, существует много способов рендеринга воды. В реальном времени рендеринг воды приемлемого качества неразрывно связан с отрисовкой отражений в текстуру. При этом существует две основные разновидности этой техники: кубические карты и фреймбуфер. То есть, в первом случае осуществляется рендеринг шести текстур – по сторонам воображаемого куба вокруг отражающего объекта. А во втором – всего одна текстура, которая затем накладывается на объект в экранном текстурном пространстве.

Второй метод может дать более качественный результат в плане разрешения, но ограничен плоскостью – то есть, его нельзя применить к объемным объектам. Кубические карты (cube map), напротив, дают реалистичные отражения на поверхностях любой сложности – и, при этом, благодаря мощностям современных GPU, могут быть почти столь же качественными, как и отражения через фреймбуфер.



Опять же, если ваша вода представляет собой плоскость с объемными волнами (а не просто анимированной картой нормалей), то только кубические карты могут дать достаточно реалистичные отражения, учитывающие эти волны.

Рендеринг самой кубической карты – задача достаточно тривиальная, поэтому мы не будем на ней останавливаться, сосредоточимся непосредственно на шейдере воды (будем использовать OpenGL 2.0).

В вершинную программу передаются две uniform-матрицы – видовая и обратная видовая. В ней выполняются привычные трансформации, а также вычисление вектора вида в мировом пространстве – это нужно для чтения из кубической карты.

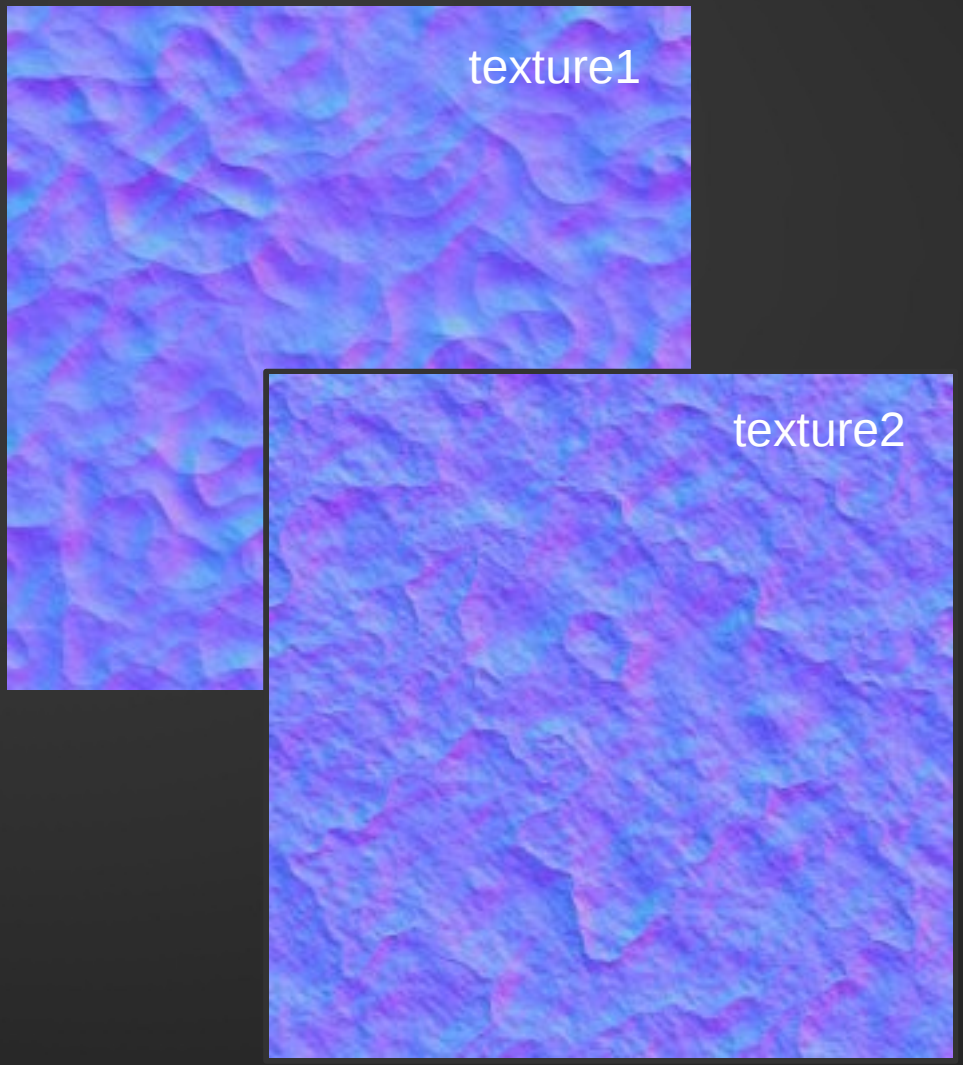
```
varying vec3 position;
varying vec3 normal;
varying vec3 tangent;
varying vec3 bitangent;
varying vec3 eye;
varying vec3 eyeTan;
varying vec3 eyeWorld;

uniform mat4 viewMatrix;
uniform mat4 invViewMatrix;

void main()
{
  gl_TexCoord[0] = gl_MultiTexCoord0;

  vec4 vertEye = gl_ModelViewMatrix * gl_Vertex;
  position = vertEye.xyz;
  eye = normalize(-position);

  vec3 worldPos = (invViewMatrix * vertEye).xyz;
  vec3 worldCamPos = (invViewMatrix[3]).xyz;
  eyeWorld = normalize(worldPos - worldCamPos);

  normal = normalize(gl_NormalMatrix * gl_Normal);
  tangent = normalize(gl_NormalMatrix * vec3(1, 0, 0));
  bitangent = cross(normal, tangent);

  eyeTan = position;
  eyeTan.x = dot(position, tangent);
  eyeTan.y = dot(position, binormal);
  eyeTan.z = dot(position, normal);
  eyeTan = -eyeTan;

  gl_Position = ftransform();
}
```

Во фрагментную программу передается собственно кубическая карта с отражениями, а также две бесшовные карты нормалей – мы просто двигаем их (вернее, их текстурные координаты) в противоположных направлениях, а затем складываем и усредняем нормали. В результате, получаются анимированные волны вполне приемлемого качества – так можно рендерить не только мелкие водоемы, но и целые моря.

Также в шейдере используется коэффициент Френеля для плавного перехода прозрачной воды в отражение – если смотреть на воду под большим углом, близким к прямому, то будет видно дно, а если смотреть под маленьким углом, то видно отражение.

```
#version 120
varying vec3 position;
varying vec3 normal;
varying vec3 tangent;
varying vec3 bitangent;
varying vec3 eye;
varying vec3 eyeTan;
varying vec3 eyeWorld;

uniform samplerCube texture0;
uniform sampler2D texture1;
uniform sampler2D texture2;
uniform mat4 viewMatrix;
uniform float scrollTime;

const vec4 waterColor = vec4(0.0, 0.05, 0.05, 1.0);
const float textureScale = 0.25;
```

```
void main()
{
  vec3 dirToLight =
   normalize(gl_LightSource[0].position.xyz - position);

  vec3 eyeN = normalize(normal);
  vec3 eyeT = normalize(tangent);
  vec3 eyeB = normalize(bitangent);
  vec3 L = vec3(dot(dirToLight, eyeT),
                dot(dirToLight, eyeB),
                dot(dirToLight, eyeN));

  vec2 uv = gl_TexCoord[0].st * textureScale;
  vec2 scrolluv1 = vec2(uv.s, uv.t + scrollTime * 0.1);
  vec2 scrolluv2 = vec2(uv.s + scrollTime * 0.1, uv.t);

  vec3 norm1 = normalize(
    texture2D(texture1, scrolluv1).rgb * 2.0 - 1.0);
  vec3 norm2 = normalize(
    texture2D(texture1, scrolluv2).rgb * 2.0 - 1.0);
  vec3 N = (norm1 + norm2) * 0.5;

  vec3 E = normalize(eyeTan);
  vec3 worldNormal = transpose(mat3(viewMatrix)) * N.xzy;
  vec3 R = reflect(normalize(eyeWorld), worldNormal);
  R = normalize(vec3(-R.x, R.y, -R.z));

  vec3 H = normalize(L + E);

  float fresnel = 1.0 - max(dot(eye, eyeN), 0.0);

  float diffuse = clamp(dot(N, L), 0.0, 1.0);
  float specular = pow(max(dot(H, N), 0.0),
    3.0 * gl_FrontMaterial.shininess);

  vec4 tex = textureCube(texture0, R);
  vec4 reflectionTerm = tex * fresnel +
    waterColor * diffuse * (1.0 - fresnel);

  vec4 col =
   reflectionTerm +
   diffuse * vec4(0.07, 0.07, 0.07, 1.0) +
   specular * gl_FrontMaterial.specular;

  col.a = max(0.9, fresnel);
  gl_FragColor = col;
}
```



Единственное, что здесь не поддерживается – это преломление. Чтобы его добавить, нужно рендерить дно в отдельный буфер и искажать его путем сдвига текстурных координат. Но это уже тема для отдельной статьи.

Тимур Гафаров

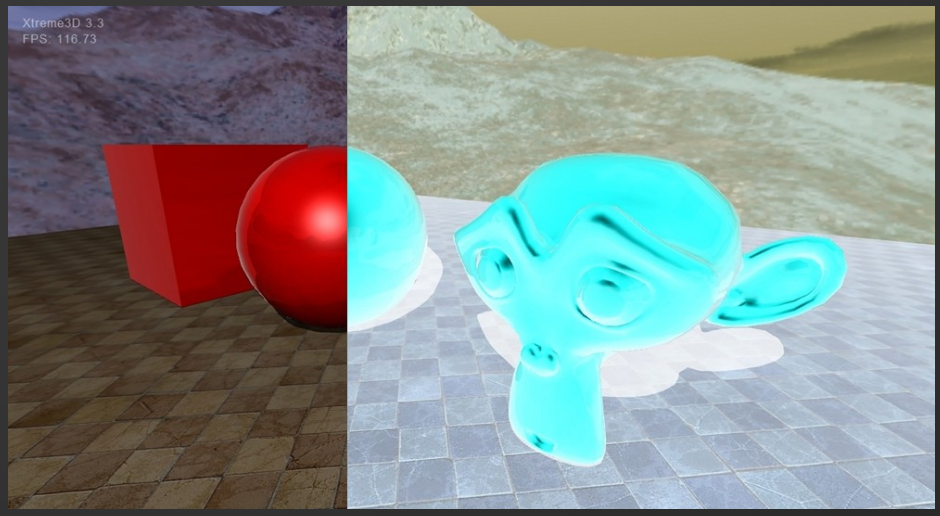
Xtreme3D 3.x

Постобработка при помощи FBO

В одном из прошлых номеров мы уже анонсировали Xtreme3D 3.x – новую версию 3D-движка для Game Maker 8. Напомним, Xtreme3D 2.0 был проприетарным проектом, и мы в сообществе http://xtreme3d.narod.ru в какой-то момент решили переписать его с нуля для дальнейшего ее развития. Теперь в истории Xtreme3D начался очередной этап: данной статьей открывается новый цикл уроков по движку!

В Xtreme3D 3.2 появилась поддержка фреймбуферных объектов (Framebuffer Objects, сокращенно FBO). Это современная технология внеэкранного рендеринга – то есть, проще говоря, рендеринга в текстуру. С ее помощью можно написать абсолютно любой эффект постобработки – то есть, пропустить отрендеренное изображение через какой-нибудь фильтр. В качестве примеров постобработки можно назвать Motion Blur, Bloom, SSAO, Depth of Field и т.д. Некоторые популярные эффекты уже реализованы на Xtreme3D – вы можете найти соответствующий пример на странице http://xtreme3d.narod.ru/examples.htm.

В Xtreme3D и раньше можно было рендерить в текстуру (ViewerCopyToTexture в Xtreme3D 2.0), а в Xtreme3D 3.0 появилась поддержка MemoryViewer – обертки над p-буферами, которые были распространены до появления FBO. Преимуществом FBO перед этими методами является значительно более высокая производительность – в отличие от MemoryViewer, тут не используется переключение контекстов, и, в отличие от ViewerCopyToTexture, не происходит копирование участков видеопамяти. Фактически, FBO позволяет просто создать в видеопамяти новый буфер кадра и рендерить в него. Затем этот буфер используется в качестве текстуры.



На этом уроке я покажу, как использовать FBO для реализации простейшего эффекта постобработки – инвертирования. Общая логика нашей программы будет выглядеть следующим образом:

1. Создаем FBO размером с наш Viewer.
2. Создаем материал и шейдер, осуществляющий инвертирование. К шейдеру прикрепляем текстуру с буфером из FBO.
3. Создаем экранный спрайт для эффекта инвертирования, покрывающий весь Viewer целиком. К нему будет применен материал с шейдером.

Рендеринг осуществляется в таком порядке:

1. Вызываем Update
2. Рендерим 3D-сцену в FBO (не рендерим 2D-объекты!)
3. Рендерим экранный спрайт
4. Рендерим 2D-объекты, если они нужны.

Для такого «нестандартного» порядка отрисовки потребуется задать особую иерархию корневых Dummycube'ов:

```
global.backAndScene = DummycubeCreate(0);
global.back = DummycubeCreate(global.backAndScene);
global.scene = DummycubeCreate(global.backAndScene);
global.front = DummycubeCreate(0);
global.fboFront = DummycubeCreate(0);
```

Обратите внимание, что мы сгруппировали global.back и global.scene под общим родителем global.backAndScene, чтобы их можно было отрендерить вместе. Также мы создали global.fboFront – это будет отдельный родитель для экранного спрайта с нашим эффектом.

Теперь можно спокойно создавать любую 3D-сцену, а также 2D-объекты, используя для этого привычные Dummycube'ы global.back, global.scene и global.front. Затем создаем FBO (учитывая, что вы уже создали Viewer и камеру):

```
fbo = FBOCreate(
  window_get_width(), window_get_height(), view);
FBOSetCamera(fbo, camera);
```

Создаем шейдер и передаем ему в параметр текстуру из FBO:

```
vp = TextRead('shaders/invert-vp.glsl');
fp = TextRead('shaders/invert-fp.glsl');
invertShader = GLSLShaderCreate(vp, fp);
paramTexture = GLSLShaderCreateParameter(
  invertShader, 'texture');
GLSLShaderSetParameterFBOColorTexture(
  paramTexture, fbo, 0);
```

Шейдеры у нас будут такие:

Вершинный шейдер – файл «shaders/invert-vp.glsl»:

```
void main()
{
  gl_Position = ftransform();
  gl_TexCoord[0] = gl_MultiTexCoord0;
}
```

Фрагментный шейдер – файл «shaders/invert-fp.glsl»:

```
uniform sampler2D texture;

void main()
{
  vec4 color = texture2D(texture, gl_TexCoord[0].xy);
  gl_FragColor = 1.0 - color;
  gl_FragColor.a = 1.0;
}
```

Создаем материал, использующий этот шейдер, и экранный спрайт с данным материалом:

```
MaterialCreate('mInvert', '');
MaterialSetOptions('mInvert', 1, 1);
MaterialSetShader('mInvert', invertShader);
```

```
fboSprite = HUDSpriteCreate('mInvert',
  window_get_width(),
  window_get_height(),
  global.fboFront);

ObjectSetPosition(fboSprite,
  window_get_width()/2.0,
  window_get_height()/2.0, 0);
```

Спрайт мы помещаем в потомки к global.fboFront.

Теперь в событии Step:

```
Update(1.0 / room_speed);
FBORenderObject(fbo, global.backAndScene);
ViewerRenderEx(view1, global.fboFront,
  true, false, false);
ViewerRenderEx(view1, global.front,
  false, true, true);
```

Смысл этого кода стоит объяснить подробнее. Сначала мы, как обычно, вызываем Update, обновляя внутренние состояния объектов. Затем, вместо того, чтобы вызывать ViewerRender, мы рендерим FBO, передавая в соответствующую функцию наш корневой Dummycube для объектов заднего и сценического плана – global.backAndScene. Тем самым мы рисуем эти объекты в текстуру, которая используется шейдером инвертирования.

Затем нам нужно отрендерить во Viewer только экранный спрайт с эффектом инвертирования, игнорируя остальные объекты. Это невозможно сделать при помощи ViewerRender, поскольку она отрендерит все корневые Dummycube'ы. Но в Xtreme3D 3.2 появилась новая функция ViewerRenderEx, с помощью которой можно рендерить отдельные объекты и иерархии – мы вызываем ее для global.fboFront.

Параметры true, false, false задают следующие опции рендеринга: очистить Viewer фоновым цветом, не делать переключение заднего и переднего буферов (мы это сделаем следующим вызовом) и не обновлять счетчик FPS (мы тоже сделаем это дальше).

Вторым вызовом ViewerRenderEx мы рендерим global.front и, соответственно, все двумерные объекты переднего плана – то есть, экранный текст и HUD-объекты, которые не должны попадать под эффект инвертирования. В этот раз мы не хотим очищать Viewer, зато хотим переключить буферы и обновить счетчик кадровой частоты.

Вы можете вызывать ViewerRenderEx сколько хотите, для любых объектов и Dummycube'ов, но помните, что переключать буферы и обновлять счетчик можно только про последнем вызове этой функции в шаге – в противном случае движок будет работать неправильно.

Как видите, постобработка в Xtreme3D 3.2 требует несколько дополнительных действий по сравнению с обычным прямым рендерингом во Viewer, однако структура сцены особо не меняется – можно адаптировать эту технологию практически к любой игре. Кроме того, можно создать несколько FBO и рендерить в них разные группы объектов, применять к полученным картинкам разные шейдеры и составлять целые конвейеры фильтров, где одна картинка поступает в другой FBO для дополнительной обработки – современные видеокарты позволяют делать подобные вещи очень быстро. Возможности поистине безграничны!

Тимур Гафаров

Мобильный FPS

Теперь любимый журнал всегда с вами!

Читайте FPS на мобильных устройствах: скачайте приложение для Android или iOS!

Available on the App Store

ANDROID APP ON Google play

Разработчик приложения: цифровое издательство St.Appler http://www.stappler.org/

Linux-гейминг

Игровые новости из мира СПО

С каждым годом все больше компаний обращает внимание на Linux как на игровую платформу. В этом году были выпущены такие игры как **XCOM 2**, **Stellaris** и **SOMA**. Количество игр для Linux в **Steam** превысило 2000. Рубеж в 1000 игр был пройден в середине марта 2015 года, 1500 игр наблюдалось год назад.

Вышел 21-й альфа-выпуск игры **0 A.D**. В этой версии появилось новое государство – империя Селевкидов, добавлены новые здания и 11 случайных карт, девять из которых отождествлены с реальными географическими регионами (Амазонка, Средиземноморье, Помпеи, Бахрейн и т.д.).

Напомним, игра является стратегией реального времени с исторической тематикой – она охватывает цивилизации, существовавшие примерно с 500 года до н.э. до 500-го года н.э.

http://play0ad.com

Увидела свет новая версия **SuperTux** – 0.5. Релиз примечателен добавлением в игру встроенного редактора уровней, а также повышением производительности и улучшением нескольких уровней. Напомним, SuperTux – это свободный 2D-платформер в стиле Mario с пингвином Туксом в главной роли.

http://supertuxproject.org

После двух лет разработки вышел **Hatari** 2.0 – эмулятор компьютеров Atari и совместимых с ними машин. В новой версии отмечаются значительные изменения в эмуляции CPU – под управлением эмулятора теперь можно запустить NetBSD для Atari ST. Добавлена поддержка компьютеров MegaST и MegaSTE.

Организация LunarG, занимающаяся разработкой Vulkan SDK, представила **VIA** (Vulkan Installation Analyzer) – инструмент валидации драйверов Vulkan. Утилита выявляет различные проблемы с установленным драйвером, а также помогает определить, поддерживает ли ваша система Vulkan. Исходники VIA распространяются по лицензии Apache License 2.0 и доступны на GitHub.

https://vulkan.lunarg.com/doc/sdk/latest/windows/via.html

После трех месяцев разработки доступен релиз свободной реализации OpenGL – **Mesa** 13.0, примечательный реализацией OpenGL 4.5 в драйверах RadeonSI, Nouveau (nvc0) и Intel (i965). Поддержка новых версий OpenGL доступна для видеокарт AMD на основе архитектуры GCN, NVIDIA на базе GPU Fermi, Kepler и Maxwell и Intel семейства Gen8+. Кроме того, в драйвере Intel для GPU семейства gen9+ (Skylake) обеспечена поддержка OpenGL ES 3.2, а для серии Gen4 (Haswell) – OpenGL ES 3.1. Первый выпуск ветки Mesa 13.0.0 имеет экспериментальный статус – после проведения окончательной стабилизации кода будет выпущена стабильная версия 13.0.2.

http://www.mesa3d.org

Компания **NVIDIA** представила новую стабильную версию проприетарного драйвера NVIDIA 375.20. Драйвер доступен для Linux (ARM, x86, x86_64), FreeBSD (x86, x86_64) и Solaris (x86_64).

Linux: полезные команды

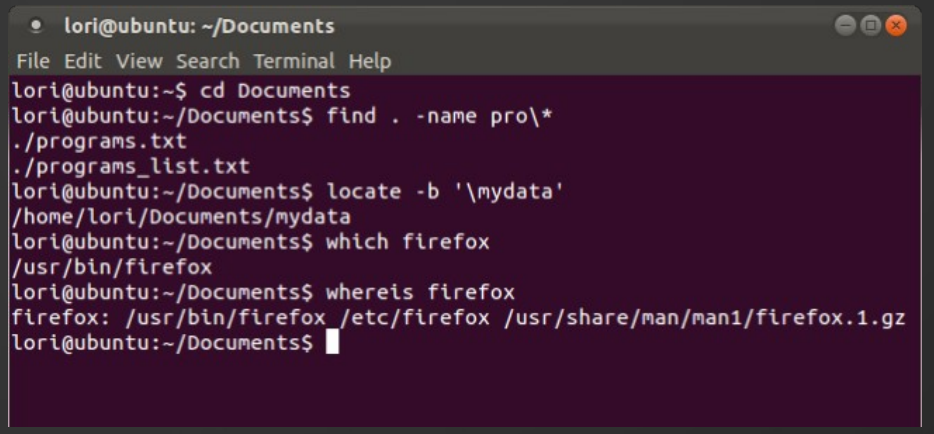
Продолжаю делиться опытом: в последние месяцы мне на своей Linux-машине довелось решать множество разных интересных задач – среди них работа с текстами и мультимедиа, шаманство в Git и многое другое...

Однажды понадобилось принять pull request, отправленный в master, в другую ветку репозитория. Это не так просто, как может показаться, но в Git нет ничего невозможного. Для этого нужно создать временную локальную ветку, принять в нее патч, а затем слить с целевой веткой:

```
$ git fetch origin pull/XXX/head:tmp_branch_name
$ git checkout branch_name
$ git merge --no-ff tmp_branch_name -m "Merged #XXX
to <branch_name> branch"
$ git push origin branch_name
```

Здесь вместо XXX подставьте номер pull request'a – он отображается веб-интерфейсом GitHub. Вместо branch_name подставьте название ветки. Вместо tmp_branch_name можно подставить свое название временной ветки.

Очень много интересного можно сделать с помощью FFMPEG – раньше, во времена Windows, я использовал для подобных целей проприетарные утилиты, часто платные и, к тому же, под каждую задачу приходилось искать новый инструмент. А теперь диву даюсь, сколько возможностей скрыты в одной-единственной программе, и притом свободной!



Начну с простого – конвертации FLAC в MP3. На основе этой команды можно написать скрипт для пакетной конвертации нескольких файлов:

```
$ ffmpeg -i input.flac -ab 320k -acodec libmp3lame
output.mp3
```

Декодируем FLAC в несжатый WAV:

```
$ ffmpeg -i input.flac output.wav
```

Нечасто используемая, но иногда нужная команда – декодирование аудиоданных из видео в заданном временном промежутке:

```
$ ffmpeg -ss 00:46:20:00 -t 00:52:19:00 -i input.mp4
-acodec flac output.flac
```

Воспроизвести WAV-файл, не выходя из консоли, можно при помощи встроенной утилиты ALSA – aplay:

```
$ aplay sound.wav
```

При помощи этой короткой команды можно узнать суммарный размер всех файлов в каталоге (физический размер каталога на диске):

```
$ du -h
```

Для сравнения текстовых файлов есть diff, а как быть с бинарными? На помощь приходит cmp:

```
$ cmp file1 file2
```

Для некоторых хакерских целей (легальных!) бывает нужно извлечь из бинарного файла все текстовые данные – hex-редактор для этого не нужен, можно просто сделать следующее:

```
$ strings < binaryFile > text.txt
```

Ну и на десерт я, как обычно, припас большой вкусный bash-скрипт – он конвертирует все текстовые файлы в текущем каталоге из кодировки Windows-1251 в UTF-8:

```
find ./ -name "*.htm" -type f |
while read file
do
  echo " $file"
  mv $file $file.icv
  iconv -f WINDOWS-1251 -t UTF-8 $file.icv > $file
  rm -f $file.icv
done
```

Тимур Гафаров

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://github.com/gecko0307/cook2

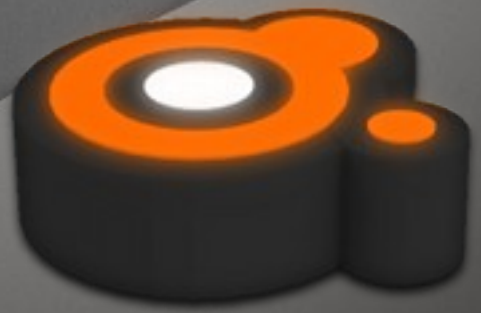
dlib

Коллекция библиотек «на все случаи жизни» для D, которая может быть использована в игровых движках и других мульти-медийных 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 архи-тектуру.

http://github.com/gecko0307/dlib

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 на любые темы, касающиеся компьютерных игр, графики, звука, программирования и т.д. на **gecko0307@gmail.com**.



http://fps-magazine.cf