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

FPS №44

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

В ЭТОМ НОМЕРЕ

Blender Новости

Начнем, как обычно, с анонса новой версии Blender. Недавно вышла версия 2.78 – ключевые новшества этого релиза включают поддержку сферического стереорендеринга в Cycles, новые возможности для системы риггинга B-Bones, переработанный инструмент Grease Pencil, начальную поддержку Alembic (открытого формата обмена данными от Sony Pictures Imageworks). Также оптимизирован расход памяти для отмены операций в режиме редактирования, улучшены различные инструменты моделирования, лепки, развесовки и рисования. Популярный аддон Archimesh от Антонио Васкеза, позволяющий генерировать дома и интерьеры с мебелью, теперь вошел в состав дистрибутива Blender, то же касается и дополнения MeasureIt от того же автора.

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

Российский CG-художник Николай Мамашев запустил краудфандинг-кампанию для производства анимированного комикса по мотивам знаменитой серии Давида Ревуа «Pepper & Carrot». Пайплайн проекта будет основан исключительно на свободном ПО – Linux, Blender и Krita, а все материалы будут доступны по лицензии CC-BY-SA.

Поддержать инициативу можно **на IndieGoGo**.

Напомним, «Pepper & Carrot» – это веб-комикс о приключениях юной волшебницы и ее котенка, все материалы которого создаются при помощи СПО и распространяются по лицензии CC-BY 4.0.

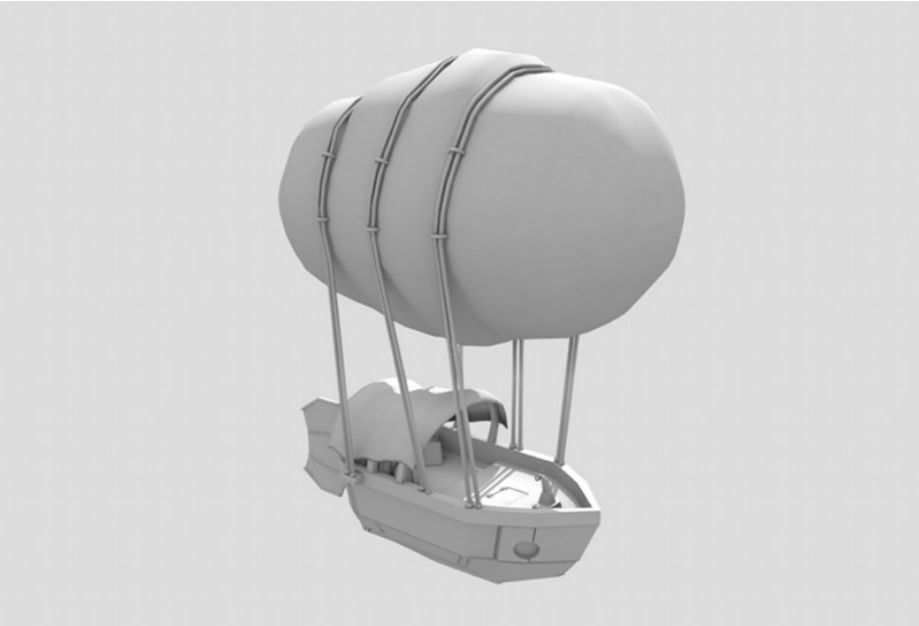
http://www.peppercarrot.com

Также этим летом состоялась премьера полнометражного фильма «Obviously: the Real and the Unreal» от канадского режиссера Томми Геренчера, все спецэффекты к которому были подготовлены при помощи Blender. Эта полуторачасовая психологическая притча, снимавшаяся аж 8 лет, рассказывает о жизни музыканта, долгое время принимавшего ЛСД и живущего в своей собственной необычной реальности.

https://www.youtube.com/watch?v=qoG8JhwRZiQ

Короткометражный фильм Circles, выполненный при помощи Blender и GIMP – 2-минутная новелла в необычном визуальном стиле о путешествии двух людей по пустыне.

https://www.youtube.com/watch?v=I_g4IMG-JfI



Интересный эксперимент по коллективному творчеству задумал один пользователь Sketchfab из Испании: он смоделировал в Blender дирижабль и предлагает всем желающим скачать и дополнить его работу чем угодно, а затем вновь выложить на сайт с тегом multiverseproject. Идею уже подхватили – будем следить и посмотрим, что получится в итоге!

Скачать оригинальный blend-файл можно **здесь**. Там же, в комментариях – ссылки на модифицированные варианты.

К игровым новостям. Как недавно выяснилось, Blender был использован при разработке игры Factorio – в частости, для рендеринга спрайтов. Factorio – это новая разновидность стратегии/симулятора, игра, в которой вы выступаете в роли руководителя завода.

https://www.factorio.com

Еще один необычный сим – это Poly Bridge, симулятор мостостроителя. В качестве движка она использует Unity, а все модели выполнены в Blender. Poly Bridge работает под управлением Windows, Linux и OSX, доступна в Steam.

http://store.steampowered.com/app/367450

Также недавно вышла мобильная игра, при создании которой использовался Blender – «À Toca do Tatu». Это игра в жанре «шарик в лабиринте», где вы управляете броненосцем, катящимся по уровню. Проект примечателен тем, что абсолютно вся графика – модели, анимация, текстуры и даже элементы интерфейса – были выполнены в Blender. Скачать игру можно в **Google Play**.

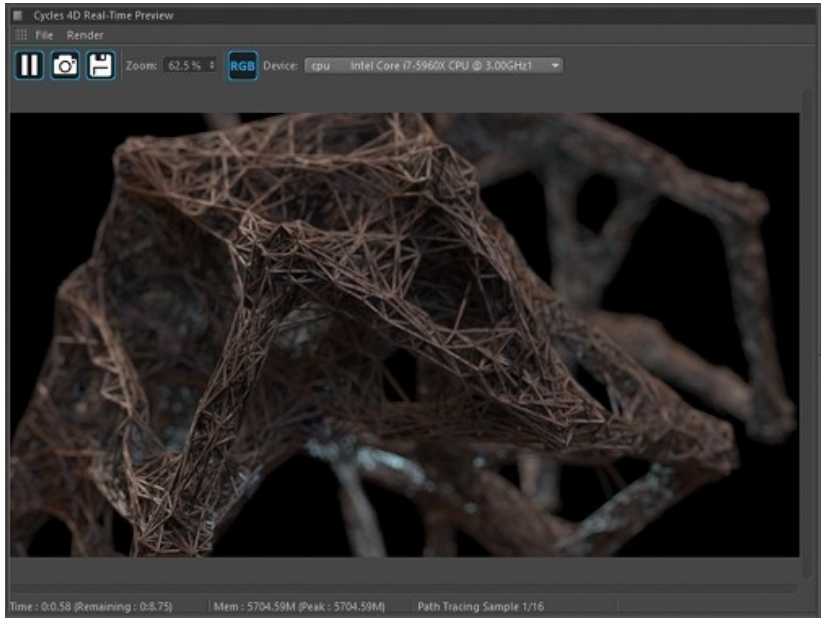
В разработке Titan Brawl, новой MOBA для iOS и Android, также был использован Blender – в частности, для моделирования персонажей.

http://omnidrone.net/titanbrawl

Blend4Web, открытый фреймворк для создания браузерных 3D-приложений в Blender, обновился до версии 16.08. Из главных нововведений можно отметить поддержку рендеринга анизотропных поверхностей, улучшенный редактор нормалей, а также существенные JavaScript-оптимизации.

https://www.blend4web.com

В одном из предыдущих номеров журнала мы писали о том, что рендер-движок Cycles вошел в состав новой версии Poser, а до этого о попытке интегрировать Cycles сообщали разработчики Rhino. И недавно стало известно, что компания Insydium занимается разработкой плагина, позволяющего использовать Cycles в Cinema4D, одном из самых популярных коммерческих 3D-пакетов. Более того: существует аналогичный проект для 3DS Max – **https://cyclesformax.net**.



RoonSehv: NeTerra

Новый квест во вселенной Myst

В «FPS» №39 '15 мы публиковали интервью с разработчиком игры Haven Moon – 3D-квеста, созданного в лучших традициях легендарной серии Myst. Интерес к этому жанру в последние годы переживает небывалый подъем: в том же 2015 году состоялся релиз одной из лучших фанатских игр по мотивам Myst – RoonSehv: NeTerra. Для нас этот проект примечателен также и тем, что все модели для него были созданы в Blender.

Myst – это уникальное явление в геймдеве. Уже первая часть серии, вышедшая в 1993 году, поразила всех проработанностью графики, уникальным дизайном и атмосферой, а о продолжениях и говорить нечего – Myst стала одной из самых популярных игровых серий 90-х и долгое время оставалась самой продаваемой игрой вообще (побить ее рекорд удалось только Sims). Именно Myst, за короткие сроки ставшая эталоном своего жанра, возвела компьютерные игры в ранг искусства.

Игровой мир серии детально проработан – здесь есть своя мифология, история, легенды. Вселенная Myst – это множество миров в пространстве-времени, называемых «эпохами», между которыми можно путешествовать при помощи книг-порталов, созданных магом-изобретателем по имени Атрус. Сюжет построен вокруг следов таинственной цивилизации Д'ни, когда-то в древности обитавшей под землей. Д'ни обладали так называемым Искусством – способностью перемещаться в параллельные миры. Атрус, в чьих жилах текла кровь Д'ни, тоже обладал Искусством – он стал одной из ключевых фигур в возрождении наследия Д'ни, автором множества книг-порталов и главой семьи, в делах которой участвует главный герой серии, безымянный персонаж, обозначенный просто как Странник.

Первая и вторая части Myst представляли собой квест из статических пререндеренных картинок – для того, чтобы повернуться, нужно было щелкнуть курсором у края экрана. Последующие же части привнесли поддержку 360-градусного обзора, а потом и полноценное 3D со свободным перемещением по виртуальному миру. Такова и игра, о которой пойдет речь в данном обзоре.

Разработка RoonSehv: NeTerra началась еще в 2004 году. Авторы игры – французская независимая студия Babel Studio. Изначально ребята планировали сделать классический пререндеренный квест в стиле ранних частей Myst, однако в 2013 году проект был переделан практически с нуля с использованием Unreal Development Kit – так игра получила современный 3D-движок и стала полностью интерактивной. По словам геймдизайнера Дени Мартина, UDK многократно ускорил разработку, и на завершение проекта ушли считанные месяцы. В результате мы имеем настоящий инди-шедевр – отличный подарок для любого поклонника Myst, да и жанра квестов в целом!

RoonSehv – это фанатский спин-офф Myst: события игры разворачиваются в одноименной эпохе, где в уединенном подземном доме в каньоне посреди пустыни жили потомки создателя этого мира. К моменту появления главного героя дом пустует – но кажется, что его последние обитатели, сестры Амелия, Лея и Флора, покинули свое жилище только вчера. Вам предстоит исследовать это загадочное подземелье, наполненное технологиями Д'ни, причудливыми механизмами, тайниками и секретными коридорами...

Игра начинается с короткой «обучалки» – в, которой, впрочем, никто особо не нуждается: управление в RoonSehv очень простое и интуитивное. Это квест от первого лица, где игрок взаимодействует с предметами и триггерами, кликая по ним мышью. Причем, левая и правая кнопки иногда выполняют разные действия: например, левая кнопка может опускать рычаг, а правая – поднимать его. Понять, какой предмет интерактивен, вам поможет курсор – при наведении на интерактивный объект он начинает мигать.

Конечно, трудно писать рецензию на квест и при этом не спойлерить – в играх, подобных Myst, любая подсказка касательно сюжета или механики может нарушить атмосферу таинственности, столь дорогую сердцу поклонника жанра.

Поэтому я не буду сообщать никаких подробностей о головоломках – достаточно сказать, что они весьма сложные и зачастую требуют знаний вселенной Myst. Например, потребуется знать цифры Д'ни – стоит заранее запастись соответствующей таблицей. Обязательно держите при себе бумагу и карандаш – потребуется что-то записать или зарисовать, провести несложные математические операции. Игровой мир не очень большой: если знать алгоритм решения всех головоломок, полное прохождение займет не более часа, однако в первый раз (если, конечно, не пользоваться подсказками и готовыми прохождениями) RoonSehv отнимет у вас не один день – а может, и не одну неделю.

Вы можете в любой момент сохраниться – доступны три сейв-слота. К сожалению, слоты никак не помечаются как занятые – есть риск забыть, куда именно вы сохранились.

Я так однажды затер важный сейв и потерял свое прохождение – к счастью, решить все головломки повторно не составило большого труда, но настроение все-таки было немного подпорчено...

Впрочем, мелкие технические недоработки меркнут на фоне дизайна игры в целом. Стилистика явно отдает дань Myst V и Uru, но технологически игра стоит поколением выше. Вы как будто попадаете в оживший мир классического Myst: то, что в 90-е было возможно лишь с использованием офф-флайн-рендеринга, теперь успешно рисуется в реальном времени. Динамические тени, вода с отражением и преломлением, крепускулярные лучи света... Игра вполне тянет на продукт коммерческого уровня, хотя и является бесплатной.

Картинка радует глаз не только спецэффектами, но и первоклассным дизайном уровней: оазис посреди пустыни ведет в мрачные подземные коридоры, а те перемежаются с интерьерами в стиле модерн – комнаты уставлены причудливой мебелью и освещены необычными лампами. Искусственные сооружения органично переплетаются с камнем и растениями. Многие элементы игрового мира подвижны, а некоторые объекты можно подбирать и носить с собой.

Звук и музыка всегда были слабым местом квестов – наверное, потому что они не играют особой роли в механике жанра, характерной неторопливым и вдумчивым геймплеем. Однако фоновая музыка и звуки окружения составляют важный элемент атмосферы Myst, и разработчики RoonSehv также уделили им особое внимание.

В игре нет навязчивых мелодий, на фоне играет размеренный эмбиент, отлично вписывающийся в атмосферу игры. Музыка меняется в зависимости от помещения, в котором вы находитесь. В игре есть, кстати, музыкальная головоломка, а также несколько музыкальных инструментов, которые тоже интерактивны – можно послушать звуки органа, патефона, китайского эрху, там-тамов и флейты.

Игру можно бесплатно скачать в каталоге GameJolt: **http://gamejolt.com/games/roonsehv-neterra/157413**. Доступна только версия для Windows, зато поддерживается несколько языков – английский, французский, итальянский и немецкий.

Сейчас, кстати, авторы работают над продолжением своего шедевра – RoonSehv: Experimentaliis. В качестве движка используется Unreal Engine 4 – сиквел будет еще круче в плане графики. А контент создается при помощи все того же Blender, что не может не радовать!

http://roonsehv.blogspot.ru

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.

Blender для начинающих

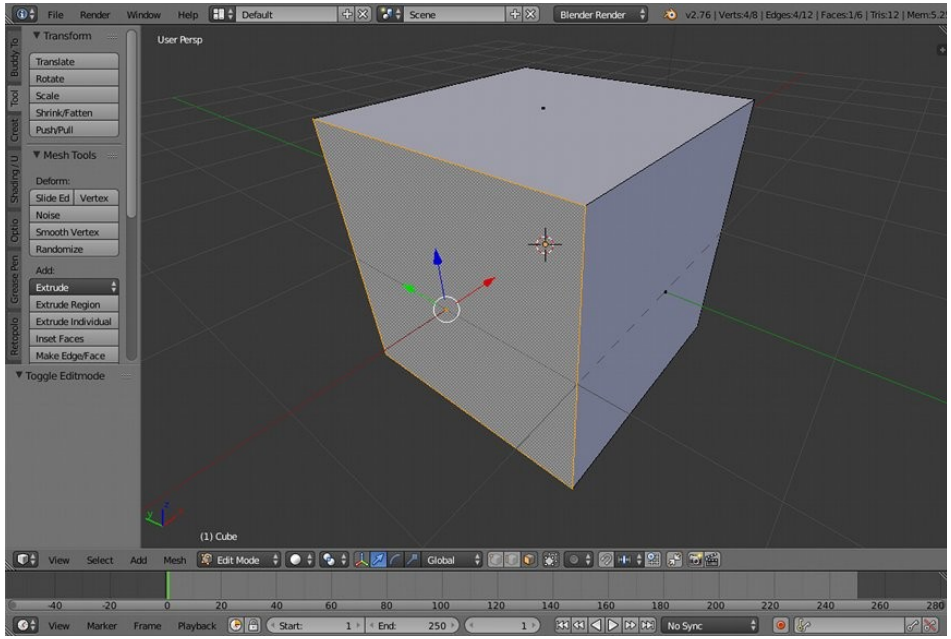
Часть 2. Редактирование мешей

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. Первую статью цикла читайте в «FPS» №43 '16.

Примитивы, которые мы можем добавлять на сцену, представляют собой меши (Mesh) – полигональные сетки, совокупности вершин, ребер и граней, составляющие форму объекта. Вершина (Vertex) – это, по сути, точка в пространстве. Ребро (Edge) – соединение между двумя вершинами. Грань (Face) – замкнутое множество ребер, в котором каждое ребро смежно с другим. В Blender поддерживаются грани, состоящие из трех и более вершин.

Вы можете просматривать и править эту структуру объекта, выделив его и нажав клавишу Tab. Эта клавиша переключает из объектного режима (Object Mode), в котором вы можете выделять и трансформировать объекты, в режим редактирования (Edit Mode). В режиме редактирования можно выделять вершины, ребра или грани правой кнопкой мыши. Переключаться между режимами выделения вершин, ребер и граней можно на уже знакомой нам нижней панели, при помощи трех кнопок с соответствующими пиктограммами.

У выделенного элемента будет отображаться такой же манипулятор с тремя осями, как и в объектном режиме – элементы можно точно так же перемещать, поворачивать и масштабировать с целью изменения геометрии объекта. Кроме того, можно удалять элементы и создавать новые, а также совершать множество других различных операций над полигональной сеткой. Весь этот процесс обычно называют моделированием полигональной сетки (Mesh Modelling).



Рассмотрим несколько распространенных операций с полигональной сеткой.

1. Выдавливание (Extrude). Если выделить грань (или группу граней) и нажать клавишу E, то можно как бы вытянуть грань из поверхности, создав новую выпуклую форму. Вместо клавиши можно воспользоваться соответствующей кнопкой на панели инструментов слева.

2. Разбивка (Sibdivide). Разбивает грань на четыре равные части, а ребро – на две.

3. Фаска (Bewel). Позволяет создать скос кромки для ребра или грани. Вызывается комбинацией клавиш Ctrl+B.

4. Нож (Knife). Вызывается клавишей K. Очень полезный инструмент – позволяет разрезать грани, создав на них новые ребра по направлению разрезания. Чтобы определить это направление, вы должны указать несколько точек подряд левой кнопкой мыши. Чтобы применить операцию разрезания, нажмите клавишу Enter.

5. Слияние (Merge). Позволяет слить две вершины в одну. Вызывается комбинацией клавиш Alt+M. При этом всплывает меню, в котором необходимо выбрать направление слияния: At First (к первой выделенной вершине), At Last (к последней), At Center (к геометрическому центру выделенных вершин).

Продолжение следует...

Уважаемые читатели!

Наш журнал регулярно выходит на протяжении 8 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!

RetopoFlow. Ретопология? Легко!

Ретопология – это изменение топологической сетки модели с сохранением ее формы. Как правило, она применяется на высокополигональных мешах, полученных из программ лепки типа ZBrush, с тем, чтобы сократить количество полигонов до оптимального и более систематично расположить вершины и ребра. В идеале меш должен представлять собой равномерную сетку из прямоугольников – с такой моделью будет проще работать и аниматорам, и текстурщикам.

Ретопология – один из самых сложных и трудоемких процессов в 3D-моделировании. Полностью автоматических методов ретопологии, которые давали бы идеальный результат «по нажатию кнопки», не существует.

В Blender есть модификатор Decimate, который способен упростить меш, соединяя близлежащие соседние вершины в одну, но он выдает очень грубую топологию, малопригодную для качественной анимации и UV-развертки. Поэтому им лучше не пользоваться. Я советую обратить внимание на аддон RetopoFlow, который включает инструменты, облегчающие процесс ручной ретопологии.

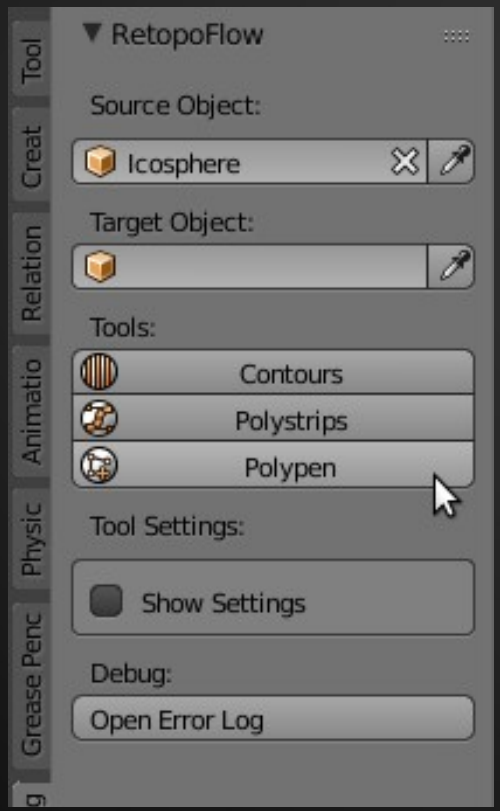
Мне лично особенно понравился в нем инструмент Polypen («полигональное перо») – с его помощью вы можете вручную «рисовать» новые вершины и полигоны на поверхности меша, с любой топологией, какую вы хотите. При этом вам не придется следить за тем, чтобы новая сетка совпадала по форме со старой – RetopoFlow делает это за вас.

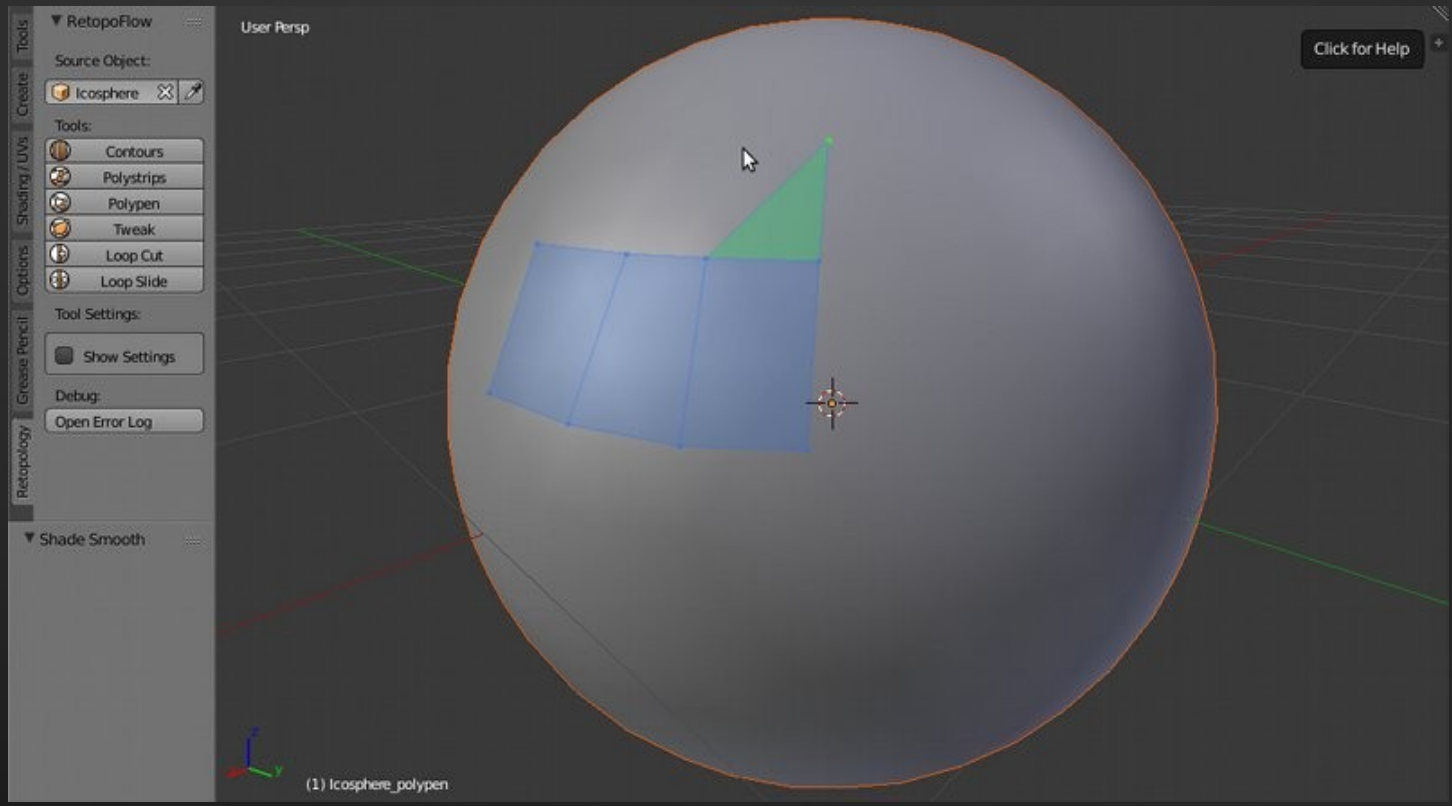
Процесс выглядит следующим образом. С установленным RetopoFlow на панели инструментов переключитесь на вкладку **Retopology**. В поле **Source Object** выберите свой высокополигональный меш. Я в целях демонстрации сделал высокополигональную сферу. Затем нажмите кнопку **Polypen**.

Вьюпорт перейдет в особый режим со своим управлением – остановимся на нем поподробнее. Зажав клавишу **Ctrl** и кликнув левой кнопкой мыши на модели, вы создадите на ее поверхности новую вершину. Правой кнопкой вы можете выделять вершины, ребра и полигоны (причем, RetopoFlow автоматически «угадывает», что вы хотите выделить – примерно как в Wings 3D). Если у вас уже выделена одна вершина, то новый клик создаст ребро. Если выделено ребро – получится треугольник. Если выделен треугольник – получится квад. Таким образом, вы можете создать сетку квадов (прямоугольников), просто кликая по модели.

Клавиша **A** снимет выделение. Клавишей **X** можно удалить выделенный элемент. Нажатие **Enter** применяет результат и выходит из режима Polypen – у вас на сцене появится новый объект с созданной вами топологией. Не лишним будет, кстати, включить для него режим **X-Ray**, чтобы он отрисовывался поверх высокополигональной модели.

Что особенно приятно: вы можете вновь нажать Polypen и вернуться к редактированию топологии. Либо вы можете переименовать объект с новой топологией, и RetopoFlow создаст еще один.

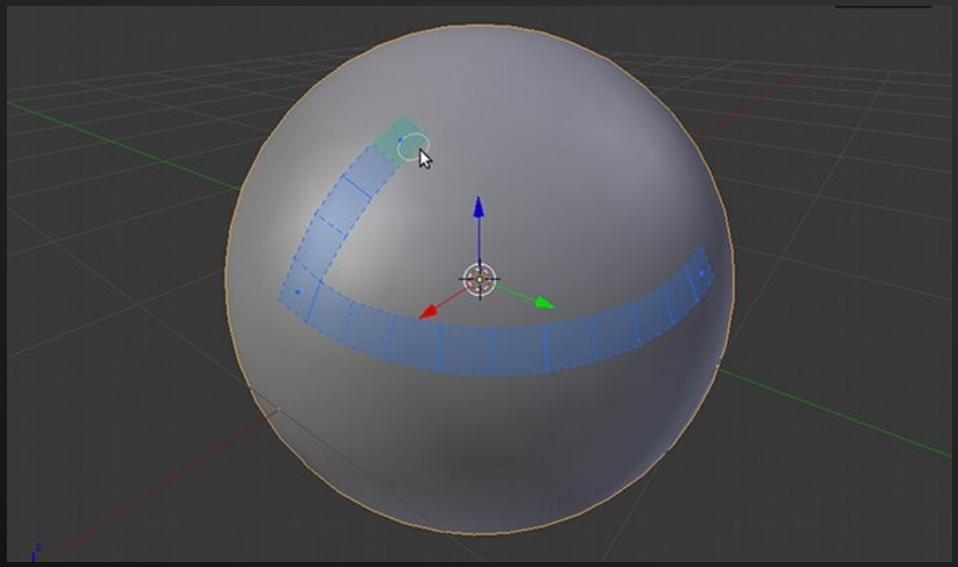




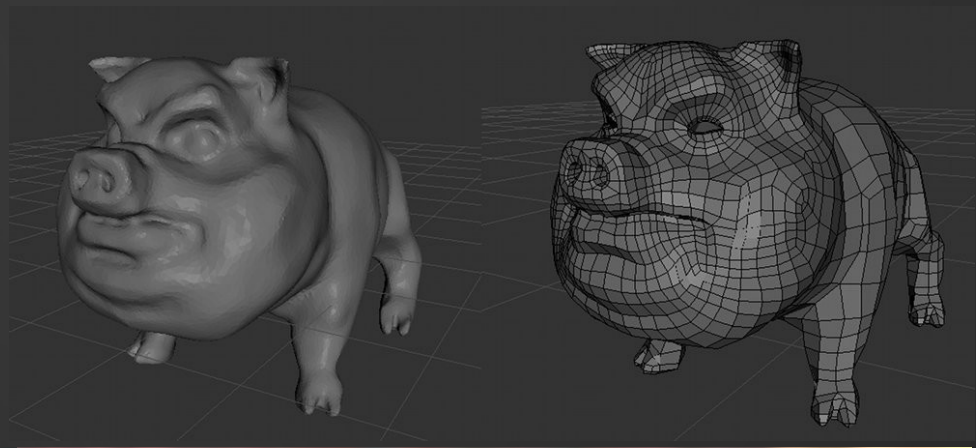
Это пригождается, когда объект становится слишком сложным (геометрия становится видна насквозь), и Polypen начинает из-за этого глючить – к примеру, неправильно ставит новые вершины. Вполне вероятно, что этот баг будет исправлен в будущем, а пока рекомендую составлять топологию несколькими небольшими патчами, которые затем можно соединить вручную в режиме редактирования меша.

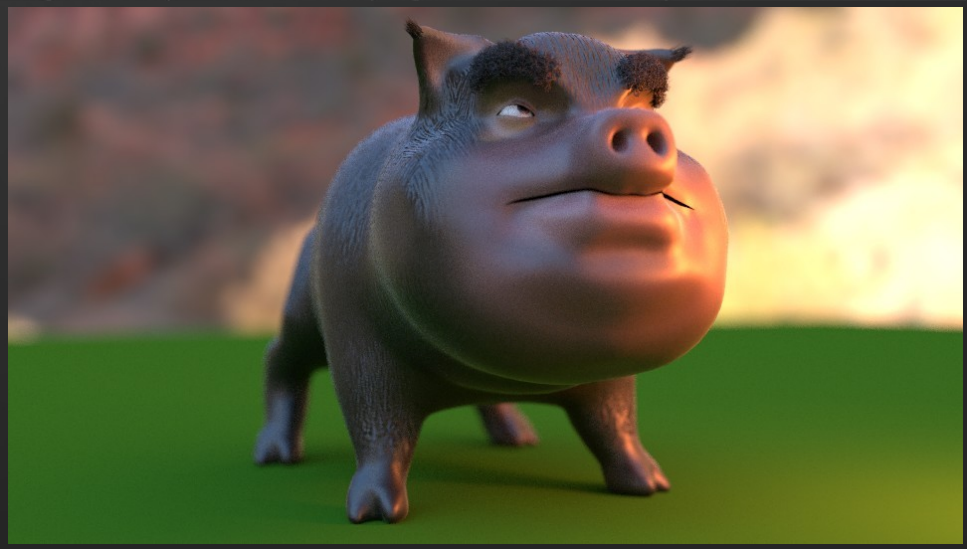
Другой полезнейший инструмент аддона – это **Polystrips**. С его помощью вы можете рисовать полосы ретопологии на поверхности модели.

А инструмент **Contours** автоматически создает контуры ретопологии вокруг меша, вдоль нарисованных вами линий. Эти два инструмента особенно хорошо себя показывают на длинных вытянутых моделях – например, на шеях, лапах или рогах животных.



В качестве демонстрации того, что можно сделать с помощью RetopoFlow, привожу свою недавнюю работу – ретопологию персонажа для анимационного проекта (высокополигональный вариант я смоделировал в Sculptris):





Скачать RetopoFlow можно на GitHub:
https://github.com/CGCookie/retopoflow

Тимур Гафаров

Blender. Рендеринг травы

Моделирование растений – одна из наиболее сложных, но и самых интересных задач в 3D-графике. Без реалистичных деревьев, кустов и травяного покрова невозможно представить себе ни одну природную сцену. И, если моделированию деревьев уделяют очень много внимания (существуют даже специализированные генераторы деревьев, работающие на фракталах и выдающие невероятно правдоподобные результаты), то трава зачастую отходит на второй план, хотя она, на мой взгляд, не менее важна.

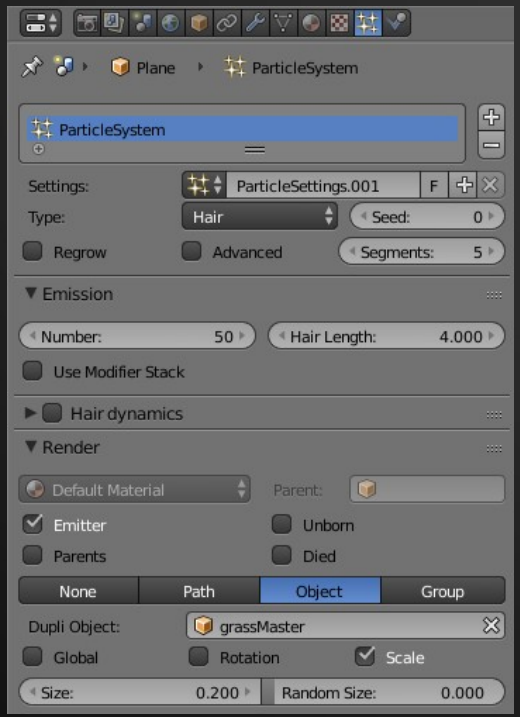
Есть два основных способа рендеринга травы. Первый, как правило, используется в графике реального времени: просто создаются плоские биллборды (прямоугольники, которые всегда ориентированы в сторону камеры) с натянутым на них изображением травинок с альфа-каналом. Получается очень дешево – дальние биллборды можно не рисовать, поэтому такой травой можно заполнить любые, даже самые огромные пространства.

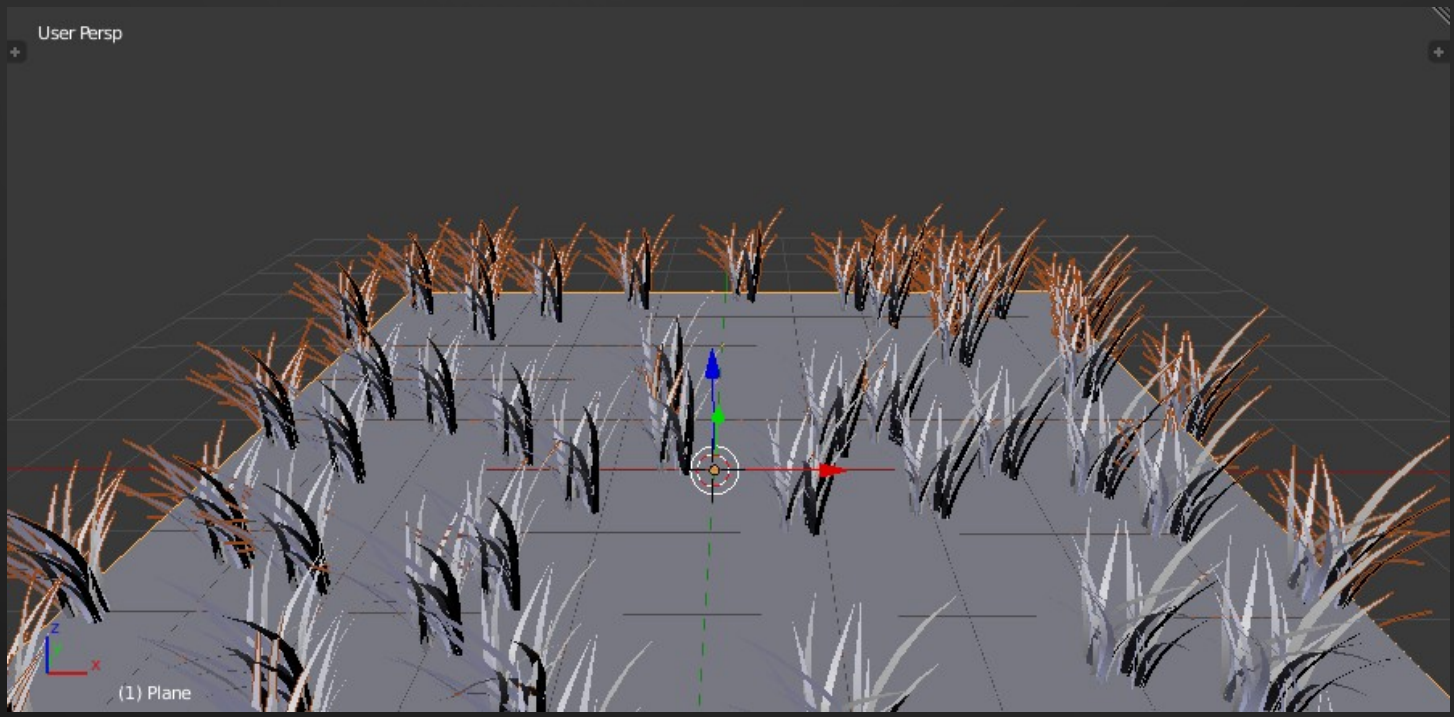
Но, естественно, у такого метода есть серьезный недостаток, который в большинстве случаев не позволяет использовать его для оффлайн-рендеринга – биллборды травы выглядят некорректно при виде сверху: если вы создаете анимационный фильм, то вряд ли вы ограничитесь обычными видами «из глаз» (хотя бывают и исключения). Поэтому приходится рендерить «честные» трехмерные травинки – этим мы и займемся на данном уроке. Если конкретно, то я продемонстрирую нехитрый метод «распыления» травинок по поверхности земли.

Модель травы выглядит тривиально – это просто три-четыре плоских вытянутых листочка, которые можно смоделировать за пять минут. Вместо отдельных травинок мы будем работать с небольшими пучками из нескольких растений – так будет гораздо проще заполнять пространство.

Создайте участок земли (сетку, скажем, 10x10) и добавьте ему систему частиц. Переключите ее тип на Hair. Во вкладке **Render** выберите **Object** и в качестве **Dupli Object** выберите модель пучка травы. Преимущество системы рендеринга волос в Blender заключается в том, что можно вместо волос использовать любые модели – так можно «распылять» не только траву, но и любые другие объекты, например, камни или какой-нибудь мусор.

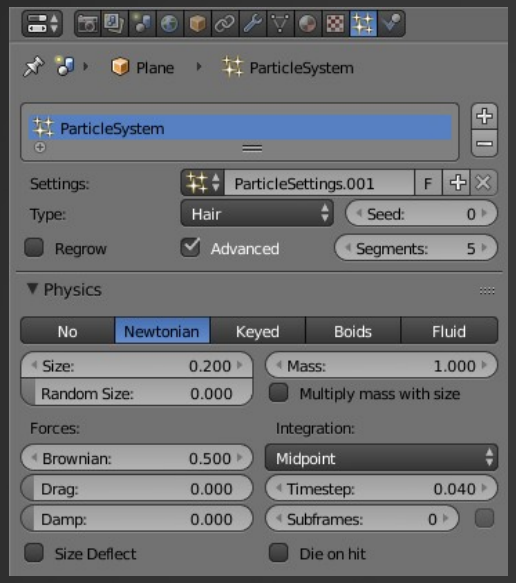
В системе частиц почему-то меняются местами оси Y и Z, поэтому клоны травы будут неправильно повернуты. Это нетрудно исправить, повернув главный пучок травы в режиме редактирования на -90 градусов по оси X.

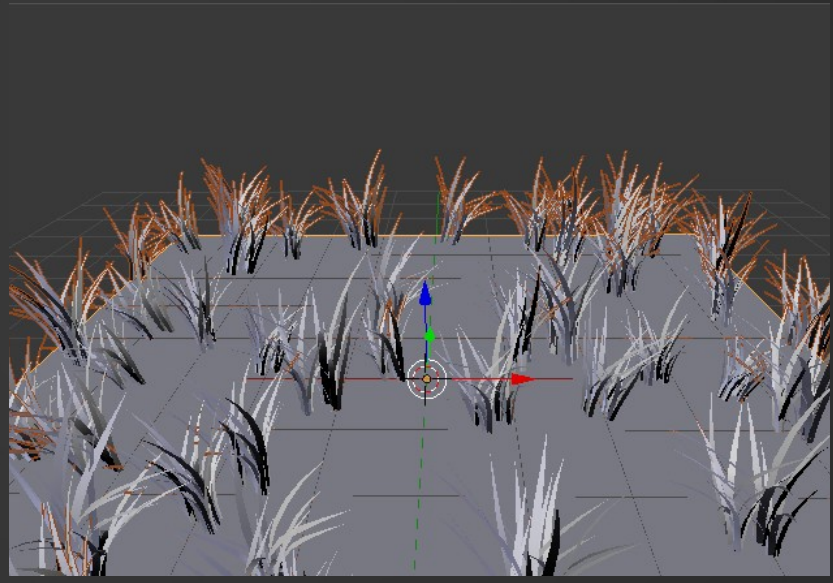


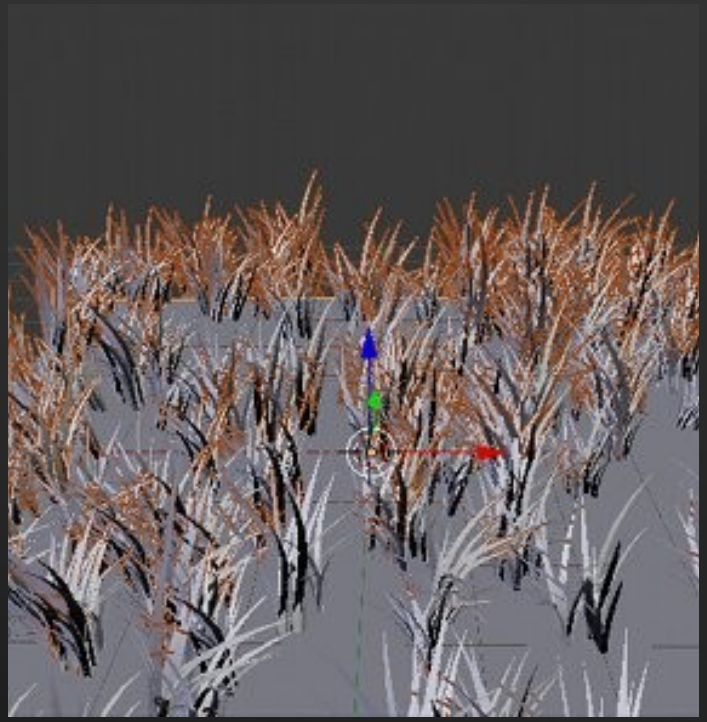


Не лишним будет добавить клонам немного случайности, чтобы трава не выглядела слишком однообразной. В настройках системы частиц поставьте галочку напротив **Advanced**, появится вкладка **Physics**.

В ней переключитесь на **Newtonian** и выставьте параметр **Brownian** равным 0.5. Это задаст небольшой случайный поворот для каждого клона.







У системы частиц есть еще одна невероятно полезная возможность – создание дочерних частиц. Если добавить каждому клону травы еще 10 дочерних в радиусе, скажем, 2, то «ковер» будет намного плотнее. Во вкладке **Children** выберите **Simple**, устновите желаемое количество дочерних частиц в параметре **Render**. Можно также изменить радиус, в пределах которого будут появляться дочерние частицы вокруг родительских.

Тимур Гафаров

Обзор дополнений Blender

Выпуск 22

Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. В этом выпуске мы представляем дополнения, связанные с освещением, материалами и рендерингом. 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****.*

Pro-Lighting Studio

Аддон, который автоматизирует процесс освещения сцены – можно выбрать пресет освещения из внушительного списка и быстро получить качественный результат. Дополнение платное, цена – $97.

Разработчик: Blender Guru
http://www.blenderguru.com/product/pro-lighting-studio

BakeTool 1.4

В предыдущих выпусках обзора мы уже писали об этом полезном дополнении: BakeTool – это набор инструментов, облегчающих работу с запеканием в Cycles (таких, как многоканальное запекание и быстрый доступ к некоторым настройкам). В новой версии BakeTool появился экспериментальный автоматический разворачиватель UV, а также обеспечена поддержка Blender 2.77. Аддон платный, цена – $14.95.

Разработчик: Vitor Balbio / Cogumelo Softworks
https://cgcookiemarkets.com/all-products/baketool

Zero Brush

Аддон, реализующий в Blender мощный и интуитивный пайплайн для раскрашивания моделей. Автоматически создает все материалы и другие необходимые данные для своей работы. Поддерживает как Blender Internal, так и Cycles. Дополнение платное, цена – $44.99.

Разработчик: Blender Sensei
https://blendersensei.com/zero-brush-1-5-cycles-bake-painting-is-here

Real Camera

Это дополнение реализует в Blender набор параметров, соответствующих реальной камере. Автоматически создает оптические блики (lens flare), хроматическую аберрацию и боке.

Разработчики: Marco Pavanello и Shaun Kreider
http://3dwolf.weebly.com/camera.html

Sprite Render Kit

Система рендеринга спрайтов из 3D-моделей, спроектированная специально для работы с анимацией. Позволяет в один клик отрендерить анимированную модель с 4 или 8 различных углов (спереди, сзади, сбоку, по диагонали и т.д). Очень полезный инструмент для разработчиков 2D-игр. Аддон платный, цена – $5.99.

Разработчик: Todor Imreorov
https://blurymind.itch.io/sprite-render-kit-for-blender-3d

Auto Drawing Tool

Комбинируя модификатор Build и NPR-рендер Freestyle, этот аддон реализует интересный эффект – анимацию рисования. Может пригодиться для создания промо-роликов и рекламной продукции.

Разработчик: squarednob
https://github.com/squarednob/auto-drawing-tool

Новинки 3D-софта

Appleseed

Appleseed – это новый свободный физический обоснованный рендер-движок, предназначенный для создания фотореалистичных изображений, анимации и VFX. Поддерживает стохастическую трассировку лучей, однонаправленную трассировку пути, многопроходный рендеринг, включает множество различных BRDF, включая GGX и Disney, а также подповерхностное рассеивание, DoF, размытие при движении, шейдеры на языке OSL и многое другое.

appleseed
open source production rendering

Движок имеет standalone-версию – appleseed.studio, также в настоящее время ведется разработка плагинов интеграции движка в 3ds Max, Maya и Blender. Appleseed работает под управлением Windows, Linux и OSX, исходники распространяются по лицензии MIT.

http://appleseedhq.net

Hydra Renderer 2.1f

Hydra Renderer, новый высокопроизводительный рендер-движок для 3ds Max, недавно обновился до версии 2.1f. Релиз включает поддержку нового алгоритма денойзинга на основе GNLM, конвертер материалов, а также множество исправлений ошибок.

Напомним, Hydra Renderer – это проект российских разработчиков, развиваемый при поддержке Лаборатории компьютерной графики и мультимедиа факультета ВМК МГУ. В данный момент проект находится в открытом бета-тестировании, принять участие в котором могут все желающие – движок распространяется бесплатно.

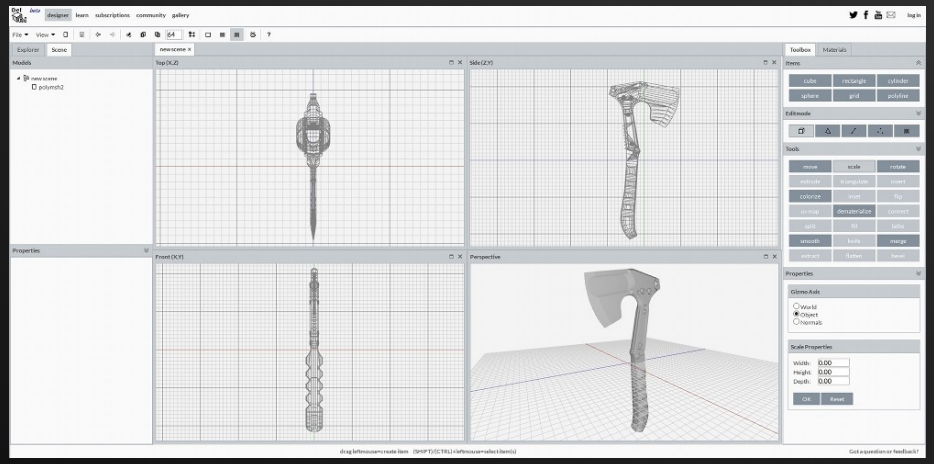
Сайт проекта: **http://abau.org/dilay/index.html**

DeleD Designer

Разрабатывается веб-версия популярного 3D-редактора DeleD – проект уже находится на стадии бета-версии, испытать которую могут все желающие. Редактор работает в браузере, на основе WebGL, предоставляя интерфейс, полностью аналогичный десктопному DeleD. Поддерживаются все основные операции с геометрическими примитивами, есть импорт и экспорт формата OBJ, а также сохранение сцен на сервере (требует регистрации).

Напомним, DeleD – это пакет для 3D-моделирования и создания игровых уровней. Будучи изначально коммерческим продуктом, в 2010 году DeleD стал бесплатным и перешел в разряд Open Source (распространяется под GPL) и с тех пор развивается сообществом.

https://beta.deleddesigner.com/#!Designer
http://www.delgine.com



Памятка читателю

В Интернете часто можно встретить вопросы о том, где скачать старые номера нашего журнала. Отвечаем. Архив всех номеров «FPS» (с 2008 по 2015 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=0B1BlzRb1uMv-bnpHNDhwZTI4eHc

В электронном издательстве **Issuu.com**:

http://issuu.com/tgafaroff/docs

Для тех, кто предпочитает скачивать с торрентов – журнал также есть на **РуТрекере**:

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

2D-графика: новости

Krita 3.0.1

Состоялся выход новой версии свободного графического редактора Krita 3.0.1. Релиз содержит все улучшения, сделанные в рамках GSoC 2016, а также появившиеся в результате фандрейзинга на KickStarter.

Это новая всплывающая палитра инструментов, предпросмотр цветовых потерь при конвертации в CMYK, улучшенный инструмент зеркального рисования, новые плагины – пороговый фильтр и вейвлет-декомпозитор, новая панель гистограммы и многое другое.

Начиная с выпуска 3.0.1, авторы Krita перешли на 6-недельный цикл разработки и теперь планируют регулярно выпускать новые релизы, предлагающие пользователям не только багфиксы, но и новые возможности.

Доступны для скачивания сборки Krita 3.0 для Windows, Linux и OSX.

https://krita.org

OpenShot 2.1

За последние месяцы обновились сразу несколько свободных программ видеомонтажа. Так, вышла новая версия OpenShot – 2.1. Релиз привносит расширенные инструменты для работы со звуком, новый интерфейс изменения свойств клипов, возможность переопределения «горячих клавиш», интерактивное руководство и многое другое.

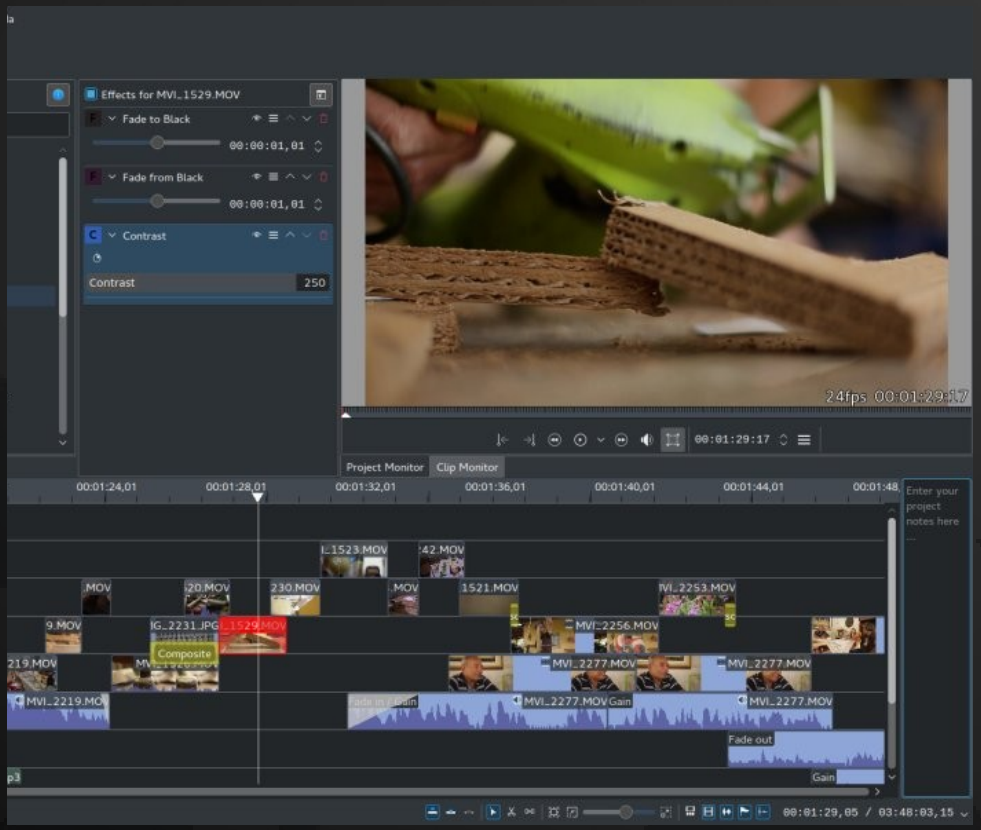
http://www.openshot.org

Avidemux 2.6.13

Также увидел свет Avidemux 2.6.13, в котором была улучшена поддержка форматов h.264 и h.265, добавлена поддержка аудиокодека AAC и шейдеров на GLSL для фильтров.

Напомним, Avidemux – это минималистичный кросс-платформенный видеоредактор, рассчитанный на решение простых задач по нарезке видео, применению фильтров и кодированию.

http://avidemux.berlios.de



Pitivi 0.97

Видеоредактор Pitivi 0.97 позиционируется как очередной выпуск на пути к первой стабильной версии. В новой версии улучшен диалог рендеринга, осуществлен переход на Gtk 3.20, GStreamer 1.8.2 и gst-transcoder 1.8.2.

Напомним, Pitivi – это программа видеомонтажа, написанная на Python с использованием фреймворка GStreamer. Pitivi поддерживает все типовые операции обработки видео и звука, неограниченное количество слоев, сохранение полной истории операций и отображение эскизов на шкале времени.

http://www.pitivi.org

Kdenlive 16.08

В Kdenlive 16.08 появился режим вставки на временной шкале, а также возможность предпросмотра эффектов на лету для произвольных областей шкалы. Реализован импорт изображений из Krita.

Напомним, Kdenlive (KDE Non-Linear Video Editor) – это свободная программа для нелинейного видеомонтажа, основанная на фреймворке MLT. Благодаря библиотеке ffmpeg, редактор поддерживает экспорт и импорт во все видеоформаты.

https://kdenlive.org

Моревна

Наконец-то вышел свободный анимационный фильм «Моревна», снятый российскими авторами в стиле аниме с сюжетом по мотивам русских народных сказок. В процессе производства фильма было использовано исключительно свободное ПО – в частности, редактор 2D-анимации Synfig.

https://morevnaproject.org

Fusion 8.2

Компания Blackmagic Design разместила в бесплатном доступе linux-версию своей знаменитой программы композитинга Fusion 8.2. Fusion развивается последние 25 лет и применялся для наложения VFX во многих известных голливудских фильмах, включая «Мстители», «Тор», «Человек-паук», «Голодные игры», «Марсианин» и «Гравитация». Пакет включает огромное количество инструментов для сведения видео, рисования, ротоскопирования, добавления 3D-частиц, наложения титров и анимации. Есть поддержка ускорения операций на GPU, а также импорт данных из Maya, 3ds Max и Cinema 4D.

Бесплатная версия, впрочем, имеет некоторые ограничения: так, отсутствует поддержка стереоскопического 3D-видео, стабилизации, плагинов OpenFX, средств для организации совместной работы и контроля версий.

https://www.blackmagicdesign.com/ru/products/fusion

Компьютерное зрение от Facebook

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.

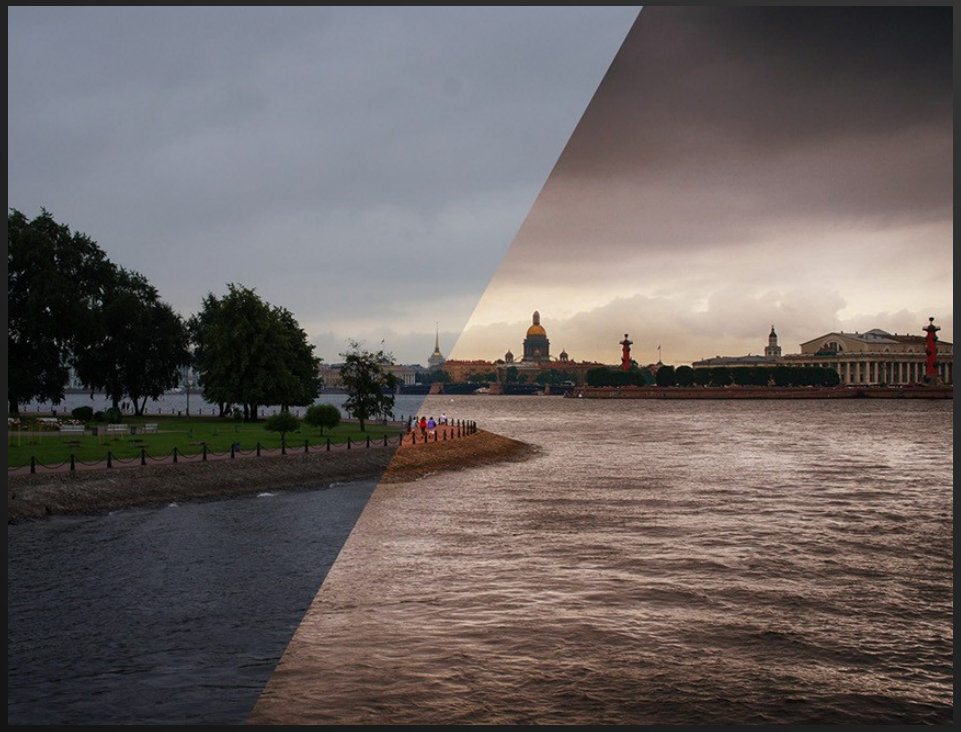
https://github.com/facebookresearch/deepmask

Nik Collection для Photoshop

В марте этого года компания Google выложила в бесплатный доступ семейство плагинов Nik Collection, который ранее разрабатывался немецкой фирмой Nik Software и стоил $500 – теперь воспользоваться богатыми функциями этого набора для профессиональной обработки фотографий может любой желающий.

Nik Collection включает семь плагинов: Analog Efex Pro, Color Efex Pro, Silver Efex Pro, Viveza, HDR Efex Pro, Sharpener Pro и Dfine. Все они совместимы с Photoshop и Lightroom от Adobe, а также с Aperture от Apple.

https://www.google.ru/intl/ru/nikcollection



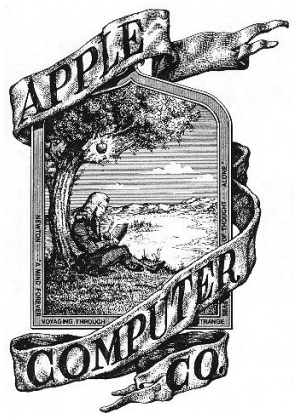
- **Analog Efex Pro** – инструменты для имитации пленочной фотографии
- **Color Efex Pro** – плагин для цветокоррекции и применения художественных фильтров
- **Silver Efex Pro** – плагин для «умного» обесцвечивания, создания качественных черно-белых снимков
- **Viveza** – выборочная цветокоррекция
- **Sharpener Pro** – повышение резкости
- **Dfine** – подавление шума.
- **HDR Efex Pro** – на наш взгляд, один из самых интересных компонентов Nik Collection. Это инструмент для применения псевдо-HDR – то есть, он позволяет создавать HDR-фотографии из одного изображения, без необходимости сведения нескольких RAW-снимков с разной экспозицией. Конечно, «честный» HDR он вряд ли заменит, но быстро повысить локальный контраст и высветлить тени с сохранением цветовой гаммы – это всегда пожалуйста. Плагин особенно хорошо себя показывает с пейзажами и интерьерами.

ИСТОРИЯ
IT-логотипов

Первый логотип Apple был создан в 1976 году Рональдом Уэйном, которого часто называют третьим сооснователем компании. Эта эмблема выглядит как некий герб, изображающий Исаака Ньютона под деревом, с которого вот-вот сорвется яблоко. Надпись на рамке гравюры гласит: «Ньютон. Разум, вечно странствующий по дивным морям мыслей в одиночку».

Логотип с Ньютоном просуществовал недолго: в том же 1976 году Стив Джобс нанял графического дизайнера Роба Джейноффа, чтобы тот сделал что-нибудь посовременнее. И работа Джейноффа стала одним из самых узнаваемых корпоративных логотипов в истории – именно он создал знаменитое «надкушенное яблоко».

Изначально оно состояло из горизонтальных полос радужных цветов – это должно было подчеркивать интернациональность и гуманизм. Такой вид логотип сохранял аж 22 года, и был пересмотрен только после возвращения Джобса в Apple в 1997 году. Форма яблока осталась прежней, а цвет стал регулярно меняться от одного продукта к другому. Объединяет их монохроматизм – однотонный цвет оказался весьма подходящим для размещения логотипа на различных поверхностях.



1976

1976

1998

2001

2007

В отличие от Apple с их «зоопарком» устройств и операционок, лицом Microsoft уже много лет является один продукт – Windows. И, соответственно, логотип Windows прочно ассоциируется с самой компанией.

Центральным элементом эмблемы Windows с самого начала был флаг, поделенный на 4 равные секции, символизирующие окна. Секции отделяет друг от друга фигура в виде креста. Впрочем, на первом логотипе Windows 1.0, созданном в 1985 году, секции были разных размеров, и на крестообразном контуре акцент еще не делался.

В Windows 3.1 (1992 г.) появился привычный нам логотип с разноцветными «окошками» – красным, синим, зеленым и желтым. Эти цвета отсылают к природным стихиям – огню, воде, флоре и фауне.

Очень похожий логотип у Windows 95 (1995 г.) – только теперь флаг стал наклоненным вправо. Достаточно серьезно логотип изменился в Windows XP (2001 г.): исчез черный контур, появилась объемная светотень, «окошки» приобрели толщину. Модуль логотипа из прямоугольного стал квадратным.

В Windows Vista (2007 г.) логотип изменился незначительно – добавилось круглое синее поле на фоне, а также модный в те годы глянцевый блеск. В Windows 7 (2009 г.) круглый фон убрали, а также убавили глянца и изменили подсветку.

А вот логотип Windows 8 (2012 г.), созданный Паулой Шер, выглядит уже по-другому – это уходящий в перспективу голубой прямоугольник, разделенный на 4 секции. Новый дизайн логотипа должен был отражать тайловый интерфейс Metro, появившийся в этой версии Windows.

1985. Windows 1.0

1992. Windows 3.1

1995. Windows 95

2001. Windows XP

2007. Windows Vista

2012. Windows 8

Будучи относительно молодой компанией, Google еще не успела значительно поменять свой логотип – с самого начала это было слово «Google», написанное разноцветными буквами. Название компании – это измененное слово «гугол», обозначающее 10 в степени 100.

Первый логотип был создан в 1997 году сооснователем Google Сергеем Брином – он, кстати, использовал для этого редактор GIMP. Изначально логотип использовал полужирное начертание, а буквы были трехмерными – 3D-текст был очень модным в 90-х. В 1998 году логотип стал плоским (хотя буквы все равно долгое время имели светотень) и приобрел более близкий к современному вид. В 1999 году шрифт стал тоньше, у логотипа появилась выраженная падающая тень – этот вариант использовался компанией 10 лет, после чего тень решили убрать. В 2013 году убрали и светотень, а в 2015 изменили шрифт – буквы стали более округлыми и минималистичными.

1997

1998

1998

1999

2010

2013

2015

У самого популярного свободного браузера была достаточно интересная молодость. Корни его уходят в Netscape, один из главных браузеров 90-х, конкурировавший с Internet Explorer. После ликвидации Netscape скупившей его AOL, его разработчики решили продолжить развивать браузер на основе сообщества, в виде свободного проекта. Изначально этот проект носил название Phoenix, но из-за правовых коллизий пришлось переименовать его в Firebird. Логотип браузера был соответствующий – он изображал огненную птицу-феникса. Но затем оказалось, что проект с таким названием тоже существует (СУБД FirebirdSQL), и поэтому пришлось переименовать проект еще раз – так родилась «огненная лиса», Firefox. Логотип, естественно, тоже переделали, и с тех пор он меняется только в незначительных деталях.

Логотип Firefox изображает рыжую лису, огибающую синий земной шар. В последующих версиях сферу сделали светлее и менее контрастной. В Firefox 3.5 хвост лисы стал объемным – языки пламени стали загибаться за сферу. В 2013 году на лисе стало меньше деталей, а Земля стала почти однотонной.

Любопытна, кстати, судьба Firefox на свободных ОС. Несмотря на то, что сам браузер свободный, его логотип является торговой маркой и не может использоваться третьими лицами без разрешения правообладателей. Поэтому мейнтейнеры дистрибутивов Linux не используют оригинальный логотип Firefox, заменяя его какой-нибудь другой эмблемой.

Phoenix / Firebird

2004

2005

2009

2013

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 2.071.2

Вышла бета-версия нового релиза референсного компилятора D – DMD 2.071.2. Это, в основном, исправляющий релиз – исправлено несколько важных багов в компиляторе, рантайме и Phobos.

http://dlang.org/download.html

LDC 1.1.0-beta2

Увидел свет второй 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

D + Asm.js

Отличные новости для веб-разработчиков: в рамках проекта dscripten развивается инструментарий для компиляции D-кода в Asm.js, низкоуровневое подмножество JavaScript, ориентированное на скорость выполнения. Компилятор основан на LDC и Emscripten — трансляторе биткода LLVM в JavaScript. Тулчейн доступен на GitHub, есть также небольшоая демонстрация браузерной игры на D.

https://github.com/Ace17/dscripten
http://code.alaiwan.org/dscripten/full.html
http://code.alaiwan.org/wp/?p=103

Ocean

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. Недавно он был полностью переведен в разряд Open Source.

Библиотека совместима с D1 и D2, но работает только под Linux. Отличительной особенностью Ocean является минимальное использование сборщика мусора.

https://github.com/jasonwhite/button

• Геймдев и мультимедиа

Video Enhancer

У D неожиданно появились приверженцы в области коммерческих мультимедийных приложений. Так, компания Infognition переписала на D свой флагманский продукт – видеоредактор Video Enhancer.

http://www.infognition.com/VideoEnhancer

plug 3.0.0

dplug, фреймворк для создания аудиоплагинов на D, обновился до версии 3.0.0. Релиз включает поддержку формата Audio Unity.
Кстати, компания Auburn Sounds выпустила уже два коммерческих VST-плагина, написанных на D при помощи dplug.

https://github.com/p0nce/dplug
https://www.auburnsounds.com/products/Panagement.html

Scone

Scone (Simple CONsole Engine) – это движок для разработки псевдографических консольных приложений. Поддерживает цветную псевдографику и пользовательский клавиатурный ввод, включает средства для создания UI и локализации. К сожалению, движок пока работает полноценно только под управлением Windows.

https://github.com/vladdeSV/scone

• Веб-разработка

vibe.d 0.7.30

Вышла новая версия веб-фреймворка Vibe.d. Релиз включает поддержку DMD 2.071.1, устраняет некоторую устаревшую функциональность и совместимость со старыми версиями DMD. Добавлена поддержка REST, улучшена подсистема HTTP и исправлены различные баги.

http://vibed.org

dlang-requests

Анонсирован проект dlang-requests – библиотека-клиент HTTP(S) и FTP, созданная по образцу Python-requests. Есть встроенная поддержка vibe.d.

https://github.com/ikod/dlang-requests

TinyRedis 2.1.0

Обновился D-драйвер к открытой документоориентированной СУБД Redis. Данный релиз вносит поддержку DMD 2.071.0, облегченные импорты, а также новый модуль tinyredis.collections.
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.

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

Тени в OpenGL
методом Shadow Mapping

Отрисовка теней в realtime-приложениях – наверное, одна из самых непростых задач, особенно для начинающих. К сожалению, растеризация не очень хорошо рассчитана на построение теней, чего не скажешь о трассировке лучей. А без теней, как сказал один мой коллега, и игра не игра. В этой статье я покажу, как реализовать на OpenGL метод теневых карт (shadow maps) – самую популярную на сегодняшний день технологию рендеринга динамических теней.

Метод, как ясно из названия, предполагает создание в видеопамяти особой текстуры, в которую будут записываться данные о тенях. В основе этого метода лежит идея о том, что освещенные точки – это те точки, которые видны из положения источника света. «Видимость» в данном случае означает, что данная точка успешно проходит тест глубины при рендеринге из положения источника света. Затененные точки, следовательно – это те, которые перекрываются освещенными.

Таким образом, shadow mapping работает в два прохода: сначала делается рендеринг из положения источника света – значения глубины объектов, которые должны отбрасывать тень, записываются в буфер. Затем делается обычный рендеринг, во время которого полученный буфер используется для проверки, попадает ли та или иная точка в тень – эта проверка осуществляется во фрагментном шейдере. Основная задача заключается в том, чтобы правильно перевести координаты точки в пространство отсечения буфера – для этого составляется специальная матрица.

В данном уроке я использую OpenGL 2.0, язык D и библиотеку dlib (для линейной алгебры).

Сначала создаем текстуру и FBO для рендеринга в буфер:

```
GLuint depthBuffer;
glGenTextures(1, &depthBuffer);
glBindTexture(GL_TEXTURE_2D, depthBuffer);
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Можно также задать цвет границ, чтобы пространство за пределами буфера не затенялось:

```
Color4f col = Color4f(1.0f, 1.0f, 1.0f, 1.0f);
glTexParameterfv(GL_TEXTURE_2D,
    GL_TEXTURE_BORDER_COLOR, col.arrayof.ptr);
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_WRAP_S, GL_CLAMP_TO_BORDER);
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_WRAP_T, GL_CLAMP_TO_BORDER);
```

Задаем настройки глубины:

```
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
glTexParameteri(GL_TEXTURE_2D,
    GL_TEXTURE_COMPARE_FUNC, GL_LEQUAL);
glTexParameteri(GL_TEXTURE_2D,
    GL_DEPTH_TEXTURE_MODE, GL_INTENSITY);
```

Создаем пустой блок данных под буфер:

```
glTexImage2D(GL_TEXTURE_2D, 0,
    GL_DEPTH_COMPONENT, 1024, 1024, 0,
    GL_DEPTH_COMPONENT, GL_UNSIGNED_BYTE, null);
glBindTexture(GL_TEXTURE_2D, 0);
```

Создаем FBO и привязываем к нему буфер глубины:

```
GLuint fbo;
glGenFramebuffers(1, &fbo);
glBindFramebuffer(GL_FRAMEBUFFER, fbo);
glDrawBuffer(GL_NONE);
glReadBuffer(GL_NONE);
glFramebufferTexture2D(GL_FRAMEBUFFER,
  GL_DEPTH_ATTACHMENT, GL_TEXTURE_2D, depthBuffer, 0);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

Теперь можно рендерить в буфер. Для это нужно построить матрицу ортогональной проекции, а также модельно-видовую матрицу, которая «смотрит» вдоль вектора направления источника света.

```
float phs = 10.0f;
Matrix4x4f projectionMatrix =
    orthoMatrix(-phs, phs, -phs, phs, -1.0f, 100.0f);
Vector3f lightPosition =
    Vector3f(0.0f, 0.0f, 0.0f);
Quaternionf lightRotation =
    rotationQuaternion(0, degtorad(-60.0f));

Matrix4x4f modelViewMatrix =
    lightRotation.conj.toMatrix4x4 *
    translationMatrix(-lightPosition);
```

Параметр phs задает полуразмер проекции – иными словами, с его помощью можно задать размер прямоугольной области, внутри которой рендерятся тени.

Не забудьте привязать lightPosition к вашей камере, чтобы тени всегда рендерились там, где они должны быть видны (в противном случае игрок может выйти за пределы проекции, где теней не будет).

Привязываем FBO, загружаем матрицы в стек и рендерим:

```
glBindFramebuffer(GL_FRAMEBUFFER, fbo);

glMatrixMode(GL_PROJECTION);
glLoadMatrixf(projectionMatrix.arrayof.ptr);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixf(modelViewMatrix.arrayof.ptr);

glShadeModel(GL_FLAT);
glColorMask(0, 0, 0, 0);
glCullFace(GL_FRONT);

foreach(obj; sceneObjects)
{
  obj.render();
}

glCullFace(GL_BACK);
glShadeModel(GL_SMOOTH);
glColorMask(1, 1, 1, 1);

glBindFramebuffer(GL_FRAMEBUFFER, 0);
```

Дальше начинается самое интересное. Перед тем, как рендерить объекты обычным образом, составляем матрицу, которая переводит точки из видового пространства камеры в пространство отсечения источника света. Чтобы не создавать лишний uniform, я решил передать ее в шейдер через последнюю текстурную матрицу OpenGL. туда же, в текстурный блок 7, я передал буфер глубины.

```
    glActiveTextureARB(GL_TEXTURE7);
    glMatrixMode(GL_TEXTURE);
    glLoadIdentity();
    glTranslatef(0.5f, 0.5f, 0.5f);
    glScalef(0.5f, 0.5f, 0.5f);
    glMultMatrixf(projectionMatrix.arrayof.ptr);
    glMultMatrixf(modelViewMatrix.arrayof.ptr);
    glMultMatrixf(invCameraMatrix.arrayof.ptr);
    glMatrixMode(GL_MODELVIEW);
    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, depthBuffer);
    glActiveTextureARB(GL_TEXTURE0_ARB);
```

Для экономии места я опущу код создания шейдера GLSL. Сам шейдер в упрощенном варианте выглядит следующим образом:

Вершинная программа:

```
varying vec4 shadowCoord;
void main()
{
  shadowCoord =
    gl_TextureMatrix[7] * (gl_ModelViewMatrix * gl_Vertex);
  gl_Position = ftransform();
}
```

Фрагментная программа:

```
varying vec4 shadowCoord;
uniform sampler2DShadow depthBuffer;
void main()
{
  float shadow = 1.0;
  if (shadowCoord.w > 0.0)
  {
    vec4 shiftedCoord =
      shadowCoord + vec4(0.0, 0.0, 0.001, 0.0);
    shadow = shadow2DProj(depthBuffer, shiftedCoord).z;
  }
  vec4 white = vec4(1.0, 1.0, 1.0, 1.0);
  gl_FragColor = white * shadow;
  gl_FragColor.a;
}
```

Шейдер выводит затененную область черным цветом, незатененную – белым. Я намеренно опустил код, отвечающий за текстуры и освещение, так как его очень легко добавить обратно.

Также обратите внимание, что я немного сдвигаю координаты по оси Z. Это нужно, чтобы бороться с так называемым shadow acne – неприятным муаровым узором, возникающим на некоторых гранях, когда не хватает точности буфера глубины.

Теневые карты очень эффективны, но они имеют и недостаток – артефакты дискретизации: иными словами, если не использовать гигантский буфер глубины, то тени получаются сильно пикселизированные. Этот эффект можно свести на нет, если во фрагметном шейдере отфильтровать выборку из буфера глубины ядром 3x3 или 4x4 – в результате чего получаются мягкие тени без пикселизации. Данное расширение метода теневых карт получило название PCF (Percentage Closer Filtering).

Вот как может выглядеть PCF с ядром 4x4 (для удобства я вывел чтение из буфера в отдельную функцию):

```
const vec2 shadowMapSize = vec2(1024.0, 1024.0);

float shadowLookup(vec4 coord, vec2 offset)
{
  vec2 texelSize = vec2(1.0) / shadowMapSize;
  vec2 v = offset * texelSize * coord.w;
  vec4 res = shadow2DProj(depthBuffer,
    coord + vec4(v.x, v.y, 0.001, 0.0));
  return res.z;
}
```

```
float shadow = 1.0;
if (shadowCoord.w > 0.0)
{
  shadow = 0.0;
  float x, y;
  const float size = 2.0;
  for (y = -size; y < size; y += 1.0)
  for (x = -size; x < size; x += 1.0)
    shadow += shadowLookup(dgl_Texture7,
      coord, vec2(x, y));
  shadow = shadow / (size * size * 4.0);
}
```

В константу shadowMapSize подставьте свой размер буфера глубины (который передавался в glTexImage2D). Лучше заменить константу на uniform и передавать это значение из приложения.

Скриншоты к этой статье были сделаны в движке **Xtreme3D**, в последних версиях которого появилась поддержка теней, реализованная описанным в статье методом.

Тимур Гафаров

Наши проекты

Cook

Программа автоматизации сборки проектов на языке D. В отличие от аналогичных инструментов, 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. Есть поддержка автоматического разрешения зависимостей из Git-репозиториев. Программа нетребовательна к ресурсам и работает даже на старых машинах с малым объемом памяти, на которых проблематично пользоваться DUB для сборки крупных проектов. Cook работает под управлением Windows и Linux.

http://github.com/gecko0307/cook2

dlib

Коллекция модулей «на все случаи жизни» для D с уклоном в сторону мультимедиа. Можно использовать ее для разработки игр и игровых движков, систем рендеринга, графических редакторов, а также научно-инженерных приложений. Библиотека идеально подходит в качестве полигона для различных экспериментов в области Computer Science, обкатки новых алгоритмов, решения математических задач, рисования графиков, разработки UI-тулкитов и т. д. С помощью dlib можно писать на D программы с полностью ручным управлением памятью, без использования сборщика мусора.

http://github.com/gecko0307/dlib

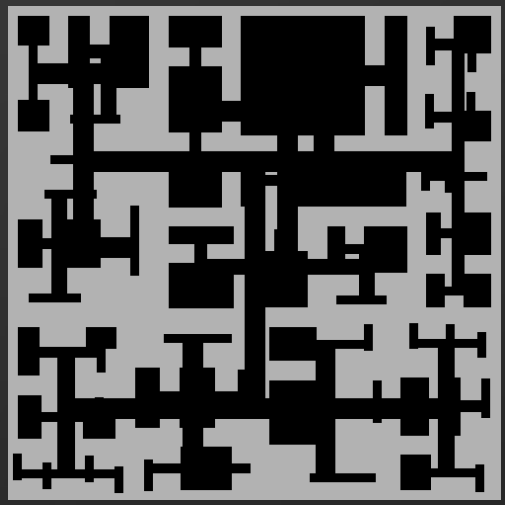
Генерируем случайные уровни

Некоторые игровые жанры совместимы с идеей процедурно генерируемого контента. Например, в разной степени элементы случайности используются в RPG – начиная от случайных характеристик персонажей и мобов и заканчивая полноценными случайными картами и radiant-квестами.

Особенно часто процедурные карты используются в «рогаликах» и играх типа dungeon crawl, представляющих собой лабиринтообразное подземелье, по которому путешествует игрок, уничтожая монстров и собирая предметы. Поэтому алгоритмы построения случайных карт, как правило, ориентированы на имитацию подземелий – то есть, за основу обычно берется изначально непустое пространство, в котором «вырезаются» интерьеры с комнатами и соединяющими их коридорами.

Сгенерировать по-настоящему интересное подземелье не так просто. В первую очередь, карта должна быть связной – то есть, должна быть возможность попасть из одной точки интерьера в любую другую. При этом желательно, чтобы возможных путей попасть из одной точки в другую было несколько – между комнатами должны быть циклические соединения.

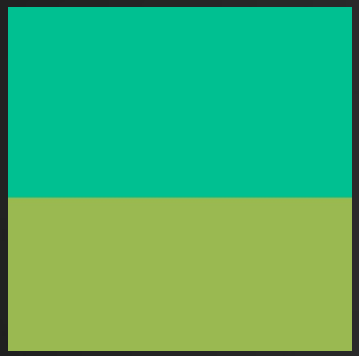
В зависимости от дизайна игры, к алгоритму генерирования могут выдвигаться и другие требования – наличие комнат разных форм и размеров, ветвлений, тупиковых коридоров и т.д.



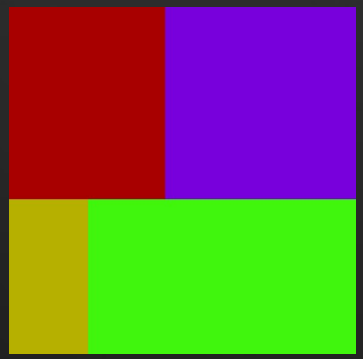
Существует несколько распространенных алгоритмов генерирования карт. Самый простой способ – расставить несколько комнат случайным образом, а затем соединить их по цепочке коридорами. Однако полученное таким образом подземелье будет линейным, и проходить его будет не очень интересно. На мой взгляд, один из самых «умных» методов – это алгоритм на основе BSP, который выдает весьма реалистичные подземелья, как на скриншоте выше.

Основная идея метода проста: строится двоичное дерево, каждый узел которого связан с разделяющей прямой или плоскостью в двумерном или трехмерном пространстве. Все объекты, лежащие с одной стороны от этой плоскости, связываются с одним поддеревом узла, а остальные – с другим.

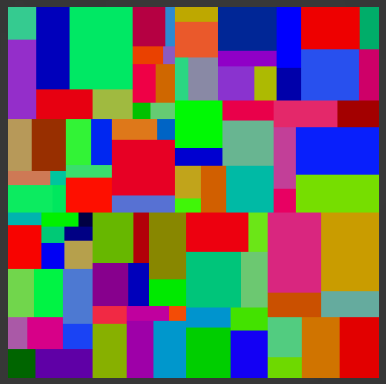
В нашем случае мы будем разбивать двумерное пространство, выделенное под будущую карту. Пусть это будет, например, квадрат 100x100. Сопоставляем этот квадрат с корневым узлом дерева.



Затем выбираем случайную разделяющую прямую – горизонтальную или вертикальную. Если разделить исходный квадрат этой прямой, получатся два прямоугольника – сопоставляем их с двумя дочерними узлами корневого.

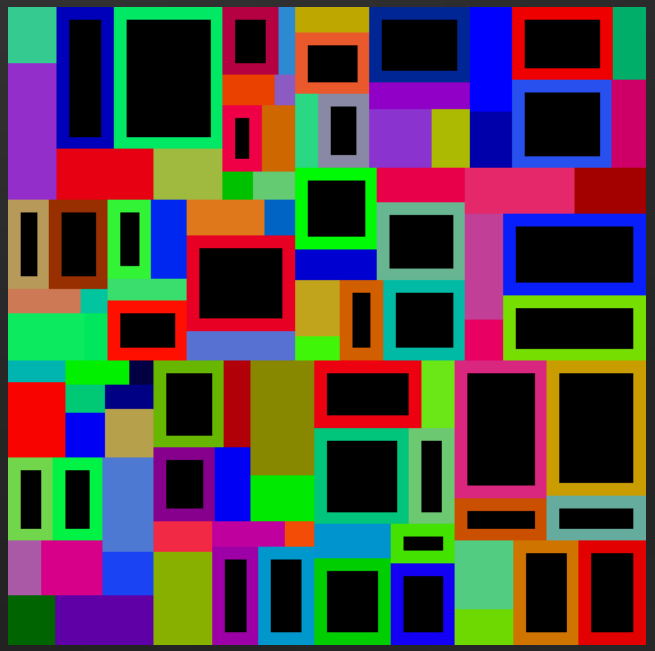


Затем рекурсивно разбиваем каждый узел еще на две части, каждый раз выбирая новую случайную разделяющую прямую, и создаем новые узлы. Логика выбора направления прямой (горизонтальное или вертикальное) может быть произвольной.



Я лично рекомендую разделять прямоугольники поперечно (то есть, вдоль короткой стороны), так получаются комнаты естественной формы, не будет уродливых пропорций.

Рекурсия продолжается, пока не будет достигнуто какое-то пороговое условие – например, прямоугольник минимальной площади. Думаю, основная идея метода уже ясна – с каждым из последних узлов (листьев дерева) сопоставляется комната подземелья. Ее можно расположить строго в центре прямоугольника или в случайном месте.



Вот так может выглядеть псевдокод для построения дерева:

```
class Room
{
  float x;
  float y;
  float width
  float height;
}

class Node
{
  float x;
  float y;
  float width
  float height;
  bool isHorizontal;
  Room room;
  Room corridor;
}

generateRooms(Node node):
{
  if not recursionThreshold
  {
    node.child1 = new Node;
    node.child2 = new Node;

    float splitPosition = randomSplitPosition(node);
    node.isHorizontal = selectSplitDirection(node);

    if node.isHorizontal
    {
      node.child1.x = node.x;
      node.child1.y = node.y;
      node.child1.width = splitPosition;
      node.child1.height = node.height;

      node.child2.x = node.x + splitPosition;
      node.child2.y = node.y;
      node.child2.width = node.width - splitPosition;
      node.child2.height = node.height;
    }
    else
    {
      node.child1.x = node.x;
      node.child1.y = node.y;
      node.child1.width = node.width;
      node.child1.height = splitPosition;

      node.child2.x = node.x;
      node.child2.y = node.y + splitPosition;
      node.child2.width = node.width;
      node.child2.height = node.height - splitPosition;
    }

    generateRooms(node.child1);
    generateRooms(node.child2);
  }
  else
  {
    node.room = new Room;
    node.room.x = node.x + WALL_WIDTH;
    node.room.y = node.y + WALL_WIDTH;
    node.room.width = node.width - WALL_WIDTH * 2;
    node.room.height = node.height - WALL_WIDTH * 2;
  }
}
```

В данном случае комнаты располагаются в центре узлов и их размеры пропорциональные размерам узлов. WALL_WIDTH – это толщина «стен», непустого пространства между комнатой и границами узла. Конечно, приведенный код далек от идеала – скорее всего, вам потребуется внести еще несколько проверок: например, чтобы не создавать комнаты для слишком маленьких узлов.

Теперь коридоры. Коридоры – это тоже комнаты, сопоставленные с каждым узлом: заметили дополнительное поле corridor в объявлении класса Node?

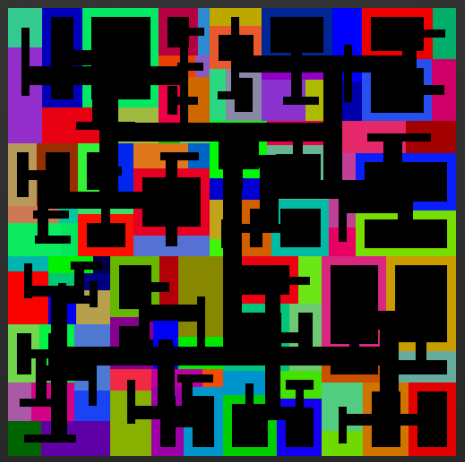
В отличие от обычных комнат, коридоры должны быть присвоены не листьям, а промежуточным узлам, так как их предназначение – соединять две комнаты, либо два других коридора. Это дает интересный эффект: если у каких-то листьев не будет комнат, получатся тупиковые коридоры, что может добавить подземелью таинственности. Например, в этих тупиках можно расположить какие-то тайники с сокровищами или тех же монстров.

Коридор строится путем соединения центров двух дочерних узлов узкой длинной комнатой. Вот простейшая функция для рекурсивного построения коридоров:

```
generateCorridors(Node node):
{
  if not node.isLeaf
  {
    float cx, cy, cw, ch;
    if node.isHorizontal
    {
      cx = node.child1.x + node.child1.width / 2;
      cy = node.child1.y + node.child1.height / 2 -
        CORRIDOR_WIDTH / 2;
      cw = node.child1.width / 2 +
        node.child2.width / 2;
      ch = CORRIDOR_WIDTH;
    }
    else
    {
      cx = node.child1.x + node.child1.width / 2 -
        CORRIDOR_WIDTH / 2;
      cy = node.child1.y + node.child1.height / 2;
      cw = CORRIDOR_WIDTH;
      ch = node.child1.height / 2
        + node.child2.height / 2;
    }
    node.corridor = new Room;
    node.corridor.x = cx;
    node.corridor.y = cy;
    node.corridor.width = cw;
    node.corridor.height = ch;

    generateCorridors(node.child1);
    generateCorridors(node.child2);
  }
}
```

CORRIDOR_WIDTH – это ширина коридора. Это может быть константа или изменяемое значение. Например, очень интересным вариантом будет варьировать толщину в зависимости от размера узла – получаются широкие магистральные коридоры, соединяющие большие комнаты, и узкие второстепенные.



Конечно, случайный генератор не является полноценной заменой живому левелдизайнеру – возможности его ограничены, и даже самый «умный» генератор не обладает и толикой творческого таланта человека.

Однако во многих случаях – в тех же играх-подземельях – возможностей гра-мотно реализованного генератора будет вполне достаточно, ведь здесь важна вариативность, а не точное воспроизведение каких-то конкретных образцов. Да здравствует случайность!

Тимур Гафаров

Разработка под PlayStation

Часть 3

Мы продолжаем серию статей, посвященных программированию под Sony PlayStation, начатую в «FPS» №36 '15 и №43 '16. Напомню, ранее мы рассмотрели простейшую программу «Hello, World» и чтение данных с CD. Настало время для самого интересного – вывода графики! ...

Для этого я подготовил такую картинку размером 128x128:

И вновь – немного теории. PSX имеет 1 мегабайт (1048576 байт) видеопамяти, которую удобно представлять в виде прямоугольника 1024x512 пикселей (при 16 битах на пиксель). В левом верхнем углу этого прямоугольника обычно находятся буферы кадра (вверху первичный, под ним – вторичный).

Текстуры размещаются в правой половине прямоугольника. Текстуры для PSX, как правило, хранят в формате TIM – этот формат ориентирован на графическое железо PSX, и в нем можно задать позицию текстуры в видеопамяти – очень необычная, кстати, для PC-программиста фича.

Для работы с TIM в Psy-Q SDK есть специальная утилита – TIM Tool. Она позволяет создавать TIM-файлы из изображений в формате BMP.

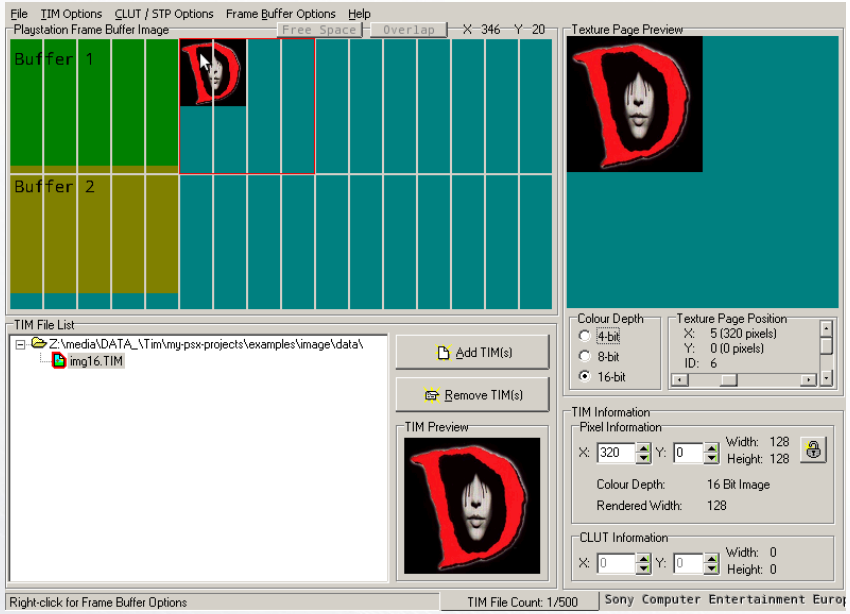
При сохранении вашей картинки в BMP укажите глубину цвета 16 бит на пиксель. Графический процессор PSX поддерживает и 24-битный цвет, однако функции рисования спрайтов и полигонов – почему-то только 16-битный.

Строго говоря, 16-битный режим в PSX называется 15-bit DIRECT. Почему же только 15 бит? Дело в том, что на каждый канал выделено по 5 битов (32 разных значения), а 16-й (старший) бит зарезервирован под прозрачность – его иногда называют STP (SemiTransparent). Такой режим позволяет выводить 32 x 32 x 32 = 32768 цветов. Для сравнения – 24-битный режим, в котором работают современные видеокарты, поддерживает 16777216 цветов.

Для экономии видеопамяти в PSX предусмотрены еще несколько текстурных режимов – а именно, индексированных, с использованием таблицы CLUT (Color Lookup Table). В будущем мы обязательно рассмотрим и их, а пока будем работать в обычном 15-битном DIRECT-режиме.

Откройте ваш BMP в TIM Tool (File → Import Image Files...). В списке Import Colour Depth выберите 16-bit. Нажмите OK. Перетащите текстуру в незанятую буферами кадра область памяти, как на скриншоте (буферы кадра выделены зеленым и болотным цветами, свободная память – бирюзовым). Во всплывающем окне при закрытии Tim Tool нажмите Yes.

В папке с исходной картинкой у вас должен появиться файл *.TIM. Будем считать, что он называется IMG.TIM.



Теперь, основываясь на коде из предыдущего урока, объявляем необходимые переменные и загружаем картинку (помним, что в C89 переменные объявляются строго в начале блока видимости):

```
Data data;
GsIMAGE image;
RECT rect;

// Тут идет инициализация графики, памяти и CD -
// такая же, как в предыдущих статьях.

readFile("\\IMG.TIM;1", &data);

GsGetTimInfo((u_long *)(data.ptr + 4), &image);
rect.x = image.px;
rect.y = image.py;
rect.w = image.pw;
rect.h = image.ph;
LoadImage(&rect, image.pixel);
```

Этот код читает картинку из оперативной памяти и помещает ее в видеопамять в качестве текстуры. Но для того, чтобы нарисовать ее на экране (в буфере кадра), нужно создать графический примитив – спрайт. Для этого я ввел специальную функцию, которая принимает на вход изображение-текстуру, позицию на экране, UV-координаты левого верхнего угла и ширину с высотой:

```
int initSprite(
  GsSPRITE* s,
  GsIMAGE* img,
  int x, int y,
  int u, int v,
  int w, int h)
{
  u_long colorMode;
  u_short tPage;

  colorMode = img->pmode & 0x03;
  tPage = GetTPage(colorMode, 0, img->px, img->py);

  s->attribute = (colorMode << 24);
  s->w = w; // size
  s->h = h;
  s->x = x; // position
  s->y = y;
  s->tpage = tPage;
  s->u = u; // UV offset inside texture
  s->v = v;
  s->mx = 0; // center
  s->my = 0;
  s->cx = img->cx;
  s->cy = img->cy;
  s->r = 128;
  s->b = 128;
  s->g = 128;
  s->scalex = ONE;
  s->scaley = ONE;
  s->rotate = 0;
}
```

Используем ее так:

```
GsSPRITE sprite;

// здесь идет предыдущий код инициализации

initSprite(&sprite, &image, 0, 0, 0, 0, 128, 128);
```

Не забудьте вместо 128, 128 поставить размеры своей картинки. PSX поддерживает спрайты размером от 1x1 до 255x255 пикселей.

Осталось нарисовать наш спрайт функцией GsSortSprite. Основной цикл будет выглядеть теперь следующим образом:

```
while (1)
{
  currentBuffer = GsGetActiveBuff();
  GsSetWorkBase(
    (PACKET*)GPUPacketArea[currentBuffer]);
  GsClearOt(0, 0, &OT[currentBuffer]);

  GsSortSprite(&sprite, &OT[currentBuffer], 0);

  DrawSync(0);
  VSync(0);
  GsSwapDispBuff();
  GsSortClear(0, 0, 0, &OT[currentBuffer]);
  GsDrawOt(&OT[currentBuffer]);
}
```

Запускаем и любуемся результатом!

Напоследок напомню, что существует такой сайт, как **http://www.psxdev.net**, там очень много полезной актуальной информации по разработке под PSX, есть активное сообщество. Если возникли вопросы, не стесняйтесь, пишите мне на почту: **gecko0307@gmail.com**, постараюсь помочь по мере своих возможностей.

Тимур Гафаров

P.S. *Только написав эту статью, я наткнулся на весьма любопытную разработку от Lameguy64, известного PSX-хакера – img2tim. Это консольный конвертер изображений в формат TIM, позволяющий обойтись без TIM Tool и, таким образом, ускорить процесс конвертирования. Обязательно рассмотрю использование этой программы в следующей части «Разработки под PlayStation», когда буду писать о индексированных режимах и CLUT. Вот ссылка на репозиторий:* ***https://github.com/Lameguy64/img2tim****.*

Полезные сервисы для пользователей GitHub. Выпуск 2

Мы продолжаем пополнять нашу коллекцию интересных сервисов и проектов, связанных с Git и GitHub, которую начали в «FPS» №37 '15. Если вы нашли что-то подходящее, просим сообщить в редакцию журнала – пишите на ***gecko0307@gmail.com****.*

Travis CI

Сервис непрерывной интеграции (continuous integration) для GitHub-проектов – автоматически собирает ваш проект под выбранные платформы после каждого коммита, что позволяет быстро обнаруживать регрессионные ошибки. Поддерживает множество языков и платформ.

https://travis-ci.org

GitPay

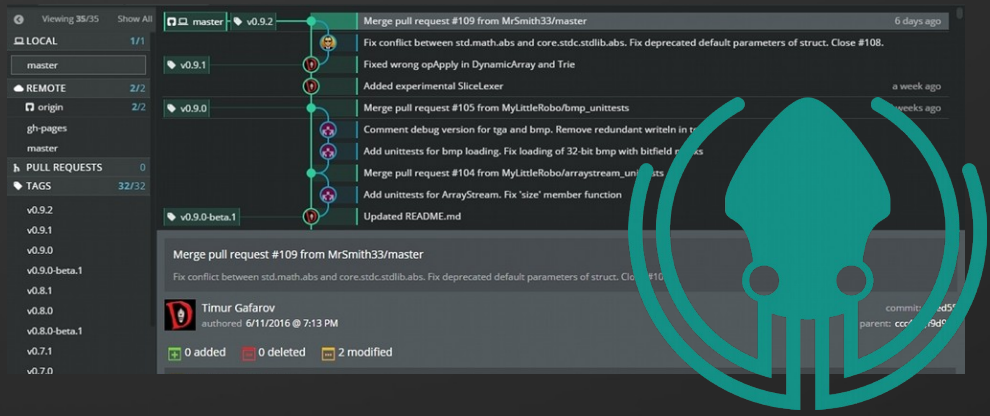
GitPay – система электронных платежей для GitHub-проектов. Деньги перечисляются через Bitcoin.

http://gitpay.org

GitKraken

GitKraken – альтернативный Git-клиент с графическим интерфейсом. Основан на веб-тулките Electron. Включает визуализацию репозиториев и веток, drag-n-drop функции, инструмент отмены в один клик и многое другое. GitKraken бесплатен (но не свободен), работает под управлением Windows, OSX и Linux. Есть также коммерческая версия GitKraken Pro с некоторыми дополнительными возможностями.

https://www.gitkraken.com



GitLab

GitLab – платформа для организации совместной работы с Git-репозиториями, которая по своим возможностям напоминает GitHub, но не привязана к конкретному сервису. GitLab распространяется в исходном коде под свободной лицензией и позволяет вам развернуть веб-сервис управления проектом на вашем сервере. Код проекта написан на языке Ruby с использованием фреймворка Ruby on Rails.

https://gitlab.com

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/

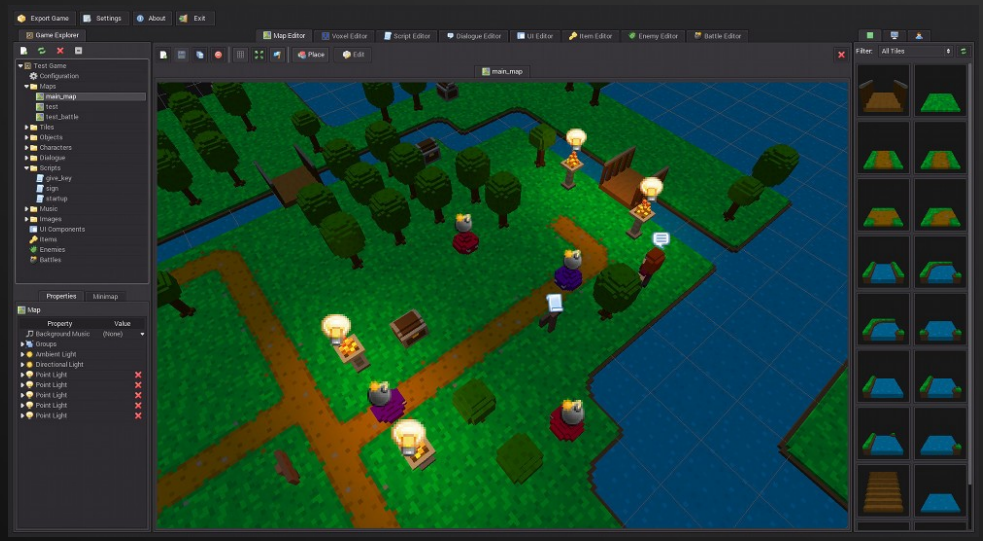
Linux-гейминг

Игровые новости из мира СПО

Знаменитый свободный движок Godot обновился до версии 2.1. В этом выпуске добавлена онлайновая библиотека ресурсов, новый API для разработки плагинов, поддержка включения в проект файлов шрифтов, система профайлинга, а также множество других нововведений.

Напомним, Godot – это активно развивающийся игровой движок, позволяющий создавать 2D и 3D-приложения под все популярные платформы (Windows, Linux, Mac OS X, Wii, Nintendo 3DS, PlayStation 3, PlayStation Vita, Android, iOS, BBX), а также Web – с использованием asm.js и NativeClient. По возможностям Godot приближается к Unity.

http://www.godotengine.org



Еще один свободный движок, Urho3D, обновился до версии 1.6. Релиз включает поддержку физически обоснованного рендеринга (PBR), variance shadow mapping, смешивания анимаций, биллбордов фиксированного размера, а также нового формата сцен на основе разметки JSON.

Напомним, Urho3D – это игровой движок для C++, примечательный поддержкой множества бэкендов: Direct3D 9, Direct3D 11, OpenGL 2.0, OpenGL 3.2, OpenGL ES 2.0 и WebGL. Есть встроенный deferred shading, аппаратный instancing, динамические тени, система частиц, рендеринг ландшафта, LOD, HDR, эффекты пост-процессинга, физика на движке Bullet, скриптинг на AngelScript, сеть, 3D-звук, встроенный GUI, поддержка Юникода при выводе текста. Исходники распространяются по лицензии MIT.

http://urho3d.github.io

Специалисты из Google и Blue Shift представили свободную реализацию упаковщика текстур на основе алгоритма ETC2 (Ericsson Texture Compression), обеспечивающего эффективное сжатие при сохранении высокого качества изображений. Поддержка формата ETC2 включена в стандарт OpenGL (начиная с OpenGL 4.3 и OpenGL ES 3.0) и не требует выплаты патентных отчислений. Исходники распространяются под лицензией Apache 2.0.

https://github.com/google/etc2comp

После трех месяцев разработки доступен релиз свободной реализации OpenGL – Mesa 12.0, примечательный реализацией OpenGL 4.2 и 4.3 в драйверах RadeonSI, Nouveau (nvc0) и Intel (i965). Поддержка новых версий OpenGL доступна для видеокарт AMD на основе архитектуры GCN, NVIDIA на основе архитектуры Fermi, Kepler и Maxwell и Intel семейства Gen8+. Кроме того, в драйверах RadeonSI и Nouveau (nvc0) обеспечена поддержка OpenGL ES 3.1. Первый выпуск ветки Mesa 12.0.0 имеет экспериментальный статус – после проведения окончательной стабилизации кода будет выпущена стабильная версия 12.0.2.

Кстати, в репозитории проекта недавно была реализована поддержка OpenGL 4.5 для драйвера i965.

http://www.mesa3d.org

Компания NVIDIA представила новую стабильную ветку проприетарного драйвера NVIDIA 370.28. Драйвер доступен для Linux (ARM, x86, x86_64), FreeBSD (x86, x86_64) и Solaris (x86_64).

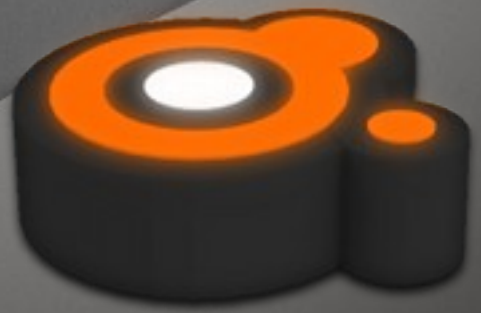
Компания AMD опубликовала выпуск унифицированного проприетарного Linux-драйвера AMD Catalyst Pro 15.302.2301 для видеокарт AMD FirePro. В набор входит графический драйвер, рантайм OpenCL, звуковые драйверы для DisplayPort/HDMI и интерфейс управления AMD Catalyst Pro Control Center.

Консорциум Khronos, занимающийся разработкой графических стандартов, анонсировал проект Vulkan Hpp, в рамках которого развивается враппер Vulkan для C++, предоставляющий большую гибкость по сравнению с основным API для C и позволяющий использовать современные возможности языка для упрощения разработки. При этом сохраняется весь спектр возможностей, доступных в низкоуровневом API. Исходники проекта доступны под лицензией Apache 2.0.

https://github.com/KhronosGroup/Vulkan-Hpp

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