41 2016 BF

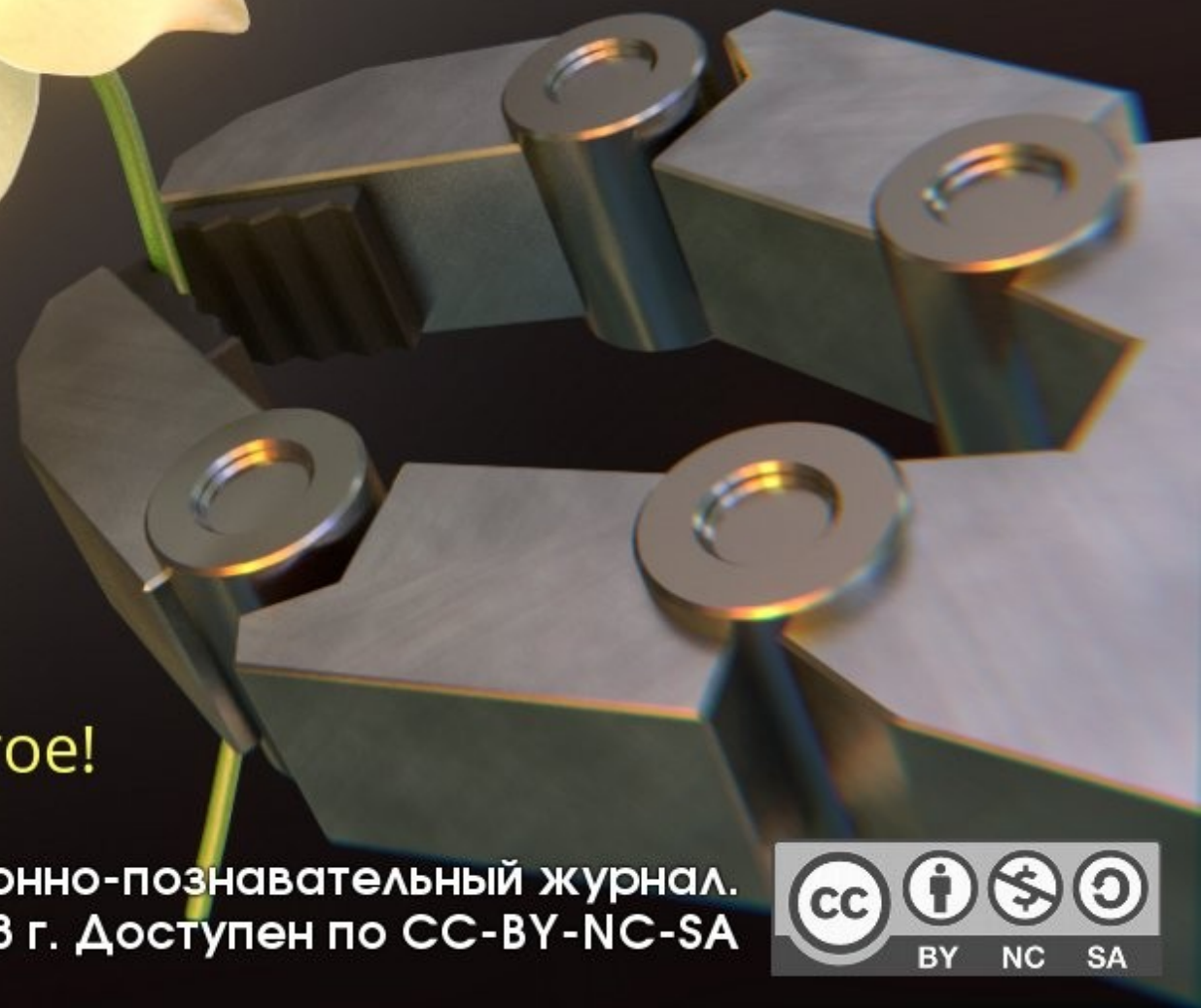
BLENDER: новости

Моделирование персонажей

Новые игры на BGE

Лучшие свободные движки

dlib и OpenCL



Vulkan

«За» и «против»

+ многое другое!

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

FPS №41

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.
и выходит 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 всегда, с анонса нового релиза Blender: недавно увидела свет версия **2.77**. Основное нововведение – это, конечно, начальная поддержка кэширования в **OpenVDB**. Это формат от Dreamworks, предназначенный для хранения объемных дискретных данных – например, результатов симуляции дыма или облаков.

Blender больше не поддерживает Windows XP. Кроме того, состоялся переход на **OpenGL 2.1** для отрисовки интерфейса – ценой отказа от поддержки устаревшего графического «железа» достигнуто значительное повышение производительности рендеринга во вьюпорте. В Blender 2.8x планируется переход на OpenGL 3.2.

Из других особенностей Blender 2.77 стоит отметить множество улучшений и оптимизаций в рендер-движке Cycles (улучшенные Subsurface Scattering, рендеринг объемных материалов и т.д.), новый инструмент булевых операций в режиме редактирования мешей, улучшенный модификатор Decimate, а также много других изменений – в режиме рисования/лепки, в системе анимации, в инструменте Grease Pencil и игровом движке. Скачать свежую версию Blender для всех платформ можно, как обычно, на **http://www.blender.org/download**.

Стало известно о участии Blender в **Google Summer of Code 2016**. Напомним, в рамках этого мероприятия ежегодно проводится отбор открытых проектов, в которых могут принять участие студенты.

В списке приоритетных задач для участников GSoC – улучшение системы трекинга, подавление шума и поддержка OpenSubdiv в рендер-движке Cycles, ускорение вершинного рисования на основе BVH, интеграция PTex, улучшение режима лепки и кривых Безье, а также многое другое. Полный список см. **здесь**.

В последнее время появляется все больше VFX-проектов, создающихся с помощью Blender – и это, конечно, не может не радовать. **«2045»** – короткометражный анимационный проект одного человека(!), 15-летнего японского подростка под ником 38912 DIGITAL. По правде говоря, в это трудно поверить – судя по качеству моделирования и анимации, да и общему профессионализму работы в целом.

Фильм повествует о недалеком будущем: в 2045 году человечество достигло технологической сингулярности – все стало производиться при помощи нанороботов. И однажды неизвестные злоумышленники, заполучив контроль над нанороботами, ставят мир на грань уничтожения.

https://www.youtube.com/watch?v=pfGpV-cJOmA

А вот еще одна научно-фантастическая сага – **«ATLAS»**, невероятно красивый 3-минутный фильм в духе «Интерстеллар», повествующий о путешествии через Солнечную систему к загадочному порталу в иное измерение. «ATLAS» был снят для конкурса, проведенного британским образовательным фондом Into Film.

https://www.youtube.com/watch?v=2XcAle1uLVs

Также недавно состоялась онлайн-премьера игрового фильма **«Electric Town»**:

https://vimeo.com/158562377

Анимационных релизов тоже немало. Так недавно вышел 8 эпизод аниме-сериала **«Deadstar»**, о котором мы уже не раз писали на страницах «FPS»:

https://www.youtube.com/watch?v=jy2a6yHdPb4

«The White Oak» – впечатляющий образец модной сегодня низкополигональной анимации, повествующий о жителях зачарованного леса и загадочном Белом дубе:

https://www.youtube.com/watch?v=nVVhdhuqm4w

Поклонники восточных единоборств и NPR-графики оценят **«Katsu Cats»** – короткометражный фильм о дерущихся котиках в японском антураже:

https://www.youtube.com/watch?v=tRf4S_ArF_A

После долгого периода стагнации, активизировался проект **YafaRay** – свободный рендер-движок, который был достойной альтернативой BI во времена Blender 2.4x. С появлением Cycles популярность сторонних рендеров для Blender резко упала, но большинство из них продолжают развиваться – жив и YafaRay, который не так давно обновился до версии 2.0. Сейчас активно разрабатывается экспериментальная ветка проекта, в которой улучшена поддержка объемных материалов, добавлена поддержка проходов, стереоскопического рендеринга и многое другое. Подробный обзор YafaRay для новых версий Blender читайте в «FPS» №20 '12.

http://www.yafaray.org

Y • A • F • A • R • A • Y

Вышла новая версия открытого фреймворка для создания браузерных 3D-приложений **Blend4Web 16.02**. В этой версии улучшена система частиц, расширена поддержка node-материалов, переработана система управления камерой.

Напомним, Blend4Web предназначен для создания трехмерного интерактивного контента, работающего в браузерах без использования плагинов. Пакет тесно интегрирован с Blender, который используется в качестве основного инструмента редактирования 3D-сцен. Воспроизведение контента осуществляется средствами WebGL, Web Audio и других браузерных технологий. Наработки проекта распространяются под лицензией GPLv3.

https://www.blend4web.com

Недавно, кстати, разработчики Blend4Web представили релиз первой полноценной браузерной игры на основе этого движка – **«Сказ о Пятигоре»**. Это фэнтезийный приключенческий боевик от третьего лица в сказочном древнеславянском антураже. Все ресурсы для игры созданы с использованием свободных инструментов – Blender, Krita, GIMP и Audacity, а исходники доступны по лицензии GPLv3.

http://www.petigor.com

Недавно открылся любопытный проект **Right-Click Select** – площадка для сбора идей и предложений по улучшению Blender. Пользователи могут размещать свои идеи, а также комментировать чужие и голосовать за них. Сайт не является официальным ресурсом Blender Foundation, поэтому публикуемые идеи могут остаться нереализованными – он скорее служит неким творческим посредником между сообществом и разработчиками-энтузиастами.

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!

Моделирование персонажей

Советы профессионалов

Дмитрий Паркин *– ветеран российского игропрома, принимавший участие во многих отечественных проектах и в нескольких западных. На прошедшем в 2008 году Dominance War III Паркин занял первое место в категории 3D.*

Я не научу вас, как моделить лучше всех на свете – это невозможно. Но я расскажу о том, что нужно знать в первую очередь. И это не на уровне владением ПО (это лишь часть работы, не самая важная), это на уровне осознания и понимания.

Прежде всего, стоит попытаться выделить своего персонажа идеей. Но не выдумывать чудище о тысячи ногах или огненную змею, покрытую адскими шипами – в общем, лучше без «шизофрении». Обязательно анализировать то, что делаешь, на кого равняешься – стараться сбалансировать своего персонажа по всем фронтам: от идеи до технического исполнения.

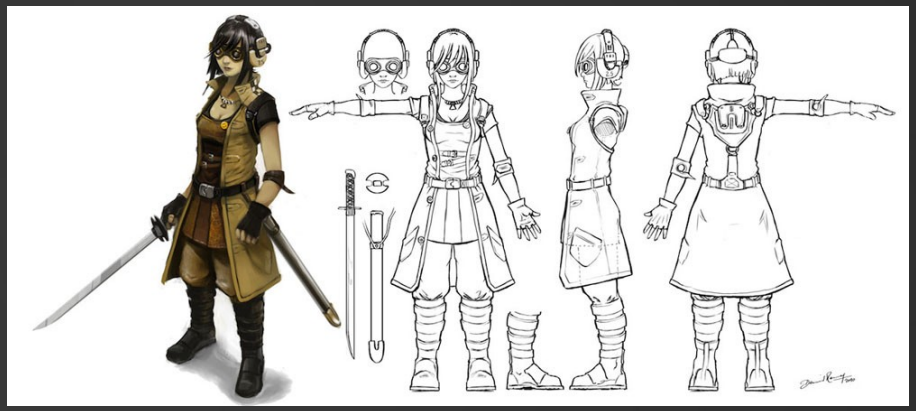
Надо рассчитывать свои силы – планку ставить высоко, но не с мыслью: «Эх, сейчас как наделаю лучше всех, чтобы в Blizzard взяли!..», а потом, на стадии моделинга, энтузиазм свалился, и все.

Нужно фильтровать идеи – не делать того, в чем не уверен на 100 процентов, не взваливать на себя кропотливую проработку узорчиков и других мелких деталей, создающих кашу – оно того не стоит.

Лучше сделать модель просто, но максимально качественно – с несколькими оригинальными деталями, которые будут хорошо читаться на общей массе.

Очень важно хорошо начать, не делать ошибок в самом начале. Оправдания вроде «а, ладно, сойдет и так, потом текстурой выправлю или в Zbrush» отметаем сразу же! Сделанные в начале ошибки или мелкие неточности – как огромная заноза: будут мешать в течение всей работы и могут сильно изгадить результат.

С первых шагов надо следить за топологией и базовой формой, швы прятать любым путем. И еще: главное – это хорошо проработанная высокополигональная модель, так как с нее запекается карта нормалей.



Пару слов о критике. Самая адекватная критика всегда идет от людей, которые вообще ничего не понимают в 3D-моделировании: например, какая-нибудь девушка без всякого умысла может точно сказать, что в персонаже не так: мол, руки у него слишком короткие, лицо глупое, ботинки смешные. Вот именно к этому стоит прислушиваться в первую очередь. Еще один важный вспомогательный момент – это референс (фотографии, скетчи, иллюстрации по теме). Кто не пользуется референсом – тот либо ламер, либо уже мега-профи!

И запомните: психологическая хитрость в том, что все можно натренировать, как в спорте: работай больше и требуй от себя больше – через некоторое время перестанешь уставать. Ведь устают не когда работа идет гладко, а когда долго не получается. Как бы не занудно это звучало, но это правда: надо много работать, чтобы что-то начало получаться сразу. Секрет прост – секрета нет!

Источник: ***http://m-cg.ru***

***Энтони Вард**, цифровой художник и аниматор из Великобритании – гуру игрового моделирования с 27-летним стажем. В 2004-2010 годах Вард выпустил несколько руководств по моделированию игровых персонажей. Вот несколько советов, которые он дал начинающим:*

1. Помните об ограничениях

Речь не только об ограничениях на количество полигонов или размеры текстур – это само собой. Большое значение имеет и то, как модель будет использована в игре. Например, если модель не будет анимирована, то и нет смысла создавать лишние edge loop'ы в местах сгибов. То же касается лица: если рот персонажа не будет открываться, то не нужны зубы и язык. Вообще, нет смысла тратить время и вычислительные ресурсы на те элементы модели, которые заведомо не будут видимы игроку.

2. Используйте референсы

При моделировании существ из реального мира это особенно важно: очень многие начинающие художники совершают ошибку, когда берутся за создание персонажа, не заглядывая в анатомическое пособие. Моделируя живое существо, помните о его внутренней пластической структуре, о скелете и мышцах.

3. Начинайте с простого

Никогда не добавляйте мелкие детали в самом начале работы. Начинайте с грубых, общих форм: например, фигуру персонажа можно составить из цилиндров. Это даст вам общее видение пропорций, которые трудно исправить, если вы уже смоделировали все до мелочей. Кстати, эту первоначальную версию модели можно использовать для риггинга (создания скелета для анимации) – таким образом, моделлер и аниматор могут работать одновременно.

4. Используйте подразбиение

Подразбиение (subdivision) – это эффективный способ моделировать сложные формы на основе более простых. Можно создать грубую версию модели из низкополигональных объектов (вплоть до параллелепипедов), а затем подразбить ее, чтобы получить гладкую поверхность.

5. Придерживайтесь качественной топологии

Расположение вершин и ребер должно быть систематичным. В идеале модель должна представлять собой равномерную сетку из прямоугольников – с такой моделью будет проще работать и аниматорам, и текстурщикам.

6. Следите за деформируемыми частями

В случае с анимированной моделью особое внимание следует уделять местам сгибов – например, локтевым и коленным суставам. В этих местах обязательно нужно добавить дополнительные полигоны.

Если вы – моделлер, текстурщик или аниматор, работали в игровых проектах и знаете профессиональные 3D-пакеты как свои пять пальцев, то приглашаем вас поделиться опытом с тысячами читателей «FPS»: присылайте свои статьи на наш постоянный адрес – ***gecko0307@gmail.com***

Лучшие игры на Blender Game Engine

Выпуск II

Представляем вашему вниманию очередную подборку свежих проектов, созданных (или создающихся) на игровом движке Blender Game Engine (BGE). Напомним, первый выпуск этого обзора был опубликован в «FPS» №34 '15.

Ballance Clone

Жанр: головоломка
Статус: В разработке

Очень интересная физическая головоломка в классическом жанре «шарик в лабиринте», восходящем к легендарной Marble Madness. Является, как ясно из названия, клоном игры Ballance 2004 года – причем, очень качественным клоном, в плане графики даже превосходящим оригинал.

От игрока требуется довести шарик до конца уровня, обходя различные препятствия и не падая в пропасть.

Особенностью Ballance Clone, как и оригинальной Ballance, является возможность в ходе игры изменить материал шарика – есть камень, древесина и бумага, причем все они имеют соответствующие физические характеристики. Игра находится в разработке.

К сожалению, у проекта пока нет официального сайта – следите за обновлениями на YouTube-канале автора.

Engine Roar

Жанр: гонки
Статус: В разработке
Сайт: http://engineroargame.blogspot.ru

Наверное, самая качественная на сегодняшний день гоночная игра, разрабатываемая на BGE – это гонки на выживание в стиле незабвенного Destruction Derby. Игра находится на ранней стадии разработки, но графика и геймплей, для инди-проекта, уже выглядят превосходно!

P_ONE

Жанр: FPS
Статус: В разработке
Сайт: http://vk.com/amazing_software

Еще одна качественная игра от российских разработчиков – студии Amazing Software: очень атмосферный сюрвайвл от первого лица.

Artefact Hunter

Жанр: action-adventure
Статус: Готова

Игра от индийских разработчиков и в индийском антураже – приключенческий боевик от первого/третьего лица, чем-то похожий на ранние части Tomb Raider. Вы играете за искателя приключений, занимающегося поиском магических артефактов в древних храмах. Скачать игру можно здесь.

Nordostrov

Жанр: FPS
Статус: Проект закрыт

Шутер от первого лица с интересной возможностью модифицировать оружие прямо во время игры – при помощи специальных компьютеров. Игра, к сожалению, так и не была завершена, но автор выложил в свободный доступ все исходники.

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!

Обзор дополнений Blender

Выпуск 19

Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. В этом выпуске мы представляем дополнения, которые пригодятся аниматорам.

Если вы разрабатываете собственное дополнение или просто нашли в Интернете чей-то интересный проект, будем очень рады, если вы напишете нам об этом и поделитесь ссылкой. Пишите на ***gecko0307@gmail.com****.*

BlenRig

BlenRig – это мощный фреймворк для автоматического риггинга человекоподобных персонажей. Поддерживает мышечную и лицевую анимацию, создает скелет или сетку для Mesh Deform.

Проект существует уже очень долго (с 2007 года) – недавно вышла уже 5-я версия. Распространяется бесплатно.

Разработчик: Juan Pablo Bouza

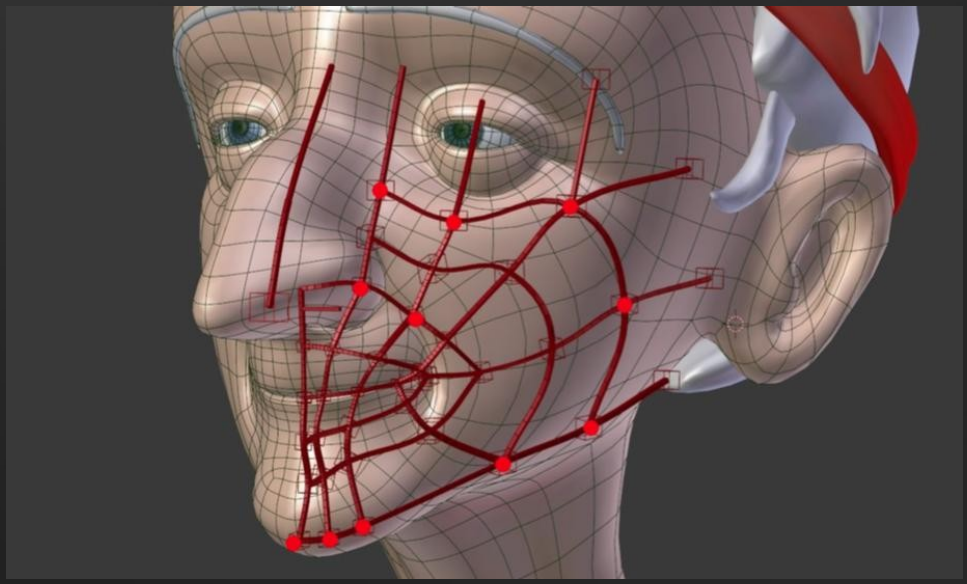
https://gitlab.com/jpbouza/BlenRig

ClothFX

Очень интересный аддон от разработчика небезызвестных SnowFX и Easy Clouds (мы писали о них в 18 выпуске обзора). Данный продукт реализует в Blender динамическое разрывание ткани – вы можете не только задать объект, который будет рвать ткань, но и контролировать саму форму и детализацию разрыва. При этом поддерживается быстрый предпросмотр результата, а также запекание симуляции. Цена дополнения (включая пожизненные обновления): $45.00.

Разработчик: AFX Lab

https://cgcookiemarkets.com/all-products/clothfx-dynamic-tearing-tool



Animation Joiner

Аддон, позволяющий соединить несколько анимированных объектов в один меш. Очень полезный инструмент для тех, кто много работает с симуляцией физики в Blender – он позволяет значительно сократить потребление памяти в сложных сценах. Дополнение бесплатное.

Разработчик: Francois Grassard
http://www.coyhot.com/blender-addon-amination-joiner

Cut-out Animation Tools

Аддон от автора Multi Object UV Editing и нескольких других популярных дополнений для Blender. COA Tools – это фреймворк для 2D-анимации. Очень полезен для рендеринга спрайтов персонажей для игр – с этим инструментом вам, по сути, не требуется специализированный 2D-пакет. В комплекте также идет импортер для движка Godot.

Дополнение бесплатное, распространяется по лицензии GPL.

Разработчик: Andreas Esau
https://github.com/ndee85/coa_tools

Animated Render Border

В Blender есть полезная функция Render Border, позволяющая рендерить не весь вид из камеры, а только заданный прямоугольный регион – это удобно для предпросмотра, когда нужно быстро посмотреть финальный рендер какого-то одного объекта или участка сцены.

Но если вы создаете анимацию движения, придется вручную менять положение рамки региона каждый кадр, что, конечно, звучит абсурдно. Эту проблему решает данное коммерческое дополнение, позволяющее автоматически двигать рамку, отслеживая перемещение объекта. Цена дополнения: $6.95.

Разработчик: Ray Maillot

https://cgcookiemarkets.com/all-products/animated-render-border

2D-графика: новости

Scribus 1.5.1

Увидела свет новая версия свободной программы для верстки Scribus – 1.5.1. Ветка 1.5 позиционируется как экспериментальная: она включает новый интерфейс пользователя на базе Qt5, измененный файловый формат проектов, полноценную поддержку таблиц и расширенные средства обработки текста. После окончательной стабилизации, на базе ветки 1.5 будет сформирован стабильный релиз Scribus 1.6.0.

Напомним, Scribus является электронной издательской системой, свободным аналогом Adobe InDesign и QuarkXPress, предоставляя профессиональные средства для предпечатной подготовки изданий.

Scribus генерирует PDF, полностью совместимые с печатными машинами, поддерживает работу со CMYK, плашечными цветами и ICC. Работает на Linux, Windows, OSX и Haiku. Исходный код проекта рас-пространяется по лицензии GPLv2.

http://www.scribus.net

OpenToonz

Стало известно об открытии исходников анимационного пакета Toonz, применявшегося, в числе прочего, для создания сериала «Футурама» и знаменитых шедевров Хаяо Миядзаки. Это стало возможным благодаря продаже проекта Toonz японской компании Dwango – в рамках сделки она обязуется опубликовать код в качестве открытого проекта OpenToonz. О поддержке проекта заявила аниме-студия Ghibli, которая также открыла свои наработки, развиваемые в Toonz Ghibli Edition.

https://github.com/opentoonz

Язык D

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 2.071.0

Вышла новая версия референсного компилятора D – DMD 2.071.0. Из значительных изменений стоит отметить исправление давнего бага с ошибочным импортом символов, помеченных как private, по полноквалифицированным именам.

http://dlang.org/download.html

LDC 0.17.1 и 1.0.0-alpha1

Вышла LDC 0.17.1 – новая версия компилятора D с LLVM в качестве бэкенда. Релиз основан на фронтенде и рантайме D 2.068.2, включает поддержку LLVM 3.5-3.8. С этой версии появилась поддержка 64-битной Windows, а также нативный компилятор под ARM. Кроме того, состоялся предварительный релиз LDC 1.0.0, который знаменует стабилизацию проекта. Он основан на фронтенде D 2.069.2.

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

• Геймдев и мультимедиа

Damage Control

Вышла новая игра, написанная на D – Damage Control, абстрактная игра в жанре tower defence – игрок защищает свою крепость от захватчиков путем управления турелями. Игра разработана на библиотеке Allegro с использованием биндинга DAllegro. Есть версии для Linux и Windows.

https://github.com/rcorre/damage_control

D + Vulkan

В этом году, как известно, вышла первая версия нового графического стандарта Vulkan, и ведущие производители видеокарт уже выпустили драйвера с поддержкой этого API. Не заставили себя ждать и энтузиасты-дишники, оперативно подготовившие биндинги Vulkan для D. Появилось сразу три таких проекта:

- **VulkanizeD** – наиболее актуальный и юзабельный биндинг, причем очень компактный: умещается в один-единственный модуль Vulkan.d. В комплекте также идут 32- и 64-битные библиотеки импорта под Windows, совместимые с DMD (библиотеки из LunarG SDK, к сожалению, несовместимы).
- **DerelictVulkan** – динамический биндинг, основанный на фреймворке Derelict. Находится в активной разработке.
- **DVulkan** – генератор биндингов из XML-спецификаций стандарта. Интересен также тем, что загружает все функции Vulkan при помощи vkGetInstanceProcAddr.

• Бизнес

GtkD 3.3.0

Вышла новая версия проекта GtkD, обеспечивающая поддержку Gtk+ 3.20 и GStreamer 1.8. Напомним, GtkD – это биндинг и объектно-ориентированная обертка над функциями кроссплатформенного графического тулкита Gtk+.

http://gtkd.org

Poodinis 6.0.0

Poodinis – это фреймворк внедрения зависимостей для D, вдохновленный такими решениями, как Spring для Java и Hypodermic для C++.

https://github.com/mbierlee/poodinis

pl0stuff

На D появился компилятор языка PL/0.

https://github.com/UplinkCoder/pl0stuff

• Веб

vibe.d 0.7.28

Вышла новая версия веб-фреймворка Vibe.d. Это минорный исправляющий релиз, не включающий серьезных нововведений.

http://vibed.org

Mondo

Mondo – это биндинг и враппер популярной документоориентированной СУБД MongoDB.

https://github.com/2night/mondo
https://www.mongodb.org

SQLite-D

Очередная реализация SQLite 3 для D – ведь СУБД никогда не бывает слишком много!

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

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

Шаблонная «магия»

ООП на структурах в dlib

Я уже достаточно давно перешел с C++ на D, и единственное, о чем я скучаю со времен «плюсов» – это инстанцирование класса на стеке. В D классы инстанцируются в динамической памяти и управляются сборщиком мусора, поэтому в тех случаях, когда нужно многократно создавать и уничтожать объекты в реальном времени, вместо классов используются структуры. Хотя, конечно, в настоящее время можно написать так:

```
scope foo = new Foo();
```

И объект foo будет создан на стеке. Но эта возможность, судя по информации на официальном сайте D, вскоре будет помечена как deprecated и, вероятнее всего, со временем будет удалена из языка. Так как же быть, если мы хотим убить двух зайцев – иметь стековые объекты, но и не отказываться от ООП?

Библиотека dlib предлагает свой вариант решения этой проблемы – прототипное ООП для структур на основе шаблонов и примесей (mixin). Реализацию этой системы вы можете найти в модуле dlib.core.oop. Стоит оговориться сразу: модуль неидеален, он не является стопроцентной заменой классам. Что он может и чего не может – перечислю в конце статьи, а сейчас начну с простого примера – системы наследования ролевых характеристик через примесь Inherit.

```
import dlib.core.oop;

struct Character
{
    int health;
}

struct Mage
{
    mixin Inherit!(Character);
    int mana;
}

struct Blacksmith
{
    mixin Inherit!(Character);
    int craft;
}
```

Конечно, в реальных RPG система характеристик реализуется не совсем так, но зато иллюстрация получилась достаточно яркой, и главное – с ее помощью можно продемонстрировать главный козырь dlib.core.oop: множественное наследование.

```
struct Player
{
    mixin Inherit!(Mage, Blacksmith);
}
```

Теперь наш Player обладает способностями и мага, и кузнеца:

```
Player player;
player.health = 100;
player.mana = 50;
player.craft = 30;
```

У множественного наследования, конечно, есть классическая проблема, которую назвают «diamond inheritance» («ромбическое наследование»): неоднозначная ситуация, когда два класса B и C наследуют от A, а класс D наследует от обоих классов B и C. Если, скажем, B и C переопределили метод из A, то какой именно из них должен вызываться из класса D?

dlib.core.oop устанавливает следующий принцип: в тех случаях, когда родительские классы (структуры) содержат одинаковые методы или поля, то дочерние всегда обращаются к первому из них. То есть, в нашем примере Mage будет иметь приоритет над Blacksmith – поле health, унаследованное Player'ом, будет принадлежать Mage.

Есть, впрочем, и возможность явным образом обратиться к конкретной родительской структуре.

Делается это так:

```
player._parent[1].health = 10;
```

В нашем примере, конечно, смысла в подобной операции мало (зачем игроку два параметра здоровья?), но в некоторых ситуациях это все-таки может понадобиться.

А теперь, как и обещал, пара слов об ограничениях dlib.core.oop. Модуль не реализует динамический полиморфизм – осуществлять диспетчеризацию в рантайме невозможно. Иными словами, функция, принимающая в качестве аргумента объект типа Character, не будет работать с Mage, Blacksmith и Player.

Эта проблема частично решается шаблонами функций, но они в данном случае тоже не являются панацеей. Например, шаблоны не помогут, если вы хотите хранить разнотипные объекты в контейнере – все упирается в статическую типизацию. Но если вам не требуется полиморфизм, а только инкапсуляция и наследование, то модуль вполне оправдает ваши ожидания.

Напомню, библиотеку dlib можно найти в репозитории на GitHub: **http://github.com/gecko0307/dlib**

Тимур Гафаров

dlib.image и OpenCL

Пакет dlib.image, как известно, предоставляет удобный набор средств для обработки изображений – в том числе, коллекцию простейших фильтров (размытие, выделение границ, свертка и т.д.). Как автор dlib, могу честно сказать, что реализация фильтров там далека от оптимальной – для маленьких изображений она еще годится, но, например, уже на фотографиях из мыльниц даже boxBlur выполняется неприлично долго. Можно, конечно, попытаться распараллеливать фильтры на несколько потоков или задействовать SIMD, но, как мне кажется, при наличии современных видеокарт все это – мертвому припарки: куда интереснее было бы перенести вычисления на GPU!

В этой статье мы рассмотрим реализацию boxBlur (прямоугольного размытия), одного из самых простых оконных фильтров, средствами OpenCL – специализированного API для параллельных вычислений на графических процессорах. На наше счастье, для OpenCL существует полноценный биндинг для D в лице проекта **DerelictCL**. Правда, он не включает поддержку OpenCL 2.0, но для наших задач хватит и версии 1.1.

Итак, подключаем необходимые модули и инициализируем биндинг:

```
import std.stdio;
import std.string;
import std.file;
import dlib.image;
import derelict.opencl.cl;

void main()
{
    DerelictCL.load();
    DerelictCL.reload(CLVersion.CL11);

    // ...
}
```

Дальнейший код размещается в функции main. Для краткости я опущу проверку OpenCL-вызовов на ошибки, но в реальном приложении, конечно, ей пренебрегать не стоит.

Начнем с загрузки входного изображения. Мы будем работать только с форматом RGBA8 и, чтобы пока не мучиться с конвертацией из других форматов, сразу сделаем соответствующую проверку:

```
auto inpImage = loadPNG("input.png");
if (inpImage.pixelFormat != PixelFormat.RGBA8)
{
    writeln("Only RGBA8 images are supported!");
    return;
}
```

Запрашиваем количество доступных платформ и их id'ы:

```
cl_platform_id[] platforms;
cl_uint numPlatforms = 0;
clGetPlatformIDs(5, null, &numPlatforms);
platforms = new cl_platform_id[numPlatforms];
clGetPlatformIDs(numPlatforms, platforms.ptr, null);
```

Платформа – это backend реализации OpenCL. Например, для GPU от NVIDIA платформой будет CUDA. В качестве рабочей берем первую полученную платформу:

```
cl_platform_id platform = platforms[0];
```

Запрашиваем количество доступных устройств и их id'ы.

```
cl_uint numDevices = 0;
clGetDeviceIDs(platform,
  CL_DEVICE_TYPE_DEFAULT, 1,
  null, &numDevices);
cl_device_id[] devices = new cl_device_id[numDevices];
clGetDeviceIDs(platform,
  CL_DEVICE_TYPE_DEFAULT, numDevices,
  devices.ptr, null);
```

В качестве рабочего берем первое устройство:

```
cl_device_id device = devices[0];
```

Создаем контекст OpenCL. Контекст – это непосредственно среда, в которой происходят вычисления:

```
cl_int ret;
cl_context context =
  clCreateContext(null, 1, &device,
  null, null, &ret);
```

Создаем очередь команд:

```
cl_command_queue command_queue =
  clCreateCommandQueue(context, device, 0, &ret);
```

Создаем входное изображение (OpenCL предоставляет встроенные средства для работы с изображениями, что очень удобно):

```
cl_image_format format = {
  CL_RGBA,
  CL_UNORM_INT8
};

cl_mem inImgMem = clCreateImage2D(context,
  CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
  &format, inpImage.width, inpImage.height, 0,
  inpImage.data.ptr, &ret);
```

Параметр CL_MEM_COPY_HOST_PTR означает, что входные данные должны быть скопированы в память устройства.

Аналогично создаем выходное изображение:

```
cl_mem outImgMem = clCreateImage2D(context,
  CL_MEM_WRITE_ONLY,
  &format, inpImage.width, inpImage.height, 0,
  null, &ret);
```

Теперь самое интересное – непосредственно фильтр. В OpenCL программы для GPU пишутся на специальном C-подобном языке, а сами программы называются ядрами (kernels). За основу для нашего блюра я взял встроенный boxBlur в dlib (dlib.image.filters.boxblur) с радиусом (полуразмером окна), равным 10. Ядро выглядит следующим образом:

```
__constant sampler_t sampler =
  CLK_NORMALIZED_COORDS_FALSE |
  CLK_ADDRESS_CLAMP_TO_EDGE |
  CLK_FILTER_NEAREST;

__kernel void blur(
  __read_only image2d_t inImage,
  __write_only image2d_t outImage)
{
  const int2 pos =
    {get_global_id(0), get_global_id(1)};

  float4 sum = (float4)(0.0f, 0.0f, 0.0f, 0.0f);

  const int radius = 10;

  for(int a = -radius; a < radius+1; a++)
  {
    for(int b = -radius; b < radius+1; b++)
    {
      sum += read_imagef(inImage, sampler,
        pos + (int2)(b, a));
    }
  }

  sum /= radius * radius * 4.0f;

  write_imagef(outImage, pos, sum);
}
```

Радиус, в принципе, можно (и нужно) передавать в ядро в виде параметра – при необходимости это легко добавить.

Загружаем программу из файла «boxblur.cl» и компилируем ее:

```
string progSrc = readText("boxblur.cl");
auto progSrcPtr = progSrc.toStringz;

size_t* lengths = cast(size_t*)[progSrc.length];
char** ptrs = cast(char**)[progSrcPtr];
cl_program program =
  clCreateProgramWithSource(context, 1, ptrs,
  lengths, &ret);
clBuildProgram(program, 0, null, null, null, null);
cl_kernel kernel =
  clCreateKernel(program, "blur", &ret);
```

Перед тем, как выполнять ядро, нужно передать ему параметры (входное и выходное изображения):

```
clSetKernelArg(kernel, 0, cl_mem.sizeof,
  cast(void*)&inImgMem);
clSetKernelArg(kernel, 1, cl_mem.sizeof,
  cast(void*)&outImgMem);
```

И последнее – копируем данные из оперативной памяти в память устройства:

```
size_t[3] origin = [0, 0, 0];
size_t[3] region =
  [inpImage.width, inpImage.height, 1];

clEnqueueWriteImage(command_queue, inImgMem, CL_TRUE,
  origin.ptr, region.ptr, 0, 0,
  inpImage.data.ptr, 0, null, null);
```

Выполняем:

```
size_t[2] work_size =
  [inpImage.width, inpImage.height];

ret = clEnqueueNDRangeKernel(command_queue, kernel,
  2, null, work_size.ptr, null, 0, null, null);
```

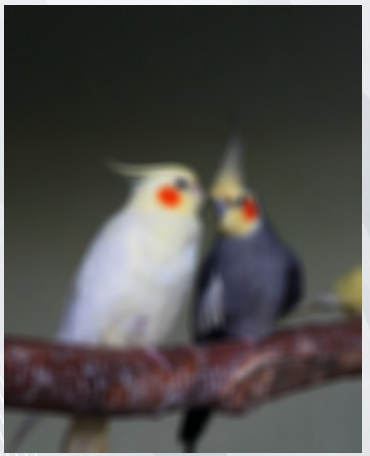
Теперь нам нужно получить результаты вычисления обратно в оперативную память. Для этого создаем новое изображение того же размера и используем функцию clEnqueueReadImage:

```
auto outImage =
  inpImage.createSameFormat(
    inpImage.width, inpImage.height);
clEnqueueReadImage(command_queue, outImgMem,
  CL_TRUE, origin.ptr, region.ptr, 0, 0,
  outImage.data.ptr, 0, null, null);
```

Осталось сохранить результат в файл:

```
outImage.savePNG("output.png");
```

На GeForce GT 740 для изображения размером 3648x2736 вся эта процедура (за вычетом операций файлового ввода/вывода) занимает около 2 секунд.



Тимур Гафаров
http://lhs-blog.info

Лучшие свободные игровые движки

Часть II

Мы продолжаем наш обзор лучших 3D-движков, распространяемых под свободной лицензией, начатый в предыдущем номере журнала. В этот раз мы рассмотрим несколько молодых и амбициозных проектов, которые можно, в той или иной степени, назвать аналогами Unity – в том смысле, что они ориентированы на работу с визуальными WYSIWYG-редакторами. Такие движки имеют на порядок более низкий порог вхождения, чем «классические» (то есть, существующие в виде библиотек и фреймворков). В большинстве случаев, вам даже не придется изучать C++, так как логика приложений в них задается динамически, на скриптовых языках.

Torque3D

Этот движок молодым назвать, конечно, нельзя – он появился еще в 2001 году и прожил долгую жизнь в качестве коммерческого продукта, а в разряд открытых проектов был переведен в 2012 году.

Движок написан на C++, позволяет создавать игры под Windows, Linux, OSX, Android, iOS, Wii, Xbox 360. Рендерит графику через Direct3D и OpenGL. Есть поддержка deferred shading, пост-процессинга, физики (враппер поверх PhysX и Bullet). Имеется свой скриптовый язык TorqueScript, с помощью которого можно задавать логику приложений без необходимости писать на C++, редактор уровней и другие вспомогательные утилиты.

Исходники распространяются по лицензии MIT.

http://torque3d.org

Xenco

Разработка японской игровой компании Silicon Studio. Ранее этот проект был известен как Paradox 3D. Движок основан на платформе .Net/Mono и работает с языком C# (интегрируется с MS Visual Studio, но это необязательно). Позволяет создавать как 3D-, так и 2D-игры. Рендеринг осуществляется через Direct3D 11 и OpenGL. Движок поддерживает Windows, Windows Phone, Android и iOS - в будущем планируется также поддержка Linux и PlayStation 4. Есть физика на основе движка Bullet, средства вывода звука и создания GUI, а также, разумеется, собственный редактор, в котором можно управлять ресурсами и создавать сцены.

Исходники Xenco распространяются по лицензии GPLv3 – следовательно, если вы модифицируете движок, то будете обязаны открыть и свою изменную версию. Возможен и dual-licensing, по индивидуальной договоренности с разработчиками.

http://xenko.com

Urho3D

Весьма интересная разработка, пригодная для создания проектов AAA-класса. Движок написан на C++. Примечательна поддержка нескольких графических 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. Движок поддерживает Windows, Linux, OSX, Android, iOS, Raspberry Pi и веб. Есть редактор сцен, полностью основанный на самом Urho3D.

Исходники распространяются по лицензии MIT.

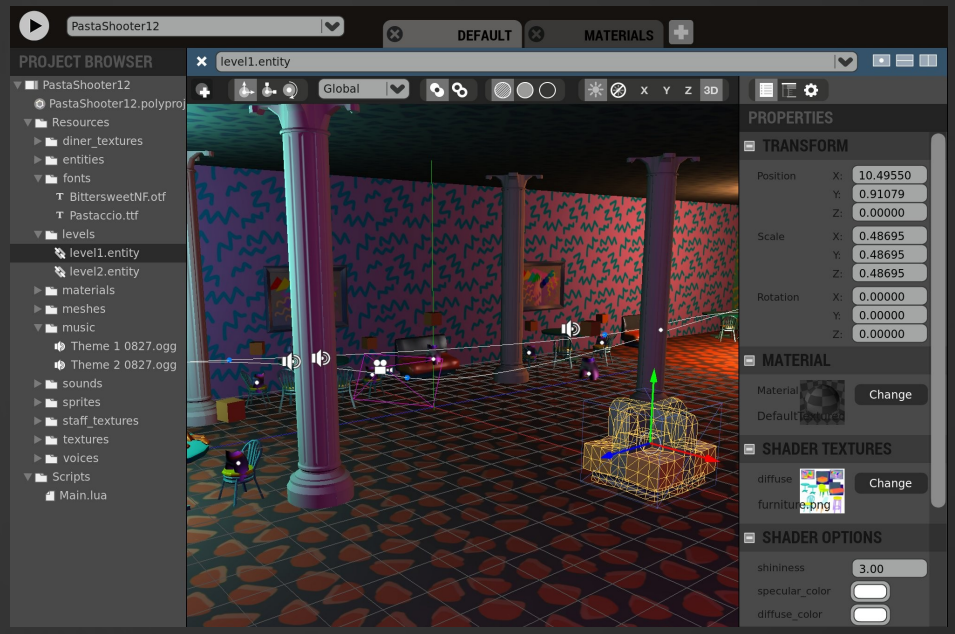
http://urho3d.github.io

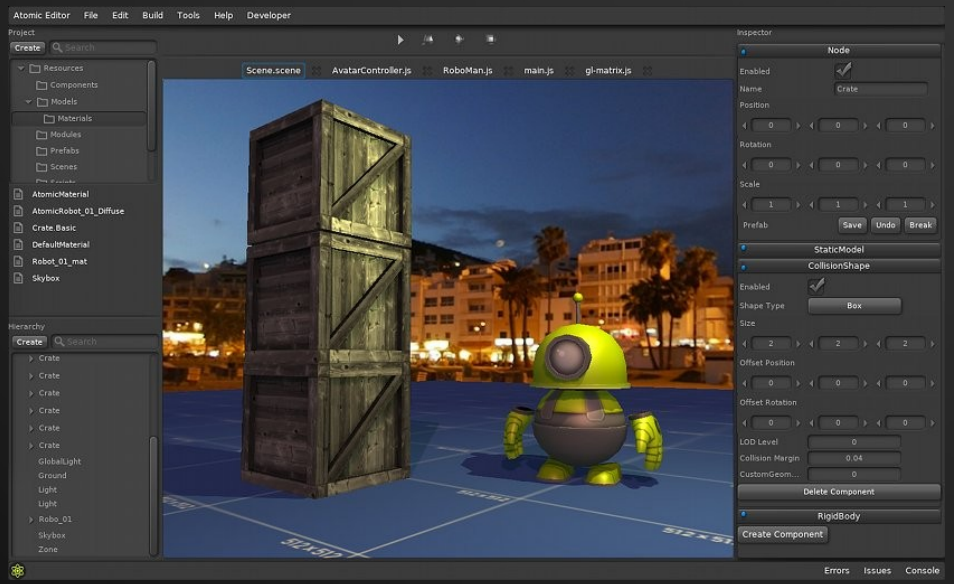
Atomic Engine

Форк Urho3D от студии Thunderbeast. Примечателен, в первую очередь, эксклюзивным Unity-подобным редактором, работа в котором донельзя интуитивна. В остальном движок практически во всем аналогичен своему прародителю. Главное отличие – отказ от AngelScript: логика приложений программируется на JavaScript. Есть поддержка форматов FBX и Blender (*.blend).

Исходники движка и редактора распространяются по лицензии MIT.

http://atomicgameengine.com





Polycode

Среда для создания игр, демонстраций и других интерактивных приложений. Работать с Polycode можно как на C++, так и в редакторе, задавая логику на языке Lua. Движок поддерживает 2D- и 3D-графику, звук, сеть, GUI. Рендеринг осуществляется через OpenGL. Есть поддержка физики на движках Bullet и Box2D. Поддерживаемые платформы – Windows, Linux и OSX (планируется также Android и iOS).

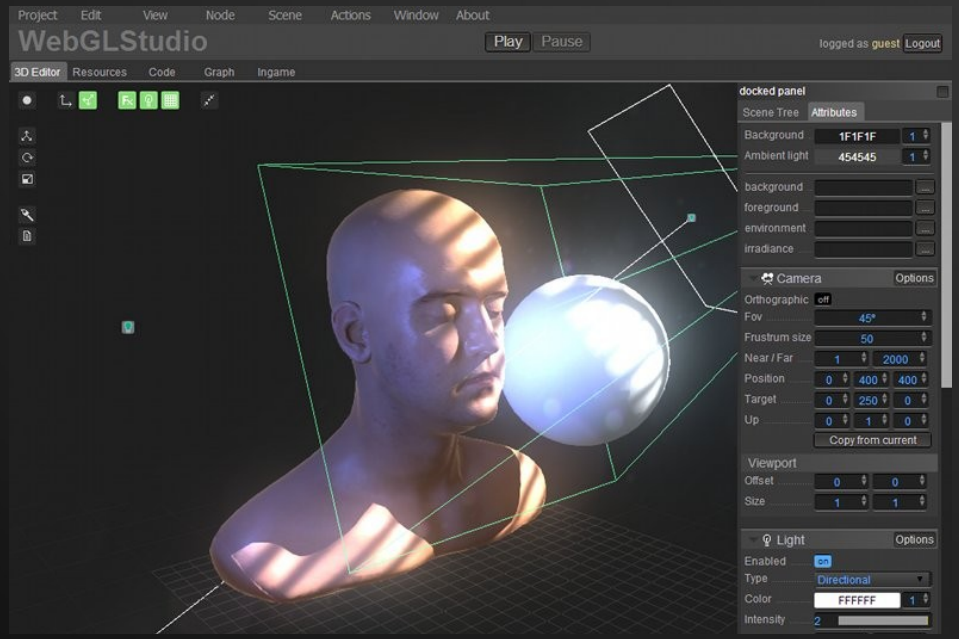
Исходники распространяются по лицензии MIT.

http://polycode.org

WebGLStudio.js

Платформа для создания интерактивных 3D-сцен прямо в веб-браузере. Позволяет редактировать сцены и материалы в визуальном режиме, создавать скрипты с описанием поведения каждого элемента сцены, редактировать шейдеры, создавать спецэффекты. Предоставляемый проектом 3D-движок (LiteScene.js), используемый для рендеринга сцен, поддерживает shadow mapping, отражения, шейдерные материалы, анимацию и различные спецэффекты. Исходники проекта распространяются по все той же лицензии MIT (интересно, почему авторы движков так любят именно ее?)

http://webglstudio.org



Ищем авторов!

Если вы – околокомпьютерный журналист (неважно, начинающий или опытный) и хотите расширить свою аудиторию, поделиться опытом и знаниями с тысячами читателей, найти единомышленников или просто высказать свою точку зрения – напишите нам! Мы будем рады опубликовать на страницах «FPS» любой материал, соответствующий тематике журнала. Это могут быть статьи, уроки, обзоры, интервью, художественная литература, авторские фото, арт, исходный код и т. д.

Вы можете выбрать одну из следующих тем: разработка компьютерных или приставочных игр, программирование, компьютерная графика, рендеринг, шейдеры, цифровой звук и музыка, геймдизайн, написание игровых сценариев, обзоры программ, движков и игровых конструкторов, отзывы и рецензии на игры, вопросы выбора лицензии, издания, дистрибьютинга или продвижения проектов, репортажи с различных конференций, выставок, встреч разработчиков и демо-пати, интересные факты из истории игр, обзоры «железа» и игровых платформ, обзоры полезных сайтов и сервисов.

Материалы принимаются в любое время на почтовый ящик редакции: **gecko0307@gmail.com**.

Прикрепленные файлы желательно архивировать в форматах zip, rar, 7z, tar.gz или tar.bz2.

Любой Читатель может стать Автором!
Желаем Вам творческих успехов!

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 Linux

Начнем наш сегодняшний выпуск большой новостью, которую большинство околоигровых СМИ почему-то обошло стороной: в феврале этого года компания The Game Creators открыла исходники своей знаменитой среды **DarkBASIC** – диалекта языка BASIC с уклоном в сторону разработки 3D- и 2D-игр.

Версия DarkBASIC Professional (а именно ее открыли авторы) работает с DirectX 9.0c, компилирует программы в машинный код, включает полноценную IDE с отладчиком. За счет поддержки расширений в виде DLL-библиотек, язык за время своего существования оброс множеством интересных дополнительных функций.

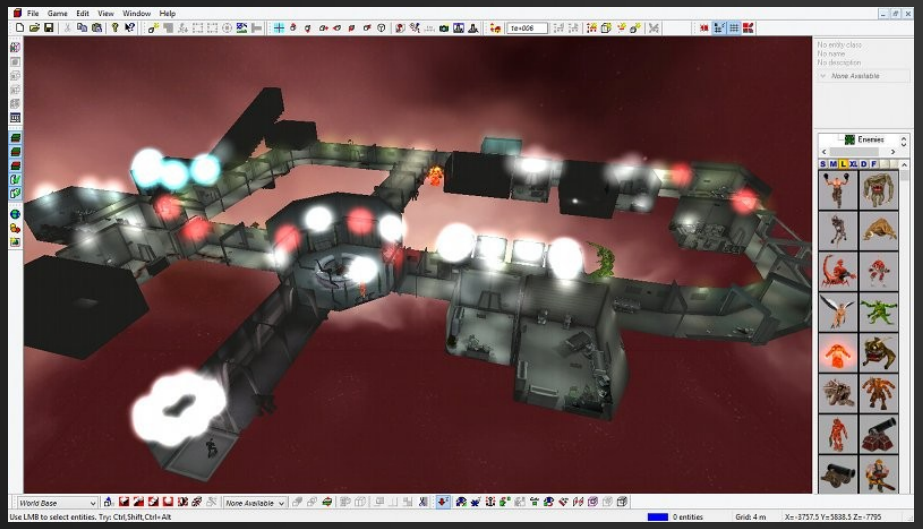
С 2010 года DBPro стала бесплатной, но распространялась под ограничительной лицензией, не позволявшей создавать с ее помощью коммерческие продукты. Теперь же компилятор, стандартная библиотека и ряд официальных расширений (таких, как система частиц, враппер PhysX, система освещения DarkLIGHTS и др.) распространяются по лицензии MIT. Единственное ограничение: исходники IDE (Synergy) выложены в неполном варианте – из-за того, что этот продукт был разработан сторонней фирмой.

Насколько DarkBASIC актуален сегодня? Конечно, конкурировать с движками AAA-класса он не в состоянии – да и сам язык BASIC чересчур архаичен, чтобы всерьез сейчас писать на нем что-то (разве только для забавы). Впрочем, исходники компилятора и 3D-движка DarkBASIC, написанные на C++, могут служить в образовательных целях – всегда интересно заглянуть в чужие наработки, чтобы узнать, как были реализованы те или иные технологии.

Ну и, конечно, просто приятно видеть, что коммерческие игровые компании продолжают старую добрую традицию, заложенную еще Джоном Кармаком, открывая свои устаревшие движки сообществу (вспомним знаменательное открытие в 2014 году исходников культового Blitz3D!).

Стоит отметить также, что ранее The Game Creators открыли другой свой замечательный проукт – конструктор шутеров от первого лица FPS Creator Classic, полностью написанный на DarkBASIC, а также множество официальных Model Pack'ов к нему. Да здравствует ретро!

https://github.com/LeeBamberTGC/Dark-Basic-Pro



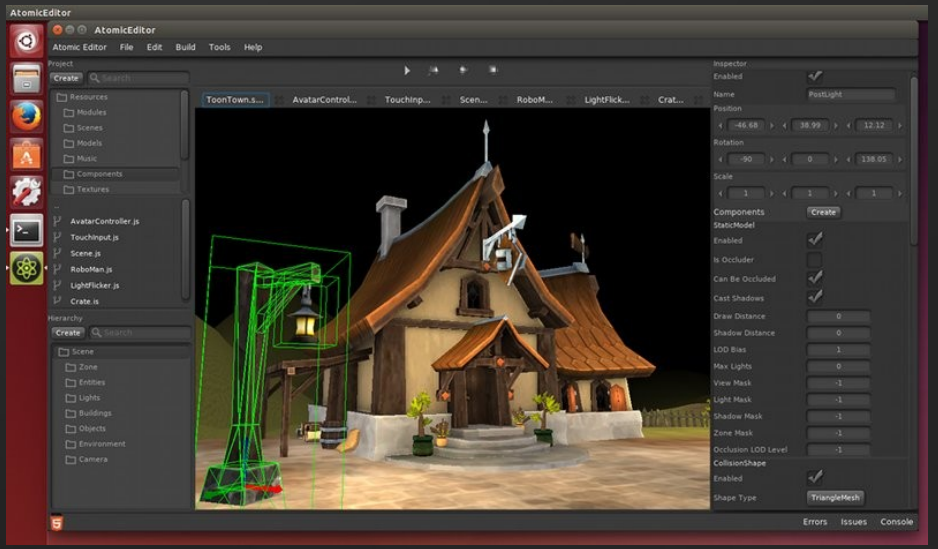
Компания Croteam, между тем, перевела в разряд открытых проектов движок **Serious Engine 1.10**, который использовался в Serious Sam: The First Encounter и Serious Sam: The Second Encounter. Эта историческая версия Serious Engine была выпущена 15 лет назад (сейчас актуальной является версия Serious Engine 4) – тем не менее, она продолжает поддерживаться сообществом, которое заинтересовано в создании модов «классических» Serious Sam.

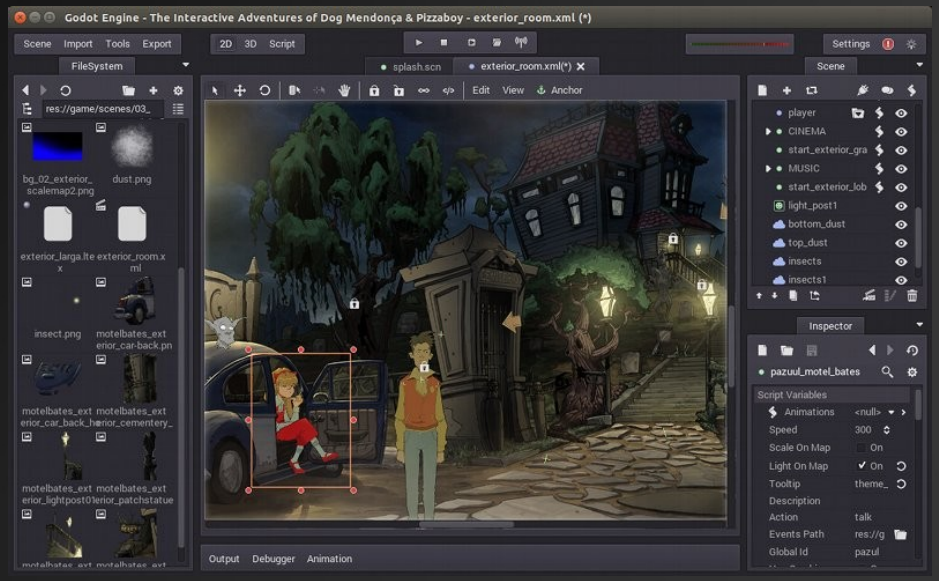
В поставку входят все компоненты, необходимые для создания собственной игры – в том числе редакторы уровней и моделей. Движок поддерживает Windows, Linux, Xbox, GameCube и PlayStation 2. Исходники распространяются по лицензии GPLv2.

https://github.com/Croteam-official/Serious-Engine

Также недавно были открыты исходники игрового движка Atomic, о котором мы уже написали выше в статье «Лучшие свободные игровые движки» – это форк движка Urho3D от студии Thunderbeast. Исходники распространяются по лицензии MIT.

http://atomicgameengine.com





Знаменитый свободный движок **Godot** обновился до версии 2.0. В этом выпуске улучшены средства создания образцов сцен, реализована возможность наследования сцен, представлен новый XML-формат для сохранения сцен и ресурсов, расширены возможности языка GDScript, добавлена поддержка аудиокодека Opus.

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

Из свободных игр за последние месяцы релизами отметились Spring и FlightGear. **Spring**, современная реинкарнация культовой стратегии Total Annihillation, обновилась до версии 101. Из новшеств отмечается возможность определения шейдеров и отрисовки карт из скриптов на языке Lua, значительное увеличение производительности, переработка поведния юнитов и многое другое.

https://springrts.com

FlightGear, свободный авиасимулятор, обновился до версии 2016.1. В рамках этого проекта, основанного еще в 1997 году, ведется работа по созданию реалистичного авиасимулятора, распространяемого по лицензии GPL. В настоящее время FlightGear моделирует более 500 самолетов, включает огромную коллекцию различных ландшафтов и аэропортов.

В новом выпуске реализован кроссплатформенный лаунчер на Qt, сокращен размер базового пакета (новые самолеты можно загрузить с помощью лаунчера), оптимизирована производительность, улучшено качество графики.

http://flightgear.org

Также был представлен первый выпуск проекта **Mininim**, в рамках которого развивается свободная реализация классической игры Prince of Persia. В текущем виде Mininim уже является полноценной и самодостаточной заменой Prince of Persia, включая необходимые игровые ресурсы и уровни.

По сравнению с оригиналом, она включает ряд новых возможностей – поддержку джойстика, игру по сети, наличие редактора уровней и многое другое. Игра поддерживает Linux и Windows, исходники проекта распространяются под лицензией GPLv3.

http://oitofelix.github.io/mininim

А эмулятор аркадных игровых автоматов MAME недавно стал полностью свободным проектом. Вместо ранее используемой несвободной лицензии MAME License выбрана модель двойного лицензирования: весь продукт теперь поставляется под лицензией GPLv2+, но большая исходных текстов также доступны и под лицензией BSD. Старая лицензия предоставляла доступ к исходным кодам MAME (т.е., была открытой), но запрещала их использование в коммерческих целях, что не соответствует общепринятым критериям свободного ПО.

Это привело к тому, что эмулятор было невозможно использовать для демонстрации старых игр в музеях (если посетители покупают билет) или для портирования игр на PC их создателями.

Процесс перелицензирования занял 10 месяцев, в ходе которых были найдены все разработчики, когда-либо передававшие свой код проекту, чтобы полу-чить от них одобрение.

Компания AMD опубликовала первую версию нового драйвера для Linux, переведенного на использование открытого модуля AMDGPU, развиваемого в рамках ини-циативы по унификации графического стека AMD для проприетарных и открытых видеодрайверов (подробнее об этом мы писали в предыдущем номере журнала).

По своим возможностям представленный драйвер примечателен поддержкой Vulkan. Также поддерживаются OpenGL 4.5, OpenCL 2.0, GLX 1.4 и средства ускорения видео VDPAU.

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

Vulkan. «За» и «против»

Конец февраля ознаменовался большим событием в мире компьютерной графики: вышла первая версия Vulkan, нового графического API от Khronos Group. Новость об этом мы публиковали в предыдущем номере журнала, в рубрике «Linux-гейминг» – хотя, на самом деле, Vulkan является полностью кроссплатформенным: на сегодняшний день этот стандарт уже реализован для Windows, Linux и Android.

Практически одновременно NVIDIA выпустила драйвер для своих видеокарт с поддержкой Vulkan, а компания Valve – соответствующий SDK. Новый стандарт ринулись осваивать не только крупные игровые студии, но и энтузиасты-одиночки вроде автора этих строк.

И вышло так, что многие в русскоязычном геймдев-сообществе оказались разочарованы: вместо «OpenGL с новыми фичами» они увидели монструозный низкоуровневый API со множеством непонятных простому смертному концепций, ничего общего с OpenGL не имеющих. Одни не усмотрели у Vulkan преимуществ перед Direct3D 12, а других сразу отпугнул пример рисования кубика в 1000 с лишним строк кода...

Но почему так вышло? Чего они ожидали, и чем является Vulkan на самом деле? Попробуем разобраться.

Для того, чтобы лучше понять все это, стоит пробежаться взглядом по истории графических API. OpenGL создавался в те времена, когда персональные компьютеры были еще довольно слабыми – ни о каком 3D на десктопе не шло и речи. Трехмерная графика оставалась уделом промышленных лабораторий и научно-исследовательских центров, оснащенных передовым на тот момент компьютерным оборудованием.

Стандарт OpenGL изначально был предложен для упрощения разработки графического ПО для рабочих станций от различных поставщиков, обладающих разной функциональностью и производительностью. Это сказалось на дизайне API – он получился достаточно абстрактным и высокоуровневым, чтобы скрыть всю «черную работу», выполняемую машиной. То, что не поддерживалось аппаратно, могло быть реализовано программно – и эта концепция вполне оправдала себя.

Наверное, в те времена мало кто ожидал, что аппартное ускорение графики оказажется весьма востребованным на потребительском рынке. Благодаря сравнительно дешевым видеоускорителям, к концу 90-х у инженеров и дизайнеров появилась возможность пересесть с дорогих рабочих станций на обыкновенные десктопы. А чтобы не пришлось при этом переписывать ПО, все компании сошлись на том, что лучше всего продолжить развивать OpenGL как единый графический стандарт для всех платформ – так оно и вышло, даже несмотря на сильную конкуренцию со стороны Direct3D.

При этом специфика API – ориентация на дизайнерский софт и научное моделирование – сохранилась на долгие годы. Видеокарты, между тем, развивались, становились мощнее и универсальнее – и, что самое главное, все больше адаптировались под нужды игр. Классический OpenGL уже не мог предоставить игровым программистам полноценный доступ к возможностям видеокарт.

Более того, балласт «обязательной» программной функциональности в драйверах начал только мешать, снижая производительность. 3-я и 4-я версии стандарта, конечно, сделали очень многое для того, чтобы решить эту проблему – но, тем не менее, инертность OpenGL с годами стала очевидной, и возникла вполне логичная потребность в более низкоуровневом API.

И вот он, этот API – Vulkan. Его главный принцип – перенести как можно больше ответственности с драйвера на программиста. Вторая идея, положенная в его основу – снизить объем данных, передаваемых покадрово из системной памяти в память видеокарты (главная проблема с производительностью у OpenGL была именно в этом). В Vulkan данные, где возможно, передаются только один раз, а изменяемое состояние сведено к минимуму. Из-за этого у Vulkan такая длинная и сложная инициализация – по сути, он весь из инициализации и состоит, а оверхеда в цикле практически нет, к чему и стремятся разработчики высокопроизводительных приложений.

Очевидно, что создатели Vulkan не собирались делать его простым с точки зрения пользователя – но он очень простой с точки зрения разработчика видеодрайверов, а это, на самом деле, куда важнее. Написать качественный Vulkan-драйвер намного проще, чем хорошую реализацию OpenGL. И это безусловное благо для всех, особенно пользователей альтернативных ОС, где проблемы с качеством видеодрайверов актуальны до сих пор.

Но нас, конечно, в первую очередь интересует клиентский код для Vulkan – как его писать, как адаптировать существующие программы для работы с Vulkan? И нужно ли это делать – есть ли смысл массово переходить на Vulkan с других API?

Те, кто ожидал, что смогут это сделать за пару вечеров, жестоко ошиблись. Дизайн существующих графических движков слишком привязан к классическому OpenGL и его машине состояний. Избавиться от «glBegin» и «glEnd» в движках не так просто, как кажется. И большой коммерческой целесообразности в этом, конечно, нет – «костыльная» поддержка Vulkan вашему движку ничего не даст, кроме необходимости лепить новые абстракции и слои совместимости. А не лепить не получится – вы не создавали свой движок с расчетом на вулкановское ручное управление видеопамятью и SPIR-V.

Наверное, появление Vulkan для игровых программистов – это знак, чтобы начать все сначала. Но перед этим стоит спросить себя: чего именно вы хотите, переходя на новый API? Если ваша цель – просто игровой проект, то возможностей OpenGL, конечно, хватит с лихвой. Смотрите на вещи трезво: вы не Valve и не id Software, вам вовсе не обязательно выжимать из видеокарты максимум. Если для вас простота программ важнее, чем оптимизация, то вряд ли использование Vulkan будет оправданным. Но если вы любите экспериментировать, и процесс для вас важнее результата – то можете смело брать Vulkan на вооружение. То же самое советуем, если вы уже являетесь опытным специалистом по традиционным графическим API – во всяком случае, новая строчка в резюме лишней не будет: специалистов по Vulkan сейчас практически нет, но они очень скоро будут востребованы.

Тимур Гафаров

Возвращение на Строггос

или Как я ставил QUAKE 2 под Windows 8

Не ошибусь, если скажу, что Quake 2 – одна из самых культовых игр за всю историю. До сих пор помню свои первые впечатления от нее – сказать, что я был поражен, значит, не сказать ничего! Но я решил написать эту статью не ради дифирамб в адрес id Software – их и без меня уже немало – а только из желания поделиться недавним опытом установки и настройки этой классической игры под современной ОС. Решил, так сказать, тряхнуть стариной...

Дело в том, что оригинальный Q2 рассчитан на ранние версии Windows (95/98) и, соответственно, на «железо» того времени. Из него невозможно выжать рендеринг в высоком разрешении – и, к тому же, без диска Quake 2 у вас не будет воспроизводиться саундтрек, а без него Q2 уже и не Q2 вовсе. К счастью, после выхода исходников игры, стали появляться современные порты, улучшающие ее с технической стороны и добавляющие совместимость с современными ОС.

Одним из лучших портов является Yamagi Quake 2, который отлично работает на 64-битных системах, включая Windows, Linux, OS X и FreeBSD.

В движок внесено огромное количество технических улучшений, но картинка и геймплей, в то же время, остаются классическими – Yamagi работает с оригинальными ресурсами Quake 2 1997 года. Одна из самых приятных фич – поддержка фоновой музыки в формате OGG Vorbis: вы можете перекодировать саундтрек Quake 2 и обойтись без компакт-диска (а можете даже задействовать в игре свою собственную любимую музыку). На сайте есть скомпилированная версия для Windows – скачайте архив и распакуйте в какой-нибудь известный каталог, например C:\Quake2. Затем нам понадобится официальное обновление Quake 2 за версией 3.20 – его можно скачать с FTP-сервера id Software:

ftp://ftp.idsoftware.com/idstuff/quake2/q2-3.20-x86-full-ctf.exe

или, например, здесь:

http://acmectf.com/downloads

Это самораспаковывающийся архив, поэтому не запускайте его, а просто распакуйте архиватором (но не в наш каталог C:\Quake2). Удалите оттуда следующие файлы:

- quake2.exe
- ref_gl.dll
- ref_soft.dll
- baseq2\gamex86.dll
- baseq2\maps.lst
- ctf\ctf2.ico
- ctf\gamex86.dll
- ctf\readme.txt
- ctf\server.cfg
- xatrix\gamex86.dll
- rogue\gamex86.dll

Объедините оставшееся содержимое с каталогом Yamagi (C:\Quake2).

Теперь вам нужен оригинальный pak-архив с ресурсами игры (pak0.pak, размер 183997730 байт). С ним чуть сложнее – в отличие от GPL-исходников, он несвободен, и распространять его не совсем легально, поэтому вам, в идеале, нужен лицензионный диск Quake 2. Но если его нет, не беда – скачивайте, например, отсюда:

http://www.mudder.net/jayna/Public/quake2/baseq2/

Думаю, подобное «пиратство» никого уже не интересует и преследоваться никем не будет. Архив нужно поместить в папку baseq2 (C:\quake2\baseq2).

На данном этапе у вас должна получиться вполне работоспособная копия игры – можете запускать quake2.exe, установить нужную вам конфигурацию видео и играть. Вероятно, придется также переназначить клавиши управления – по умолчанию стоит старомодная конфигурация. На всякий случай запускайте игру, предварительно переключившись на английскую раскладку клавиатуры (хотя в оконном режиме переключение раскладки должно работать прямо во время игры).

Наконец, музыка. Придется опять побыть «пиратом» и скачать оригинальный саундтрек где-нибудь на торрент-трекерах. Благо, авторы музыки – немецкая индастриал-группа Sonic Mayhem – выпустили ее отдельным альбомом. Вот список композиций в том порядке, в каком они шли на компакт-диске:

1. Operation Overlord
2. Rage
3. Kill Ratio
4. March of the Stroggs
5. The Underworld
6. Quad Machine
7. Big Gun
8. Decent into Cerberon
9. Climb
10. Final Showdown

Их нужно перекодировать в формат OGG и поместить в папку baseq2\music. При этом имена файлов должны выглядеть следующим образом: 02.ogg, 03.ogg, 04.ogg... 11.ogg. Нумерация начинается с 02, поскольку первый трек на Mixed Mode CD – это данные. Если музыка не воспроизводится, зайдите в настройки игры и проверьте, включена ли опция «OGG music».

Ради разнообразия можно поставить саундтрек из дополнения Quake 2: Ground Zero, также записанный Sonic Mayhem – он, на мой взгляд, ничем не хуже оригинального.

Можно создать плейлист – создайте текстовый файл baseq2\music\playlist (без расширения) и запишите в нем построчно имена файлов в том порядке, в котором они должны воспроизводиться. Yamagi также включает несколько консольных команд, при помощи которых можно управлять воспроизведением музыки прямо во время игры – ищите информацию в официальном README.txt.

Есть также возможность немного улучшить качество картинки – фанатами был создан retexturing pack, мод с улучшенными текстурами. Найти его можно тут:

http://deponie.yamagi.org/quake2/texturepack

Распакуйте его в папку baseq2 (должна получиться папка baseq2/textures с текстурами внутри).

Стоит также покопаться в настройках графики – Yamagi поддерживает анизотропную фильтрацию и мультисэмплинг. А если вам и этого недостаточно, то имеется набор консольных команд для более гибкого управления графикой – например, этими двумя командами включаются стенсильные тени:

gl_shadows 1
gl_stecilshadow 1

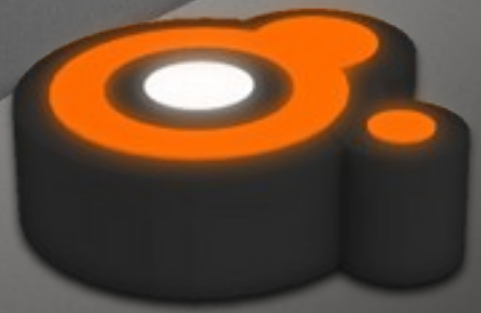
Вот, вроде бы, и все. Желаю удачи!

Тимур Гафаров

P.S.: полную историю легендарной серии Quake читайте в «FPS» №32 '14.

Это все!

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



http://fps-magazine.cf