38 2015 ДЕЛО

BLENDER:
Новости

Основы видеомонтажа

Интервью с Давидом Ревуа

Локальный контраст в GIMP

Физический движок своими руками. Часть VI

Эффективная проверка столкновений MPR

Vulkanизация

Лучшие МИСТИЧЕСКИЕ ИГРЫ

+ многое другое!

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

FPS №38

FPS – бесплатный, свободно распространяемый электронный журнал, посвященный разработке игр и другим видам цифрового творчества.

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

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

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

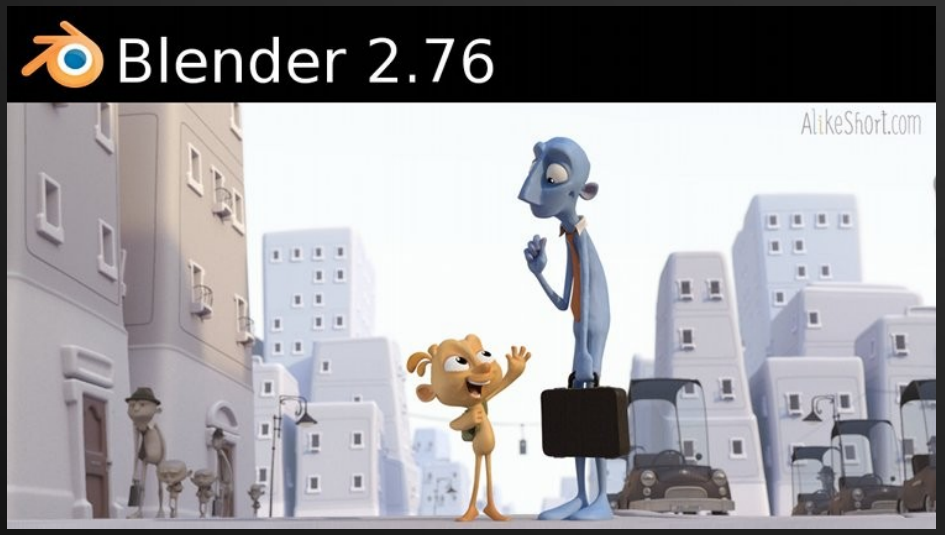
Наш сайт: **http://fps-magazine.cf**

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

Blender Новости

Вышел **Blender 2.76**. Основным нововведением данной версии является, конечно, долгожданная интеграция Open-Subdiv – технологии подразбиения мешей от Pixar. Среди других значимых особенностей можно отметить повышение производительности вьюпорта и файлового менеджера, улучшенную поддержку GPU от AMD в Cycles, новые инструменты моделирования Flatten faces и Edge offset, а также поддержку субтитров в редакторе видео – кстати, вводную статью по видеомонтажу в Blender читайте далее в этом номере журнала!

Скачать Blender 2.76 для всех платформ можно на **http://www.blender.org/download**.



23-25 октября в Амстердаме состоялась XIV международная **Конференция Blender**. По устоявшейся традиции, она прошла в здании старинного театра Де Балье в самом центре города. На конференции представили свои доклады ведущие специалисты по кинематографу, мультипликации, 2D/3D-графике и разработке игр.

Особенно пристальное внимание было уделено тренду года – виртуальной реальности. Темами других презентаций стали рендеринг в Cycles, анимация и motion-capture, композитинг и VFX, 3D-печать, использование Blender в архитектуре, промышленном дизайне и профессиональном геймдеве.

Работают, не покладая рук, авторы **Blend4Web** – свободной платформы для создания браузерных 3D-приложений с использованием Blender. В новой версии 15.08 появился граф логики приложения – на данный момент он позволяет управлять анимацией, осуществлять выбор трехмерных объектов, производить простейшие математические операции и условные переходы. Кроме того, улучшен API, физический движок, проделано множество багфиксов и оптимизаций производительности.

Кстати, известно, что Blend4Web используется в NASA: в честь 3-летия с момента высадки Curiosity на поверхность Марса, Лаборатория реактивного движения NASA подготовила при помощи Blender и Blend4Web интерактивное веб-приложение, в котором отражены наиболее значимые моменты этой космической программы.

Состоялся окончательный релиз **LuxRender** 1.5 – свободного физически корректного рендер-движка. В этой версии дебютирует новый микроядерный OpenCL-движок, компромиссный трассировщик путей, поддержка много-проходного и адаптивного рендеринга, а также интеграция движка Embree для ускорения трассировки лучей. Обновилась и интеграция в Blender: она теперь поддерживает интерактивный рендеринг во вьюпорте, как Cycles.

Студия Pixar в очередной раз порадовала OpenSource-сообщество новым анонсом, пообещав открыть исходники реализации **USD** – это событие запланировано на лето 2016 года. USD (Universal Scene Description) представляет собой формат описания сцен для обмена данными между графическими приложениями – по сути, аналог Alembic от Sony Pictures Imageworks. USD поддерживает два формата: usda – читаемый ASCII-формат, который напоминает язык описания заголовочных файлов, и usdb – бинарный формат для хранения спецификаций в БД.

В состав планируемого к открытию пакета USD, в числе прочего, будет включен встраиваемый движок 3D-визуализации и плагины для некоторых ключевых систем создания визуальных эффектов. Помимо базового программного интерфейса на C++, планируется предоставить биндинги для языка Python. На сайте проекта (**http://graphics.pixar.com/usd**) сейчас доступны демонстрационные видео, обзорная статья и прочая документация, раскрывающая базовые принципы системы.

Команда энтузиастов работает над римейком хоррор-игры **P.T.**, интерактивного тизера Silent Hills, выпущенного для PlayStation 4. Проект получил название **PuniTy**.

Весь 3D-контент для него создается в Blender, само приложение – в Unity. Есть 32- и 64-битные версии для Windows, а также для Linux и Mac OS X.

Подробнее – на **официальной странице**.

Внимание сообщества Blender также заслуженно привлек любопытный экспериментальный проект **Pheromones** – двух-минутный фильм о роли социальных сервисов в процессе создания 3D-контента. Посмотреть ролик можно на Vimeo:

https://vimeo.com/134535702

Вышел 47 номер **«BlenderArt»** – старейшего и самого известного англоязычного PDF-журнала, посвященного Blender. В этом выпуске читателей ждет много полезной информации: уроки, статьи жара «making-of», галерея работ, интервью с Тоном Розендалем и другие материалы. Выход номера, кстати, совпал с 10-летним юбилеем издания, так что поздравляем наших зарубежных коллег с этой знаменательной датой.

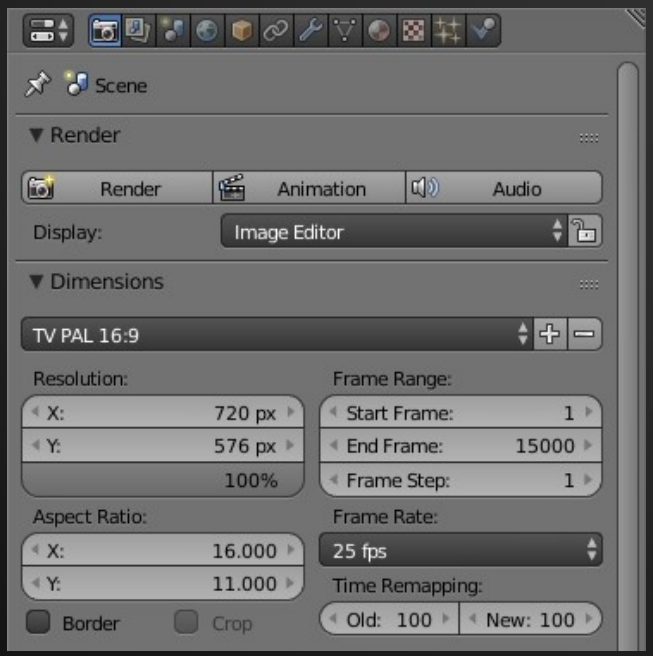
Бесплатных PDF-журналов, создаваемых энтузиастами на некоммерческой основе, в мире не так уж и много, поэтому каждый такой юбилей – это, конечно, большое событие для всех нас. Желаем «BlenderArt» долгой жизни и побольше благодарных читателей, а редакции и авторам (и лично главному редактору Сандре Гилберт) – дальнейших творческих успехов!

http://blenderart.org

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!

Видеомонтаж в Blender

Если у вас есть установленный Blender, вам, откровенно говоря, не нужен отдельный пакет для нелинейного видеомонтажа – можете даже не думать, чем заменить Adobe Premiere или Sony Vegas. Blender включает полнофункциональный редактор видео (Video Sequence Editor, сокращенно VSE), о существовании которого почему-то все забывают, когда речь заходит о бесплатных аналогах коммерческим монстрам. А, между тем, им вполне можно заменить тот же Premiere, забыв, как страшный сон, все его тормоза, глюки с падениями и порчей проектов...



Хорошей новостью для пользователей Adobe Premiere будет то, что интерфейс Blender построен по тому же принципу – те же неперекрывающиеся масштабируемые рабочие области – поэтому переход должен быть практически безболезненным.

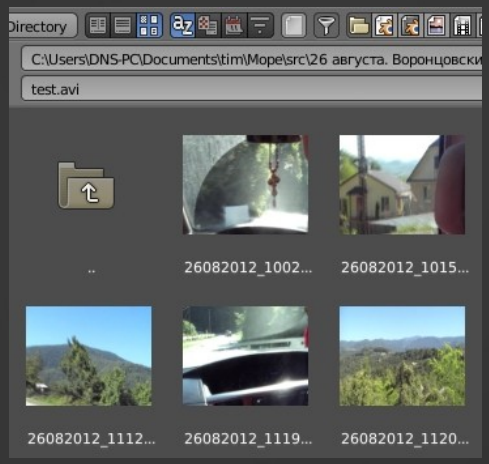
Перед началом работы рекомендую сразу задать выходной формат видео: в свойствах рендеринга на вкладке **Dimensions** выберите нужный пресет из списка или выставьте свои параметры. Частоту кадров (**Frame Rate**) укажите ту же, что и в исходных видеозаписях, которые вы собираетесь использовать.

Номер конечного кадра (**End Frame**) рекомендую сразу поставить большой – 10000-20000, в зависимости от длительности вашего фильма. Нужное количество кадров можно посчитать, умножив частоту на время в секундах. Например, для 10-минутного видео количество кадров будет 600*25 = 15000. Впрочем, номер начального и конечного кадров можно выставить и во время работы – кстати, это очень удобная возможность, если надо в целях предпросмотра отрендерить в финальном качестве не весь фильм, а только небольшой его фрагмент.

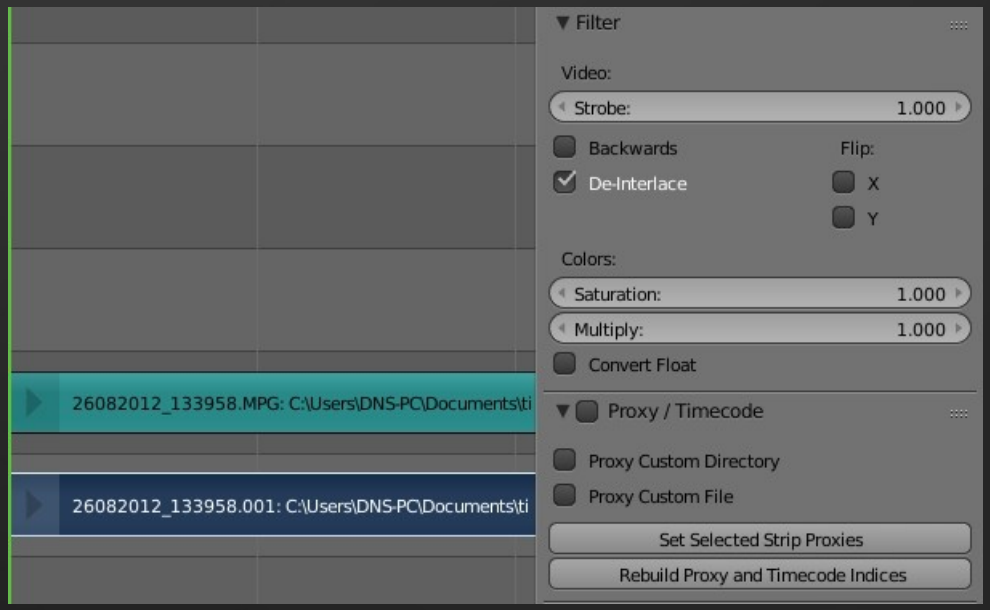
Переключите режим интерфейса на **Video Editing**. В правой части окна появится область предпросмотра видео, слева – редактор графов, внизу – секвенсор (монтажный стол) и шкала времени. На шкале времени удобно переключить единицы измерения – с кадров на секунды (**View → Show Seconds**).

Добавьте на секвенсор видеоклип: **Ctrl+A** (или меню **Add → Movie**). Выберите видео – для удобства можно переключить отображение файлов в режим миниатюр: Blender умеет создавать миниатюры не только изображений, но и видео. Добавленное видео отобразится на секвенсоре в виде двух лент (**Strip**), синей и зеленой – соответственно, для изображения и звука. Можно воспроизвести содержимое секвенсора кнопками на шкале времени.

Некоторые видеокамеры снимают в чересстрочной развертке, и такие записи на устройствах с прогрессивной разверткой (ЖК-мониторы) отображаются с неприятными артефактами в виде горизонтальных полос. Чтобы избавиться от этого эффекта, нужно сконвертировать клип в прогрессивный формат – применить так называемый деинтерлейсинг.



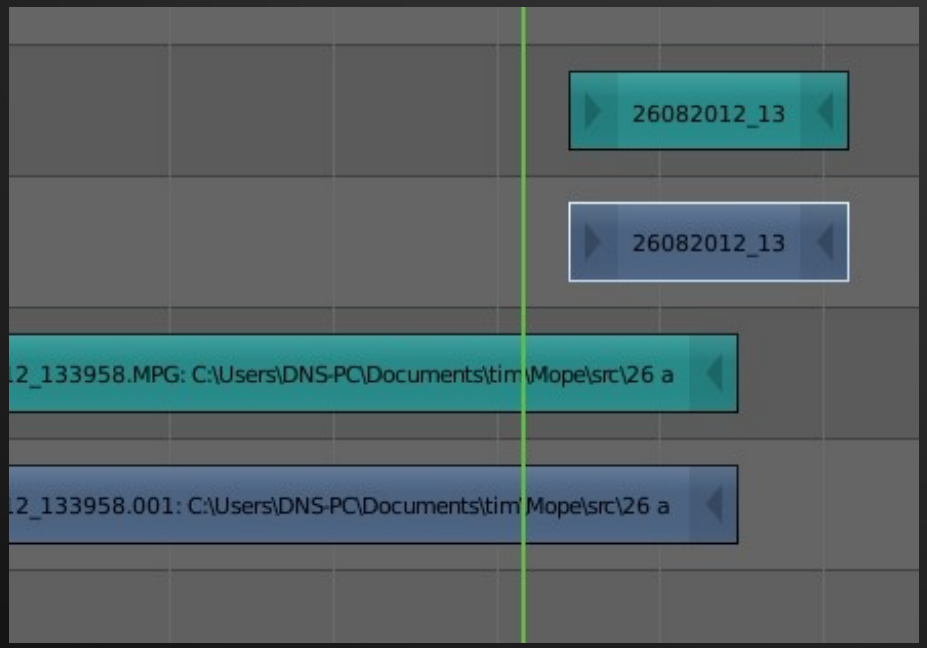
Для этого снимите выделение с лент (клавиша **A**), выделите видеоленту (синего цвета) правой кнопкой мыши и на панели свойств ленты поставьте галочку напротив **Deinterlace** на вкладке **Filter**. Можно применить деинтерлейсинг сразу ко всем лентам (**Strip → Deinterlace Movies**).



Вы можете двигаться вдоль и поперек секвенсора, зажав среднюю кнопку мыши. Колесико изменяет масштаб. Перемещать ленты можно при помощи клавиши **G**. Если сразу после нее нажать **X** или **Y**, вы ограничите перемещение ленты – только вдоль оси X или Y.

Нажатие **Ctrl** позволит выровнять позицию ленты относительно соседних – полезно, когда нужно вставить ее точно в начало или в конец другой ленты.

Разрезать ленту можно клавишей **K** – перед этим нужно установить позицию среза (текущий кадр) левой кнопкой мыши.



Можно добавить плавный переход между двумя лентами: для этого нужно расположить их на разных дорожках, одну над другой, как на скриншоте – последующая лента должна перекрывать предыдущую.

Длительность перекрытия вы выбираете на свое усмотрение. Разместив ленты, выберите сначала предыдущую, затем, зажав **Shift**, следующую (важен именно такой порядок).

Выберите **Add → Effect Strip... → Cross**. Дорожкой выше появится новая лента красного цвета, которая будет представлять собой эффект плавного перехода.

Чтобы отрендерить фильм, вернитесь на панель свойств – на вкладке **Output** выберите целевой каталог и формат, а затем на вкладке **Render** нажмите **Animation**.

Тимур Гафаров

Видеокомпозитинг в Blender:

VSE Compositor Node Proxy

Возможности VSE, как правило, недооцениваются: столкнувшись с неинтуитивным интерфейсом добавления эффектов и переходов для видеоклипов, пользователи начинают думать, что для серьезного видеомонтажа Blender мало годится. Но, на самом деле, VSE способен значительно на большее, чем просто расстановка и нарезка клипов.

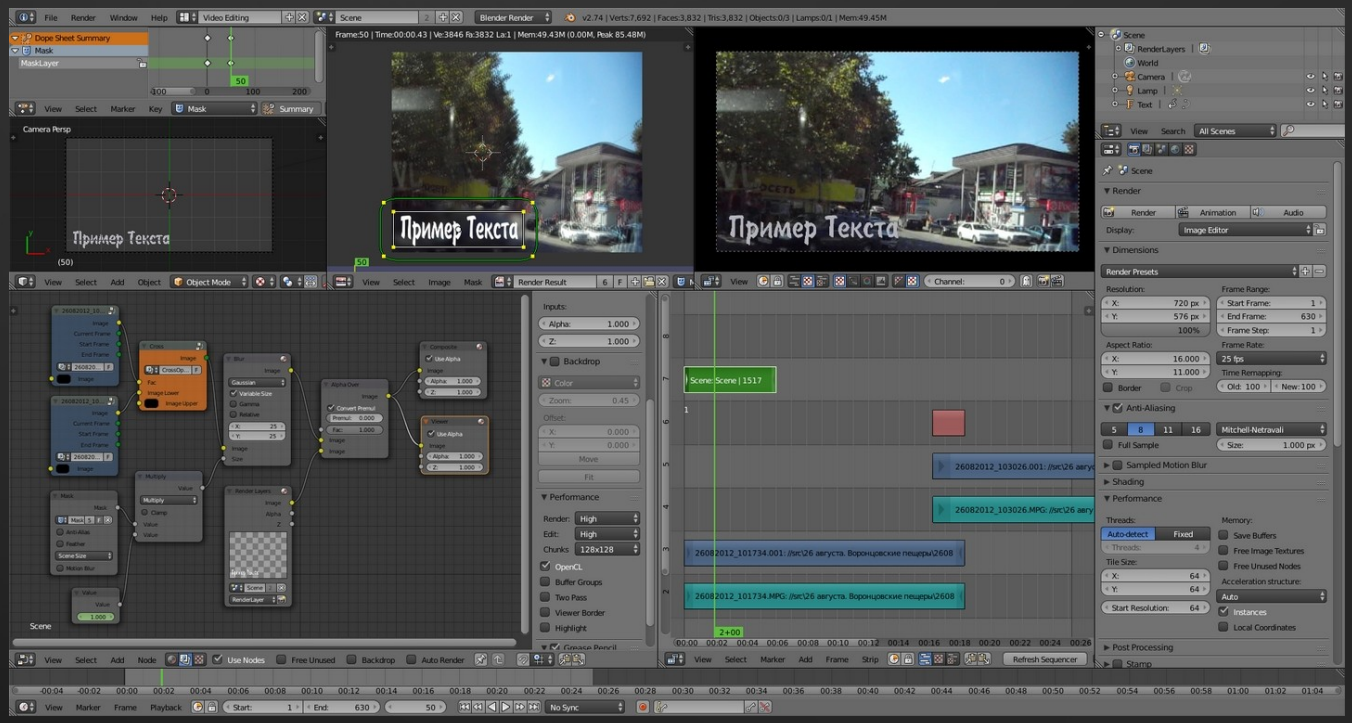
При этом везде можно использовать анимированные данные. Возможности поистине неограниченные - связка VSE/Compositing делает Blender на порядок мощнее, чем большинство других программ видеомонтажа, даже коммерческих!

Скачать скрипт можно на PasteAll:
http://www.pasteall.org/51922

Описание от автора:
http://blenderartists.org/forum/showthread.php?338248-VSE-Compositor-Node-proxy

Так, при помощи замечательного скрипта VSE Compositor Node Proxy можно организовать перенаправление текущего обрабатываемого кадра напрямую в редактор узлов композитинга – а уже там вы можете делать с кадром все, что вам угодно: накладывать на него другие изображения и видео, текст и трехмерные объекты!

Можно производить цветокоррекцию, пропускать кадр через фильтры в любых комбинациях, а при помощи векторных масок – частично менять цвет или размывать в нужных местах.



Обзор дополнений Blender

Выпуск 16

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

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****, либо в наше сообщество:*

https://plus.google.com/communities/103327597951489946649

Cabinet Library

Авторы Fluid Designer, еще одного известного инструмента для дизайна интерьеров, выпустили первый коммерческий аддон: Cabinet Library – это библиотека полностью настраиваемых компонентов для создания кухонь, ванных комнат и других подсобных помещений. Цена аддона – $249.99.

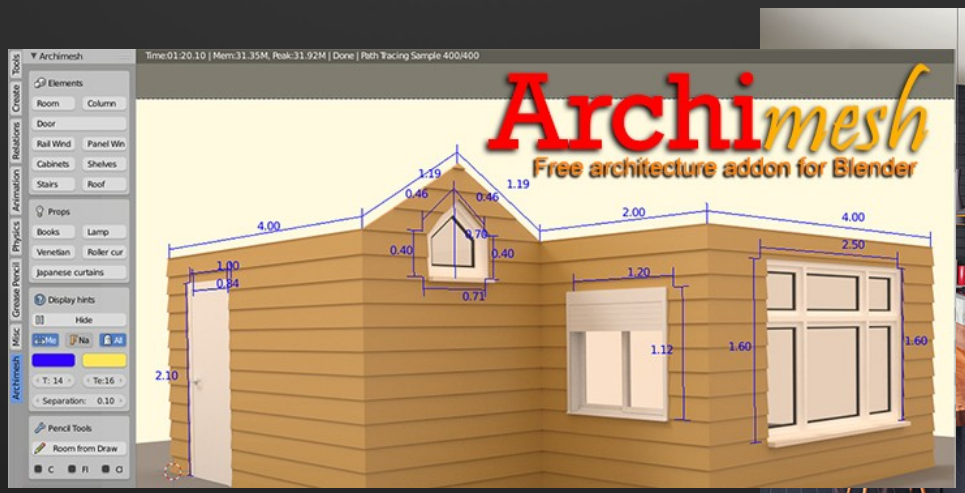
Автор: Microvellum

http://www.microvellum.com/product/cabinet-library-2

Archimesh 1.1

Замечательное дополнение Archimesh, позволяющее генерировать дома и интерьеры с мебелью, обновилось до версии 1.1.

Релиз включает удобные инструменты для измерения расстояний – кроме того, многие объекты интерьера теперь поддерживают редактирование после создания.



Автор: Антонио Васкез

https://github.com/Antonioya/blender/tree/master/archimesh

MeasureIt

Инструмент для измерений от автора Archimesh. Способен, например, отобразить расстояние между двумя вершинами одного меша, расстояние между двумя объектами, расстояние от объекта до начала координат и т.д.

Автор: Антонио Васкез
https://github.com/Antonioya/blender/tree/master/measureit

ParticleLink

Дополнение, позволяющее создавать соединения между частицами. Может здорово пригодиться анимационным дизайнерам.

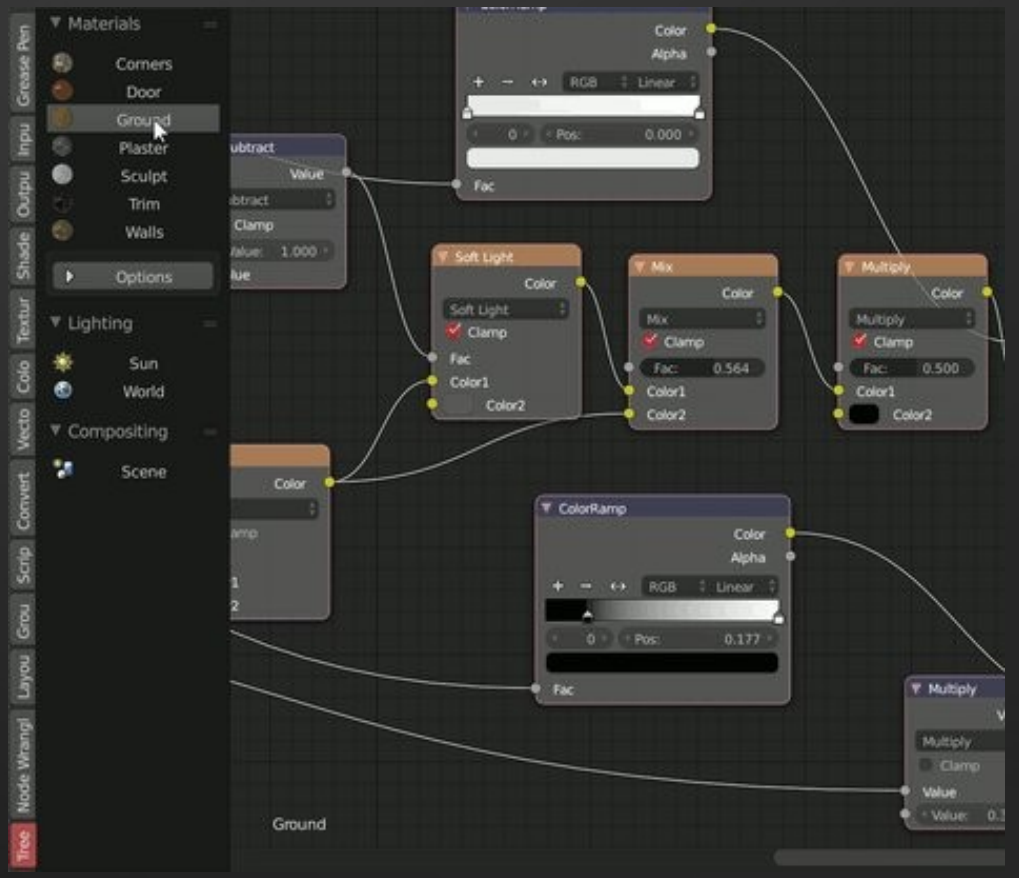
Автор: Лукас Шеллер

Ссылка

Matalogue

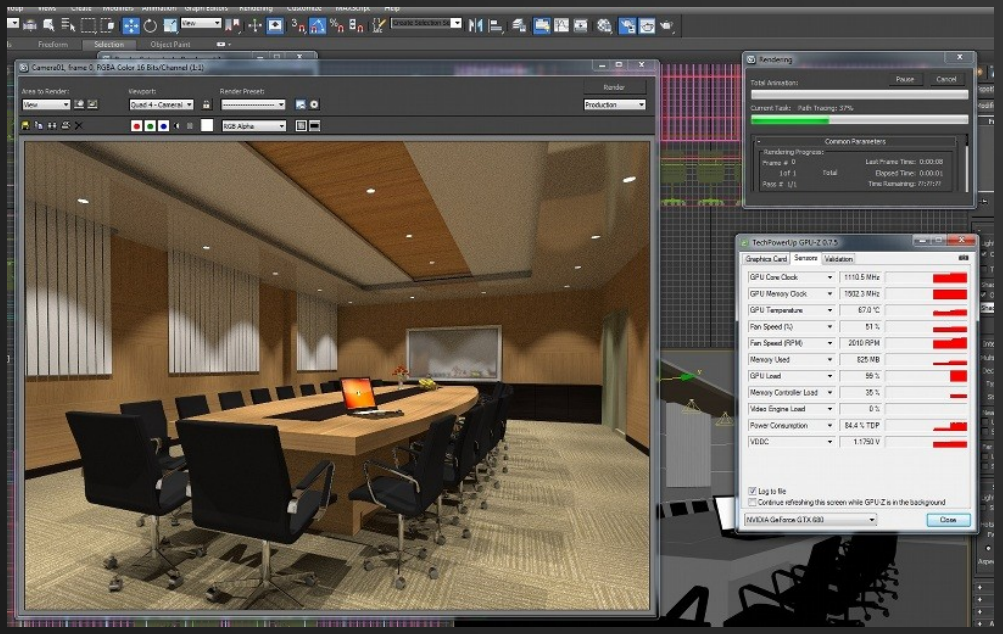
Любопытный аддон, который будет полезен тем, кто много работает в редакторе узлов – он реализует боковую панель, при помощи которой можно быстро переключаться между материалами и другими деревьями узлов.

Автор: Грег Зааль
https://github.com/gregzaal/Matalogue



Hydra Renderer

Группа российских разработчиков при поддержке Лаборатории компьютерной графики и мультимедиа факультета ВМК МГУ анонсировала свою новую разработку – высокопроизводительный физически корректный рендер-движок Hydra Renderer для 3ds Max.



В данный момент проект находится в режиме свободного бета-тестирования, принять участие в котором могут все желающие – движок распространяется совершенно бесплатно!

Сайт проекта: **http://raytracing.ru**

Реализованные возможности:

- Работает в качестве плагина для всех версий **Autodesk 3ds Max с 2013 по 2016**. Разработчики не исключают в будущем появления поддержки Blender

- Есть **standalone-версия**

- Поддержка платформ **NVIDIA CUDA**, **OpenCL**

- Движок работает **полностью на GPU** (кроме построения BVH, для этого используется CPU)

- Поддержка алгоритмов **Path Tracing**, **SPPM**, **Adaptive Manifolds for Real-Time High-Dimensional Filtering**. В настоящий момент Hydra для рендеринга использует адаптивную трассировку путей (Path Tracing) и стохастические прогрессивные фотонные карты (SPPM)

- **Fresnel**, **Translucent/Double-Sided**, **Faloff**, процедурные текстуры, Shadow Matte, IES-источники (Photometric Web), DOF, контроль экспозиции (Tone Map), каустика

- Кроме собственных материалов, плагин поддерживает **материалы Standart**, **VRay**, **Mental Ray Arch & Design** и **Corona**.

- В ближайшем будущем планируется поддержка **работы на CPU**, **Instancing**, Blend и Vray Blend, а также торрент-рендер.

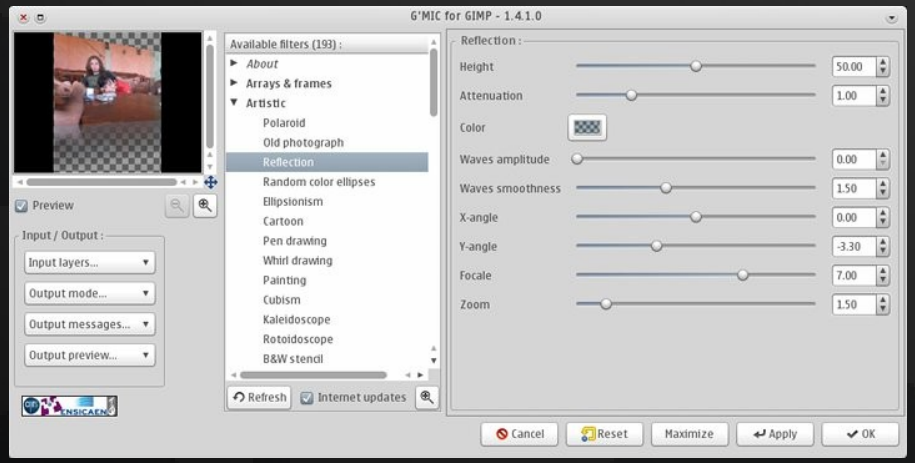
2D-графика: новости

GEGL + G'MIC

Хорошая новость от команды разработчиков G'MIC: началась работа по внедрению в плагин поддержки 16- и 32-битных каналов GEGL, которые есть в нестабильной ветке GIMP и планируются к «официальному» релизу в версии 3.0.

Напомним, G'MIC – это один из лучших плагинов для GIMP, включающий более 400 качественных фильтров. Проект существует также в виде самостоятельного фреймворка для обработки изображений, аналогичного ImageMagick или GraphicsMagick.

http://gmic.eu



Synfig на IndieGoGo

Сразу после выпуска версии 1.0 команда разработчиков Synfig запустила очередную кампанию по сбору средств на создание нового рендер-движка для программы. Основной приоритет – производительность. Планируется собрать $3000, помочь проекту можно на IndieGoGo:

https://www.indiegogo.com/projects/synfig-free-animation-software

Напомним, Synfig – это один из самых мощных свободных пакетов для работы с векторной анимацией. Из особенностей программы можно выделить средства для композитинга на основе слоев и полную поддержку контурных градиентов, которые позволяют добавлять мягкое затенение анимации без необходимости рисовать его на каждом кадре. Пакет используется для создания отечественного аниме «Моревна».

Код проекта распространяется под лицензией GPLv2. Сборки Synfig формируются для Linux, Windows и OS X.

http://www.synfig.org

Интервью с Давидом Ревуа:

Linux, Krita и цифровая живопись

В «FPS» №17 '11 мы уже публиковали беседу с Давидом Ревуа – французским иллюстратором и концепт-художником, использующим в своей работе свободное ПО. Пользователям Blender он известен тем, что был арт-директором фильма «Синтел», а также создавал концепт-арт для «Космической прачечной». В предыдущем интервью говорилось о видеокурсе «Chaos & Evolutions», работе над «Синтел», о «редакторе года» Alchemy и других свободных графических пакетах – сегодня речь пойдет, главным образом, о Krita: Давид уже несколько лет активно сотрудничает с разработчиками этого редактора. Интервью взяла пресс-служба Krita.

– Расскажи, пожалуйста, немного о себе.

– Мне 33 года и я CG-художник из Франции. Мне довелось поработать в различных областях искусства: я занимался традиционной живописью, иллюстрацией, концепт-артом, а также преподаванием. Вероятно, вы уже видели какие-то мои работы в Интернете – например, концепт-арт к открытым фильмам («Синтел», «Стальные слезы», «Космическая прачечная»), а также иллюстрации к настольным играм («Создатель вселенных» Филипа Фармера, «Lutinfernal», «BobbySitter») и книгам (Fedeylin, Club of Magic Horse).

Что могу сказать особенного о себе? Я редко принимаю чьи-то чужие идеи. Поэтому я, например, не смотрю телевизор, не интересуюсь политикой и религией. Само собой, я не пользуюсь проприетарным ПО.

– Ты занимаешься искусством профессионально или в качестве хобби? Или и то, и другое?

И то, и другое. Сейчас я работаю над собственным интернет-комиксом «Pepper & Carrot», который делается в Krita и финансируется читателями. Управление подобным проектом – достаточно непростое дело, но оно стоит затраченных усилий.

– В каких жанрах ты работаешь?

– В самых разных, но в последнее время склоняюсь к фэнтези.

– Какие художники тебя вдохновляют? Есть ли у тебя ролевая модель?

– Ролевой модели, как таковой, нет, но меня очень вдохновляют художники, стирающие границы между различными видами искусства – например, Йошитака Амано, который объединил концепт-арт, иллюстрацию и живопись.

– Как ты пришел в компьютерную графику?

– Впервые это произошло в 1992 году, когда я сел за Deluxe Paint II под MS DOS. Тогда я был еще ребенком, и мне повезло с ранних лет иметь дома компьютер. К счастью, другие члены семьи компьютера боялись, поэтому он был полностью в моем распоряжении. Подчеркиваю специально для молодого поколения, читающего эти строки: представьте – Windows 3.1, нет Интернета, графика VGA, 640x480 пикселей при 256 цветах!

– И когда же произошел окончательный переход из традиционного искусства в CG?

– В 18 лет я ушел из родительского дома, нигде особо не учился и не испытал влияния какой-либо школы. Работал сначала уличным портретистом в Авиньоне. Когда иссякал приток туристов, я брался за живопись – моя маленькая квартира была буквально забита холстами. Но работы маслом слишком долго сохли, и, если заказчику что-то не нравилось, приходилось начинать все сначала.

Где-то в начале «нулевых» я вышел в Интернет и открыл для себя цифровую живопись – я был потрясен работами CG-художников: прекрасные цвета, богатые оттенки...

До Интернета приходилось идти на выставки или покупать альбомы, чтобы увидеть новинки – а здесь я мог хоть каждый день любоваться тысячами новых произведений. Кроме того, повсеместно открывались форумы для художников, где мастера делились своими приемами и секретами.

Я, недолго думая, купил графический планшет и начал рисовать на нем. Тогда я слабо разбирался в графическом ПО, и свои ранние цифровые работы делал в Photoshop Elements, который прилагался к планшету.

Благодаря цифровой живописи, я мог экспериментировать с сюжетами, которые никогда бы не продал на холсте. Вскоре я встретил интернет-издателей, заинтересованных в моем творчестве, и начал работать все больше – сначала в полноценном Photoshop, потом в Corel Painter. В 2003 году я продал все свои последние полотна и перестал заниматься традиционной живописью.

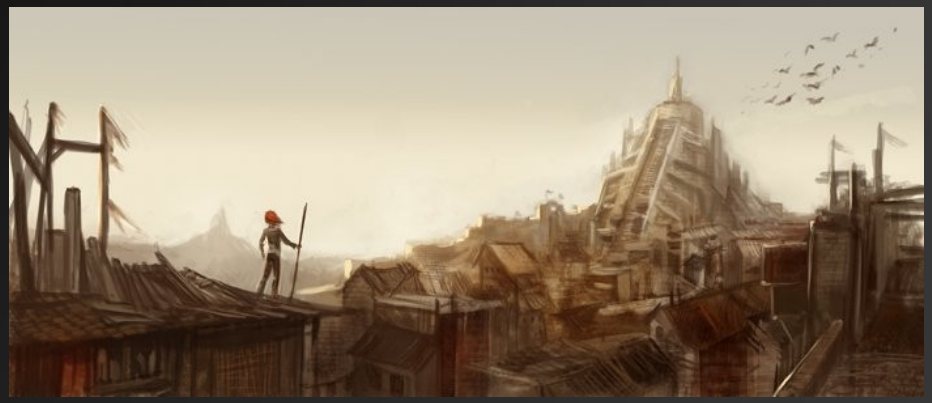
– Как ты узнал про Krita?

– Где-то в 2007 году, на одном из форумов. Тогда Krita была доступна только под Linux, а я тогда еще был пользователем Windows – на Linux я перешел только в 2009-м. Помню, как попытался установить Linux, но с первого раза так и не вышло. В итоге мне все-таки удалось установить в дуалбут Mint 4.0: к этому времени я уже увлекся оупенсорсом – в частности, Blender.

С Krita я впервые познакомился близко в 2009 году, когда работал арт-директором в «Синтел» и изучал подряд все графические редакторы под Linux (GIMP, GIMP-Painter, MyPaint, Qaquarelle, Gogh, Krita, Drawpiles). Мне действительно хотелось использовать чисто OpenSource при создании концепт-арта для фильма – эта «безумная» идея и стала поворотным моментом в моей карьере.

Перейдя полностью на Linux, моими привычными инструментами стали GIMP-Painter 2.6, MyPaint и Alchemy. Я выпустил множество уроков и несколько DVD-курсов по работе в этих пакетах.

Не все, правда, было позитивно и радужно. GIMP-Painter, форк GIMP с расширенным набором инструментов для рисования, было достаточно сложно установить в большинстве linux-дистрибутивов, поскольку они по умолчанию предлагали обычный GIMP, и две версии редактора с трудом уживались на одной системе. Обычный GIMP меня не устраивал – даже когда разработчики выпустили версию 2.8, то в нее, к сожалению, не вошли замечательные возможности из GIMP-Painter. MyPaint одно время тоже переживал не лучшие годы – главный разработчик ушел из проекта. В мире свободных графических редакторов наблюдалась стагнация – дошло даже до того, что я засомневался, был ли мой переход на Linux правильным шагом.



Тогда я начал вкладывать все больше времени и сил в развитие Krita. В то время это был не самый популярный редактор – версии 2.2 и 2.3 совершенно не годились для серьезной работы, и несколько лет я занимался тем, что исправлял баги, занимался популяризацией и поддержкой в сообществе, и самое главное – продолжал рисовать в Krita. И это был хороший выбор – мне пришлась по душе философия проекта, направленная на создание действительно хорошего инструмента для цифровых живописцев.

– А каким было твое первое впечатление?

– В 2009 году в Krita было невозможно рисовать на холсте 1000x1000. Было, конечно, много хороших функций: CMYK, кистевой движок, однооконный интерфейс, выделения, трансформации и т.д. Но все они были недоделаны и не годились для реального использования.

Проекту недоставало пользователей, бета-тестеров. Я горжусь тем фактом, что с тех пор сделал более 200 багрепостов в трекер Krita. Теперь я, по сути, один из разработчиков – недавно я сделал свой первый коммит.

– Что тебе больше всего нравится в Krita?

– Звучит по-гиковски, но это экспорт через командную строку:

$ krita in.kra -export-filename out.png

Эта возможность сильно ускоряет мою работу – например, у меня есть скрипты для сохранения документов Krita в JPEG с низким качеством, в PNG и т.д. Автоматизированный экспорт широко используется в моем последнем проекте, «Pepper & Carrot».

– Как по-твоему, что в Krita требует улучшения? Есть ли недостатки, которые тебя раздражают?

– Нужна более высокая стабильность. Я приглашаю всех пользователей, которые хотят помочь проекту, сообщать обо всех багах и падениях – не стоит ждать, пока кто-то это сделает за вас, или надеяться, что разработчики сами заметят ошибки.

– Что в твоих глазах отличает Krita от других инструментов?

– Krita, как и прочие пакеты для цифрового рисования или лепки, сильно отличается от всего остального ПО. Она должна работать в реальном времени – но, когда я рисую, то чувствую, что она не попадает в мой ритм. Наверное, поэтому все так рады знать, что повышение производительности является для команды Krita первоочередной задачей.

– Какую работу, сделанную в Krita, ты бы назвал своей любимой? И почему?

– Последний эпизод «Pepper & Carrot». Поскольку художник постоянно совершенствует свое мастерство, лучше всего о нем говорит его последняя работа. Более старые вещи – такие, как портрет Чарльза Дарвина – ближе к моему уровню 2012 года.

– Какие техники и кисти ты использовал при создании этой работы?

– Я рисовал своим обычным набором кистей. Рисовал напрямую «из головы», без использования дополнительных слоев – как в спидпейнтинге. Потом доработал детали, но старался не заглаживать работу слишком сильно.

– Где люди могут увидеть твои работы?

– В моем онлайн-портфолио:
http://www.davidrevoy.com

– Чем бы ты еще поделился с читателями?

– Вы можете со мной связаться через Twitter, Google+ или DeviantArt, пообщаться на тему Krita. У меня также появился канал на YouTube, где я размещаю видеоуроки по Krita – не стесняйтесь, комментируйте, оставляйте предложения и замечания. Я также подключен к IRC-каналу #krita на FreeNode – мой ник там «deevad». Увидимся!

Оригинал интервью:
https://krita.org

Проект, над которым сейчас работает Давид Ревуа – это свободный веб-комикс «Pepper & Carrot», все материалы которого создаются при помощи СПО и распространяются по лицензии Creative Commons Attribution 4.0. У проекта нет издателей – он финансируется читателями напрямую.

Комикс, повествующий о приключениях юной волшебницы Пеппер (в официальной локализации ее назвали Перчинка), ориентирован на широкую аудиторию – он понравится и детям, и взрослым. Отдельно стоит подчеркнуть, что «Pepper & Carrot» выходит сразу на нескольких языках – на английском, французском, немецком, русском и многих других.

Проект уже растянулся на 11 эпизодов – и, судя по всему, это только начало. Поклонники творчества Ревуа подхватили идею – и сегодня «Pepper & Carrot» уже продается в печатном варианте, есть EPUB-версия для электронных читалок, а также две фан-игры. Студия SpringBox объявила о начале работы над экранизацией комикса – нас ждет еще один качественный анимационный фильм, сделанный при помощи Blender!

Подробности – на официальном сайте:
http://www.peppercarrot.com

Помочь проекту материально можно на Patreon:
https://www.patreon.com/davidrevoy

Фотолаборатория

Локальный контраст в GIMP

Локальный контраст – это очень эффектный фильтр, позволяющий сделать фотографию более живой и четкой. Он часто используется HDR-фотографами вкупе со сведением экспозиции.

Суть локального контраста сводится к тому, что повышается контраст не изображения в целом, а отдельных деталей – при этом сохраняется крупномасштабная светотень, поэтому не придется идти на компромисс между засветкой и чернотой. Это достигается избирательным пересечением отдельных пикселей в гистограмме, чего невозможно добиться, повышая контраст с помощью кривых гамма-коррекции.

Локальное улучшение контраста работает аналогично повышению резкости с помощью «нерезкой маски» – однако в данном случае маска создается с использованием более размытого изображения.

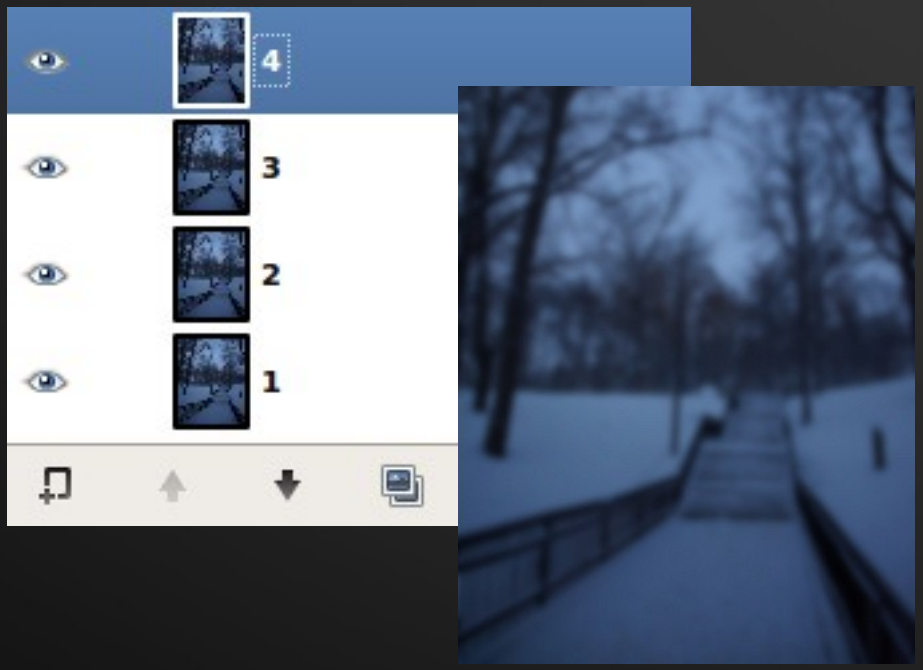
«Формула» локального контраста выглядит следующим образом:

маска = оригинал - размытая копия

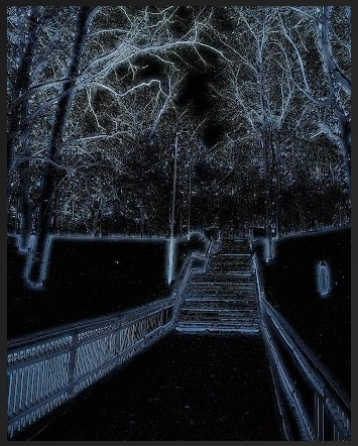
результат = оригинал с повышенным контрастом * маска

Получить эффект локального контраста в GIMP можно буквально в три шага. Попробуем применить его к фотографии справа. Откройте изображение и продублируйте слой три раза (должны получиться 4 копии исходной фотографии).

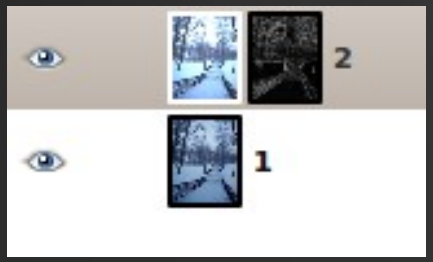
Переключитесь на верхнюю копию и примените к ней размытие. Я обычно использую для этого соответствующую операцию GEGL, так как она быстрее, чем встроенное размытие GIMP (**Инструменты → Операция GEGL... → gaussian-blur**). В этом примере я задал размер размытия 20 по горизонтали и вертикали.



Смените режим смешивания слоя на «Вычитание» и объедините его с предыдущим – должен получиться своеобразный контурный негатив. Этот слой мы назовем маской локального контраста.



Выделите слой (**Ctrl+A**) и скопируйте (**Ctrl+C**), а затем отключите его видимость.



Теперь перейдите к предыдущему слою (второй копии, если считать снизу) и повысьте ему контраст. Это можно сделать как с помощью инструмента «Яркость-Контраст», так и с помощью кривых. Добавьте слою маску (**Слой → Маска → Добавить маску слоя**).

Переключитесь на маску и вставьте скопированное ранее изображение. Вуаля! Мы получили фотографию с усиленным контрастом вдоль контуров. Чтобы усилить эффект, можно сделать маску более яркой – это лучше делать при помощи все тех же кривых гамма-коррекции.

Тимур Гафаров

Недаво сайт 500px.com, одна из крупнейших фотогалерей в Интернете, опубликовал советы от профессиональных фотографов. Не о том, какой техникой снимать или чем обрабатывать снимки, а о том, как фотография из технологического процесса становится искусством. Где искать вдохновение? Как передать эмоции через снимок? Что значит – быть хорошим фотографом?..

Элия Локарди

«Лучший фотограф – это тот, который получает больше всего удовольствия от своей работы. Следуйте своему творческому видению, своему желанию творить. Главное – любить свое дело, а все остальное придет со временем...»

Колби Браун

«Лучший совет, который мне когда-либо давали – находить страсть во всем, что делаешь. Страсть порождает вдохновение, а сильнее вдохновения нет ничего».

Тони Хьюитт

«Меня научили всегда следовать плану. Если вы запланировали съемку, но погода испортилась, не отменяйте – идите в любом случае, ведь вы не знаете, что можете пропустить!»

Ник Рейнс

«Один фотоколлекционер посоветовал мне научиться – когда не нужно снимать. Не снимайте все подряд – снимайте только удачные сцены с хорошим светом. Предпочитайте качество количеству...»

Советы профессиональных фотографов

Варина Патель

«Каждый из нас увлекается фотографией по своим причинам. Для меня это – возможность показать зрителям мой внутренний мир. Когда я снимаю, то ищу «дух места» – иногда это конкретный объект, иногда просто мимолетное чувство, эмоциональный отклик. Найти это – намного важнее, чем просто бездумно щелкать затвором. Можно ли без слов передать то чувство, когда ты просыпаешься теплым весенним утром? Можно ли передать силу летней грозы или тишину свежевыпавшего снега? Мой совет прост. Не делайте просто фотографию – затроньте душу своего зрителя...»

Джей Патель

«Помните, почему вы занимаетесь фотографией. Не ради комментариев, лайков, рейтинга, популярности на 500px. Просто потому, что вы любите фотографию.

Поэтому – снимайте то, что вам нравится, игнорируйте критику, не обращайте внимания на неудачи. Наслаждайтесь творческим опытом!»

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 2.068

Вышла новая версия референсного компилятора DMD 2.068.0. Одно из главных нововведений – директива pragma(inline), позволяющая указывать внутри кода, какие функции следует инлайнить, а какие нет. Появилась поддержка элементов кортежа в объявлении типа. Также была улучшена реализация встроенных ассоциативных массивов. Кроме того, исправлено множество багов.

Отметим также, что в будущем релизе 2.069 используемый сейчас фронтенд, написанный на C++, наконец-то будет заменен фронтендом, написаным на самом D. Таким образом, кодовая база заморожена до полной замены фронтенда – серьезных изменений в компиляторе в следующей версии не будет.

http://dlang.org/download.html

• IDE

Coedit 1 – обновление

Вышло обновление для Coedit 1, стабильной версии новой IDE для D. Обновление, в основном, касается багфиксов. Напомним, Coedit – это кроссплатформенная среда разработки для D, работающая с компилятором DMD. В Coedit интегрирован демон автозаполнения DCD, а также свой файловый браузер. Есть собственный репозиторий с пакетами, в который входят самые популярные D-библиотеки.

https://github.com/BBasile/Coedit

DDT 0.13.0

Вышла новая версия DDT 0.13.0 «Candy Kingdom». Реализована поддержка Build Targets, исправлено несколько ошибок. Напомним, DDT (D Development Tools) – это плагин поддержки D для среды разработки Eclipse.

https://github.com/bruno-medeiros/DDT
http://www.eclipse.org

Visual D 0.3.42

Райнер Шутце анонсировал новую версию Visual D – проекта по интеграции D в среду разработки MS Visual Studio. В новой версии интегрирован минимизатор кода DustMite, добавлена поддержка Visual Studio Performance Wizard, улучшена поддержка формата отладки DWARF и совместимость с Visual Studio 2015.

http://rainers.github.io/visuald/visuald/StartPage.html

• Геймдев и мультимедиа

dmech 0.2.6

Обновился до версии 0.2.6 физический движок dmech. Теперь движок может быть скомпилирован как динамическая библиотека (как под Windows, так и под Linux) – у dmech появился C-интерфейс, позволяющий использовать его с любым языком, способным линковаться с C-библиотеками (прилагается пример использования dmech в программе на C). Пока, правда, реализован далеко не весь API, но начало положено. Кроме того, добавлена поддержка деактивации и возможность отключить трение для тел. К dmech теперь прилагаются три демо-приложения, которые можно собрать под Windows, Linux и Mac OS X (включая 64-битные системы) – релиз включает соответствующие бинарные сборки. Также появилась небольшая документация к движку в виде уроков.

https://github.com/gecko0307/dmech

dlib 0.7.0

Вышло подряд несколько обновлений коллекции библиотек dlib, в том числе очередной крупный релиз 0.7.0. Начиная с версии 0.6.1 в dlib.core.memory доступен встроенный профайлер памяти. Состоялось серьезное обновление dlib.image – с версии 0.6.2 декодер PNG больше не отражает изображение по вертикали. В версии 0.6.4 обновился пакет dlib.math, были оптимизированы перемножение матриц и доступ к элементам вектора. Появился новый контейнер std.container.dict – универсальный ассоциативный массив.
Версия 0.7.0 включает реализацию кроссплатформенных потоков выполнения (threads), пакет обработки текста (dlib.text), обновленный парсер XML и новые независимые от сборщика мусора контейнеры. Исправлено несколько серьезных ошибок. Также проект обзавелся поддержкой сервиса непрерывной интеграции Travis-CI.

https://github.com/gecko0307/dlib

• Веб

Vibe.d 0.7.24

Вышла новая версия веб-фреймворка Vibe.d. Из основных нововведений релиза стоит отметить поддержку D 2.068 и генератор веб-интерфейсов vibe.web.web.

http://vibed.org

MQTT

Клиент MQTT (Message Queue Telemetry Transport) для Vibe.d. MQTT – это простой и легковесный сетевой протокол обмена сообщениями, работающий поверх TCP/IP.

https://github.com/chalucha/vibe-mqtt

Asynchronous

Реализация асинхронного программирования, порт библиотеки asyncio для Python. Асинхронная концепция программирования заключается в использовании функций с отложенным результатом (доступным не сразу же, а через некоторое время). В асинхронном программировании все функции – неблокирующие, то есть, их вызов не приводит к прерыванию текущего процесса. Каждая функция выполняется в отдельном потоке, а результат передается в функцию-обработчик, которая вызывается автоматически в нужное время.

https://github.com/dcarp/asynchronous

• Системная разработка

DerelictCocoa

Анонсирован Derelict-биндинг к Cocoa, объектно-ориентированному API операционной системы Mac OS X.

https://github.com/p0nce/DerelictCocoa

DDWM

D-порт DWM, минималистичного тайлового менеджера окон под X Window System.

https://bitbucket.org/growlercab/ddwm
http://dwm.suckless.org

• Наука

Sargon

Набор из нескольких библиотек: реализация алгоритма сжатия LZ77, чисел с плавающей запятой половинной точности (IEEE 754) и другие.

https://github.com/DigitalMars/sargon

SciD переехал

Репозиторий проекта SciD переехал с dsource на GitHub. Напомним, SciD – это библиотека вычислительной математики для D, включающая методы численного интегрирования и дифференцирования, инструменты линейной алгебры (в том числе удобный интерфейс к LAPACK), а также средства для решения нелилейных уравнений.

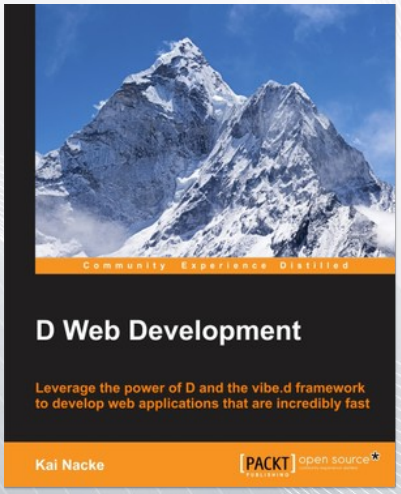
https://github.com/DlangScience/scid

Литература •

«Programming in D» на бумаге

В продажу поступила бумажная версия знаменитой книги Али Цехрели «Programming in D», перевод которой на английский мы так долго отслеживали. В последней редакции книга учитывает все нововведения D 2.068.
Цена на Amazon.com – $28.50. Электронные версии (PDF, EPUB, AZW3) по-прежнему доступны бесплатно на сайте автора.

http://ddili.org/ders/d.en



«D Web Development»

На Packt Publishing доступна для предзаказа новая книга по D – «D Web Development», посвященная веб-разработке и, в частности, работе с фреймворком Vibe.d.

Цена электронной версии – €16.00, бумажной – €24.99. Выход намечен на январь 2016 года.

https://www.packtpub.com/web-development/d-web-development

Искусственный интеллект **OBI**:
инновация или мошенничество?

В начале сентября ньюсгруппу digitalmars.d.announce всколыхнуло необычное сообщение: некий Асам Обиома (Asame Obiomah) заявил о разработке революционного искусственного интеллекта на языке D под кодовым названием OBI (Organic Big data Intelligence). Сам он называет его громким термином «искусственное сознание»: по заверениям автора, его система способна не только к обучению и принятию решений, но и к... эмпатии и переживанию эмоций!

Обиома утверждает, что OBI, в перспективе, можно будет использовать в качестве персонального компьютерного помощника для управления повседневными задачами, взаимодействие с которым осуществляется на человеческом языке. Цель проекта – ни много ни мало, «поселить» искусственный интеллект в каждом доме и в каждом электронном устройстве!

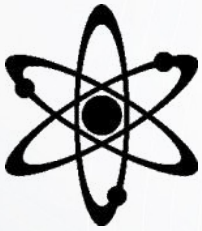
Некоторые участники сообщества D восприняли эту новость как некую форму мошенничества – раздувание IT-сенсации с целью поиска инвесторов (сразу родилась соответствующая шутка – «нигерийский софт»). Действительно, подобные случаи – не такая уж редкость: вспомним знаменитый проект Unlimited Detail, технологию рендеринга, которая якобы произведет революцию в компьютерной графике. На деле же она является лишь разновидностью воксельной графики, которая, конечно, вряд ли в ближайшее время вытеснит полигональную. Но это не помешало авторам разрекламировать Unlimited Detail до такой степени, что проект получил крупный государственный грант в два миллиона долларов!

Что же перед нами на этот раз? Сайт проекта, **http://www.okeuvo.com**, не изобилует подробностями – там размещена лишь общая информация и краткие сведения об авторе.

На странице «About» указано, что Асам Обиома является призером Microsoft DesignIT 2006 года. Такая премия действительно существует, но ничего о лауреате с таким именем мы не нашли. И вообще в Интернете никакой конкретной информации об этом человеке нет – никаких научных публикаций или проектов в области ИИ. Даже профиль Обиомы на LinkedIn почему-то оказался удален. Все это, естественно, не может не выглядеть подозрительно...

Да и заявлять о «искусственном сознании» само по себе чересчур опрометчиво – с учетом того, что сознание на сегодняшний день является малоизученным феноменом, как и многие другие свойства человеческого интеллекта.

Впрочем, делать какие-либо выводы пока рановато. Первый релиз OBI назначен на 2016 год: поживем – увидим...



Физический движок своими руками

Часть VI

Мы продолжаем цикл статей о разработке физического движка, начатый в FPS №№ 24, 25, 26, 27 и 31. Мы рассмотрели ключевые компоненты игровой физики: интегратор, обнаружение столкновений и солвер с трением и корректировкой позиций. На самом деле, математический аппарат, лежащий в основе симулятора твердых тел, достаточно тривиален, чего нельзя сказать о практической реализации. Наивная реализация, приведенная в предыдущих статьях нашего цикла, конечно, не может считаться эффективной. Рассмотрим типовые приемы, которые используются в большинстве физических движков для обеспечения стабильности и максимальной производительности.

Кэширование контактов

Этот компонент относится к технической стороне реализации и прямого отношения к физике не имеет, но без него не обходится ни один сложный физический движок. Дело в том, что контакты, полученные на этапе обнаружения столкновений, нужно как-то хранить в памяти, чтобы затем передать солверу. Более того, нужно учитывать, что некоторые контакты существуют на протяжении длительного времени – например, если тела покоятся одно на другом.

Иными словами, такой контакт после решения не должен удаляться из памяти. Это необходимо для «горячего» запуска солвера (warmstaring) – методики, позволяющей заметно сократить количество итераций, за которое сойдется решение системы. Мы обязательно рассмотрим ее в одной из следующих статей.

Для хранения контактов используется специализированный контейнер – одним из неплохих вариантов является хэш-таблица, оптимизированная для работы в реальном времени. Такая таблица должна ассоциировать контакт (или, в общем случае, множество контактов) с парой тел.

В качестве хэш-функции при этом используется функция нумерации пар (pairing function) – функция, уникальным образом сопоставляющая двум числам одно. Одна из лучших – это функция Шудзика (Matthew Szudzik, «An Elegant Pairing Function»), которая выглядит следующим образом:

```
uint szudzikPair(uint a, uint b)
{
    return a >= b ? a * a + a + b : a + b * b;
}
```

Параметрами a и b в нашем случае будут уникальные идентификаторы двух тел. Можно просто присваивать им порядковый номер при создании, а можно генерировать идентификатор каким-то другим способом – скажем, брать адрес в памяти или вычислять хэш от имени. Если тела создаются и хранятся централизованно, то лучше, конечно, использовать порядковый номер.

Реализация самой хэш-таблицы выходит за рамки данной статьи – в Интернете можно найти достаточно много информации и исходников по этой теме.

Контактные площадки

Когда происходит столкновение тел ребрами или гранями (например, один бокс лежит на другом), одной точки контакта для адекватного описания ситуации становится мало, нужно строить приближенное контактное пятно – контактную площадку (или так называемое контактное многообразие – contact manifold), состоящую из двух и более точек. Эти точки затем последовательно решаются в солвере. Нахождение подобной структуры – отдельная непростая задача, которую все решают разными методами. Главная проблема тут заключается в том, что алгоритмы обнаружения столкновений (ОС) дают нам всего одну точку на пару столкнувшихся тел.

Перспективный метод, который естественным образом «дружит» с алгоритмами ОС – persistent contact manifold. Он получает контакты по одному и постепенно, в несколько шагов, строит площадку.

Основная идея тут заключается в том, что решение первого найденного контакта приведет к появлению второго, решение второго – к появлению третьего, и так далее, пока не будут найдены все крайние точки, описывающие контактное пятно. Когда точки начнут повторяться, алгоритм должен это обнаружить и перестать добавлять их в площадку.

Возникает вопрос – когда именно удалять точки из площадки? Ведь тела рано или поздно перестанут сталкиваться. Один из вариантов – хранить в описании точки контакта ее координаты относительно позиции двух тел на момент обнаружения этой точки. После интегрирования, когда тела переместятся в новые позиции, нужно произвести проверку: если точка относительно новых позиций тел сместилась на какое-то пороговое расстояние, то, значит, в этой точке тела уже не сталкиваются, и ее можно удалять.

У этого достаточно простого и эффективного метода есть недостаток – в случае, когда одно тело лежит на другом, как определить, покоится ли оно или скользит по поверхности? Если скользит, то алгоритм воспримет смещение точек относительно тел как разрыв контактов – и удалит их, хотя, по идее, этого происходить не должно. Некоторые авторы утверждают, что на практике такая погрешность пренебрежима – при скольжении контакты просто будут исчезать и сразу появляться снова (из-за взаимопроникновения вследствие силы тяжести). В большинстве случаев ничего страшного при этом не произойдет, но разрыв контактов в таких ситуациях может сводить на нет преимущества «горячего старта» в системах с множеством лежащих друг на друге тел – например, когда стопка из боксов скользит по наклонной поверхности.

Существует, впрочем, и другой метод построения контактной площадки, предложенный Дирком Грегориусом, одним из разработчиков Havok, и реализованный нашим соотечественником Александром Санниковым. Он позволяет получить всю площадку за раз и основывается на опорной функции выпуклых тел. Подробнее – в статье на Gamedev.ru:

http://www.gamedev.ru/code/articles/convex_collisions

Избавляемся от дрожания

Дрожание тел (или так называемый джиттеринг) – это неприятный артефакт, присущий импульсной физике. Визуально оно проявляется как заметная вибрация или подергивание покоящихся тел, из-за чего они начинают самопроизвольно смещаться или поворачиваться. Это может происходить из-за передачи телам лишнего или неправильного импульса, либо из-за погрешностей корректировки позиции. Также очень часто дрожание возникает вследствие нестабильности солвера в системе с множеством ограничений или большими разницами масс.

Полностью устранить дрожание в импульсном движке невозможно. Но в большинстве ситуаций удается сделать тела более стабильными путем повышения итераций солвера. Кроме того, с дрожанием покоящихся тел можно бороться при помощи очень простого хака – введения нижнего порога скорости: если скорость тела по модулю становится ниже этого порога, она считается равной нулю и не учитывается при интегрировании.

Порог обычно подбирается экспериментально, в зависимости от масштабов симулируемой сцены.

Если порог слишком высок, то тела могут останавливаться в неестественных состояниях – например, замирать на ребре. А если порог слишком низок, дрожание может не исчезнуть.

Статические меши

В играх удобно представлять статическую геометрию не телами, а мешами – сетками треугольников. Но вот беда – меш, представляющий, к примеру, интерьер или ландшафт, не является выпуклой геометрией, и к нему не подходят традиционные алгоритмы ОС (такие, как GJK/EPA или MPR). Но отдельный треугольник – выпуклый, поэтому можно реализовать ОС с мешем путем перебора составляющих его треугольников и проверки с каждым из них.

Для больших мешей прямой перебор, конечно, неэффективен – куда лучше построить из меша иерархическую структуру, которая будет сужать область перебора до минимума. Из подобных структур особенно популярны octree (октодерево) и его обобщенный вариант BVH (иерархия ограничивающих объемов).

Все пространство, которое занимает меш, рекурсивно разбивается на некоторое число подпространств (параллелепипедов). Каждый параллелепипед становится узлом дерева – узел хранит треугольники, входящие в соответствующий узлу параллелепипед. Разбиение продолжается, пока не будет достигнута определенная глубина рекурсии.

После этого листья дерева будут содержать итоговые списки треугольников, перебор которых в реальном времени уже выглядит вполне реалистично. Обход дерева, предполагающий рекурсивную проверку на пересечение двух AABB (ограничивающих параллелепипедов узла и выпуклого тела) также займет пренебрежимо малое время (в худшем случае, O(n), где n – высота дерева).

Подробнее о построении BVH можно прочитать в FPS №22 '13 («Оптимизация проверки столкновений»).

Но на этом дело не закончено. Если наше выпуклое тело – скажем, сфера – катится или скользит по плоской поверхности, составленной из нескольких треугольников, то неизбежны заметные «застревания» в местах стыков ребер. Этот неприятный артефакт можно устранить, если решать в данной ситуации только один контакт из двух – например, с наибольшей глубиной проникновения.

Кэш процессора

Одним из «узких мест» в компьютерных вычислениях являются операции чтения и записи памяти: большую часть своего времени процессор тратит именно на них. Чтобы повысить эффективность вычислений, CPU обычно снабжается кэшем – собственной очень быстрой памятью небольшого объема, в которую помещаются копии часто используемых данных. Когда процессору нужно обратиться к данным в памяти, он сначала проверяет, доступна ли их копия в кэше. Если да, то процессор считывает их оттуда, что намного быстрее обращению к основной оперативной памяти.

При программировании высокопроизводительных приложений, таких, как физические движки, важно научиться писать так называемый «cache-friendly» (дружественный с кэшем) код. Например, данные в памяти должны располагаться «плотно» – то есть, строго друг за другом.

Дело в том, что процессор, при чтении данных из памяти, автоматически загружает в кэш некоторое количество данных, идущих после. Это позволяет, например, ускорить обработку массивов – элементы массива, если они достаточно малы, поступают в кэш порциями. Размер этой «порции» (она называется строкой кэша – cache line) зависит от модели процессора – наиболее часто встречаются строки кэша в 32, 64 и 128 байт. Чтобы с выгодой использовать строки кэша, рекомендуется, например, в структурах объявлять поля в порядке уменьшения размера типов.

Однако даже если соседствующие данные умещаются в этот объем, не каждая группа байт может быть кэширована по причине дополнительного ограничения, известного как адресное выравнивание (address alignment). Группа соседствующих байт может помещена в строку кэша тогда и только тогда, если ее начальный адрес выровнен по границе, равной размеру строки. Например, для 32-байтной строки такими границами будут 0, 32, 64, 95 и т. д.

В процессорах Intel обычно используются 64-битные строки, поэтому есть смысл выравнивать структуры директивой align(64).

Тимур Гафаров

Проверка столкновений
алгоритмом MPR

Проверка столкновений – одна из самых важных частей игрового движка. В том или ином виде, обнаружение пересечения между объектами – например, между двумя параллелепипедами или между сферой и лучом – используется в любой динамической игре. Особенно важно точное обнаружение пересечения в играх жанра Action – таких, как шутеры, гонки, файтинги и пр.

Для нахождения факта пересечения двух выпуклых тел существует несколько популярных алгоритмов. Проще всего обнаружить пересечение двух сфер – если сумма их радиусов превышает расстояние между их центрами, то сферы пересекаются. С другими телами дело обстоит уже гораздо сложнее. Для прямоугольных параллелепипедов обычно используют алгоритм **SAT** (теорема о разделяющей оси). В современных физических движках используются универсальные алгоритмы, работающие для любых выпуклых тел – **GJK** и **MPR**. Они во многом похожи – основаны на одной и той же теоретической базе. В этой статье мы рассмотрим MPR, как новейший и самый перспективный. К тому же, по GJK в Интернете и так достаточно информации, а MPR не описывается практически нигде, кроме как в книге «Game Programming Gems 7». Русскоязычных описаний этого алгоритма до сих пор и вовсе не существовало, что я и хочу исправить данной статьей.

Алгоритм MPR (**Minkowski Portal Refinement**) основан на постулатах так называемой выпуклой геометрии – раздела геометрии, изучающей выпуклые множества в евклидовом пространстве, которыми являются такие тела, как сфера, параллелепипед, цилиндр и др. Для экономии места, не буду здесь давать определение выпуклости, так как его можно найти в учебниках или в Википедии.

Одна из ключевых операций выпуклой геометрии – это **сумма Минковского**. Сумма Минковского множеств A и B – это сумма всех элементов A и всех элементов B. Сумма Минковского выпуклых множеств обладает некоторыми интересными свойствами, на которые и опираются алгоритмы обнаружения столкновений – в частности, они оперируют **разностью Минковского** (разностью всех элементов A и всех элементов B), которое также называют конфигурационным пространством препятствий (**Configuration Space Obstacle, CSO**).

С разностью Минковского связана особая теорема, которая используется в MPR и GJK. Звучит она следующим образом: **если два выпуклых множества в евклидовом пространстве пересекаются, то их разность Минковского содержит начало координат**. Доказать это несложно. Если два выпуклых множества пересекаются, у них по определению должен быть хотя бы один общий элемент – назовем его X.

Следовательно, их разность Минковского должна содержать элемент X-X – иными словами, нулевой элемент, или, в случае с точками евклидового пространства, начало координат.

Если мы можем высчитать CSO для двух выпуклых оболочек, проверка пересечения между ними сводится к определению, включает ли получившаяся оболочка CSO начало координат. Но в большинстве случаев явным образом найти A-B не представляется возможным – в частности, для таких тел, как сфера или цилиндр – ведь их оболочки содержат бесконечное количество точек. Но, к счастью, для решения нашей задачи это необязательно. Вместо этого мы можем найти конечное подмножество CSO, которое гарантированно включает в себя начало координат.

Простейшим таким подмножеством для двумерного случая будет треугольник, для трехмерного – тетраэдр. Обобщение треугольника и тетраэдра обозначают термином **симплекс**. Построить симплекс внутри выпуклой оболочки можно при помощи **опорной функции** (support function) – эта функция, заданная на множестве, сопоставляет произвольному направлению самую дальнюю точку множества в данном направлении относительно центра. Например, для сферы эта функция будет выглядеть как $sup(n) = n * r$, где r – радиус сферы, а n – направление (единичный вектор). Таким образом, мы можем строить симплексы, не имея на руках самого множества, а имея только его математическое описание, выраженное опорной функцией.

Очень важную роль здесь играет замечательное свойство суммы Минковского – ее опорная функция равна сумме опорных функций слагаемых множеств:

$$sup_{A+B} = sup_A + sup_B$$

То же самое справедливо и для CSO:

$$sup_{A-B} = sup_A - sup_B$$

Следовательно, мы можем оперировать только опорными функциями, что позволяет использовать один и тот же алгоритм с любыми возможными выпуклыми телами, а также их суммами и разностями Минковского в любых комбинациях.

Алгоритм MPR выглядит следующим образом (составлено на основе статьи «XenoCollide: Complex Collision Made Simple» – Gary Snethen, «Game Programming Gems 7»):

1. Начнем строить симплекс с известной точки, которая гарантированно входит в CSO – назовем ее **внутренней точкой**. Этой точкой, например, является разность центров двух тел:

```
Vector3f v0 = object2.position - object1.position;
```

Эта точка очень важна, так как с ней связана вся дальнейшая логика.

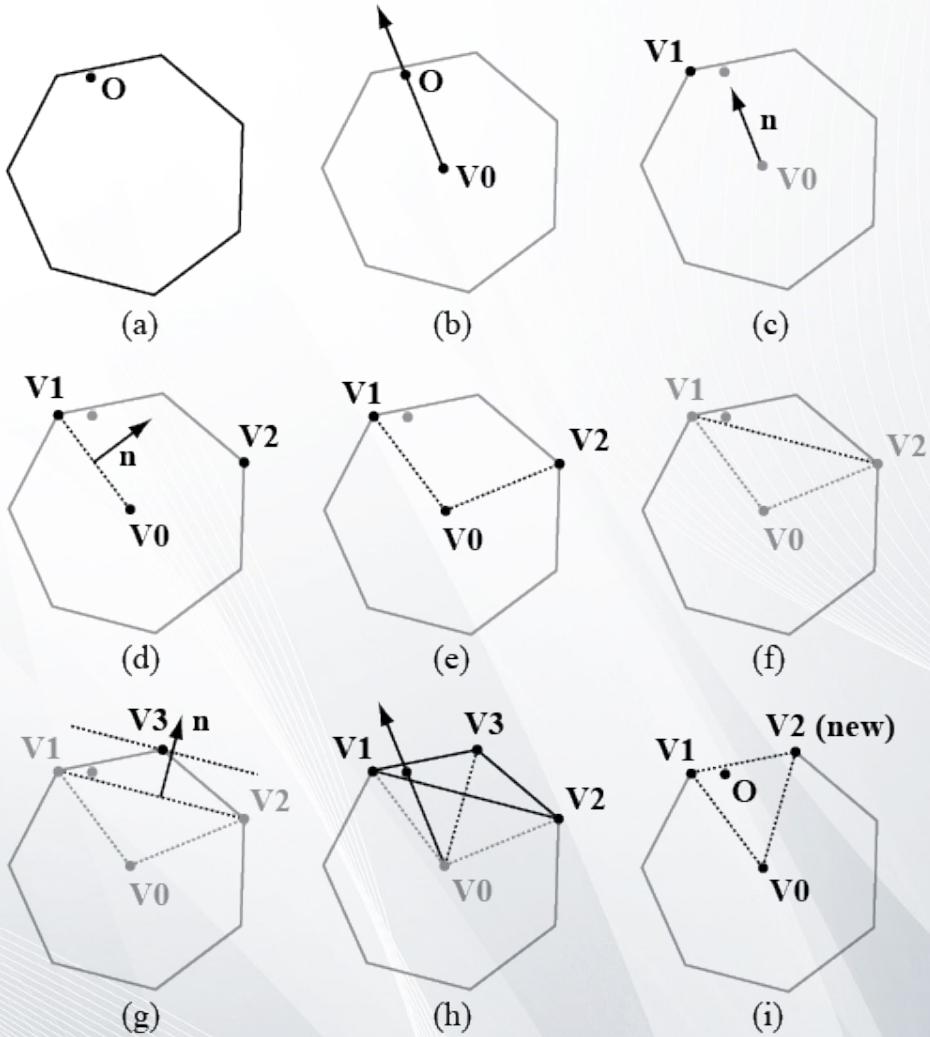
Из определения выпуклости следует, что, если луч с началом во внутренней точке, проведенный через начало координат (назовем его лучом O), пересекает оболочку CSO до того, как достигнет начала координат, то тела не пересекаются (начало координат лежит вне CSO). И, соответственно, наоборот – если луч O пересекает оболочку CSO после того, как достигнет начала координат, то тела пересекаются.

2. Построим начальный симплекс. Для этого требуется найти еще три точки на оболочке CSO, не лежащие на одной прямой. Они образуют треугольник, который в терминологии MPR называется **порталом**. Мы можем найти их, используя луч O:

```
// Опорная точка CSO в направлении луча O:
Vector3f v1 = csoSupport(normalize(-v0));

// Опорная точка CSO в направлении,
// перпендикулярном плоскости, на которой лежат
// начало координат, внутренняя точка и точка v1:
Vector3f v2 = csoSupport(normalize(cross(v1, v0)));

// Опорная точка CSO в направлении,
// перпендикулярном плоскости, на которой лежат
// внутренняя точка, точки v1 и v2:
Vector3f v3 = csoSupport(
    normalize( cross(v2 - v0, v1 - v0) )
);
```



3. Данный этап алгоритма называют **Фазой 1**. Необходимо определить, пересекает ли луч O треугольник, образованный точками v1, v2, v3. Это можно сделать, например, путем проверки, на какой стороне лежит начало координат от трех плоскостей, образованных ребрами треугольника и внутренней точкой.

Если начало координат находится с внутренней стороны всех трех плоскостей, то, значит, мы нашли нужный портал и можем переходить к **Фазе 2** (см. шаг 5). Если нет, то нужно построить новый портал (см. шаг 4).

4. Если начало координат лежит на внешней стороне одной из плоскостей, нужно использовать нормаль этой плоскости, чтобы найти новую опорную точку на CSO в направлении этой нормали. Этой точкой следует заменить одну из двух других точек треугольника – получится новый портал, после чего следует новая проверка (см. шаг 3).

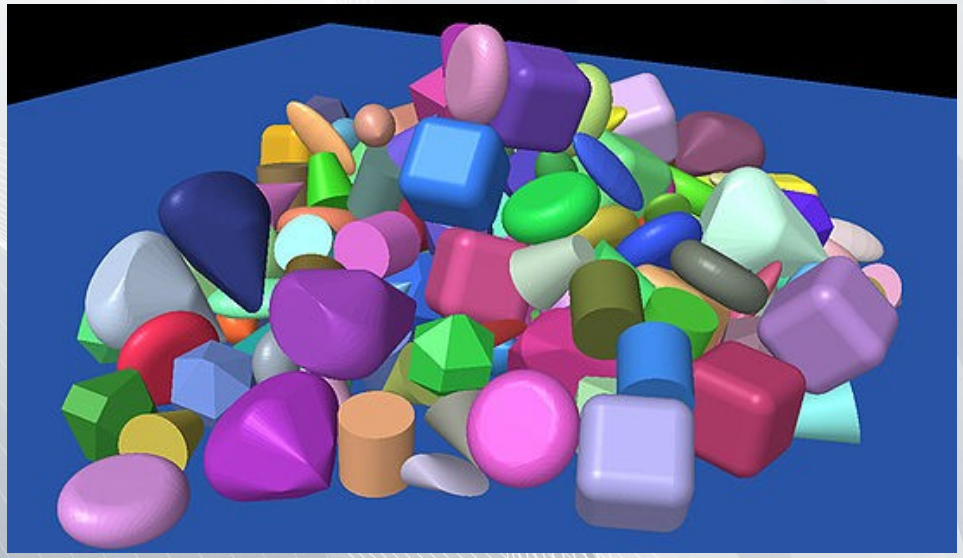
5. **Фаза 2**. На данном этапе мы имеем тетраэдр, образованный точками v0, v1, v2, v3 и лежащий внутри CSO. Если этот тетраэдр содержит начало координат, то и CSO содержит его (мы это доказали предыдущей проверкой). Следовательно, мы можем проверить, содержит ли тетраэдр начало координат – если да, то алгоритм завершает свою работу и возвращает положительный результат. В противном случае, переходим к шагу 6.

6. Мы убедились, что тетраэдр не содержит начало координат, но пока не можем с уверенностью сказать, что начало координат не лежит внутри CSO на внешней стороне портала. Делаем следующее. Берем нормаль портала во внешнем направлении и получаем еще одну опорную точку на CSO. Если начало координат лежит на внешней стороне плоскости, образованной этой нормалью и опорной точкой, то она гарантированно лежит вне CSO.

Следовательно, алгоритм в этом случае завершает работу и возвращает отрицательный результат.

7. В противном случае, нам нужно уточнить данные – в терминологии MPR, улучшить портал (отсюда «Portal Refinement» в названии алгоритма). Мы знаем, что начало координат находится где-то между порталом и выше-упомянутой плоскостью, но все еще не можем утверждать, что она лежит внутри CSO. Следовательно, нам нужно найти новый портал, более близкий к оболочке CSO.

Для этого делаем следующее. Зная, что луч O, покинув тетраэдр, гарантированно пересекает один из трех треугольников, образованных ребрами портала и опорной точкой, находим этот треугольник и берем его в качестве нового портала.



8. Повторяем проверку (шаг 6). Повторяем, если нужно, в течение нескольких итераций – алгоритм в этом месте нужно обязательно ограничить конечным количеством итераций, либо условием завершения (в противном случае, время завершения алгоритма будет недетерминированным, что, конечно, недопустимо в игровых приложениях).

Условием завершения может быть, например, оценка расстояния от портала до опорной точки – если оно достигает минимального порога, алгоритм завершается, возвращая какое-то заранее определенное для такого случая значение. Для физических движков авторы MPR рекомендуют возвращать отрицательный результат, так как небольшое проникновение двух столкнувшихся тел менее критично для зрителя, нежели видимый зазор между ними.

В теории алгоритм достаточно простой и визуально интуитивный, но на практике эффективно реализовать его не так просто. Референсных реализаций MPR в свободном доступе существует не очень много – одна из них, например, входит в состав физического движка dmech:

https://github.com/gecko0307/dmech

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, 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

Трюки из C в D

Практикум хакера

В этой небольшой статье я расскажу, как при помощи некоторых очевидных и не очень способах, часто применяемых в языке программирования C, сделать что-нибудь с некоторыми (в основном, числовыми) значениями в D.

Проверка числа на четность/нечетность. В некоторых случаях, требуется проверить является ли число четным или нечетным. Казалось бы, тут все очевидно, и первый прием, который приходит в голову, это что-то вроде:

```
x % 2 == 0
```

Однако есть способ поинтереснее: используя битовую операцию «**&**» можно проверку на четность реализовать гораздо изящнее, сэкономив при этом часть машинных операций.

Чтобы проверить число на четность можно воспользоваться вот таким решением:

```
!(x & 1)
```

А для проверки на нечетность достаточно применить вот такой прием:

```
x & 1
```

Насколько мне известно, такой трюк работает для целых чисел и аналогичных им по свойствам типов (по идее, и на char тоже сработает).

Приведение любого типа к bool. Иногда возникает ситуация, когда нужно привести некоторое значение к типу bool, а первой мыслью, которая возникает, обычно бывает такая:

```
cast(bool)x
```

Но операция, используемая в таком решении, не всегда безопасна, и порой требуется от нее отказаться. В этом случае к нам приходит на помощь, такая операция как отрицание «**!**» – дело в том, что любое ненулевое значение компилятор трактует как true (если это значение используется в условиях и циклам по условиям), а нулевое значение трактуется как false.

При этом для типа T величина, определяемая свойством T.init, гарантировано интерпретируется компилятором, как false. Таким образом, можно применить следующее экстраординарное выражение:

```
!!x
```

Думаю, вы поняли, как оно понимается компилятором.

Является ли число степенью двойки. Подобная задача решается самыми разными способами: от банальной сверки с таблицей степеней до приемов со скоростными логарифмированием по основанию 2, но все-таки есть один неочевидный трюк:

```
(n - 1) & n
```

Скоростное умножение/деление на 2. Иногда в программе требуется сделать так, чтобы операции с числами стали более легковесными и требовали как можно меньше тактов процессора. Умножение и деление принадлежат к тем операциям от которых, в таком случае, лучше отказаться.

Когда-то, еще в начале компьютерной эры, быстродействие компьютеров оставляло желать лучшего, и именно тогда был придуман трюк, позволяющий по быстрому множить или разделить на 2: для умножения на 2 достаточно сдвинуть целое число на 1 разряд влево (**<<**), а для обратной операции необходимо выполнить сдвиг на одну позицию вправо (**>>**). Кроме того, сдвиг влево или вправо на N ячеек позволяет сделать умножение или деление на 2 в степени N. Секрет прост: в набор инструкций современных процессоров встроены атомарные операции для проведения сдвига.

Битовые маски. Допустим, у вас есть некоторое значение, для которого необходимо узнать, какое значение имеет его N-ый бит. В D есть тип byte, но нет типа bit (хотя раньше был) – что делать?

Для выяснения того, какое значение принимает N-ый бит числа, лучше всего воспользоваться побитовым «И», которое сопоставляет соответствующие биты некоторых двух чисел.

Допустим, у нас есть число 147, а нужно выяснить какое значение (0 или 1) принимает 5-ый бит этого числа: для этого мы воспользуемся приемом, который называется наложением битовой маски. Наложение маски, заключается в следующем – берем исходное число и некоторое другое число (которое и называется битовой маской), которое будет служить своеобразным шаблоном для сопоставления по битам, и применяем к этим числам побитовое «И». Если вспомнить таблицу истинности «И», то мы увидим, что для двух значений мы получим истину (т.е. единицу в числовой интерпретации) только в том случае, если оба значения истинны. Таким образом, зная, что исходное число мы изменять не вправе, и именно из него мы будем выделять значение нужного бита, мы придем к выводу, что второе число (битовую маску) мы должны сформировать таким образом, чтобы в ее двоичном представлении на нужном месте (которое по номеру соответствует выделяемому биту) стояла 1, а в остальных местах (разрядах двоичного представления) был ноль.

Возвращаясь к нашему примеру выделения 5-ого бита из 147, приходим к выводу, что первое число – это 147 (в двоичном виде выглядит как 10010011), а второе число в двоичном виде выглядит примерно так: 00010000 (прошу заметить, что число двоичных разрядов в битовой маске равно количеству двоичных разрядов в исходном числе).

Таким образом, выделение 5-ого бита выглядит так:

```
147 & 0b00010000
```

При этом происходит следующее:

Число	1 0 0 1 0 0 1 1
Битовая маска	0 0 0 1 0 0 0 0
Результат	0 0 0 1 0 0 0 0

Получается результат – 00010000, что соответствует числу 16 в десятеричной системе счисления. Если вспомнить степени двойки, 2 в 4 степени равно 16 (дело в том, что нумерация битов начинается с правого конца и с нуля, в нашем случае: отсчет был начать с правого конца, но с единицы; а потому степень равна 4 (5-1), не 5, как следовало ожидать).

Далее, если вспомнить, что любое ненулевое значение понимается компилятором, как единица, то подобные операцию с применением битовой маски можно применять в условиях, так как установленный в ноль бит приведет к выдаче нулевого значения.

А что, если нам нужно не узнать, какие биты в числе установлены, а установить их? В этом случае, нам способна помочь такая операция, как побитовое «ИЛИ», обозначаемая в D как **|**. При этом все выглядит так же, как и в случае с побитовым «И»:

```
147 | 0b00010000
```

Ну и в виде таблицы это выглядит так:

Число	1 0 0 1 0 0 1 1
Битовая маска	0 0 0 1 0 0 0 0
Результат	1 0 0 1 0 0 1 1

Быстрый обратный квадратный корень. Ребята, а теперь адская жесть: вычисление быстрого обратного корня с помощью метода Ньютона на C – при этом код был портирован из реальных исходников игры Quake 3. Тут, как говорится, без комментариев:

```
float Q_rsqrt(float number)
{
  long i;
  float x2, y;
  const float threehalfs = 1.5f;
  x2 = number * 0.5f;
  y = number;

  // дьявольский хакинг на битовом уровне
  // для чисел с плавающей запятой:
  i = *(cast(long*)&y);

  // что за...?
  i = 0x5f3759df - (i >> 1);
  y = *(cast(float*)&i);

  // 1-ая итерация
  y = y * (threehalfs - (x2 * y * y));

  // 2-ая итерация
  y = y * (threehalfs - (x2 * y * y));
  return y;
}
```

Олег Бахарев
http://lhs-blog.info

Dev-C++ от Orwell

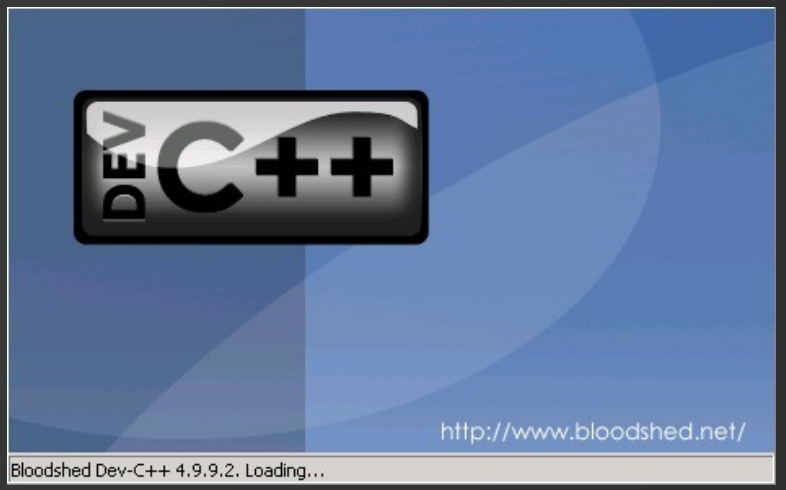
Возвращение легенды

Со средой разработки Dev-C++ у меня связаны самые теплые воспоминания – это была моя первая IDE под C++, на ней я учил язык, писал свои первые программы. К сожалению, с 2005 года этот замечательный проект оказался заброшен – сейчас оригинальная Bloodshed Dev-C++ даже не работает толком под современными версиями Windows.

Недавно мне понадобилось собрать программку на C, и не желая связываться с MS Visual Studio, я решил, как говорится, тряхнуть стариной – и с удивлением обнаружил, что Dev-C++ вновь активно разрабатывается!

Для тех, кто не в курсе: Dev-C++ – это свободная IDE под Windows, работающая с компилятором GCC, очень легковесная и интуитивная в использовании, прекрасный инструмент для образовательных задач, а также для работы над небольшими проектами.

Несмотря на открытость исходников, с OpenSource-проектами подобные случаи «воскрешения из мертвых» происходят не очень часто, поэтому я не мог не написать о данном радостном событии в журнал.



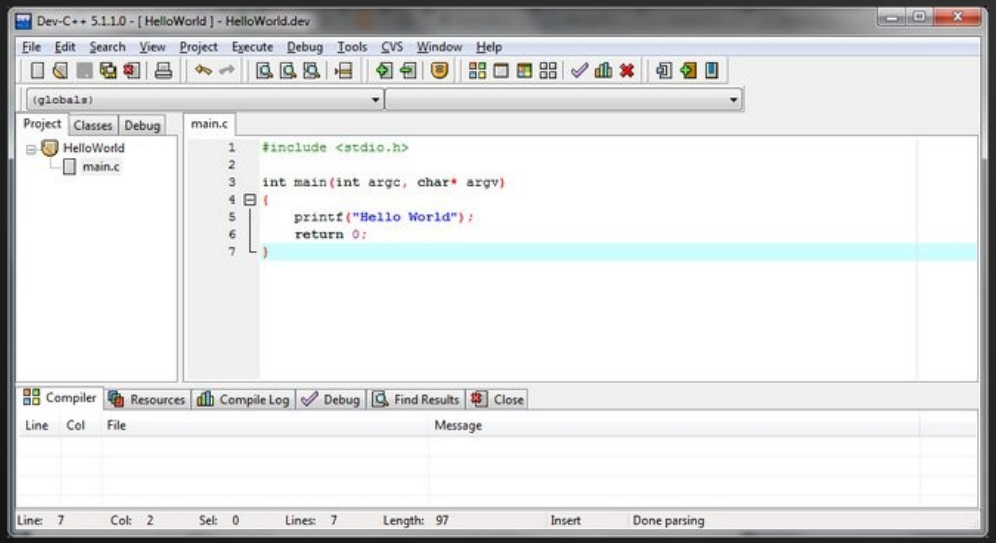
У Dev-C++ уже существовал один форк – wxDev-C++, попытка переписать среду на wxWidgets и добавить RAD-конструктор. Мне RAD не нужен, поэтому я этим форком особо не интересовался. В 2011 году среда была форкнута программистом под ником Orwell – так появилась ветка 5.0.

На сегодняшний день актуальна уже версия 5.11, которую я себе и установил. Посмотрим, насколько проект изменился за эти годы...

Главное, что хочется отметить – это, конечно, поддержка 64-битных систем. Причем, среда может работать и в 32-битном режиме, что очень удобно. В качестве компилятора используется TDM-GCC 4.9.2 – форк MinGW, отличающийся несколькими дополнительными патчами.

Тулчейн прекрасно работает под современными версиями Windows, сама IDE стала заметно быстрее и стабильнее. К сожалению, не появилась поддержка проектов MSVS современного формата (*.vcproj). Как известно, Dev-C++ поддерживает только импорт *.dsp, а проекты в этом древнем формате сегодня уже встретишь нечасто.

Естественно, обновились хедеры Win32. В комплекте со средой поставляется также SDK новейших версий Direct3D. Чтобы скомпилировать приложение D3D9, нужно прибегнуть к небольшой хитрости: скопируйте библиотеки libd3d9.a и libd3dx9_43.a из каталога MinGW64/x86_64-w64-mingw32/lib в папку с проектом и добавьте их в опции линкера в настройках проекта (вкладка Parameters). Никаких SDK от Microsoft ставить не надо, все необходимое есть в дистрибутиве Dev-C++.



OpenGL, кстати, также работает «из коробки». Кроме того, мне удалось с ходу прикрутить свежий релиз SDL2 – и базовый «джентльменский набор» разработчика игр, по сути, готов!

Единственное, что слегка огорчило – сломалась совместимость со старыми devpak'ами, пакетами-расширениями. За годы силами сообщества был сформирован достаточно большой архив devpak'ов на все случаи жизни – есть, например, большой репозиторий на сайте **http://devpaks.org**. В Orwell'овской же версии изменились стандартные каталоги заголовочных файлов и библиотек, поэтому расширения устанавливаются некорректно.

Впрочем, менеджер пакетов Packman жив и здоров, и вы всегда можете пересобрать пакеты, если есть на то большое желание.

Orwell Dev-C++ распространяется в нескольких вариантах – есть инсталляторы с компилятором и без, есть portaple-версии с компилятором и без. Обратите внимание – проекту требуются переводчики: смело открывайте Russian.lng в папке Lang и переводите то, что не переведено.

Скачать среду и узнать последние новости по проекту можно в блоге автора: **http://orwelldevcpp.blogspot.ru**

Тимур Гафаров

Мобильный FPS

Теперь любимый журнал всегда с вами!

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

Available on the App Store

ANDROID APP ON Google play

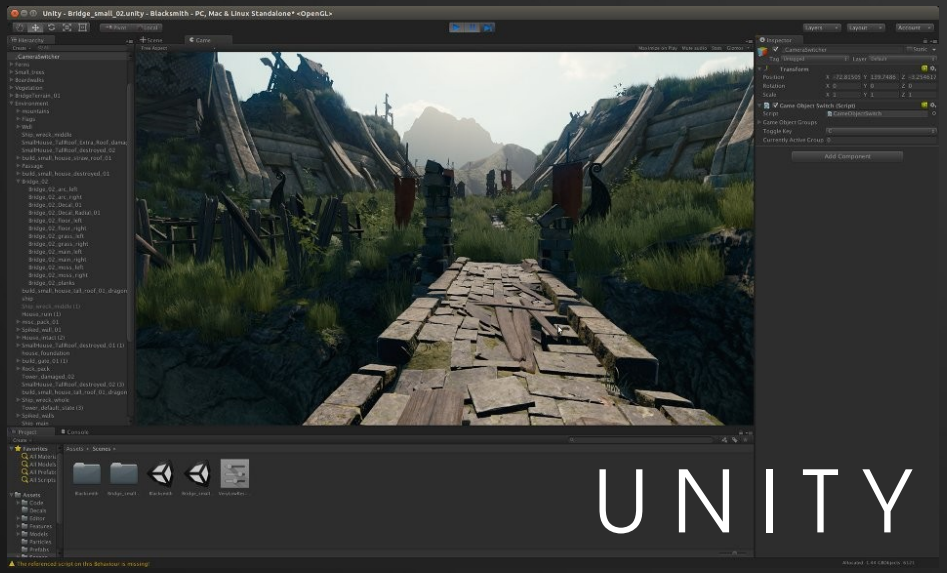
Разработчик приложения: цифровое издательство St.Appler http://www.stappler.org/

Linux-гейминг

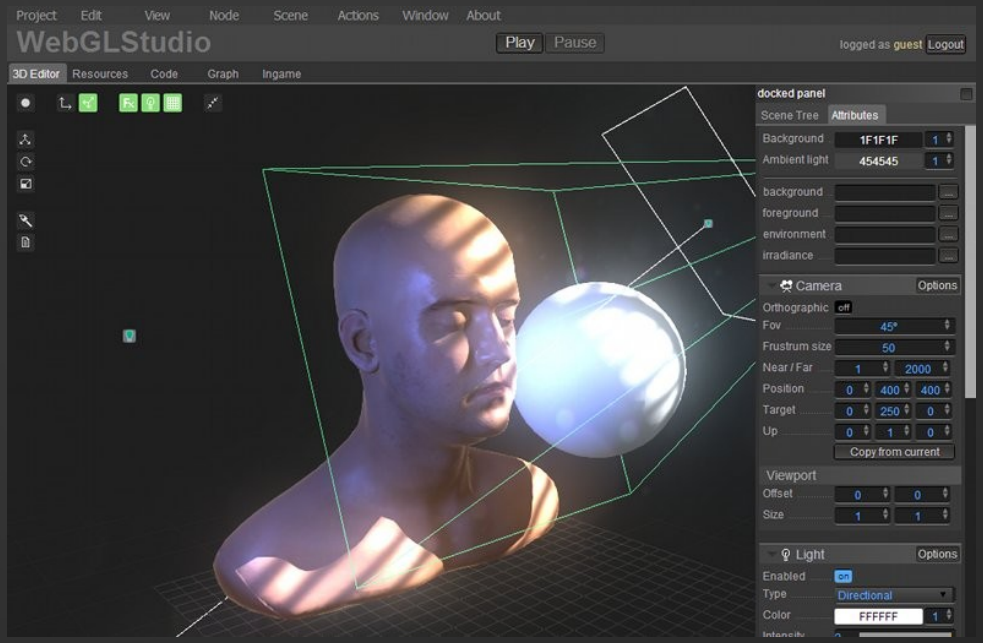
Игровые новости из мира Linux

Откроем наш сегодняшний блок новостей недавней сенсацией: знаменитый редактор игр **Unity** наконец-то доступен для Linux!

Разработчики выпустили экспериментальную сборку на основе Unity 5.1.0f3, включающую рантайм-компоненты для создания игр под Linux, OS X, Windows, WebGL, WebPlayer, Android, Tizen и SamsungTV. Сборка сформирована для 64-битной Ubuntu 12.04, либо более новых версий. Для работы требуется современная видеокарта NVIDIA, AMD или Intel с драйверами от производителя.



Поясним, что сих пор редактор Unity выпускался только для Mac OS X и Windows – притом, что проекты можно было экспортировать под Linux уже достаточно давно.



Еще один интересный проект, анонсированный в последние месяцы – это 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 под лицензией MIT.

http://webglstudio.org

Свободный транспортный мультисимулятор Rigs of Rods обновился до версии 0.4.5 – аккурат в собственный день рождения: 11 августа проекту исполнилось 10 лет.

Rigs of Rods представляет собой полноценный авто-, авиа-, ЖД- и судоходный симулятор, основанный на движке OGRE. Ключевая особенность игры – собственный физический движок Beam, использующий физику мягких тел для симуляции движения и деформации транспортных средств.

Игроку предоставлена полная свобода действий – «песочница» с огромным количеством карт, автомобилей, кораблей, поездов, самолетов и вертолетов. При этом все виды транспорта имеют крайне реалистичную модель движения и повреждений, вплоть до симуляции полного разрушения.

Rigs of Rods начинал свое развитие как исследование Пьера-Мишеля Рикорделя в области физики мягких тел. Проект сразу привлек внимание сообщества – энтузиасты начали создавать для него карты и транспортные средства. Универсальность физического движка сделала игру единственной в своем роде, революционной для своего времени.

Игра пережила «клиническую смерть» в начале 2013 года из-за ухода всех активных разработчиков в проект BeamNG. Лишь к началу 2015 года собралась новая команда энтузиастов, продолжившая разработку – сейчас они ищут добровольцев-программистов C++, AngelScript и Lua.

http://www.rigsofrods.com

Еще один любопытный анонс: обновление игры Minetest, свободного клона Minecraft. Minetest отличается полностью открытым исходным кодом, использованием движка Irrlicht и задействованием языка Lua для игровой логики сервера и написания модификаций.

В новой версии, 0.4.13, добавлен кинематографический режим и сглаживание камеры, реализовано настраиваемое автоматическое масштабирование текстур и задействование фильтров во время загрузки, возможность переопределения текстур, добавлен интерфейс для настройки сглаживания.

Код Minetest распространяется под лицензией LGPL, а игровые ресурсы – под CC-BY-SA 3.0. Готовые сборки имеются для Linux, FreeBSD, Windows и Mac OS X.

http://www.minetest.net

Число игр для Linux в Steam, между тем, достигло полутора тысяч. Отметка в 1000 игр была пройдена в марте нынешнего года, 500 – весной прошлого года, 250 – в декабре 2013 года, 100 – в начале лета 2013 года. Впрочем, доля Linux-пользователей среди клиентов Steam держится на уровне одного процента, что в два раза меньше, чем наблюдалось в 2013 году после бета-тестирования Steam для Linux.

Состоялся релиз свободной реализации OpenGL – Mesa 11.0. Выпуск примечателен реализацией OpenGL 4.1 в драйверах RadeonSI и Nouveau (nvc0) для видеокарт AMD на основе архитектуры GCN (HD 7700-7900, HD 8000, Rx 240-290, Rx 300) и NVIDIA на базе GPU Fermi и Kepler (GeForce 400/500/600).

В драйвере Intel i965 полностью обеспечена поддержка специфичных расширений OpenGL 4.2, но пока остаются нереализованными несколько расширений OpenGL 4.0 и 4.1. Первый выпуск ветки Mesa 11.0.0 имеет экспериментальный статус – после проведения окончательной стабилизации кода будет выпущена стабильная версия 11.0.1.

В Wine, тем временем, началась реализация поддержки Direct3D 11 – речь идет об очередном экспериментальном выпуске Wine 1.7.50. Также в новой версии закрыто множество отчетов об ошибках, связанных с работой различных игр и других приложений.

Vulkanизация

Известно, что концерн Khronos Group активно работает над спецификацией графического API нового поколения, известным как Vulkan. Судя по пресс-релизам и докладам на конференциях, новый API предлагает больше возможностей, чем традиционные OpenGL и Direct3D – это, например, улучшенная поддержка многопоточности, платформонезависимый байт-код для шейдеров, гибкая работа с видеопамятью и многое другое (более подробный обзор читайте в «FPS» №35 '15).

Выход первой спецификации Vulkan намечен на конец этого года – и, пока все ждут этого знаменательного события, хочется на основе существующей информации подвести первые итоги, отделив факты от кривотолков. Что же значит появление Vulkan для разработчиков игр? Нужно ли будет всем в срочном порядке переходить на новый API? Попробуем разобраться.

Является ли Vulkan полной заменой OpenGL?

Этот вопрос, наверное, вызывает больше всего дискуссий – неужели век «классического» OpenGL подошел к концу? Мы считаем, что беспокоиться пока не о чем – если верить заявлениям Khronos Group, оба API будут мирно сосуществовать. OpenGL в обозримом будущем никуда не денется и будет развиваться параллельно с Vulkan.

Слухи о том, что Khronos прекратит работу над OpenGL, вероятно, пошли из-за появления кодовых названий «OpenGL NG» и «glNext» – можно было подумать, что Vulkan станет некой новой версией OpenGL, ломающей обратную совместимость. Но на деле это, конечно, не так. Vulkan – это просто новый API, спроектированный для несколько иных задач, нежели OpenGL. Vulkan более низкоуровневый и его использование будет оправдано далеко не во всех ситуациях, поэтому о полном отказе от OpenGL говорить пока рано.

И даже если Khronos прекратит обновлять спецификации OpenGL и OpenGL ES, производители оборудования все равно будут поддерживать эти API для обеспечения совместимости с тоннами существующего ПО – ведь в этом заинтересованы многочисленные разработчики и пользователи игр, пакетов 3D-моделирования, CAD-систем и т.д.

Впрочем, история компьютеров знает случаи, когда API практически полностью исчезали за считанные годы – например, так было с Glide, «фирменным» интерфейсом видеокарт Voodoo от 3dfx. Он был популярен в 90-х, но очень быстро оказался вытеснен Direct3D и OpenGL. После 1999 года не выходило ни одной коммерческой игры, зависящей от Glide, и этот API канул в лету.

Сегодня Glide уже не поддерживается производителями видеокарт – существуют лишь врапперы и эмуляторы, позволяющие запускать Glide-игры на современном оборудовании (например, путем трансляции в OpenGL).

Не повторит ли теперь OpenGL печальную судьбу своего предшественника? Нельзя исключать, что в будущем реализации OpenGL будут точно так же представлять собой обертку над Vulkan. И это подводит нас к следующему вопросу.

Заменит ли Vulkan WebGL?

WebGL – достаточно молодой стандарт: рынок браузерных 3D-игр только начал развиваться, и разработчики, конечно, примут на ура любую инновацию, которая позволит им повысить производительность своих продуктов. Между тем, WebGL 1.0, основанный на OpenGL ES 2.0, начинает устаревать – не за горами выход второй версии спецификации. Не будет ли она основана на Vulkan?

Большинство специалистов считают, что нет. Vulkan – небезопасный API, в нем нет встроенных механизмов обработки ошибок, поэтому неправильно написанная программа для Vilkan может вызвать падение приложения. А любая веб-технология сегодня обязана работать в «песочнице» – то есть, в виртуализированном безопасном окружении. Поэтому появление «WebVulkan» до сих пор под вопросом.

Более реалистичным представляется то, что будущие реализации WebGL будут основаны уже не на OpenGL, а именно на Vulkan. Фактически, сегодня уже можно говорить о едином графическом стеке на всех платформах, ведь одна из задач Vulkan – унификация графики на десктопных и мобильных системах.

Потребует ли Vulkan обновления графического железа?

Еще один волнующий всех вопрос. На самом деле, Vulkan – это не какая-то «технология будущего», создающаяся для компьютеров завтрашнего дня. Этот API, по сути, просто сужает прослойку между видеокартой и прикладным ПО – при этом речь идет о обычных современных GPU: Vulkan рассчитан на любое железо, способное потянуть OpenGL 4.1 / OpenGL ES 3.1 или выше.

Что насчет Mantle? Есть ли связь между Mantle и Vulkan?

Определенно, связь есть. Судя по исследованиям хакеров, Vulkan основан на Mantle, причем вплоть до повторения названий функций – например, vkBeginCommandBuffer вместо grBeginCommandBuffer. Будущее самого Mantle не совсем ясно – скорее всего, Vulkan полностью его заменит.

*Сегодня я хочу рассказать о уже знаменитом симуляторе **Banished**, выпущенном в 2014 году разработчиками **Shining Rock Software**. Игра интересна своим бережным отношением к ресурсам вашего компьютера – занимает на жестком диске всего 200 Мб, а в этот маленький размер умещается огромный опасный мир, в котором вам необходимо помочь выжить группе поселенцев...*

BANISHED

Суровые будни колонистов

Нам не рассказывают всю предысторию героев. Небольшая группа поселенцев оказывается на новой неизведанной земле, на которой им предстоит выжить любой ценой. Размер ресурсов в начале игры зависит от выбранного вами уровня сложности: на легком уровне в вашем распоряжении будет амбар и склад с неким ограниченным набором провианта и бревен, а на сложном – жалкая тележка.

И с этим приходится работать дальше: строить дома, дороги, возводить лесопилки и каменоломню...

Карта для игры генерируется всегда случайным образом, тут тоже вам предлагается выбор: строить свое поселение на равнине или в горной местности. Естественно, на последней выжить труднее. Также для упрощения игрового процесса для особо ленивых можно выключить бедствия. А бедствия там были очень даже серьезные: во время моей игры половину моих жителей сметало ураганам, а в другой раз все заболели и умерли – а все из-за того, что у меня не было больницы.

Постройка каких-либо зданий требует много ресурсов, которые приходится добывать потом и кровью: вырубать леса, работать в шахтах, сеять кукурузу и картофель. В игре также существует и режим торговли. Если построить порт на берегу реки, туда могут приплыть торговцы, с которыми можно обменяться товарами, прикупить семена новой садовой культуры или новый вид домашнего животного.

Хочется отметить и изменяющуюся погоду: если летом не заготовил нужное количество дровишек, зимой приходится замерзать.

Осенью часто идет дождь, зимой – снег: природа живая и за ней наблюдать очень интересно!..

Человечки в игре тоже живые и интересные: они передвигаются, работают, размножаются. Если вы хотите, чтобы численность вашего поселка увеличилась, стройте дома – молодые люди совсем не хотят жениться и заводить детей в доме со своими родителями (жизненно!).

Еще игровой составляющей является некий черный юмор разработчиков: так, в игре можно построить кладбище. Занимательное зрелище: когда умирает кто-то из жителей поселения, на кладбище тут же появляется новое надгробие.

Геймплей у игры интересный: управление достаточно простое, есть режим обучения, милый саундтрек, который, однако, быстро надоедает.

Какой-то конкретной цели у игры нет – просто выжить и увеличить численность ваших изгнанников до максимального количества. А это ой как непросто: максимум, чего я смогла достичь – это жалкие сто жителей. Хотя, может, плохой из меня стратег...

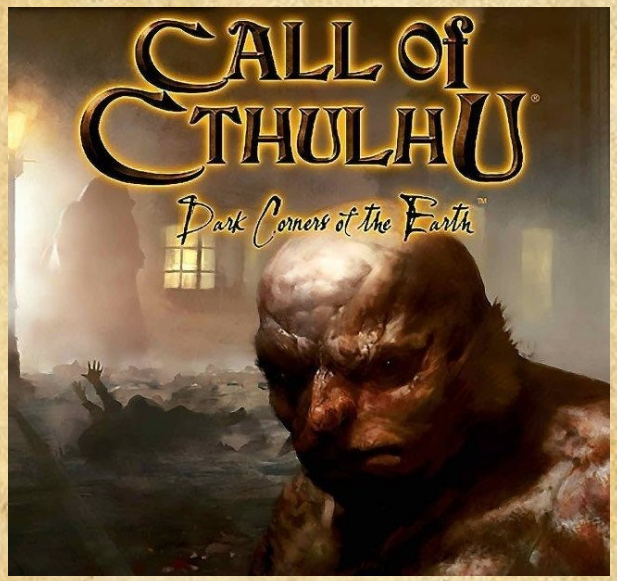
Наталия Чумакова

http://www.shiningrocksoftware.com

КУЛЬТОВЫЕ ИГРЫ

Лучшие мистические игры

Интерес к мистике не утихает даже в наш технологичный век – наверное, желание прикоснуться к тайне, к чему-то пугающему и загадочному свойственно самой природе человека. Бум мистической литературы и кино во второй половине XX века – самое яркое тому подтверждение. Хоррор, темное фэнтези, мистический триллер – все эти жанры не могли не повлиять и на компьютерные игры: речь в данном обзоре пойдет именно об этом.



«Call of Cthulhu: Dark Corners of the Earth» («Зов Ктулху: темные уголки Земли») – самая известная игра, созданная по мотивам творчества Говарда Лавкрафта, легендарного американского писателя-мистика.

Влиянием Лавкрафта отмечено практически все мистическое искусство XX века: им зачитывались Роберт Блох, Стивен Кинг, Нил Гейман и многие другие писатели, тоже ставшие культовыми. Загадочный «Некрономикон» из его романов превратился в современный миф – многие верят, что эта книга существует на самом деле. У Лавкрафта было множество единомышленников, которые еще при жизни писателя продолжили развивать придуманную им вселенную ужасов.

Идейным последователем Лавкрафта можно считать и Ганса Руди Гигера, швейцарского художника, создавшего собственное направление в фантастической иллюстративной живописи. Очевидно, что при создании знаменитого «Чужого» и других своих персонажей, Гигер вдохновлялся лавкрафтовскими демонами и чудовищами – кстати, у него есть серия альбомов, которая называется «Necronomicon».

Сегодня наследие Лавкрафта не прекращает вдохновлять людей творчества на новые произведения в жанре «темного фэнтези», объединяющие научную фантастику и сверхъестественные ужасы. Такова и игра «Зов Ктулху», которая дает вам возможность столкнуться с непознанным, почувствовав себя главным героем лавкрафтовских рассказов – это детективная история, где вам в роли частного сыщика предстоит разгадать тайну оккультной секты, поклоняющейся морскому богу Дагону.

Действие разворачивается в Иннсмауте – вымышленном американском городе из рассказа «Тень над Иннсмаутом», который признан одним из лучших произведений Лавкрафта. Всю игру можно считать иллюстрацией к этому рассказу. Впрочем, дотошные знатоки творчества писателя нашли в ней отсылки и к другим его новеллам («Кошмар в Ред-Хуке», «Шепчущий во тьме», «Музыка Эриха Цанна» и т.д.).

Игру отличает очень мрачная атмосфера. Иннсмаут – это, по сути, «город смерти»: ветхие трущобы как будто разваливаются на глазах, по углам чердаков и подвалов валяются трупы, которые никто не думает убирать, а по улицам, как привидения, одиноко слоняются угрюмые и неразговорчивые обитатели города.

Игра удачно сочетает боевые действия и детективную составляющую: вам придется не только бороться с чудовищами, но и разгадывать достаточно сложные головоломки. Окружение полностью интерактивно – можно изучить любой предмет, заглядывать в шкафы и ящики, собирать улики и документы. Герой ведет дневник, в который заносятся все интересные факты, собранные по ходу расследования. Игра замечательно озвучена и полностью переведена на русский язык.

Еще одна игровая серия, которую также причисляют к «лавкрафтовским ужасам», – это Penumbra («Полутень»). Первая игра в этой серии, «Penumbra: Overture» («Полутень: увертюра») частично отсылает к роману Лавкрафта «Хребты Безумия».

Главный герой в поисках исчезнувшего отца попадает в заброшенную шахту в Гренландии, где разворачивается настоящий кошмар. Шахта оказывается населена загадочными монстрами – по ходу своих поисков герой узнает, что некое тайное общество занималось в этом месте изучением аномальных явлений, однако все его члены погибли при странных обстоятельствах.

Penumbra, как и предыдущая игра, также делает главную ставку на решение головоломок: в игре даже нет огнестрельного оружия – от врагов вам придется отбиваться подручными средствами. В игре можно подбирать и переносить с места на место любые предметы – на этом основана вся ее механика. В целом, Penumbra является классическим образцом хоррора: ведущие в неизвестность темные коридоры, скрипучие двери, жуткие звуки из-за угла – и, разумеется, отсутствие пути назад...

Вторая часть, «Penumbra: Black Plague» («Полутень: черная чума»), проливает свет на историю «проклятой шахты»: выясняется, что люди потревожили во льдах Гренландии загадочное существо по имени Туурнгайт, которое наслало на них болезнь, превращающую людей в мутантов. Главный герой становится «избранным», способным доказать Туурнгайту, что человечество имеет право на существование...

Одна из самых знаковых мистических игровых серий – это, конечно, японская «Silent Hill». Первая и третья части серии были экранизированы, причем первый фильм стал культовым сам по себе, вне связи с игрой. Кстати, режиссер фильма Кристоф Ган тоже является поклонником Лавкрафта – в 1994 году он снял фильм «Некрономикон», который и привлек внимание японских киноинвесторов.

Действие игры происходит в «проклятом городе» Сайлент Хилле: вы, в роли писателя Гарри Мэйсона, в поисках исчезнувшей приемной дочери оказываетесь в своеобразном инфернальном параллельном мире – город, впитавший огромное количество зла и человеческих страданий, постепенно погрузился в ад, который засасывает каждого, кто окажется поблизости. Все страхи человека здесь принимают материальную форму, рисуя перед ним самые ужасные видения...

По сюжету первой части «Silent Hill», дочь писателя, Шерил, попадает в руки местной секты «охотников на ведьм». Выясняется, что много лет назад здесь... заживо сожгли на костре маленькую девочку Алессу. Она выжила, но ее душа разделилась на две части: «светлая» сторона Алессы – это и есть Шерил, а «темная» – запертое в подвалах неподвижное обгоревшее тело. Ее страдания стали источником ужасного проклятия, преследующего всех жителей Сайлент Хилла. Шерил попадает в руки сектантов, и ваша задача в игре – пройти через все кошмары, которые создает перед вами город, и спасти девочку.

Игру отличает сложный философский сюжет, уникальный дизайн и неповторимая атмосфера – критики отмечают сильное влияние фильма «Лестница Иакова», а также таких мэтров литературы и кино, как Стэнли Кубрик, Альфрен Хичкок, Рэй Брэдбери, Стивен Кинг и, конечно, Говард Лавкрафт. Визуально игра отсылает к Босху и японской иллюстративной графике. Руководитель проекта Кэйитиро Тояма отмечает, что полагался на свои увлечения оккультизмом, НЛО и творчеством Дэвида Линча. В игре очень много оккультной символики, отсылок к Каббале, к религиозному мистицизму, средневековой черной магии, а также современным ужасам и фэнтези. Даже улицы Сайлент Хилла названы в честь знаменитых писателей-фантастов!..

Не обойдем стороной и мистические квесты. Лучшим из них (и одним из лучших графических квестов вообще) считается «Myst» – большая игровая серия, включающая пять номерных выпусков, три переиздания и несколько спин-оффов. Myst и его продолжения в совокупности разошлись в количестве около 12 миллионов копий – серия долгое время считалась самой продаваемой игрой в мире. Ее впоследствии смогла превзойти только одна игра – The Sims.

Вселенная Myst – это множество миров в пространстве-времени, называемых «эпохами», между которыми можно путешествовать при помощи книг-порталов, созданных таинственным магом по имени Атрус.

Игра начинается с того, что безымянный главный герой, обозначенный просто как Странник, находит необычную книгу под названием «Myst», и она переносит его на одноименный остров. Изучая его и решая головоломки, Странник находит книги-порталы, ведущие в другие миры. Постепенно выясняется, что Атрус, как и его сыновья Сиррус и Акенар, попал в книгу-ловушку, и вашей задачей становится освободить их.

Геймплей в Myst, как и в большинстве аналогичных игр, очень размеренный и неторопливый. За этой игрой можно проводить долгие часы, вдумчиво читая книги в местных библиотеках, изучая свойства причудливых механизмов или просто любуясь живописными видами.

Игровой мир очень атмосферен и детально проработан – здесь есть своя мифология, история, легенды. Каждый уголок здесь наполнен мистической символикой, имеются, к примеру, отсылки к иудейской религии – в частности, Бог-творец в космологии Myst зовется Яхво.

Первая и вторая части игры 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.

Ну а поклонников мистической литературы явно не разочарует квест по мотивам знаменитого романа Брэма Стокера – «Dracula: Resurrection» («Дракула: воскрешение»). Это, фактически, продолжение книги: действие происходит через 7 лет после того, как Джонатан Харкер и его товарищи убили графа Дракулу в Трансильвании. Выясняется, что вампир все-таки не был уничтожен, и Мина вновь начинает ощущать на себе его зов. Не в силах сопротивляться, девушка уезжает к нему, и вы, в роли Джонатана, должны снова отправиться в замок Дракулы, чтобы спасти возлюбленную.

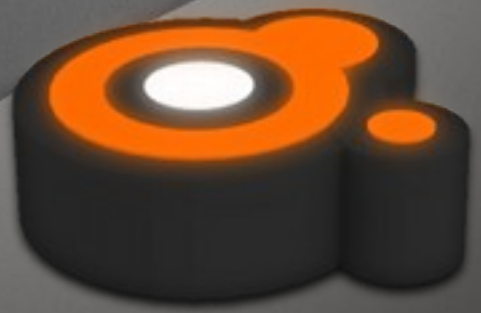
Окружение полностью пререндеренное, игра перемежается кинематографическими сценами. Игра очень атмосферна, она с головой погружает вас в мрачный мир восточноевропейской глубинки начала XX века. Головоломки достаточно сложные, чтобы впечатлить современного игрока: вам придется взламывать замки и сейфы, искать артефакты и секретные двери в потайные комнаты.

Тимур Гафаров

Конечно, описанными здесь тайтлами список мистических игр не ограничивается – если вам кажется, что мы забыли упомянуть ту или иную важную игру, напишите нам об этом на ***gecko0307@gmail.com****, и мы обязательно опубликуем продолжение обзора!*

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