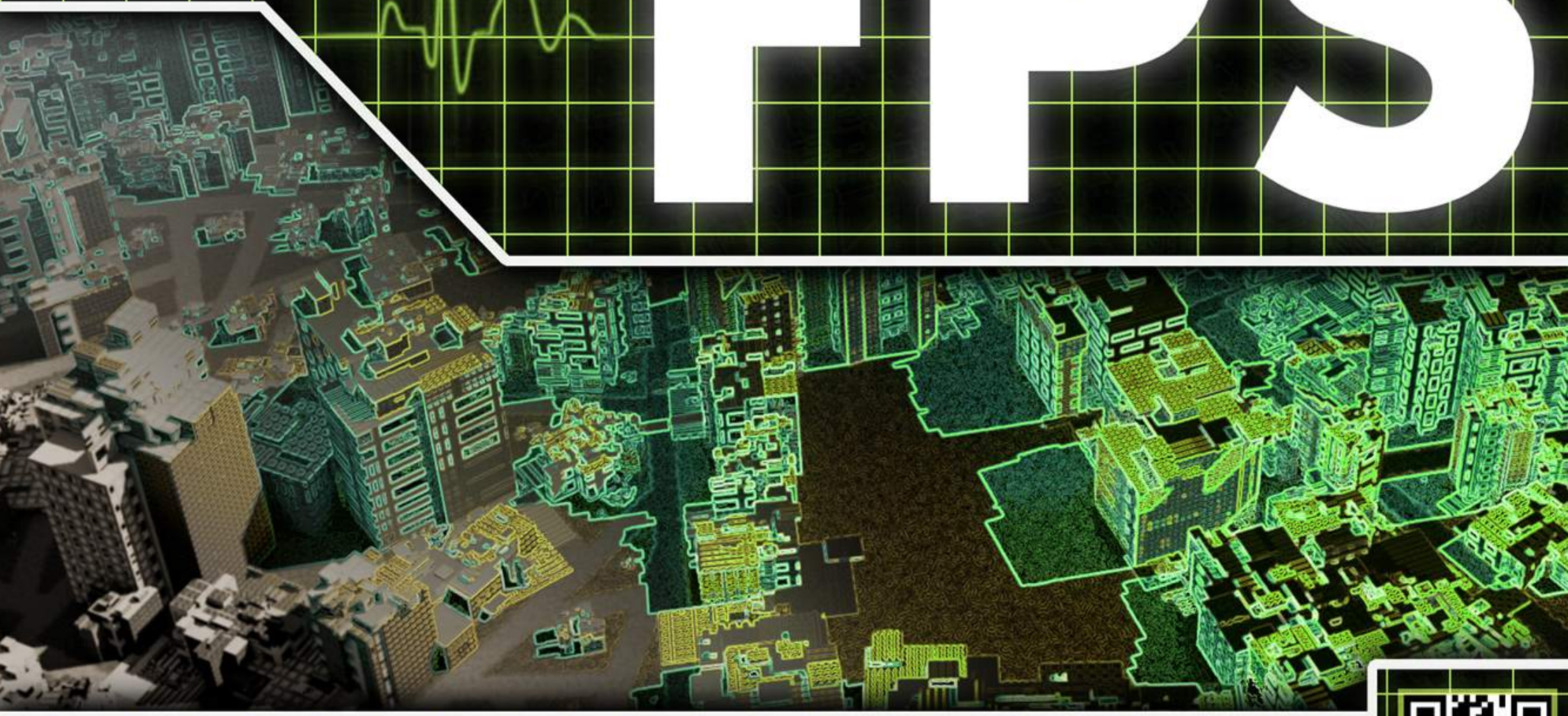
№17 • 2011

FPS



BLENDER: Обзор дополнений
Интервью с Давидом РЕВУА

ALCHEMY: Инструмент импровизатора
Институт разработки игр DIGIPEN

...и многое другое!

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

Нас читают:

gcup.ru
Все для начинающего и профессионального разработчика игр

xtreme3d.narod.ru
Сайт движка Xtreme3D

make-games.ru
Портал создания игр

gamer-club.ucoz.com
Все для создания игр без программирования и не только

mizzystic.ru
Крупнейший информационный Game Maker портал

xbobr.at.ua
Создание игр, программ, музыки

dapf.us
Форум дизайнеров и программистов

esate.ru
Мультимедиа-сообщество

www.mobkiosk.com
"Мобильный киоск"

dogames.ru
Мастерская игр

gamecreate.ru
Создай свою игру!

igrostroenie.net.ru
Игровые движки и ресурсы

gacon.ucoz.ru
Сайт, посвященный разработке игр

bigslava.3dn.ru
Сообщество творческих людей

antistarforce.com
Игровой портал

Главный редактор: Тимур Гафаров
Дизайн и верстка: Тимур Гафаров
По вопросам сотрудничества обращаться по адресу gecko0307@gmail.com.
Официальный сайт журнала: http://fpsmag.zymichost.com

ЭЛЕКТРОННЫЙ ЖУРНАЛ о разработке компьютерных игр

№ 17
2011

В этом выпуске:

Blender

Новости

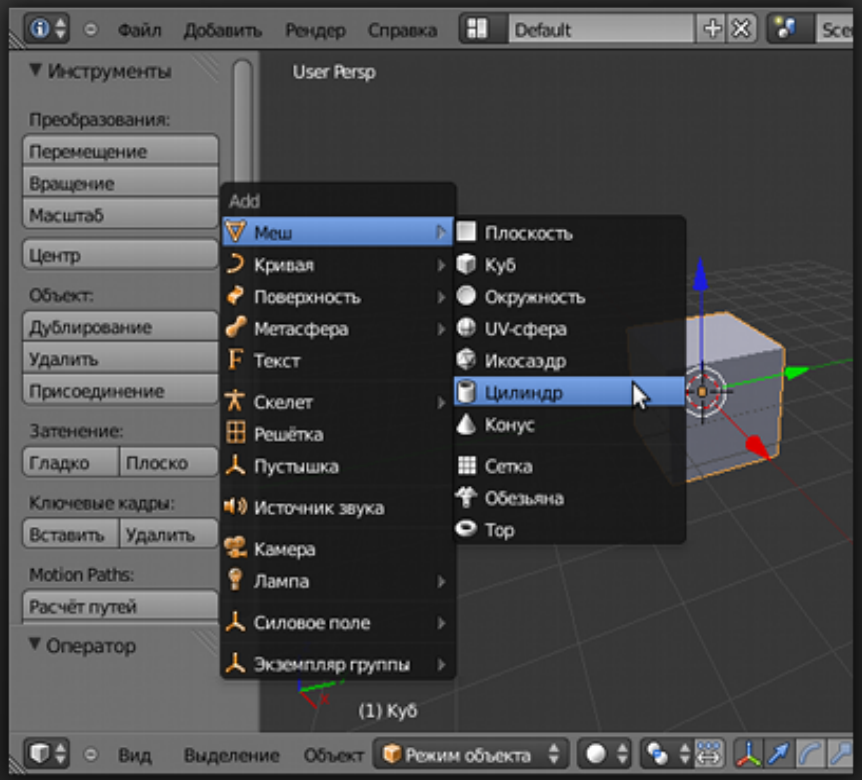
В ДЕКАБРЕ состоялся релиз Blender 2.61. Из существенных нововведений этой версии следует отметить:

- технологию отслеживания движений (Motion Tracking) из томатной ветки (Tomato Branch), которая позволяет восстановить анимацию камеры с видеосъемки и синхронизировать соответствующим образом трехмерные объекты для их последующего наложения в композиторе;

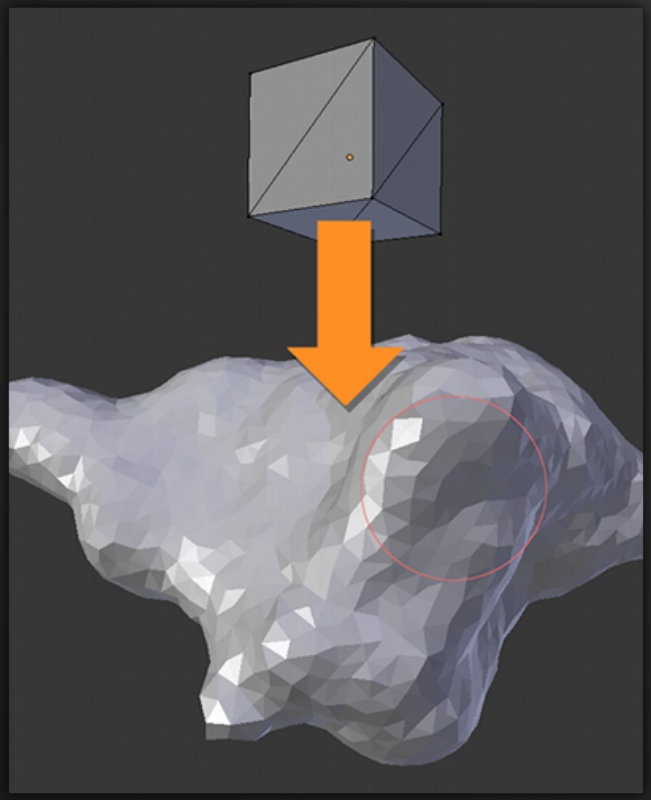
- новый движок рендеринга Cycles, который теперь доступен наряду с Blender Internal. Cycles представляет собой мощный интерактивный трассировщик лучей с поддержкой аппаратного ускорения на GPU. Подробнее об этом проекте читайте ниже;

- систему динамического рисования. Это новый модификатор, позволяющий превратить трехмерные объекты в своеобразные «холсты» и «кисти». При их соприкосновении во время анимации, на «холсте» остаются следы, которые влияют на материал объекта. Эта система значительно облегчает динамическое создание сложных эффектов – таких, например, как появление дождевых луж, следов на снегу, грязи, пыли, инея на объектах и т.д.;

- систему симуляции океана. Существующий симулятор жидкостей в Blender мало подходит для моделирования морских волн – в то время, как морской пейзаж был и остается одним из самых излюбленных мотивов в компьютерной графике. А реалистичная анимация моря в мультфильмах – это и вовсе «высший пилотаж» (вспомните хотя бы Pixar и «В поисках Немо»). Данный проект призван помочь художникам в этом нелегком деле.



Многие пользователи Blender в Росии явно будут рады практически полной русификации интерфейса и полной поддержке кириллицы в текстовых полях ввода. И, если профессионалы в русском переводе изначально не нуждались (обычно говорят, что программы такого уровня лучше не переводить – слишком много специфических терминов), то преподаватели и составители учебников уж точно вздохнут с облегчением.



Программист Николас Бишоп, автор системы BMesh, написал в блоге про свой новый проект по реализации динамической топологии в Blender.

Суть идеи в том, что при лепке топология обновляется в режиме реального времени – во время каждого штриха, а не по его завершении, что существенно повышает удобство моделирования. Реализация Бишопа никак не зависит от схожего проекта Unlimited Clay.

Исходники проекта доступны на Gitorious. Возможно, скоро также появятся соответствующие сборки Blender на GraphicAll.org. Все подробности расписаны в блоге разработчика.

18 сентября в русскоязычном сегменте Интернета состоялся очередной удаленный семинар по использованию Blender, который на этот раз был посвящен архитектурной визуализации. Ведущий семинара – Игорь Безуглый, директор и главный визуализатор компании «Слон».

Игорь уже пять лет занимается архитектурной визуализацией зданий «на местности» и в рамках семинара рассмотрел типичные ошибки, допускаемые при моделировании экстерьеров и интерьеров. Также были подробно рассмотрены вопросы постановки и настройки камеры для композитинга трехмерных сцен и статичной двумерной графики.

Медленно, но верно началась работа над небезызвестным проектом Mango – новым открытым фильмом Blender. В ноябре у всех желающих был шанс войти в команду по работе над фильмом – для этого необходимо было отправить свое портфолио в Blender Institute. Кроме того, недавно на сайте проекта появились первые концептуальные наброски. Один из авторов, кстати – Давид Ревуа, арт-директор фильма «Синтел» (интервью с ним читайте в этом номере журнала).

David Revoy - www.davidrevoy.com - CC-BY Blender Foundation

Gecko
gecko0307@gmail.com

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

Спецпредложение от «FPS»!

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

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

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

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

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

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

Cycles @ Blender 2.61

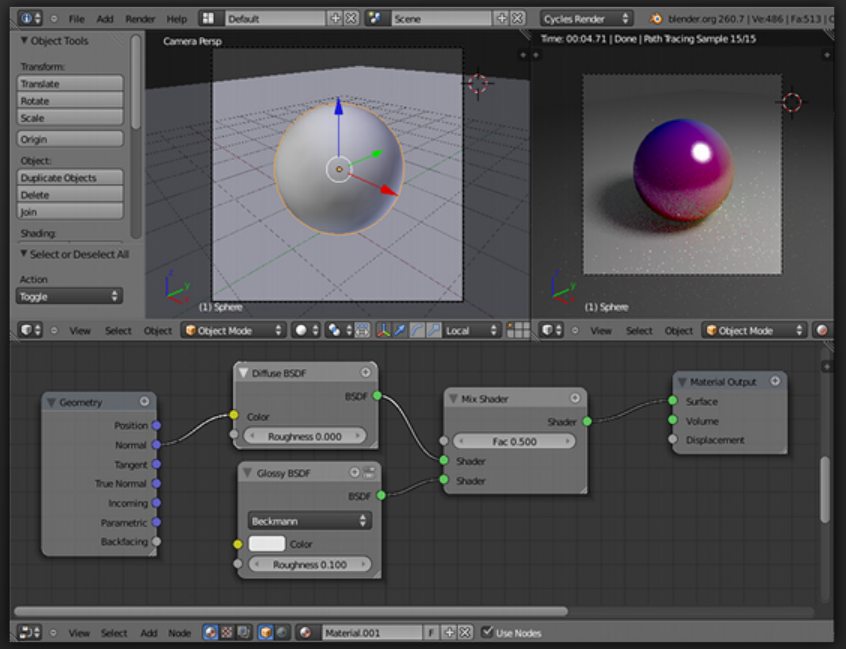
Ключевое нововведение Blender 2.61 – это, без сомнения, «официальная» интеграция в пакет нового движка рендеринга Cycles. В «FPS» №16 '11 мы уже успели познакомить читателей журнала с базовыми принципами работы Cycles на примере пользовательской сборки Blender. Теперь пришло время для, так сказать, более пристального взгляда.

Внесение чего-либо в основную ветку Blender – всегда знаковое событие. Это свидетельствует об определенном уровне надежности и качества того или иного инструмента. И Cycles в этом смысле – не исключение, хоть в официальном чейнджлоге и стоит оговорка «first preview release». Налицо значительное повышение производительности: разработчики честно сдержали заявленное на октябрьской конференции Blender в Амстердаме обещание повысить скорость работы движка в 2 раза.

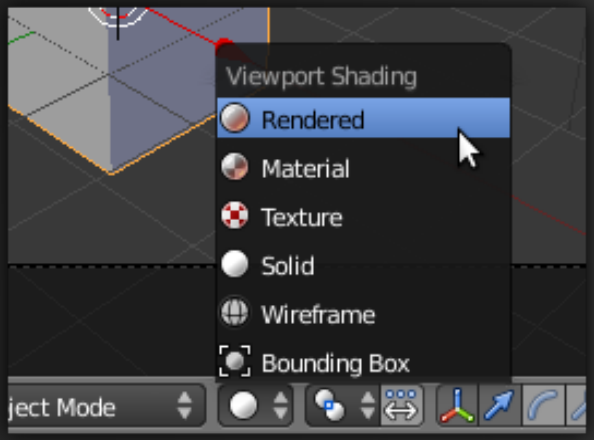
Не лишним будет обнадежить консервативно настроенных пользователей – Cycles не собирается в будущем вытеснять Blender Internal; старый встроенный движок останется полностью доступен.

Одна из самых интригующих особенностей Cycles – интерактивный рендеринг. Не любите постоянно нажимать F12 и долго ждать результата? Значит, Cycles – для вас! Система рендеринга работает практически в реальном времени, и эффект от изменения настроек сразу же становится виден на экране – с учетом теней, отражений, глобального освещения, каустики и т.д.

Эта технология является ответом Blender на V-Ray RT для 3ds Max. Cycles тоже поддерживает рендеринг на GPU при помощи CUDA, что в разы повышает скорость вычислений на современных видеокартах от nVidia. В перспективе планируется также поддержка OpenCL. Мгновенная реакция на действия пользователя может ощутимо помочь художнику на этапе настройки материалов и освещения. Такой подход привносит совершенно новые ощущения и опыт работы над сценой!



Чтобы включить интерактивный рендеринг, активируйте Cycles и в любом окне просмотра сцены переключите метод отображения на Rendered. Можно улучшить качество предпросмотра – для этого необходимо повысить количество сэмплов в поле **Samples > Preview** на вкладке **Integrator** в настройках сцены.

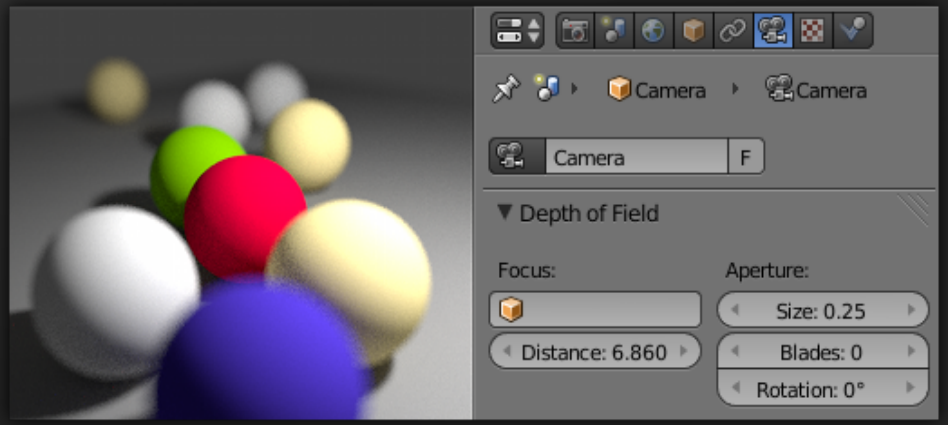


Из числа значимых «новинок» Cycles в Blender 2.61 можно отметить поддержку микрорельефа (в терминологии Cycles – смещения). Смещение задается либо в настройках материала на вкладке **Displacement**, либо непосредственно в редакторе узлов. Эта возможность на данный момент помечена как экспериментальная.

Кстати, использование узлов для создания материалов – еще одна фундаментальная концепция Cycles. Это роднит его с семейством программируемых движков в стиле RenderMan. Понятие материала теперь максимально приближено к понятию шейдера: пользователю предоставляется свобода создания собственных композитных шейдеров на основе различных входных данных и встроенных моделей освещения (**BSDF**). Это могут быть как фотореалистичные материалы, так и полностью творческие.

Помимо шейдеров поверхности материала, в Cycles предусмотрена возможность работы с его объемом. Шейдер объема (**volume shader**) описывает поведение света по мере его проникновения внутрь объекта. Это необходимо для реалистичного моделирования полупрозрачных материалов с эффектом подповерхностного рассеивания (**Subsurface Scattering**).

Многие пользователи наверняка оценят встроенную поддержку **DOF** (**Depth Of Field**) – глубины резкости, которую ранее приходилось делать в композиторе. Теперь она интегрирована с объектом камеры: просто укажите фокусное расстояние (или объект, на который нужно сфокусировать камеру), повысьте радиус апертуры и наблюдайте результат в реальном времени. И никакой пост-обработки!



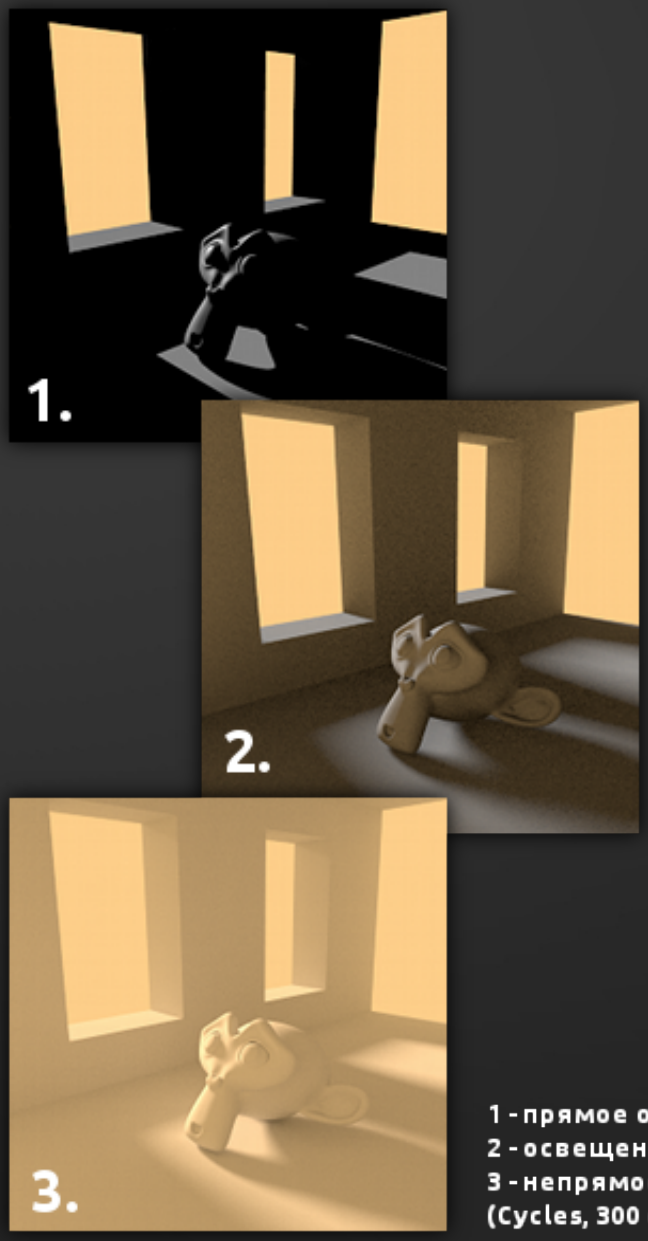
В Cycles также учитываются стандартные лампы Blender. На данный момент есть поддержка ламп типа Point, Sun и Area. В качестве источников света также можно использовать объекты-излучатели с шейдером Emission в материале. Кроме того, при рендеринге учитывается свет окружения.

Существенный недостаток Cycles, присущий вообще всем физически корректным рендерам – высокий уровень цветового шума при малом количестве сэмплов (менее 1000). В частности, много лишнего шума дают зеркальные бликующие материалы. Разработчики знают об этой проблеме и уже наметили **план действий** по ее устранению. Не исключено, что в будущем специально для этого в композиторе появится фильтр шумоподавления.

Ссылки к статье:
Cycles на wiki.blender.org
Коллекция уроков

Gecko
gecko0307@gmail.com

P.S. Несмотря на относительную молодость проекта, в русскоязычном Blender-сообществе Cycles уже успели наградить собственным прозвищем – «суслик» :)



1 - прямое освещение (BI)
2 - освещение среды (BI)
3 - непрямое освещение (Cycles, 300 сэмплов)

Обзор дополнений Blender

Выпуск 1

Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. Для старой версии программы (серия 2.4x) уже создано огромное множество расширений – от простых скриптов экспорта до комплексных инструментариев и профессиональных коммерческих плагинов. Для 2.5 и 2.6 их пока не так много – но эта ситуация быстро меняется. Этим выпуском мы открываем серию обзоров полезных дополнений для Blender 2.5 и выше, которые могут заинтересовать пользователей, использующих программу в качестве инструмента для разработки игр или создания игрового контента...

Suicidator City Generator

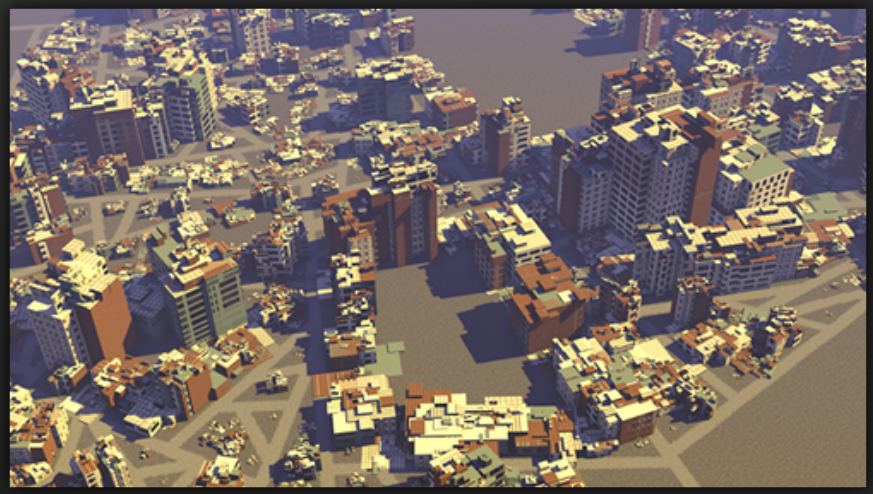
Suicidator City Generator (SCG), разработка французского программиста Арно Кутюрье – это дополнение для Blender 2.6x, предназначенное для процедурного моделирования городов. С его помощью вы можете за пять минут создать свой собственный мегаполис – с небоскребами и автострадами.

SCG «на лету» генерирует город в соответствии с заданными параметрами. Можно застроить свою «столицу мира» простыми боксами, а можно – весьма детализированными конструкциями со множеством полигонов. Основная концепция SCG – случайность. Каждый сгенерированный город по-своему уникален и не похож на другие.

SCG хорошо оптимизирован и экономно расходует оперативную память. Его можно использовать даже на слабых компьютерах – в то время как обладатели мощных конфигураций могут генерировать огромные города, простирающиеся «до горизонта». Полученные при помощи SCG дома и улицы можно редактировать и доводить до совершенства встроенными средствами Blender.

Дополнение стоит € 14.95. Есть бесплатная версия – с ограничениями на размер текстур и площадь моделируемого города.

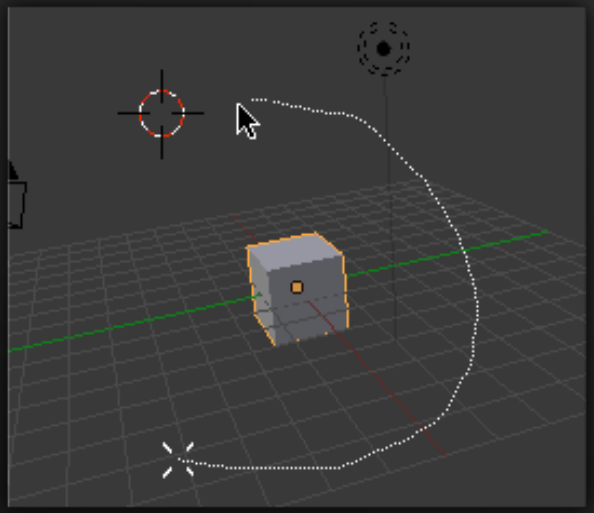
http://arnaud.ile.nc/sce/



Mouse Gesture

Не знаю, как другие, но я лично скучаю по жестам мышью из Blender 2.4x, которые позволяли перемещать объекты, нарисовав прямую линию, и вращать их, нарисовав эллипс – это экономило сотни нажатий клавиш. По всей видимости, эту возможность так и не собираются возвращать в новой версии пакета. Во всяком случае, она уже который год ждет своего часа в официальном ToDo по интерфейсу Blender 2.5. Сейчас уже вышла 2.6, но воз и ныне там...

Однако мало кому известно: решение этой проблемы существует, и уже достаточно давно. Chromoly, известный Blender-программист, выпустил дополнение Mouse Gesture, которое возвращает привычные жесты мышью. Более того, к основным жестам трансформации (прямая линия = перемещение, окружность = вращение, уголок = масштабирование) добавились новые – жесты переключения проекции.



На панели свойств (клавиша **N**), на вкладке **Mouse Gesture**, можно изменить режим жестов с **Transform** на **View** – и переключаться на вид сверху (вертикальная линия снизу вверх), спереди (вертикальная сверху вниз), сбоку (горизонтальная слева направо) или с камеры (горизонтальная справа налево).

Ссылка на скачивание

Дополнение работает со всеми версиями Blender начиная с 2.56. Распространяется на условиях GNU GPL.

MD5 Exporter

Многие игровые движки поддерживают MD5 – формат моделей idTech 4 / Doom 3. MD5 позволяет хранить модели со скелетной анимацией, причем анимация хранится в отдельном файле от самой модели (используются расширения *.md5mesh и *.md5anim). Формат хорошо изучен, в Интернете легко можно найти исходники на C/C++ для загрузки и рендеринга MD5, например, при помощи OpenGL.

Изначально для создания моделей в этом формате использовался пакет Maya, но сейчас существуют «неофициальные» экспортеры и для всех остальных 3D-редакторов. Есть такой экспортер и для Blender – у него достаточно длинная «биография» версий; сейчас он благополучно портирован под Blender 2.5+.

http://www.katsbits.com/tools/

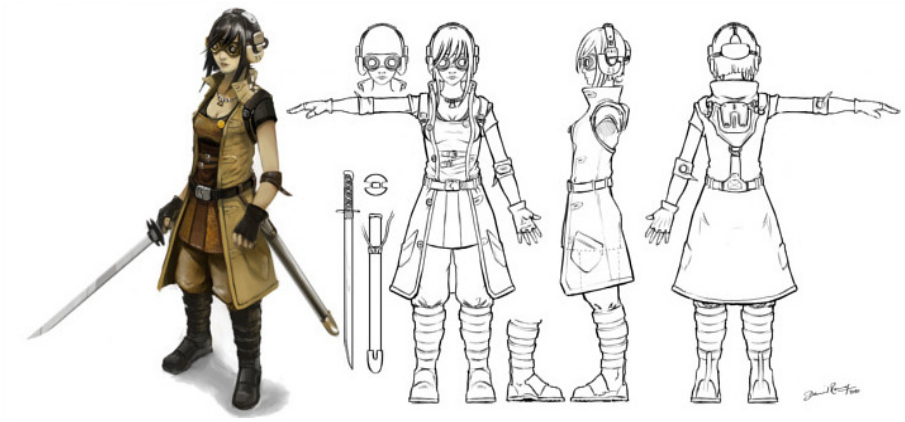
Gecko
gecko0307@gmail.com

КОНЦЕПТЕР

Интервью с Давидом РЕВУА

Давид Ревуа (David Revoy) – известный в кругах пользователей Blender художник-фрилансер. Он специализируется на книжной и игровой иллюстрации, но более известен широкой аудитории как арт-директор фильма «Синтел».

В 2010 году он выпустил обучающий DVD-курс по работе со свободными графическими пакетами – **«Chaos & Evolutions»**, в котором были представлены видеоуроки в формате HD, демонстрирующие процесс рисования персонажей. Другой популярный видеокурс Ревуа – «Blend & Paint» – объясняет основы интерфейса Blender, моделирование, настройку освещения и рендеринг, а также перенос отрендеренной картинки в MyPaint и GIMP для дальнейшей обработки.



Журнал «BlenderArt» недавно опубликовал интервью с Давидом Ревуа в рамках выпуска, целиком посвященного моделированию и анимации персонажей.

Давид, многим читателям ты известен именно как автор концепт-арта для «Синтел». Но ведь это не первая твоя работа. Расскажи немного о себе.

Мне 29 лет, я занимаюсь фрилансом в своей домашней студии в Тулузе во Франции. Мои работы можно увидеть на обложках книг. Еще я занимаюсь иллюстрацией настольных игр, так что у меня достаточно насыщенная творческая жизнь.

Сейчас я пользуюсь только свободным софтом и Linux (хотя и преподаю работу с Photoshop и Painter в колледже), потому что мне по душе социальная и культурная революция вокруг свободного программного обеспечения. «Синтел» – мой первый проект в качестве арт-директора. Работа в таком проекте на такой должности для меня как «свободного художника» была мечтой.

Твой видеокурс для Blender Institute удачно продается и получает положительные отзывы. Оглядываясь на этот проект, каким ты видишь его естественное развитие?

Во-первых, мне очень приятна похвала, да и читать отзывы об этом курсе всегда интересно. Что касается продолжения, я задумал «Chaos & Evolutions» именно как курс, который максимально отвязан от конкретных инструментов. Его содержание скорее отражает мою методику работы и, если хотите, философию относительно искусства. Мне хочется верить, что этот курс будет полезен на протяжении еще очень долгого времени. Надеюсь, что видеоролики распространятся по разным интернет-ресурсам.

В книге Джонатана Уильямсона «Character Development in Blender 2.5» используется модель женского персонажа из «Chaos & Evolutions». Расскажи подробнее, как так получилось?

Действительно, этот персонаж – из «Chaos & Evolutions». Около года назад Джонатан попросил меня создать дизайн для двух персонажей. Я предложил уже существующую модель из моего видеокурса – она была доступна под CC-BY (как и все другие художественные работы на диске). Вторым персонажем стала Кара, которую вы, наверное, уже видели ее на Blender Cookie.

Ты сотрудничал с Уильямсоном при работе над несколькими проектами – в том числе и над его книгой. Есть ли у вас еще какие-нибудь совместные планы на будущее?

Работать с Джонатаном – сущее удовольствие. Конечно, будучи «креативщиками», мы постоянно обсуждаем те или иные новые идеи. Но я немного суеверен и предпочитаю не распространяться раньше времени о проектах, которые еще не распланированы как следует.

Как ты считаешь, повысила ли работа над «Синтел» и видеокурсами твой уровень признания и зрительской симпатии?

Откровенно говоря, да – если говорить о сообществе Open Source. Но у меня впереди еще достаточно работы, чтобы добиться признания широкой публики. Я стараюсь как можно чаще обновлять свою галерею на DeviantArt, а также «свечусь» в известных CG-коммьюнити, чтобы не зацикливаться на Open Source.

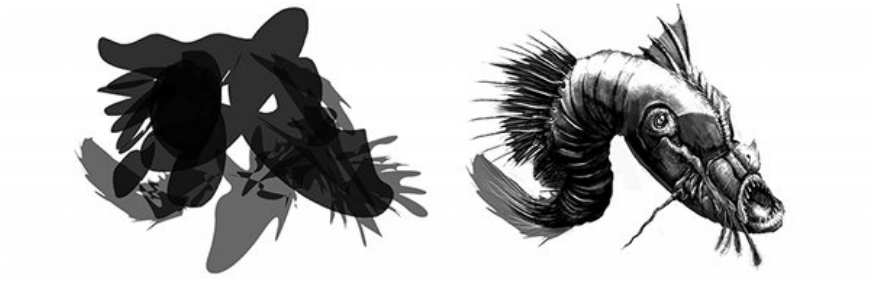
При работе над «Синтел» ты пользовался только свободными инструментами – GIMP, MyPaint, Alchemy. Какое у тебя осталось впечатление от работы с ними?

Gimp-painter 2.6.8 и MyPaint (еженедельная сборка с Git master) в Linux Mint работают действительно стабильно. Я полностью доверяю им при работе на заказчика. По моему личному опыту они даже стабильнее, чем моя предыдущая конфигурация: Adobe Photoshop CS2 и Corel Painter XI.5 на Windows XP Pro SP3. В этом смысле это действительно профессиональные продукты. За последние месяцы не припомню ни единого падения или потери данных. Я могу сделать в них всю свою работу от начала до конца без «сюрпризов» посередине. Разработчики – просто молодцы!

Программа Alchemy, по сути, является экспериментальной – в том смысле, что позволяет делать совершенно необычные вещи. Но экспериментальный софт хоть и помогает придумать и опробовать какие-то новые идеи, все-таки остается экспериментальным. Большинство пользователей все равно продолжает пользоваться более традиционными решениями. Как ты считаешь, есть ли у Alchemy шанс завоевать сердца художников? Или проекту это и не нужно?

Лично мне кажется, что инструменты вроде Alchemy больше подходят именно для цифровой живописи, чем для традиционной (вроде имитации диффузии акварели или неровностей от мазков маслом). Но и как цифровой инструмент Alchemy порой слишком «экстремален».

Я практически уверен в том, что авторы других программ для цифровой живописи со временем перенесут к себе разные интересные идеи, заложенные в Alchemy (к примеру, в Autodesk SketchBook уже появилось динамическое симметричное рисование). Ну а сам Alchemy наверняка продолжит двигаться в исходном направлении: как инструмент виджеинга и прочих экспериментальных направлений в искусстве. Такие программы – совершенно потрясающий источник идей.



Насколько я могу судить, заметный рост интереса к MyPaint в последнее время во многом обязан твоему деморолику про спидпейнтинг. Достаточно многие высказанные тобой идеи были реализованы в версии 0.8.0, а твой набор кистей сейчас используется в программе по умолчанию. Ты продолжаешь взаимодействовать с разработчиками? Если да, то на каком уровне?

Сейчас я общаюсь в основном с разработчиками MyPaint и Krita – главным образом, потому что часто пересобираю эти программы из исходников. Так что я могу судить о том, что происходит в этих проектах. Что касается GIMP, я все время пользуюсь им в своей работе как фрилансер, и по этой причине не слишком расположен тестировать нестабильные версии. Наверное, поэтому со стороны может показаться, что я несколько прохладно отношусь к программе, хотя на самом деле меня очень волнует выпуск версии 2.8.

Что касается именно взаимодействия, я не то чтобы очень активен; я просто читаю списки рассылки и участвую в обсуждениях на IRC и форумах. Как обычный пользователь я стараюсь помогать проектам чем могу и когда могу. Иногда я просто прошу помощи, когда та или иная программа «падает», иногда я просто говорю «Здорово!», чтобы воодушевить программистов, иногда я хочу их о чем-то попросить, для чего я просто рисую прототип функции, чтобы сразу было понятно, что мне надо. Со временем дружеские, хоть и виртуальные, отношения с разработчиками только крепнут, и меня это очень радует. Я действительно ценю тот огромный труд, который они вкладывают в свои программы. Они настоящие мастера своего дела!

Если говорить о всей экосистеме свободных программ для творчества, чего тебе не хватает больше всего? Какие функции можно было бы реализовать достаточно легко, но при этом сделать твою работу заметно проще?

Самым важным мне кажется повышение производительности при работе над большими изображениями. Многие художники привыкли рисовать большими кистями (512×512) по большим холстам (6000×6000 и выше) и при этом работать с большим количеством слоев и разными режимами смешивания. Работать над такими рисунками в свободных приложениях, даже при наличии современных мощных компьютеров, все еще сложновато. Так что, с моей точки зрения, именно оптимизация производительности программ помогла бы привлечь внимание профессиональных художников.

*За предоставленную информацию благодарим портал **linuxgraphics.ru** и журнал **BlenderArt***

Blender: настольная книга

«Blender. Настольная книга» – это проект от журнала «FPS» по созданию полноценного русскоязычного электронного руководства по основам работы в Blender 2.5 и 2.6. Целевая аудитория – начинающие пользователи программы (как перешедшие со старых версий, так и начинающие знакомство с Blender «с нуля»). Книга будет представлять собой сборник статей, охватывающих различные аспекты использования Blender, скомпонованных по принципу «от простого к сложному».

Издание будет распространятся бесплатно, по лицензии Creative Commons BY SA. На данный момент активно ведется подготовка текста книги.

К работе над книгой приглашаются все желающие! На почтовый ящик редакции (gecko0307@gmail.com) принимаются статьи и уроки, а также общие советы и предложения. Кроме того, книге нужны графические материалы: авторские художественные работы, интересные скриншоты, демонстрационные рендеры, схемы, диаграммы и т.д. Весь Ваш вклад в книгу обязательно будет учтен, и Ваше имя будет указано в списке авторов.

Подробности на сайте проекта:

http://fpsmag.zymichost.com/book

Как часто вам приходилось хаотично водить карандашом по бумаге в поисках вдохновения? Этим, несомненно, «страдают» все художники без исключения. Увидев в случайных штрихах или пятнах интересный силуэт, они часто используют его в качестве основы для будущего произведения искусства.

В мире цифровой живописи тоже возникает необходимость в таких набросках – именно для этого была создана программа Alchemy. Это уникальный графический редактор, направленный на содействие свободному творчеству и упрощение поиска идей для концепт-арта. Чаще всего Alchemy используется для создания оригинальных абстрактных композиций. Художник набрасывает форму пятнами и линиями, если что-то получается – продолжает развивать идею в других программах.

Будучи сугубо экспериментальным инструментом, Alchemy предлагает совершенно новые способы рисования на компьютере, к которым даже трудно подобрать какие-либо термины. Например, программа позволяет контролировать форму линий с микрофона, может самостоятельно генерировать случайные штрихи и поддерживает симметричное рисование для быстрого создания лиц. Практически все инструменты и эффекты имеют случайную составляющую.

Процесс работы в Alchemy достаточно прост и интуитивен. По большому счету, это ни что иное, как векторный редактор с нарочно урезанной функциональностью. В нем, например, нет ластика, отмены и возможности масштабировать изображение. Единственная панель инструментов предоставляет доступ ко всем основным функциям программы: выбор модулей, управление их параметрами, изменение типа линии, ее цвета и прозрачности.

Модули делятся на две категории: создающие модули, отвечающие за характер линий, форм, фигур и т. д. (одновременно может быть активен только один такой модуль) и модули эффектов – они модифицируют уже имеющиеся или рисующиеся в данный момент формы, а также могут добавлять новые формы на основе имеющихся (одновременно могут быть активны несколько таких модулей). Alchemy также поддерживает пользовательские модули, написанные на Java.

Alchemy разрабатывается в Японии при поддержке Exploratory Software Project. В настоящее время программа имеет статус бета-версии, хотя и вполне стабильна. Программа кроссплатформенная (Java-приложение). Распространяется по лицензии GNU GPL.

Официальный сайт проекта – **al.chemy.org**.

DigiPen

Институт разработки игр

Разработка компьютерных игр, казалось бы, только вчера вышедшая из гаражей и хакерского андеграунда в большой бизнес, стала де-факто одновременно и индустрией, и искусством, и наукой. Как промышленность, она подразумевает экономический подход, как искусство – участие творчески одаренных, талантливых людей с креативным мышлением, как наука – собственный академический аппарат и систему подготовки специалистов. И, если с первыми двумя условиями в нашей стране худо-бедно справляются, то с последним ситуация достаточно плачевна. На Западе десятки учебных заведений предлагают программы обучения, связанные с разработкой игр. В России, к сожалению, ни один вуз не готовит таких специалистов. Приходится все изучать самому. В этом отношении мы находимся в той же ситуации, что и «первопроходцы» игровой индустрии.

Собственно, в самообразовании как таковом нет ничего плохого. Просто без соответствующего диплома в глазах окружающих вы так и останетесь чудаком-любителем, какими бы знаниями и опытом не располагали. В нашей стране, где даже к художникам-профессионалам относятся как минимум несерьезно (как, на художника тоже нужно учиться?!), сказать, что вы занимаетесь разработкой игр – примерно то же самое, что назвать своей профессией, скажем, алхимию. А все потому, что у нас «этому не обучают». Это проблема чисто психологического характера. Но если она вам мешает, и отсутствие «корочки» все же не дает покоя, стоит обратить внимание на зарубежные учебные заведения.

Пожалуй, самым «раскрученным» среди них можно считать Технологический институт DigiPen (DigiPen Institute of Technology). Это коммерческий колледж в США, специализирующийся на информатике, вычислительной технике и дизайне с акцентом на разработку компьютерных игр. Учившиеся в этих стенах художники, аниматоры, программисты и менеджеры еще будучи студентами создавали такие хиты, как Synaesthete, Narbacular Drop и Tag: Power of Paint. DigiPen по праву можно назвать «кузницей инди-легенд»...

В 1988 г. Клод Комер основал корпорацию DigiPen в Ванкувере (Канада) как компанию, которая занимается компьютерной симуляцией и анимацией. Поскольку спрос на продукцию вырос, DigiPen столкнулась с недостатком в квалифицированных кадрах, и в 1990 г. она начала предлагать специальную программу обучения в области трехмерной компьютерной анимации. В конце года, опираясь на успех и образовательный опыт подготовки учащихся к работе в студиях по производству 3D-анимации, DigiPen начала сотрудничать с американским отделом Nintendo с целью предоставить высшее образование тем, кто заинтересован работать в игровой индустрии. Результатом стало основание в 1994 г. DigiPen Applied Computer Graphics School (Школы прикладной компьютерной графики DigiPen). Для первого потока студентов была подготовлена двухлетняя программа курса «Программирования двумерных и трехмерных видеоигр».

Вскоре в DigiPen осознали необходимость предоставления программы обучения на уровне степени бакалавра. На основе своих профессиональных знаний и опыта, инженеры DigiPen разработали программу обучения на четыре года, которые они представили для авторизации в Совет Вашингтона по высшему образованию (Washington State Higher Education Coordinating Board, HECB).

В мае 1996 г. НЕСВ предоставил DigiPen разрешение на выдачу диплома адъюнкт-бакалавра наук по специальности «интерактивное моделирование в режиме реального времени». Это стало первой в мире академической программой подготовки специалистов в области разработки компьютерных игр.

В 1998 г. открылся Технологический институт DigiPen в Редмонде (где, кстати, также находится главная штаб-квартира Microsoft!), который начал предлагать следующие программы обучения:

- Младший научный сотрудник, «интерактивное моделирование в режиме реального времени»;

- Бакалавр наук, «интерактивное моделирование в режиме реального времени».

В 1999 году была добавлена специальность «прикладное искусство в трехмерной компьютерной анимации». DigiPen прекратила свою образовательную деятельность в Канаде, и отделение Редмонда стало единственным образовательным учреждением DigiPen. Первая церемония выпуска состоялось 22 июля 2000 г. – выпускались шесть младших научных сотрудников и пять бакалавров.

В ноябре 2002 г. институт получил аккредитацию от комиссии ACCSCT (Accrediting Commission of Career Schools and Colleges of Technology). В 2004 г. DigiPen добавил следующие программы:

- Бакалавр в области вычислительной техники;
- Бакалавр изящных искусств в производстве анимации;
- Магистр в области компьютерных наук.

В 2007 г. число студентов достигло 900. DigiPen продолжает стремиться к высокому уровню обучения, которого требует индустрия, и до сих пор считается лидером образования в сфере цифровых интерактивных развлечений. Работы студентов DigiPen участвуют в различных фестивалях и конкурсах независимых видеоигр. Все они доступны для скачивания на официальном сайте института.

Кстати, всем известная игра Portal была разработана корпорацией Valve совместно со студентами DigiPen. Предшественником Portal является инди-проект Narbacular Drop, который разрабатывался группой студентов института. Игра вышла, и до сих пор распространяется бесплатно. Игровой процесс в ней состоит из перемещений по подземельям и преодолевания ловушек с помощью системы порталов. Игрок может открыть два связанных портала на любой неметаллической поверхности: стене, полу или потолке.

Впоследствии создатели Narbacular Drop, выступавшие под названием Nuclear Monkey Software, устроили презентацию своей игры в офисе Valve, в ходе которой растроганный Гейб Ньюэлл вскочил со стула, чуть ли не обнял студентов и заявил, что им немедленно стоит войти в состав компании. Дебютным проектом нового отделения Valve, состоявшего из четырех программистов и трех дизайнеров из института DigiPen, и стала легендарная Portal...

Тимур ГАФАРОВ
tgafaroff@gmail.com

Корпорация Zenimax, которой сейчас принадлежит компания id Software, открыла под лицензией GPL исходные тексты наработок, связанных с игрой Doom 3 – включая игровой движок id Tech 4. Помимо Doom 3, на базе id Tech 4 построены и более современные игры – такие, как Quake 4 и Prey. Ресурсы игры (карты, текстуры, звуки и т.д.) по-прежнему являются интеллектуальной собственностью компании Zenimax и распространяются по отдельному соглашению EULA.

К сожалению, из-за наличия, подлежащей лицензированию технологии формирования теней, запатентованной компанией Creative Labs, представленный код не включает в себя реализацию метода Depth Fail для предварительного рендеринга теней. Данная функциональность известна как «Carmack's Reverse». По словам Джона Кармака, для обхода патента требуется добавить четыре строки и изменить две строки кода.

По инициативе юристов компании Zenimax, код был открыт не под лицензией GPL 2, как это делалось ранее, а под GPL 3. При этом текст лицензионного соглашения был немного изменен. Изменения были внесены в 15 раздел «Отказ от гарантий» и в 16 раздел «Ограничение ответственности». Было добавлено упоминание о непредоставлении лицензии на использование торговых марок и логотипов, а также упоминание о том, что, создавая производные проекты на основе представленного кода, вся ответственность перекладывается на разработчика производного продукта.

Открытие Doom 3 продолжило сложившуюся ранее традицию, по которой id Software открывает код своих устаревших игровых движков, с момента выпуска которых прошло пять лет. Например, в прошлом году был открыт код игры Return To Castle Wolfenstein и связанного с ней движка id Tech 3 / Quake 3. В настоящий момент открытыми являются движки игр Wolfenstein 3D, Doom, Quake, Quake 2 и Quake 3, что дает возможность энтузиастам создавать на их базе новые свободные игры. О планах по открытию движка id Tech 5, на основе которого построены Rage и Doom 4, пока ничего не известно.

Исходный код Doom 3 доступен на GitHub: **https://github.com/TTimo/doom3.gpl**

Источник: **www.opennet.ru**

Язык D: новости

> Компилятор DMD обновился до версии 1.072 и 2.057 (для D1 и D2 соответственно). Ключевая особенность релиза – поддержка 64-битных версий Mac OS X. Кроме того, добавлена поддержка классов, интерфейсов и исключений в CTFE. Исправлено множество багов. Также стало известно, что с 31 декабря 2011 г. прекращается рзработка DMD 1.x. Всем владельцам проектов на D1 рекомендуется начать портировать их на вторую версию языка. Компиляторы D1 останутся доступными, но никакой официальной поддержки иметь не будут.

http://www.digitalmars.com/d/1.0/changelog.html
http://ftp.digitalmars.com/dmd.1.072.zip

> xfBuild – утилита для сборки проектов, изначально написанная на D1/Tango – портирована на D2/Phobos. xfBuild считается одной из самых быстрых систем автоматизации сборки для D, так как использует многопоточность и кэширует зависимости между модулями.

https://github.com/AndrejMitrovic/xfBuild
https://bitbucket.org/h3r3tic/xfbuild

> Вышел Mono-D – плагин для среды MonoDevelop, предоставляющий поддержку языка D: подсветку синтаксиса и управление проектами. На данный момент доступна альфа-версия Mono-D.

http://mono-d.sourceforge.net
http://monodevelop.com

> Появился DMagick – объектно-ориентированный API для популярной библиотеки обработки изображений Image-Magick (наподобие Magick++ и RMagick для C++ и Ruby соответственно). DMagick написан на D2 и работает с ImageMagick версии 6.6.0 и выше.

http://code.mikewey.eu/p/DMagick/
http://www.imagemagick.org

> LDC – компилятор D с LLVM в качестве бэкенда – теперь может собирать динамические библиотеки. Для этого используется флаг «-shared». Правда, официального релиза новой версии пока не было объявлено.

http://www.dsource.org/projects/ldc

Воспроизведение WAV-файлов через OpenAL

Практически в любой игре есть звуковые эффекты – даже если нет фоновой музыки. За примером далеко ходить не надо: возьмите, скажем, тот же пасьянс «Паук» в Windows. Поэтому необходимо предусмотреть хотя бы самый простой способ загружать и воспроизводить звук из файла. Ну а формата аудиофайлов проще, чем WAV, наверное, не найти.

Немного истории. Формат Waveform (или просто WAV) был разработан в 1991 году совместно Microsoft и IBM в качестве стандартного формата-контейнера для хранения звукового потока на PC. Это реализация спецификации RIFF (Resource Interchange File Format) – метода хранения данных в виде порций (chunks), близкого к 8SVX и AIFF для компьютеров Amiga и Macintosh соответственно.

Под Windows этот формат чаще всего используется в качестве оболочки для несжатого звука, когда для каждого отсчета амплитуды сигнала выделяется определенное число бит. Однако в контейнер WAV можно поместить звук, сжатый почти любым кодеком (но с воспроизведением таких файлов могут возникать проблемы).

Мы рассмотрим простейший декодер WAV, поддерживающий только несжатый звук – PCM. Вывод звука будет осуществляться через OpenAL.

```
module main;

import std.stdio;
import std.stream;
import std.conv;

import derelict.openal.al;
```

Объявим необходимые структуры. В первую очередь, заголовок-дескриптор, состоящий из идентификатора (4 байта), целочисленного обозначения размера порции и типа формата (4 байта). В идентификаторе содержится текст «RIFF», в типе формата – «WAVE».

```
struct WAVHeader
{
    ubyte[4] id;
    int size;
    ubyte[4] type;
}
```

Структура WAVChunk содержит информацию о порции данных (идентификатор и размер):

```
struct WAVChunk
{
    ubyte[4] id;
    int size;
}
```

WAVFormatChunk хранит метаданные о звуковом файле (тип сжатия, количество каналов, частоту дискретизации, битрейт и др.):

```
struct WAVFormatChunk
{
    short compression;
    short channels;
    int sampleRate;
    int bytesPerSecond;
    short blockAlignment;
    short bitsPerSample;
}
```

Удобнее оформить декодер в специальный класс WAVSound, экземпляры которого будут хранить отдельные звуки.

```
class WAVSound
{
    protected:
    WAVFormatChunk format;
    ubyte[] data;
    ALuint buffer;
    ALuint source;
    /* ... */
}
```

В конструктор передается имя WAV-файла. Для загрузки данных из файла мы будем использовать std.stream.File.

```
public this(string filename)
{
    auto wavFile = new std.stream.File(filename);

    ubyte[WAVHeader.sizeof] wavHeaderArray;
    WAVHeader wavHeader;

    wavFile.read(wavHeaderArray);
    wavHeader = cast(WAVHeader)wavHeaderArray;
```

Благодаря тому, что в D можно легко преобразовывать массив байтов в структуру оператором приведения (cast), загрузка данных не представляет никакой сложности.

После загрузки дескриптора, мы вводим проверку на валидность:

```
    if (wavHeader.id!="RIFF" || wavHeader.type!="WAVE")
        throw new Exception(
            filename ~ " is not a valid RIFF/WAV file");
```

Аналогично – читаем следующую порцию (это должны быть метаданные):

```
    ubyte[WAVChunk.sizeof] wavChunkArray;
    WAVChunk wavChunk;

    wavFile.read(wavChunkArray);
    wavChunk = cast(WAVChunk)wavChunkArray;
```

```
    if (wavChunk.id == "fmt ")
    {
        ubyte[WAVFormatChunk.sizeof] wavFormatChunkArray;
        wavFile.read(wavFormatChunkArray);
        format = cast(WAVFormatChunk)wavFormatChunkArray;
    }

    if (format.compression != 1)
        throw new Exception(
            filename ~ " is not a PCM WAV file");
```

Наконец, загружаем непосредственно аудиоданные и закрываем файл:

```
    wavFile.read(wavChunkArray);
    wavChunk = cast(WAVChunk)wavChunkArray;

    if (wavChunk.id == "data")
    {
        data = new ubyte[wavChunk.size];
        wavFile.read(data);
    }

    wavFile.close();
```

Перед тем, как перейти к работе с OpenAL, следует выбрать правильный формат аудио:

```
    ALenum alformat;
    if (format.channels == 1)
    {
        if (format.bitsPerSample == 8)
            alformat = AL_FORMAT_MONO8;
        else if (format.bitsPerSample == 16)
            alformat = AL_FORMAT_MONO16;
    }
    else if (format.channels == 2)
    {
        if (format.bitsPerSample == 8)
            alformat = AL_FORMAT_STEREO8;
        else if (format.bitsPerSample == 16)
            alformat = AL_FORMAT_STEREO16;
    }
```

А вот теперь создаем буфер и источник звука в OpenAL:

```
    alGenBuffers(1, &buffer);
    alGenSources(1, &source);

    alBufferData(buffer,
                 alformat,
                 data.ptr,
                 data.length,
                 format.sampleRate);

    alSource3f(source, AL_POSITION,        0.0, 0.0, 0.0);
    alSource3f(source, AL_VELOCITY,        0.0, 0.0, 0.0);
    alSource3f(source, AL_DIRECTION,       0.0, 0.0, 0.0);
    alSourcef (source, AL_ROLLOFF_FACTOR,  0.0          );
    alSourcei (source, AL_SOURCE_RELATIVE, AL_TRUE      );
    alSourcei (source, AL_BUFFER,          buffer       );
}
```

Для управления воспроизведением необходимо еще ввести несколько методов:

```
void play()
{
    alSourcePlay(source);
}

void pause()
{
    alSourcePause(source);
}

void stop()
{
    alSourceStop(source);
}

void rewind()
{
    alSourceRewind(source);
}

bool playing()
{
    ALenum state;
    alGetSourcei(source, AL_SOURCE_STATE, &state);
    return (state == AL_PLAYING);
}

void free()
{
    alDeleteSources(1, &source);
    alDeleteBuffers(1, &buffer);
}
```

Привожу простой пример использования класса WAVSound:

```
void main()
{
    DerelictAL.load();

    ALCchar* defaultDevice = cast(ALCchar*)alcGetString(null,
        ALC_DEFAULT_DEVICE_SPECIFIER);
    ALCdevice* device = alcOpenDevice(defaultDevice);

    assert (device !is null, "failed to open audio device");
    writefln("OpenAL device: %s", to!string(defaultDevice));

    ALCcontext* context = alcCreateContext(device, null);
    alcMakeContextCurrent(context);
    alcProcessContext(context);

    scope(exit)
    {
        alcMakeContextCurrent(null);
        alcDestroyContext(context);
        alcCloseDevice(device);
    }

    WAVSound snd = new WAVSound("sound.wav");
    snd.play();
    while(snd.playing) { }
    snd.free();
}
```

Тимур ГАФАРОВ
tgafaroff@gmail.com

Проблемы быстродействия

Быстродействие никогда не выйдет из моды. Чтобы обеспечить максимальное быстродействие программы, ее необходимо оптимизировать. Об этом знают все программисты, занимающиеся разработкой игр. Игры должны работать быстро, иначе они будут плохо продаваться. С каждым годом игроки желают иметь все более высокое быстродействие. Каждая новая игра-бестселлер устанавливает новые стандарты и поднимает планку еще выше.

Впрочем, на играх свет клином не сошелся. Скажем, разработчики САПР, графических редакторов и образовательных программ в меньшей степени озабочены вопросами оптимизации и быстродействия, чем игровые программисты. Но все эти приложения тоже должны работать достаточно быстро. Никому не нравится работать с медленными программами. Пользователи желают, чтобы события на компьютере происходили сразу же, а не через несколько секунд. Вполне достойный, но плохо оптимизированный пакет может проиграть в конкурентной борьбе.

Не стоит полагать, что каждый пользователь, знакомясь с новой программой, говорит себе: «Пусть эта штука работает побыстрее, а не то...» Быстродействие программы чаще оценивается на подсознательном уровне. Работа с быстрой программой, мгновенно реагирующей на все ваши команды, доставляет радость. Хорошее быстродействие вселяет в пользователя уверенность и желание работать дальше. Медленные программы лишь испытывают наше терпение. Каждый, кому приходится работать с ними, мечтают поскорее закончить свои мучения. Пользователи предпочитают, чтобы любая программа (текстовый или графический редактор, игра или любое другое приложение) быстро и адекватно реагировала на их действия.

В этой статье речь пойдет о некоторых практических аспектах быстродействия, знакомство с которыми позволит вам поднять свои программы на новый уровень.

Эксперты по оптимизации программ написали многие тома на эту тему. Из их исследований мы узнали много полезного о том, как написать программу, выполняющую свои функции за минимальное время. Мы не будем рассматривать такую оптимизацию – эта тема слишком обширна.

Усилия, затраченные на оптимизацию, окупаются лишь в некоторых частях программы. Не стоит напрасно тратить время на возню с однократно выполняемым кодом, однако оптимизация кода в циклах и часто вызываемых функциях оказывается жизненно важной.

Аппаратная часть быстрее программной. Приложение, которое работает в режиме 1024x768x24 и при этом обеспечивает вывод 60 кадров в секунду, было бы невозможно написать без аппаратного ускорения. Так как большинство движков автоматически использует все возможности для аппаратного ускорения, вам даже не придется беспокоиться на этот счет.

Я упоминаю об этом лишь по одной причине: если компьютер пользователя не обладает возможностями аппаратного ускорения (или такие возможности слабы), вам не удастся почти ничего сделать. Не стоит беспокоиться о подобной ситуации, потому что проблема заключается в аппаратной части, а не в вашей программе. Если можно, постарайтесь добиться оптимального быстродействия за счет поддержки видеорежимов с низким разрешением. После этого можете считать, что сделали все возможное.

Нехватка видеопамяти. На большинстве новых видеокарт установлено свыше 128 Мб памяти, но многие карты имеют лишь 64 Мб. Это не так уж много. С увеличением разрешения проблема становится еще острее. В зависимости от объема установленной памяти, можно рекомендовать использование различных видеорежимов, обеспечивающих хорошее быстродействие. Когда свободная видеопамять кончается, текстуры создаются в системной памяти. Хотя системная память и обладает некоторыми преимуществами по сравнению с видеопамятью, быстродействие не относится к их числу.

FPS – еще не все. В наши дни часто приходится слышать о частоте вывода кадров, или FPS (Frames Per Second, количество кадров в секунду). Этот показатель стал критерием для сравнения графических приложений. Однако, все же не стоит переоценивать важность этой характеристики. FPS показывает, с какой частотой приложение обновляет информацию, выводимую видеокартой на экран. Тем не менее может возникнуть ситуация, при которой частота генерации содержимого для новых кадров превышает частоту смены кадров, установленную для видеокарты и монитора.

В этом случае часть кадров пропадет, потому что видеокарта не будет успевать выводить (а монитор – отображать) генерируемые данные. В идеальном варианте приложение должно генерировать кадры со скоростью, соответствующей кадровой частоте видеокарты и монитора, и для этого есть несколько причин.

Во-первых, такая синхронизация предотвращает эффект расхождения, потому что монитор всегда выводит полностью сформированное изображение. Во-вторых, нет смысла генерировать кадры быстрее, чем человеческий глаз может их воспринимать. Частота смены кадров на мониторе выбирается с учетом человеческого восприятия; следовательно, если приложение генерирует кадры с частотой их смены монитором, то оно выводит максимум визуальной информации, воспринимаемой человеческим глазом. Сказанное оказывается особенно справедливым для видеорежимов с частотой смены кадров в 60 Гц и выше.

Долой аппаратную зависимость! Если в приложении используются возможности, поддерживаемые лишь некоторыми видеокартами, такое приложение называется аппаратно-зависимым (device-dependent). Приложение не должно полагаться на присутствие конкретной видеокарты или ее специфические возможности, если только вы не пишете демонстрационную программу для производителя этих видеокарт. Подобная зависимость ограничивает рынок сбыта и раздражает пользователей, чьи видеокарты не обладают необходимыми аппаратными возможностями.

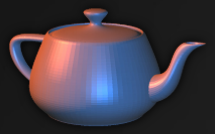
Хороший способ избежать аппаратной зависимости – протестировать приложение на максимальном количестве разных аппаратных конфигураций.

При этом может выясниться, что вы пользуетесь возможностями своей видеокарты, не поддерживаемыми большинством других видеокарт. Всегда лучше узнать об этом самому, чем услышать от недовольных потребителей.

Перестановка кресел на «Титанике». Вспоминая времена своей учебы, я понимаю, что тогда мы слишком беспокоились о быстродействии. Например, один профессор подсчитывал, сколько дополнительных инструкций уходит на вызов функции. Он любил рассуждать о разнице в быстродействии при использовании вызовов функций в стиле C и Pascal и о преимуществах развертки циклов. Он вырос в эпоху перфокарт и ассемблера. Его учили тщательно строить программу на ассемблерном уровне, где приходится учитывать каждый байт, поэтому вполне естественно, что он не выносил напрасной траты байтов и команд процессора. Действительно, вызов функции требует нескольких лишних тактов, а уменьшение количества аргументов экономит память в стеке, но что из того? Преимущества, получаемые от вызова функций и от использования аргументов, оправдывают затраты.

По сравнению с техникой, применявшейся в эпоху перфокарт, современный компьютер невероятно быстр и мощен. Экономить каждый байт в современном программном пакете – то же самое, что пытаться переставлять кресла на палубе «Титаника». Никакого толку от этого занятия не будет – хуже того, оно отвлечет вас от решения настоящих проблем с быстродействием.

Джефф ДАНТЕМАНН
«Графика для Windows средствами DirectDraw»



Модели освещения

Анизотропное распределение Гейдриха-Сейделя

Традиционные модели освещения (например, Ламберта и Фонга) рассматривают так называемые изотропные поверхности. Для этих поверхностей их свойства в точке не зависят от ориентации плоскости – то есть, поворот поверхности вокруг нормали к точке не изменяет интенсивность отраженного от нее света.

Однако в реальной жизни мы обычно сталкиваемся с другим классом поверхностей – анизотропными. Простейшим примером анизотропии является компакт-диск. Структура концентрических бороздок на его поверхности оказывает очень сильное влияние на то, какой мы ее видим. Блики на диске вытянуты перпендикулярно направлению бороздок. Поскольку эти бороздки условно считаются бесконечно тонкими, то для определения освещенности анизотропных поверхностей исходят из следующего соображения: из всего множества нормалей в точке выбирается та, которая дает наибольшую яркость.

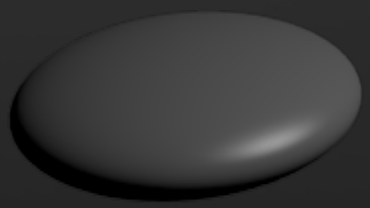
Одной из самых простых анизотропных моделей считается распределение Гейдриха-Сейделя (авторы – Вольфганг Гейдрих и Ганс-Петер Сейдель). Оно основано на модели Фонга.

Его можно использовать при моделировании поверхностей с микроскопическими параллельными бороздками или волокнами, например, полированного металла, волос и некоторых тканей.

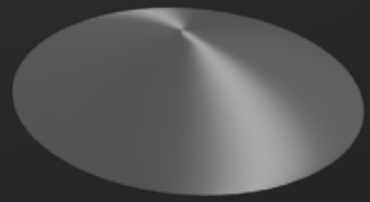
Интенсивность зеркально отраженного света в этой модели задается следующей формулой:

$$k_{spec} = [sin(L,T)sin(V,T) - cos(L,T)cos(V,T)]^n$$

где n – экспонента Фонга, V – вектор наблюдателя, L – вектор источника света и T – вектор направления бороздок в данной точке поверхности. Зачастую его задают в пространстве касательных и переводят в мировые координаты путем умножения на TBN-матрицу.



Изотропная поверхность



Анизотропная поверхность

Реализация на GLSL*

Вершинная программа:

```
varying vec4 V_eye;
varying vec4 L_eye;
varying vec4 N_eye;

void main(void)
{
    gl_Position = ftransform();

    V_eye = gl_ModelViewMatrix * gl_Vertex;
    L_eye = gl_LightSource[0].position - V_eye;
    N_eye = vec4(gl_NormalMatrix * gl_Normal, 1.0);

    V_eye = -V_eye;
}
```

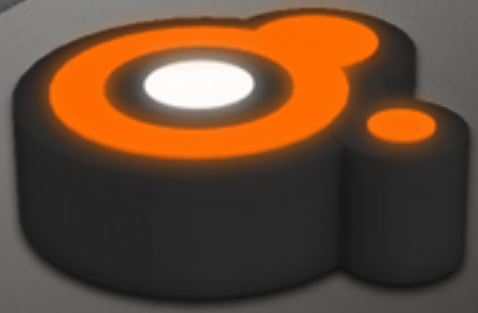
Фрагментная программа:

```
varying vec4 V_eye;
varying vec4 L_eye;
varying vec4 N_eye;

vec3 reflect(vec3 N, vec3 L) {
    return 2.0*N*dot(N, L) - L;
}

void main(void) {
    vec4 Ca = gl_FrontMaterial.ambient;
    vec4 Cd = gl_FrontMaterial.diffuse;
    vec4 Cs = gl_FrontMaterial.specular;

    vec3 V = normalize(vec3(V_eye));
    vec3 L = normalize(vec3(L_eye));
    vec3 N = normalize(vec3(N_eye));
    float diffuse = clamp(dot(L, N), 0.0, 1.0);
    vec3 R = reflect(N, L);
    vec3 D = normalize(vec3(1.0, 1.0, 0.3));
    vec3 T = cross(N, cross(D, N));
    float cosLT = dot(L,T);
    float cosVT = dot(V,T);
    float sinLT = sqrt(1-cosLT*cosLT);
    float sinVT = sqrt(1-cosVT*cosVT);
    float k = clamp(
        pow((sinLT*sinVT - cosLT*cosVT), 16.0), 0.0, 1.0);
    float specular = k + pow(max(dot(R, V), 0.0 ), 64.0);
    gl_FragColor = Ca + (Cd*diffuse) + (Cs*specular);
    gl_FragColor.a = 1.0;
}
```

** - предполагается, что в приложении уже включены и настроены соответствующие параметры OpenGL (позиция источника света, свойства материала и др.) Приведенная реализация в целях упрощения не учитывает интенсивность источника света. Исходный код предоставлен как общественное достояние (Public Domain) и может быть использован для любых целей безо всяких ограничений.*

Это все!

Надеемся, номер вышел интересным. Если так, поддержите FPS! Отправляйте статьи, обзоры, интервью и прочее на любые темы, касающиеся игр, графики, звука, программирования и т.д. на gecko0307@gmail.com.



http://fpsmag.zymichost.com