#10
2012

Независимый электронный журнал
о разработке компьютерных игр.
Издается с 2008 г.

http://fpsmag.zymichost.com

FPS

Веб-сервер «своими руками»
Часть I

Полная SOPA!

BMesh: шаг вперед

Подавление шума в Cycles

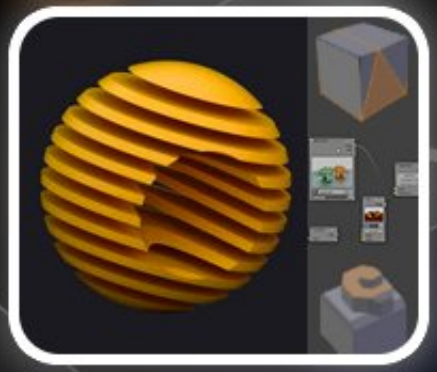
BLENDER. Обзор дополнений
Победители DOMINANCE WAR V
Функциональное программирование

POD или не POD?
«Типографская печать» на GLSL
...и многое другое!

Главный редактор: Тимур Гафаров
Дизайн и верстка: Тимур Гафаров

fpsmag.zymichost.com

Blender



BMesh. Шаг вперед
Подавление шума в Cycles
Обзор дополнений. Выпуск 2

Dominance War V

Полная SOPA!

Язык D

Новости
Функциональное программирование
POD или не POD?
Веб-сервер своими руками. Часть I

GLSL

Эффект типографской печати

BMesh: шаг вперед...

Согласно официальному расписанию, к выходу Blender 2.63 разработчики планируют внесение в основную ветку проекта системы **BMesh** Николаса Бишопа, которая привносит поддержку N-гонов (полигонов с более чем четырьмя ребрами) и множество новых средств для работы с полигональной сеткой. Ранее из всех свободных программ такое было реализовано разве что только в Wings3D. Это настолько значительное и долгожданное событие, что пользователи пакета шутят: вот что майя имели в виду в своем пророчестве о 2012 годе – внесение BMesh в trunk!

Система BMesh является одним из самых востребованных экспериментальных проектов, разрабатываемых для Blender. Это, по сути, новое ядро полигонального моделирования, представляющее собой улучшенный подход к операциям редактирования полигональной сетки. Множество таких операций ранее были либо принципиально недоступны в Blender, либо опирались на «костыли», и пользоваться ими было практически невозможно...

К этому относится, например, отсутствие прямой поддержки полигонов с более чем четырьмя вершинами. Хотя наличие таких элементов в финальной геометрии, как правило, не приветствуется, в процессе создания топологии возможность манипулирования различными много-угольниками освобождает моделлера от рутинного (и совсем необязательного в процессе творчества) «причесывания» сетки. BMesh призван исправить эту ситуацию, внося полноценную поддержку N-гонов и всех необходимых операций над ними. Для полигонального моделлинга BMesh является очень значительным шагом вперед, он без сомнения упростит и ускорит работу с геометрией, что привлечет к Blender немало новых пользователей.

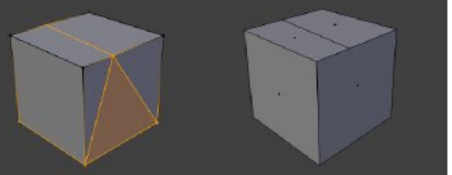
Из важных особенностей BMesh стоит обратить внимание на новый режим удаления **Dissolve**. Теперь можно удалить элемент, не разрушая образованной им геометрии. В Blender и раньше было нечто подобное – удаление кольца ребер (**Edge Loop**), но Dissolve представляет собой гораздо более универсальный инструмент, позволяющий удалять из геометрии вершины, ребра и полигоны.

Наконец-то появился полноценный инструмент «фаска» – **Bevel**, которого так не хватало всем пользователям! Использовать его очень просто: выделите вершины, ребра или грани, которые хотите «заточить», нажмите клавишу W и выберите Bevel.

Другой полезный инструмент – «нож» (**Knife**) – позволяет вырезать на гранях произвольные отверстия. В режиме редактирования нажмите клавишу K и начинайте рисовать контур разрезания на поверхности объекта. Точки добавляются левой кнопкой мыши. Замкнув контур, нажмите Enter, чтобы применить действие «ножа».

BMESH

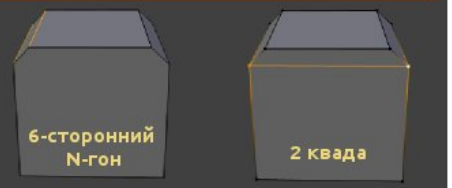
I. В чем суть проблемы?



В текущей реализации любое действие, которое дает грани с более чем четырьмя вершинами, разбивает меш на треугольники. Это достаточно неудобно и только запутывает пользователя…

В случае с BMesh триангуляции не происходит, так как грани с более чем четырьмя вершинами (N-гоны) полностью поддерживаются.

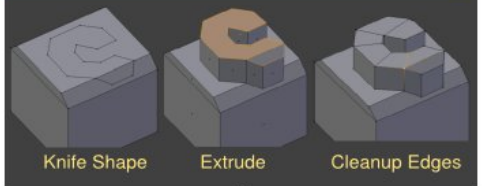
II. Для чего полезны N-гоны?



Они полезны на начальном этапе редактирования, так как делают его неразрушающим.
Можно осоредоточиться на форме и привести сетку в порядок позже.

В данном случае нам для этого достаточно выделить вершины и соединить их.
Без BMesh пришлось бы удалить несколько треугольников и заполнить их заново.

III. Нужно ли переучиваться?



Нет. В основном, вы будете моделировать так, как привыкли.
Новая система просто поможет в некоторых ситуациях сделать работу быстрее и проще.

BMesh поможет работать с мешами со сложной топологией. Даже относительно простая форма (как на рисунке выше) в текущем EditMesh труднодостижима.

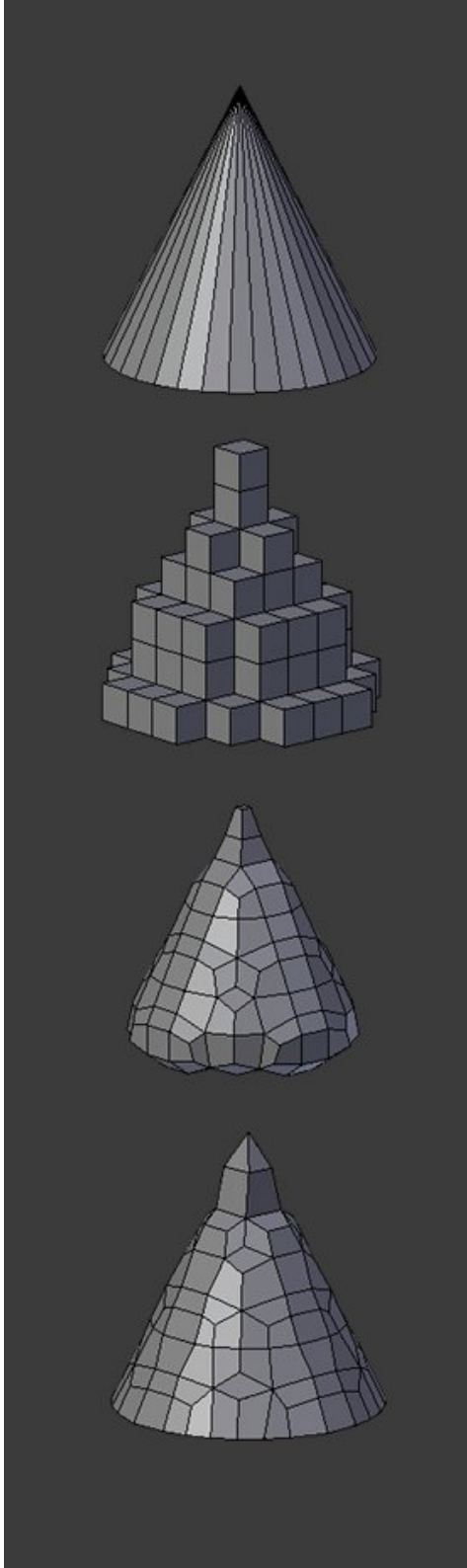
В сообществах Blender ведутся дискуссии о производительности BMesh по сравнению с EditMesh – по всей видимости, придется смириться с некоторым повышением потребления памяти. К тому же, GPU не поддерживает N-гоны аппаратно – при отрисовке через OpenGL они все равно преобразуются в треугольники. Поэтому новая полигональная система немного уступает в скорости старой. Это, правда, не будет особо заметно для обладателей мощных конфигураций. Да и выгоды от использования новых инструментов с лихвой оправдывают все эти минусы (кто моделировал в Wings3D, тот поймет...)

В данный момент ведется адаптация BMesh для поддержки Cycles (они все еще несовместимы между собой), а также подготовка к полной миграции всех инструментов со старого EditMesh на новую систему – большинство существующих модификаторов сетки, например, Boolean, еще не умеют корректно работать с BMesh.

Нельзя не отметить, что стабилизация BMesh откроет дорогу многим другим разработкам, в том числе – **Unlimited Clay**, альтернативной реализации технологии динамической тесселяции при лепке, наподобие той, которая используется в Sculptris. Кстати, один из «дочерних» проектов Бишопа, так или иначе пересекающихся с Bmesh – модификатор **Remesh** – недавно был перенесен в основную ветку. Он предназначен для переразбития сетки в более равномерную топологию, оптимизированную для последующей скульптурной лепки. Модификатор умеет устранять мелкие отсоединенные фрагменты геометрии, а также заполнять отверстия в сетке.

Кроме того, в вышедшей недавно новой версии Blender 2.62 была внесена библиотека булевых операций **Carve**. Она ускоряет и повышает точность вычислений, связанных с логическими операциями. Carve поддерживает неразрушающую модификацию объектов и интерактивную анимацию «разрезания».

А BMesh уже сейчас можно попробовать в действии. Соответствующие сборки есть на **GraphicAll**.



Cycles. Подавление шума

– Видишь суслика?
– Нет.
– И я нет. А он есть...

В «FPS» №17 '11 мы уже упоминали о том, что Cycles дает достаточно много шума при малом количестве сэмплов. Это особенно заметно в сценах с тусклым освещением и большим количеством полутеней. Сообществом активно ведется работа по решению этой проблемы. Недавно **Майк Фарнсворт** опубликовал патч, позволяющий заметно снизить уровень шума путем применения выборки по значимости (**Importance Sampling**, сокращенно IS).

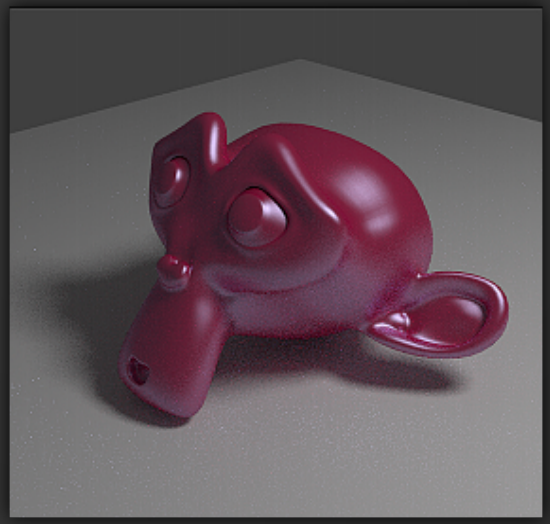
Выборка по значимости – один из методов уменьшения дисперсии случайной величины, который используется для улучшения сходимости процесса моделирования методом Монте-Карло. Идея базируется на том наблюдении, что некоторые значения случайной величины в процессе моделирования имеют большую значимость (вероятность) для оцениваемой функции, чем другие. Если эти «более вероятные» значения будут появляться в процессе выбора случайной величины чаще, дисперсия оцениваемой функции уменьшится. Следовательно, базовая методология IS заключается в выборе распределения, которое способствует выбору «более вероятных» значений случайной величины. Такое «смещенное» распределение изменяет оцениваемую функцию, если применяется прямо в процессе расчета.

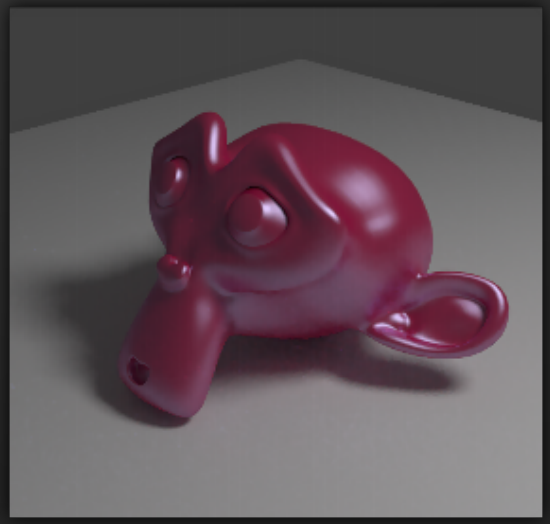
За счет использования карты значимостей (importance map), предложенный Фарнсвортом патч позволяет достичь более качественных результатов при меньшем количестве сэмплов. Правда, скорость рендеринга остается той же, но развитие ядра Cycles не стоит на месте – есть все основания полагать, что движок будет сочетать в себе как высокую производительность, так и профессиональное качество рендеров.

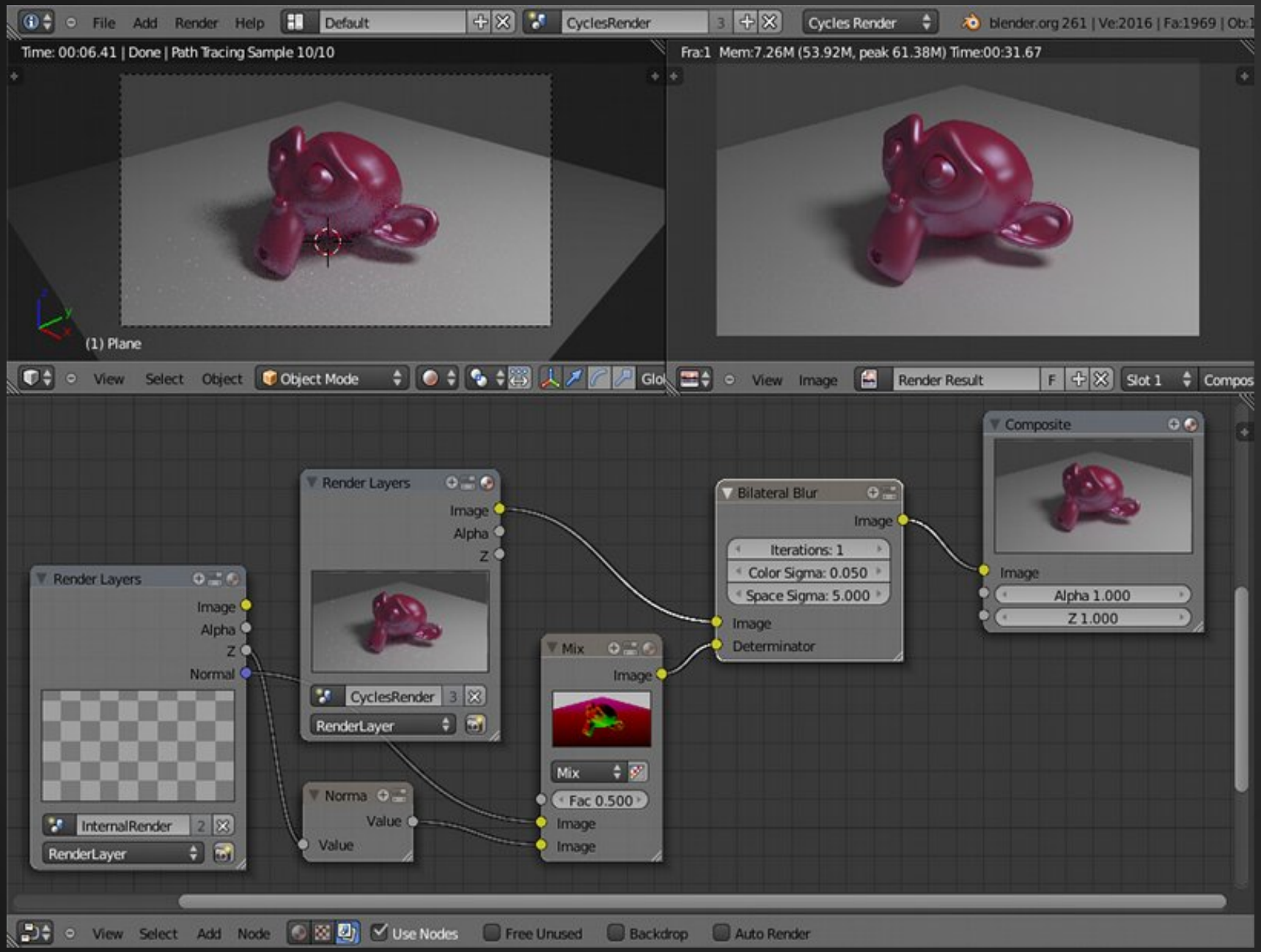
Для подавления шума на полученных при рендеринге изображениях можно также использовать пост-обработку. Для этого в редакторе узлов предусмотрен специальный фильтр – двустороннее размытие (Bilateral Blur). Этот алгоритм основан на обычном гауссовом распределении, но при рассчете средневзвешенного значения он использует разность интенсивности цветов. Иными словами, он не размывает края. Поэтому такой тип размытия часто называется Edge Preserving Blur или адаптивным сглаживанием.

Вся хитрость заключается в том, чтобы передать фильтру в разъем **Determinator** подходящую картинку, на которой будет четкая цветовая разница между объектами – скажем, вектора нормалей в мировом координатном пространстве, «запеченные» в значения RGB. Для этого придется рендерить сцену дважды – встроенным движком и Cycles. К сожалению, для одной сцены можно выбрать только один движок, поэтому необходимо создать новую и добавить на нее ссылки на объекты (**Add scene > Link Objects**). Выберите для этой сцены **Blender Internal** в качестве движка и включите вывод нормалей для активного слоя рендеринга. Затем перейдите на Cycles-сцену и добавьте узлы пост-обработки как на скриншоте.

Gecko
gecko0307@gmail.com






File Add Render Help
Default
CyclesRender 3
Cycles Render
blender.org 261 | Ve:2016 | Fa:1969 | Ob:1
Time: 00:06.41 | Done | Path Tracing Sample 10/10
Fra:1 Mem:7.26M (53.92M, peak 61.38M) Time:00:31.67
(1) Plane
View Select Object Object Mode
View Image Render Result F Slot 1
Render Layers
Image
Alpha
Z
Normal
InternalRender 2
RenderLayer
CyclesRender 3
RenderLayer
Norma
Value
Value
Mix
Image
Mix
Fac 0.500
Image
Image
Bilateral Blur
Image
Iterations: 1
Color Sigma: 0.050
Space Sigma: 5.000
Image
Determinator
Composite
Image
Alpha 1.000
Z 1.000
View Select Add Node
Use Nodes
Free Unused
Backdrop
Auto Render

Обзор дополнений Blender

Выпуск 2

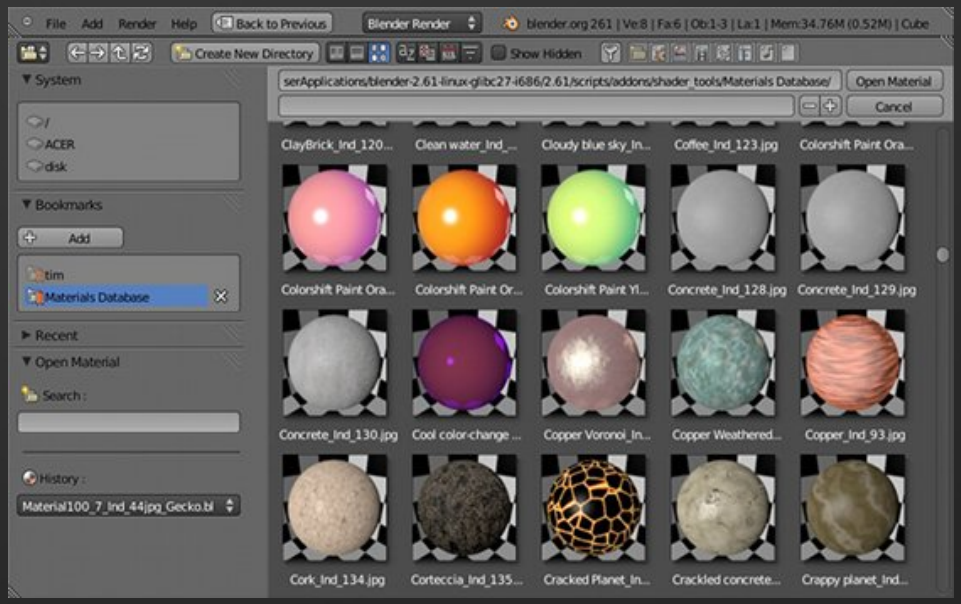
Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. Наш журнал отслеживает выход новых полезных дополнений для Blender 2.6x, которые могут заинтересовать пользователей, использующих программу в качестве инструмента для разработки игр или создания игрового контента...

ShaderTools

Начинает сбываться мечта многих пользователей: недавно было выпущено дополнение, которое реализует библиотеку материалов! ShaderTools позволяет хранить материалы отдельно от blend-файлов и индексировать их по связанной картинке через встроенный файловый менеджер Blender. Для ускорения доступа они по умолчанию хранятся в локальной базе данных SQLite, но любой материал можно сохранить в виде отдельного файла – чтобы передать другим пользователям. После импорта в проект, материал не требует установленного ShaderTools – полученный *.blend можно без проблем передавать другим людям.

ShaderTools находится на стадии beta. Пока нет поддержки Cycles, и английская локализация оставляет желать лучшего... Но в стандартной базе дополнения уже имеется огромная коллекция из 450 (!) готовых материалов. И это только начало.

Вполне возможно, в будущем появится поддержка сетевых хранилищ – только представьте: мгновенный доступ к сотням пользовательских материалов на серверах прямо из окна программы!



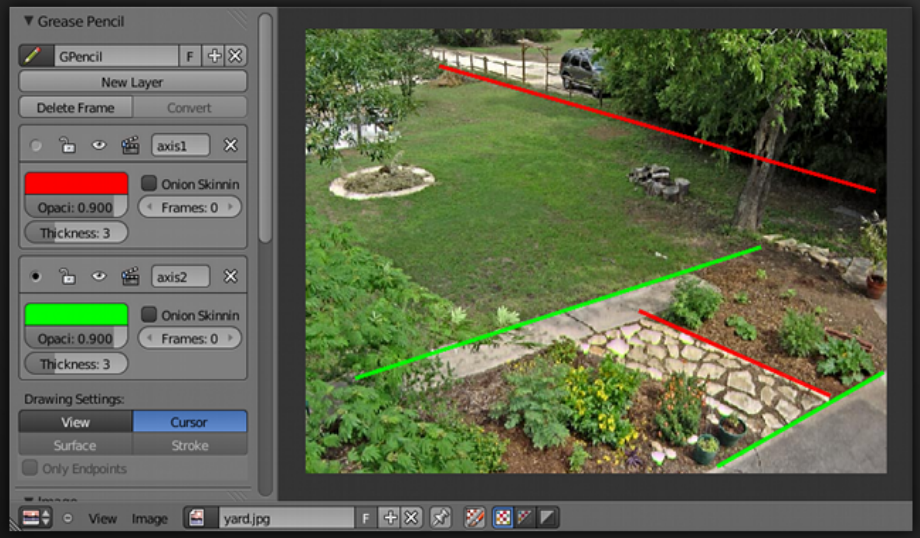
Дополнение доступно по лицензии GNU GPL v2.

http://shadertools.tuxfamily.org
https://github.com/tinangel/ShaderTools

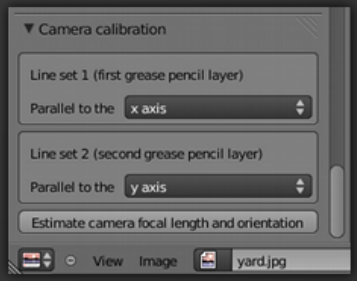
Blam

Blam – это дополнение, призванное облегчить синхронизацию камеры при моделировании поверх фотографий, наподобие того, как это делается в Google SketchUp. Вы наносите на изображении направляющие линии перспективы, и Blam выполнит за вас все необходимые вычисления для точной «подгонки» камеры с учетом фокусного расстояния. Дополнение также позволяет восстановить геометрию из ее перспективной проекции.

Использовать Blam достаточно просто: загрузите фотографию и в редакторе UV/Изображений создайте два слоя Grease Pencil с двумя разными цветами. Удерживая Ctrl+D, нанесите на изображение отрезки, которые передают перспективу на снимке. Отрезки одного слоя должны быть параллельными.



Отрезки двух слоев должны располагаться под прямым углом по отношению друг к другу – иными словами, вы должны показать оси X и Y. Я использовал «красный» слой для X и «зеленый» для Y.

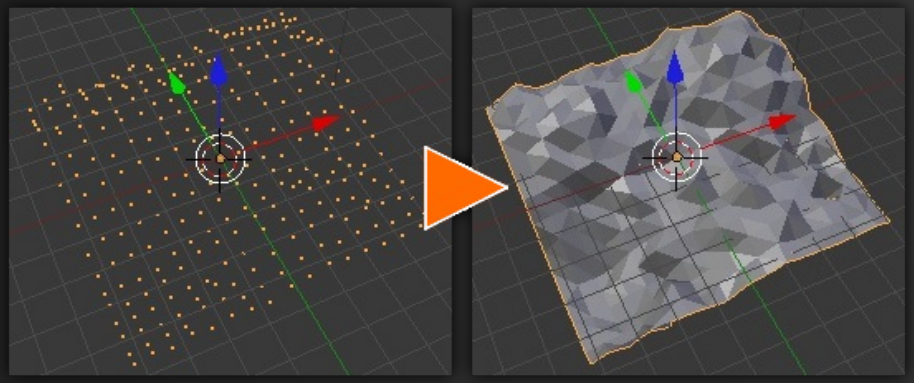


После этого нажмите кнопку **Estimate camera focal length and orientation** на вкладке **Camera calibration** – и готово! Вам остается только выбрать фотографию в качестве фонового изображения для сцены.

http://code.google.com/p/blam/

Point Cloud Skinner

А это дополнение будет невероятно полезно для исследовательских работ, задач статистики и различных экспериментов: оно позволяет создать непрерывную поверхность из массива (облака) отдельных точек. Это могут быть, например, топологические данные о ландшафте или результат вычислений при какой-либо симуляции.



Тред на BlenderArtists.org

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.

После долгих лет войны ткань пространства и времени была разорваны на куски. Боги Земли и Воздуха, демоны, Жизнь и Смерть боролись за власть, но никто не мог победить. Этот хаос продолжился бы, но одна сила оказадась сильнее, чем сами боги – сила Бессознательного. Все живое во Вселенной, от самых маленьких до самых массивных форм, требовало перемен. Они хотели положить конец бесконечному циклу войны. И конец настал...

Так завершается

Dominance War – ежегодный конкурс для художников по персонажам. Это уникальный в своем роде товарищеский турнир, на котором собираются представители крупнейших арт-коммьюнити всего мира, чтобы побороться за звание чемпиона – ведь это не просто почетный титул, а также и крупный денежный приз. Кроме того, за этим конкурсом следят «гиганты» игровой индустрии – вполне вероятно, что победителей примут на работу, например, в Blizzard или в Epic Games. Dominance War располагает обмениваться идеями, техниками, помогает художникам приобрести новые навыки и методы для достижения потрясающих результатов.

В декабре 2011 г. были объявлены победители пятого по счету соревнования.

Золото получили:

«God of Judgement» – Yanbo2k1, Китай (**LeewiArt**)
«Monggoo Mogo» – Сань Хун Ан, Южная Корея (**CGLand**)
«God of the Undead» – Джим Кристофер, США (**CGSociety**)

Серебро:

«God of the Zuris» – Бруно Брито Камара, Бразилия (**GameArtisans**)
«God of Punishment» – Ванг Чжоу, Китай (**LeewiArt**)
«God of Evil» – Чао Янг, Китай (**LeewiArt**)

Бронза:

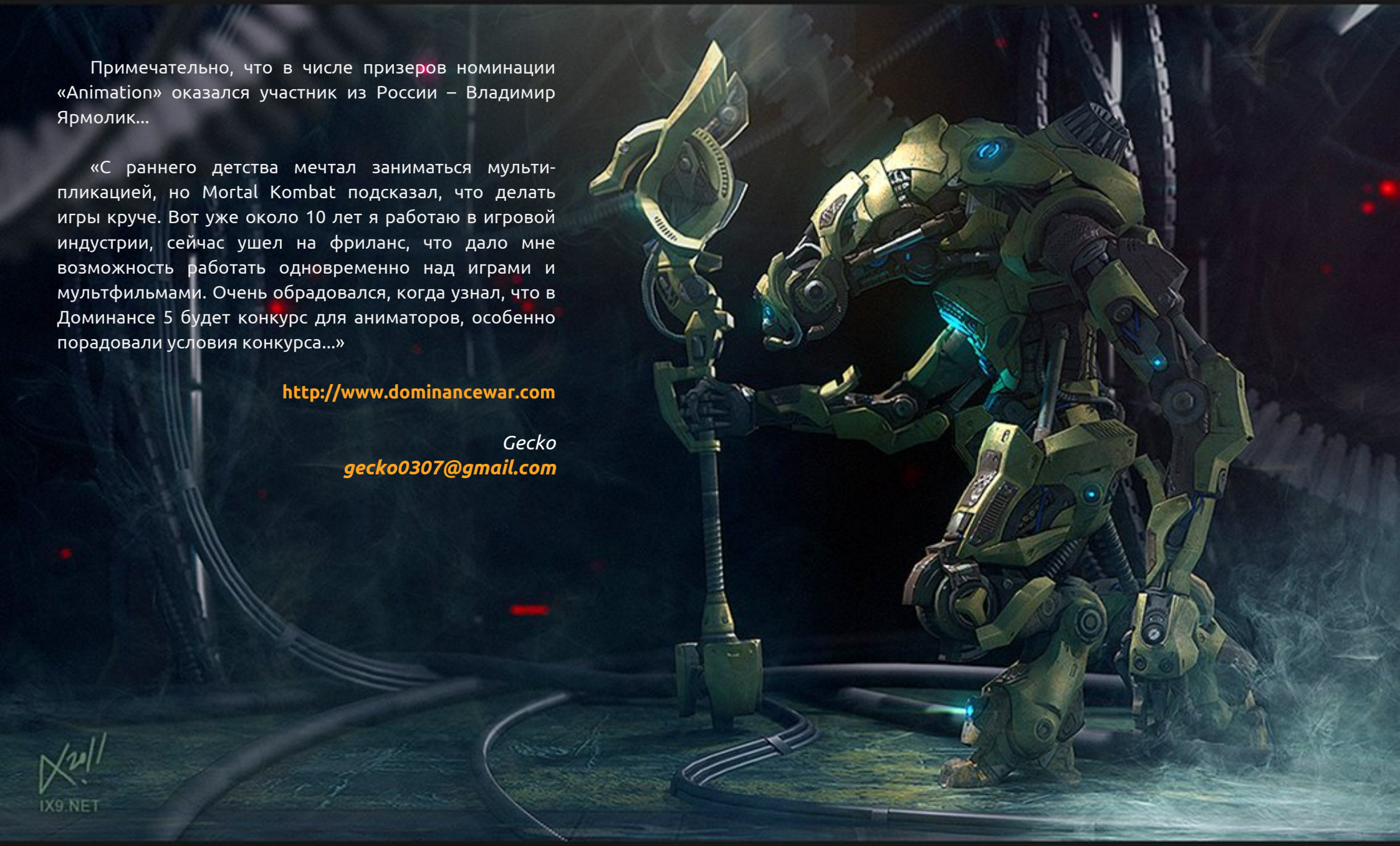
«Gotterdammerung» – Тейлор Кингстон, Канада (**3DTotal**)
«Goddess of Dragons» – Джан Гуан, Китай (**CGSociety**)
«Uroboros» – Владимир Ярмолик, Россия (**3DTotal**)

Примечательно, что в числе призеров номинации «Animation» оказался участник из России – Владимир Ярмолик...

«С раннего детства мечтал заниматься мультипликацией, но Mortal Kombat подсказал, что делать игры круче. Вот уже около 10 лет я работаю в игровой индустрии, сейчас ушел на фриланс, что дало мне возможность работать одновременно над играми и мультфильмами. Очень обрадовался, когда узнал, что в Доминансе 5 будет конкурс для аниматоров, особенно порадовали условия конкурса...»

http://www.dominancewar.com

Gecko
gecko0307@gmail.com



Язык D: новости

DMD 2.058 и 1.073

В феврале состоялся релиз новых версий компилятора DMD 2.058 и 1.073 для D2 и D1 соответственно. Из основных нововведений стоит отметить введение краткого синтаксиса для лямбда-выражений (=>), поддержку инструкций AVX во встроенном ассемблере, улучшенное использование регистров XMM под Mac OS X и многое другое. Кроме того, было исправлено множество багов в компиляторе и в стандартной библиотеке.

Напоминаем, что официальная поддержка D1 будет прекращена 31 декабря 2012 года. Всем разработчикам рекомендуется начать портировать свои проекты на D2.

http://dlang.org/download.html

DDevel

Русскоязычных пользователей наверняка заинтересует DDevel – развивающаяся социальная сеть для разработчиков на D. На сайте есть поддержка групп, микроблогов, закладок, агрегатор активности, вики-движок, а также собственный игровой сервис DDevel Games. Есть сообщества для пользователей D, C++, ShiVa3D, Ogre, Unreal Development Kit, G3D и др.

http://www.ddevel.org/

Tango портирована на D2

В связи со скорым прекращением официальной поддержки D1, сообществом D была проведена работа по адаптации легендарного фреймворка Tango на актуальную вторую версию языка. На данный момент этот проект близок к завершению, большинство модулей успешно портированы. Библиотека компилируется под Linux, Windows и Mac OS X. Поддержка остальных платформ должна быть стабилизирована в ближайшем будущем. Также планируется добавить возможность собирать динамическую версию библиотеки.

Этот порт полностью совместим с последней версией компилятора DMD. Tango можно использовать как замену Phobos или «в паре» с ней – в качестве дополнительной библиотеки, как, например, Boost в C++.

Напомним, что Tango на протяжении многих лет оставалась де-факто основной стандартной библиотекой D1. Одна из самых популярных книг по D – **Learn to Tango with D** раскрывает основы языка на примере именно этой библиотеки.

https://github.com/SiegeLord/Tango-D2
http://www.dsource.org/projects/tango

Кросс-компилятор GDC для Android

В наши дни приобрели актуальность ARM-платформы и операционные системы, совместимые с ними – например, Android. Соответственно, важной стала и поддержка архитектуры ARM со стороны компиляторов различных языков. Недавно на странице проекта GDC появилась практическая инструкция по сборке кросс-компилятора GCC/GDC для компиляции программ на D в нативный ARM-код с использованием Android NDK. В качестве хост-системы предлагается Linux.

Эта инициатива пока носит сугубо экспериментальный характер, но как «proof-of-concept» вполне годится. Первый шаг сделан. Кто знает, может быть, в ближайшем будущем появится полноценная версия GDC для Android.

Внимание: потребуется пропатчить GDC и отказаться от использования Phobos, так как Android не соответствует стандарту Posix.

https://bitbucket.org/goshawk/gdc/wiki/GDC on Android
http://www.android.com
http://ru.wikipedia.org/wiki/ARM_(архитектура)

D:GamesVFS

Для D появилась виртуальная файловая система, предназначенная для разработчиков игр – D:GamesVFS. Она пока находится на ранней стадий разработки: API библиотеки нестабилен, нет поддержки архивов, равно как и возможности удалять файлы и директории. Исходный код проекта доступен по лицензии Boost.

https://github.com/kiith-sa/D-GameVFS

DuckFuck – интерпретатор BrainFuck на D

Хорошая новость для любителей «эзотерических» языков: на D появилась гибкая система интерпретации и трансляции кода FuckFuck и других производных BrainFuck. Проект включает консольный интерпретатор для этих языков. Есть поддержка прямой трансляции на D во время компиляции.

BrainFuck – это полный по Тьюрингу язык программирования, содержащий всего восемь команд. Программы на нем писать очень сложно, за что его иногда называют «языком для мазохистов»...

http://bitbucket.org/Abscissa/duckfuck
http://esoteric.voxelperfect.net/wiki/Brainfuck
http://esoteric.voxelperfect.net/wiki/Fuckfuck

Visual D 0.3.30

Райнер Шутце, разработчик Visual D – проекта по интеграции D в среду разработки Microsoft Visual Studio – выпустил новую версию. Внесены улучшения в подсветку синтаксиса, отладчик, систему конфигурации и сборки. Добавлено экспериментальное автодополнение на основе семантического анализа кода.

Visual D совместим с со всеми версиями Visual Studio 2005-2011.

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

Derelict3

Репозиторий Derelict – коллекции привязок к мультимедийным библиотекам – постепенно начинает переезд на GitHub. Проект получил название Derelict3 – одновременно была начата работа по улучшению механизма загрузки расширений OpenGL и совместимости с последними версиями стандарта. Кроме того, был добавлен биндинг к GLFW – популярной кроссплатформенной библиотеке для работы с OpenGL и устройствами ввода. GLFW используют такие известные игровые движки, как Horde3D и BlendELF.

http://www.dsource.org/projects/derelict
https://github.com/aldacron/Derelict3
http://www.glfw.org

Cook 1.8.5a

Вышла новая версия Cook – программы автоматизации сборки проектов на D. Этот релиз обеспечивает совместимость с последними версиями DMD. Кроме того, была создана новая экспериментальная ветка программы – Chiefcook. В ней планируются фундаментальные нововведения – к примеру, полная переделка парсера зависимостей модулей.

http://code.google.com/p/cook-build-automation-tool

Goldie 0.7

Свободный набор инструментов для парсинга Goldie, совместимый с GOLD Parser Builder, обновился до версии 0.8. Были внесены улучшения в API, исправлено несколько багов, улучшена поддержка новых версий DMD. Репозиторий проекта переехал с DSource/SVN на BitBucket/Git.

http://www.semitwist.com/goldie

D:YAML 0.4

Вышла новая версия D:YAML – парсера языка разметки YAML. Проект ставит целью создание полноценной YAML-библиотеки для D2/Phobos. Этот релиз улучшает совместимость с DMD 2.057 и исправляет баги. Кроме того, был упрощен API библиотеки.

https://github.com/kiith-sa/D-YAML

Mono-D 0.2.9

Mono-D – плагин для среды MonoDevelop, предоставляющий поддержку языка D в этой IDE – обновился до версии 0.2.9. Из числа нововведений стоит отметить новые опции политики, внутренний рефакторинг парсера, улучшение совместимости с DMD 2.058 и т.д.

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

DMD 2.x в репозитории Arch Linux

В январе в общественном репозитории дистрибутива Arch Linux старая версия компилятора DMD была заменена на актуальную ветку 2.x.

http://www.archlinux.org/packages/?q=dmd

DVM 0.4.0

D Version Manager (DVM) – менеджер версий компиляторов D – обновился до версии 0.4.0. Основное нововведение – поддержка сборки DMD, druntime и Phobos из исходников с GitHub. Для этого нужно создать локальную копию репозитория и выполнить команду «dvm compile (имя папки)».

https://bitbucket.org/doob/dvm

Плагин D для Pelles C

Вышел открытый плагин для популярной IDE Pelles C, предоставляющий поддержку языка D. На данный момент он находится на стадии альфа, но уже годится для использования.

Pelles C – бесплатная легковесная интегрированная среда разработки программ на C, работающая на платформах Windows и Pocket PC.

http://my.opera.com/run3/blog/2012/02/09/d-programming-in-pelles-c

Краткий экскурс
в функциональное программирование

В свое время Платон предложил идею, что все в нашем мире всего лишь приближение к идеальному. Он осознал, что мы понимаем идеальное, ни разу в жизни не столкнувшись с ним. И Платон пришел к выводу, что идеальные математические формы должны жить где-то в другом мире, и что мы каким-то образом узнаем о них – будто у нас есть связь с этим миром.

Например, совершенно ясно, что не существует идеальной окружности, которую мы можем наблюдать. Однако мы можем описать идеальную окружность при помощи уравнений. Тогда что такое математика? Почему Вселенная описывается математическими законами? Можем ли мы все явления в нашей Вселенной описать с помощью математики?

Надо сказать, что утвердительный ответ на последний вопрос довольно сомнителен. Ученым пришлось признать: далеко не очевидно, что все во Вселенной подчиняется законам, которые можно описать с помощью математики... Математика, на самом деле – головоломка. Сначала мы создаем базовый непротиворечивый набор принципов и правил. Затем мы можем сочетать их, чтобы получить еще более сложные правила. Ученые называют этот метод «формальной системой»...

«Отцом» функционального программирования считается математик Алонзо Черч. Он и Алан Тьюринг, Джон фон Нейман и Курт Гедель сконцентрировали свои исследования на формальных системах. Они не слишком старались сохранить связь с физическим миром – гораздо больше они интересовались абстрактными математическими головоломками. Они искали ответы на вопросы о вычислениях.

Если бы у нас была машина бесконечной вычислительной мощности, какие задачи мы бы смогли решать? Могли бы мы их решать автоматически? Могли ли некоторые задачи остаться неразрешимыми, и почему? Могли ли разные машины с разной архитектурой быть равными по мощности?

Черч разработал формальную систему, которая получила название **лямбда-исчисления**. Эта система была на самом деле «языком программирования» для воображаемой машины.

Язык был основан на функциях, которые принимали функции в качестве параметров и возвращали функции в качестве результата. Функции обозначались греческой буквой **λ**, откуда и название. Используя эту формальную систему, Черч смог исследовать вопросы, приведенные выше, и дать убедительные ответы на них.

Алан Тьюринг работал в аналогичном направлении. Он разработал другую формальную систему – которую сейчас называют «машиной Тьюринга» – и с ее помощью пришел к тем же самым выводам, что и Черч. Позже Тьюринг доказал, что его «машина» и лямбда-исчисление эквивалентны по мощности.

Функциональное программирование (ФП) – это практическая реализация идей Алонзо Черча. Не все идеи из лямбда-исчисления оказались практически полезными, поскольку оригинальное лямбда-исчисление не было зажато рамками физических ограничений. Таким образом, ФП – это множество идей, но не множество жестких правил.

На сегодняшний день существует целый набор функциональных языков, и многие из них делают одно и то же самыми различными способами. В этой статье я постараюсь изложить некоторые из наиболее широко распространенных концепций функциональных языков, используя примеры, написанные на языке D, который является, наверное, самым удачным компромиссом между функциональными и императивными языками.

Лямбда-исчисление было создано, чтобы исследовать задачи, связанные с вычислениями. Таким образом, ФП, прежде всего, имеет дело с вычислениями – и, что самое интересное, использует функции для выполнения вычислений. Функция – это базовый «кирпич» в ФП. Функции используются почти везде, даже в простейших вычислениях. Даже переменные – и те заменены функциями. В функциональных языках переменные не могут быть изменены, они принимают значения только один раз.

Это может показаться жутко странным. Но функциональные языки предлагают совершенно новые способы абстракции, после которых вам в большинстве случаев не захочется модифицировать переменные. Один из таких способов – это возможность работать с **функциями высшего порядка (higher-order functions)**. Это функции, принимающие в качестве аргументов другие функции или возвращающие другую функцию в качестве результата. **Функция первого порядка (fist-order function)** – функция, которая может принимать на вход в качестве входного параметра только значения «простых» (не функциональных) типов, а также возвращать значения таких «простых» типов в качестве результата.

Чистое ФП подразумевает отсутствие состояния вычисления и немодифицируемость памяти. Грубо говоря, вся программа в чистом ФП представляет собой одну большую «формулу». В ФП есть понятие **чистой функции (pure function)** – функции, которая не имеет побочных эффектов ввода-вывода и памяти, зависит только от своих параметров и возвращает только свой результат.

```
pure int add(int x, int y)
{
    return x + y;
}
```

На самом деле, функциональные программы, конечно же, могут хранить состояние. Правда, в отличие от императивных языков, они используют для этого не переменные, а функции. Состояние хранится в параметрах функции, в стеке.

В ФП функция является **объектом первого порядка (first-class object)**. Это значит, что функцию можно создать, можно присвоить переменной, можно передать в другую функцию, можно вернуть из функции. Для этого используются лямбда-выражения и делегаты.

Лямбда-выражение (lambda-expression, λ) – это специальный синтаксис для объявления **анонимных функций (anonymous function)**. Это особый вид функции, которая объявляется в месте использования и не получает уникального идентификатора для доступа к ней. Обычно при создании она либо вызывается напрямую, либо ссылка на функцию присваивается переменной, с помощью которой затем можно косвенно вызывать данную функцию.

```
auto d = (int x, int y){ return x + y; };
```

В этом контексте «лямбда» можно понимать как «функция» – просто в математической нотации одну греческую букву писать легче.

Делегат – безопасный указатель на функцию.

```
int bar( int delegate(int) d )
{
    return d(10);
}

int foo(int a)
{
    return a + a;
}

int i = bar(&foo);
```

Кстати, в D 2.058 была внесена поддержка специального краткого синтаксиса для объявления лямбда-выражений:

```
auto lambda = (int x) => x * x;
```

Анонимные функции обладают свойством **замыкания (closure)** – то есть, имеют доступ в окружающий контекст. Такая функция может быть определена в теле другой функции.

При этом вложенная внутренняя функция содержит ссылки на локальные переменные и аргументы внешней. Каждый раз при выполнении внешней функции происходит создание нового экземпляра внутренней функции, с новыми ссылками на переменные внешней функции.

Замыкание связывает код функции с ее лексическим окружением (местом, в котором она определена в коде). Лексические переменные замыкания отличаются от глобальных переменных тем, что они не занимают глобальное пространство имен. От переменных в объектах они отличаются тем, что привязаны к функциям, а не объектам.

```
auto getAdder(int x)
{
    return (int y)
    {
        return x + y;
    };
}
```

В ФП часто используется прием, который называется **каррированием (currying)** – преобразование функции от пары аргументов в функцию, берущую свои аргументы по одному. Классический пример каррирования – преобразование функции, которая возводит число в степень, к функции, которая возводит число в квадрат:

```
auto square = (int x) => pow(x, 2);
int i = square(10);
```

В функциональных языках эта техника уже встроена изначально. Вам не нужно вручную создавать функцию, которая «обертывает» оригинал, за вас это сделает сам язык. В D каррирование реализовано в стандартной библиотеке Phobos:

```
import std.functional;

auto foo = (int x, int y) => x * y;
alias curry!(foo, 10) bar;
int i = bar(5);
```

Другая интересная техника – **отложенные вычисления (lazy evaluation)**. Их еще называют «ленивыми». Это концепция, согласно которой вычисления следует откладывать до тех пор, пока не понадобится их результат. Она, в частности, позволяет сократить общий объем вычислений за счет тех вычислений, результаты которых не будут использованы.

```
void dotimes(int n, lazy void exp) {
    while (n--) exp();
}
int x = 0;
dotimes(10, writeln(x++));
```

Один из основных приемов ФП – композиция (composition), комбинирование двух функций с тем, чтобы получить третью.

```
T delegate(S) compose(T, U, S) (T delegate(U) f,
                                U delegate(S) g)
{
    return (S s) { return f(g(s)); };
}

auto mult10 = (real x) { return x * 10; };
auto div2 = (real x) { return x * 0.5; };

auto mult10div2 = compose(mult10, div2);

writeln( mult10div2(30) );
```

Программировать в функциональном стиле означает решать, *что* вычислять, а не *как* вычислять. Это ведет к тому, что функциональные программы гораздо короче императивных, их намного проще читать.

Например, используя приведенные ниже шаблоны функций можно написать выражение в истинно функциональном духе. Оно выдает сумму всех четных чисел от 1 до 100 включительно:

```
int i = range(1,101).where(predicate!(int,"x%2==0")).sum;
```

```
void forevery(T) (T[] arr, void delegate(ref T) func)
{
    foreach(i, v; arr) func(arr[i]);
}

T[] range(T) (T minb, T maxb)
in
{
    assert (minb < maxb);
}
body
{
    T[] r = new T[maxb - minb];
    foreach(i, v; r) r[i] = minb + i;
    return r;
}

T[] where(T) (T[] arr, bool delegate(T i) func)
{
    auto newArr = arr.dup;
    size_t j = 0;
    arr.forevery((ref T v)
    {
        if (func(v))
        {
            newArr[j] = v;
            j++;
        }
    });
    newArr.length = j;
    return newArr;
}
```

```
bool delegate(T) predicate(T, string s) ()
{
    return mixin("("~T.stringof~"x){return("~s~");}");
}

T sum(T) (T[] numbers...)
{
    T result = 0;
    foreach(n; numbers) result += n;
    return result;
}
```

В некоторых языках предусмотрены объекты, которые можно использовать как функции – их называют **функторами (functor)**. В D функтор легко объявить, перегрузив для класса оператор вызова (opCall):

```
class Functor
{
    int a;
    this(int ia)
    {
        a = ia;
    }
    int opCall(int b)
    {
        return a + b;
    }
}

auto f = new Functor(10);
int i = f(20); // i == 30
```

В D можно, например, специализировать функторами шаблоны:

```
auto foo(T) (T func)
{
    return func(20);
}

int i = foo(f); // i == 30
```

Этот же шаблон, кстати, принимает и «настоящие» функции:

```
auto d = (int x) => x * x;
int i = foo(d); // i == 400
```

У ФП множество преимуществ перед традиционным императивным программированием. Например, если функциональная программа не работает должным образом, то вы всегда сможете воспроизвести проблему – поскольку ошибки в функциональной программе не зависят от посторонних участков программы, которые исполнялись до этого.

Вы наверняка сталкивались с ситуацией, когда в императивной программе ошибки не воспроизводятся регулярно. Есть даже соответствующая поговорка из области программистского юмора: «в зависимости от фазы Луны...» Поскольку работа функций зависит от внешнего состояния, сформированного побочными эффектами других функций, при отладке вы можете вообще начать двигаться в направлении, никак не связанном с ошибкой.

В функциональной программе такое невозможно – если возвращаемое значение функции неправильное, оно всегда неправильное, независимо от того, какой код исполнялся до вызова функции.

Но самое главное – функциональная программа уже готова для исполнения в параллельном режиме без дополнительных модификаций. Вам не нужно беспокоиться о взаимоблокировках (deadlocks) и борьбе за ресурсы (race conditions), вам просто не потребуется использовать мьютексы и другие механизмы синхронизации. Никакой блок данных не модифицируется дважды одним и тем же потоком, не говоря уже о двух и более. Это означает, что вы легко можете добавлять сколько угодно потоков, не утруждая себя мыслями о привычных проблемах, которые неизбежно сопровождают многопоточные приложения!

Источники:
http://www.rsdn.ru
http://ru.wikipedia.org

POD или не POD?

Для более полного понимания объектной модели D, следует помнить, что в этом языке понятие структуры строго отделено от понятия класса. Это уходит корнями в давний вопрос POD-типов.

Все типы данных делятся на две группы: объектные типы и все остальные. Группа объектных типов содержит две подгруппы – POD и не-POD. POD – это аббревиатура от «Plain Old Data», что можно перевести как «Простые данные». В старых языках, например, в стандартном Pascal и C все типы данных рассматриваются как простые.

Для не-POD-типа нельзя сделать никаких предположений о том, как устроен объект, так как это не оговорено стандартом языка. Внутри такого объекта в произвольном месте могут располагаться служебные области, неподконтрольные программисту. Формат представления не-POD-объектов не стандартизирован для того, чтобы не связывать руки производителям компиляторов.

POD-типы, напротив, совершенно прозрачны. Данные в них располагаются строго так, как это было объявлено программистом. Простые данные не требуют автоматического управления. Переменные такого типа можно копировать простыми процедурами копирования байтов памяти.

К POD-типам в C++ относятся все встроенные арифметические типы, перечисления (т.е. типы, объявленные с помощью ключевого слова enum), указатели, а также некоторые классы/структуры. Чтобы считаться POD, структура в C++ должна удовлетворять следующим требованиям:

- не должны содержать пользовательских конструкторов, деструктора или копирующего оператора присваивания;
- не дожны иметь базовых классов;
- не должны содержать виртуальных функций;
- не должны содержать защищенных (protected) или закрытых (private) нестатических членов данных;
- не должны содержать нестатических членов данных не-POD-типов (или массивов из таких типов).

Ключевые слова «class» и «struct» в C++ могут объявлять как POD, так и не-POD тип – в зависимости от семантики. Очевидно, что такой подход представляет собой, мягко говоря, просчет в дизайне языка...

В D слово «struct» **всегда** объявляет POD-структуру, а «class» – не-POD. В отличие от C++, в D для POD-структур поддерживаются конструкторы и деструкторы.

```
struct S
{
    int a, b;
    this(int ia, int ib)
    {
        a = ia;
        b = ib;
    }
}

S s = S(10, 5);
```

Кроме того, необходимо помнить, что структуры D по умолчанию передаются по значению, а классы – всегда по ссылке, в то время как в C++ и то, и другое по умолчанию передается по значению.

Gecko
gecko0307@gmail.com

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

Веб-сервер своими руками
Часть I

В последнее время возрос интерес к веб-программированию. В этой связи современные языки программирования часто рассматриваются на примере написания несложного веб-сервера. Попробуем написать такой на языке D – он будет слушать порт 80 и передавать содержимое запрошенного файла при его наличии в папке, откуда был запущен. Но для начала – немного теории...

Для обеспечения сетевых коммуникаций используются сокеты. **Сокет** – это интерфейс двунаправленной связи, который используется для взаимодействия программ с друг с другом – на той же машине или на других, через сеть. Обычно используются сокеты Беркли, история которых уходит корнями в легендарную операционную систему BSD (Berkeley Software Distribution).

Сокеты существуют в областях связи (доменах). Домен сокета (не путать с доменными именами Интернета) – это абстракция, которая определяет структуру адресации и набор протоколов. Сокеты могут соединяться только с сокетами в том же самом домене. Обычно используются **UNIX**-сокеты и Интернет-сокеты (**INET**). Домен UNIX обеспечивает адресное пространство сокетов для локальной вычислительной системы. Для организации связи между процессами на различных системах используется домен INET. Коммуникации этого домена используют стек протоколов **TCP/IP**.

BSD 4.2 была первой ОС, в которой протокол TCP/IP был включен в состав ядра, и в которой был предложен программный интерфейс этого протокола – сокеты. Они, таким образом, представляют собой API между прикладными программами и сетевыми уровнями. API сокетов Беркли сформировал де-факто стандарт абстракции для сетевых сокетов во всех современных системах и языках. Реализация TCP/IP из BSD была заимствована при создании MS Windows и Apple Mac OS.

TCP/IP работает в качестве стека (stack) – это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Например, TCP работает поверх IP. Стек TCP/IP включает в себя протоколы пяти уровней:

- прикладной: HTTP, FTP, SMTP, DNS и др.
- транспортный: TCP, UDP, SCTP, DCCP
- сетевой: IP, ICMP, IGMP
- канальный: Ethernet, IEEE 802.11, SLIP, Token Ring и др.
- физический: провода, радиосвязь и т.д.

Физический уровень описывает среду передачи данных (коаксиальный кабель, витая пара, оптическое волокно или радиоканал), физические характеристики такой среды и принцип передачи данных (разделение каналов, модуляцию, амплитуду и частоту сигналов, время ожидания ответа и т.д.)

Канальный – уровень межсетевых интерфейсов – описывает, каким образом данные передаются через физический уровень.

Уровень сетевого взаимодействия занимается передачей данных из одной сети в другую. В качестве основного протокола сетевого уровня в стеке используется **IP**, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества объединенных локальных сетей. Поэтому протокол IP хорошо работает в сетях со сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи.

Протокол IP доставляет блоки данных, называемых **датаграммами** или **пакетами**, от одного IP-адреса к другому. IP-адрес является уникальным 32-битным идентификатором компьютера (а, точнее, его сетевого интерфейса). Данные для датаграммы передаются IP-модулю транспортным уровнем. IP-модуль предваряет эти данные заголовком, содержащим IP-адреса отправителя и получателя и другую служебную информацию, и сформированная таким образом датаграмма передается на уровень доступа к среде передачи (например, одному из физических интерфейсов) для отправки по каналу передачи данных.

В то время, как задачей сетевого уровня является передача данных между произвольными узлами сети, задача транспортного уровня заключается в передаче данных между любыми прикладными процессами, выполняющимися на любых узлах сети. Самые распространенные транспортные протоколы – это **TCP**, **UDP**, **SCTP**.

В зависимости от семантики взаимодействия (способа передачи пакетов от отправителя к получателю) различают несколько типов сокетов:

- Поточный сокет (**STREAM**). Обеспечивает надежный двухсторонний последовательный поток байтов без фиксированной длины. В домене Интернета этот сокет использует протокол TCP.

- Датаграммный сокет (**DGRAM**). Поддерживает двухсторонний поток сообщений. Приложение, использующее такие сокеты, может получать сообщения в порядке, отличном от последовательности, в которой эти сообщения посылались. В домене Интернета этот сокет использует протокол UDP.

- Сокет последовательных пакетов (**SEQPACKET**). Обеспечивает двухсторонний последовательный надежный обмен датаграммами фиксированной максимальной длины. Для этого типа сокета не существует специального протокола.

- Простой сокет (**RAW**). Обеспечивает доступ к основным протоколам связи.

TCP – это транспортный механизм, предоставляющий поток данных с предварительной установкой соединения, и, за счет этого, дающий уверенность в достоверности получаемых данных. Он осуществляет повторный запрос данных в случае их потери и устраняет дублирование при получении двух копий одного пакета. Реализация TCP, как правило, встроена в ядро ОС.

Пакеты, поступающие на транспортный уровень, организуются операционной системой в виде множества очередей к точкам входа различных прикладных процессов. В терминологии TCP/IP такие системные очереди называются **портами**.

Протоколы прикладного уровня обеспечивают взаимодействие сети и пользователя. Этот уровень позволяет приложениям пользователя иметь доступ к сетевым службам – таким, как обработчик запросов к базам данных, доступ к файлам, пересылка электронной почты. Также он отвечает за передачу служебной информации, предоставляет приложениям информацию об ошибках и формирует запросы к уровню представления.

В обычной клиент-серверной модели процесс либо ожидает входящих данных («слушает порт»), либо посылает данные на известный открытый порт. В первом случае процесс выполняет роль **сервера**, во втором – **клиента**.

Порты нумеруются начиная с нуля. По умолчанию приложению выдается порт с произвольным номером – например, ближайшим свободным. При необходимости приложение может запросить конкретный номер порта. Так, у многих протоколов прикладного уровня определены стандартные порты, используемые серверами по умолчанию:

- **HTTP** – TCP-порт 80
- **SMTP** – TCP-порт 25
- **FTP** – TCP-порт 21
- **DNS** – UDP-порт 53

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста. Он был разработан в 1992 г. в CERN инженером Тимом Бернерсом-Ли, который стоял у истоков Всемирной паутины. Работа по этому протоколу происходит следующим образом: программа-клиент устанавливает TCP-соединение с сервером и выдает ему HTTP-запрос. Сервер обрабатывает этот запрос и выдает HTTP-ответ клиенту.

Интерфейс между пользователем и сетевым приложением называется агентом пользователя (**User Agent**). Для WWW роль агента пользователя играет HTTP-клиент – веб-браузер.

Мы напишем HTTP-сервер – программу, которая «понимает» HTTP-запросы и отвечает на них.

В стандартной библиотеке D – Phobos – есть простой кроссплатформенный интерфейс к сокетам в модуле std.socket.

```
import std.stdio;
import std.socket;
```

При создании сокета, как правило, определяют три параметра: область связи, тип взаимодействия и транспортный протокол:

```
auto listener = new Socket(AddressFamily.INET,
                           SocketType.STREAM,
                           ProtocolType.IP);
```

В std.socket уже предусмотрен стандартный TCP-сокет, так что проще будет написать так:

```
auto listener = new TcpSocket;
```

Привязываем сокет к порту 80 и «слушаем»:

```
listener.bind(new InternetAddress(80));
listener.listen(10);
```

Нам необходим цикл, в котором будут обрабатываться входящие сообщения:

```
Socket currSock;
uint bytesRead;
char[1000] buf;

while(true)
{
    currSock = listener.accept();
    if ((bytesRead = currSock.receive(buf)) > 0)
    {
        writeln("Client message:");
        writeln(buf[0..bytesRead]);
    }
    currSock.close();
    buf.clear();
}
```

«Костяк» сервера готов – программа принимает запросы от клиента и просто выводит их в консоль. Попробуйте запустить ее и ввести в адресную строку браузера «localhost/» без кавычек.

Сервер должен вывести что-нибудь вроде

```
Client message: GET / HTTP/1.1 Host: localhost
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.9.0.3) Gecko/2008101315 Linux Mint/6 (Felicia)
Firefox/3.0.3 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8 Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate Accept-Charset:
ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300
Connection: keep-alive
```

В следующей части статьи мы рассмотрим функцию, которая будет «расшифровывать» такие запросы и правильно отвечать на них.

Gecko
gecko0307@gmail.com

Шейдеры на все случаи жизни

Эффект типографской печати на GLSL

Один из наиболее любопытных методов нефотореалистичного рендеринга (NPR) – имитация типографской печати. А, точнее, передача тона круглыми растровыми точками, которые используются в полиграфии. Эти точки, сливаясь на расстоянии, создают ощущение цветовых/тональных переходов.

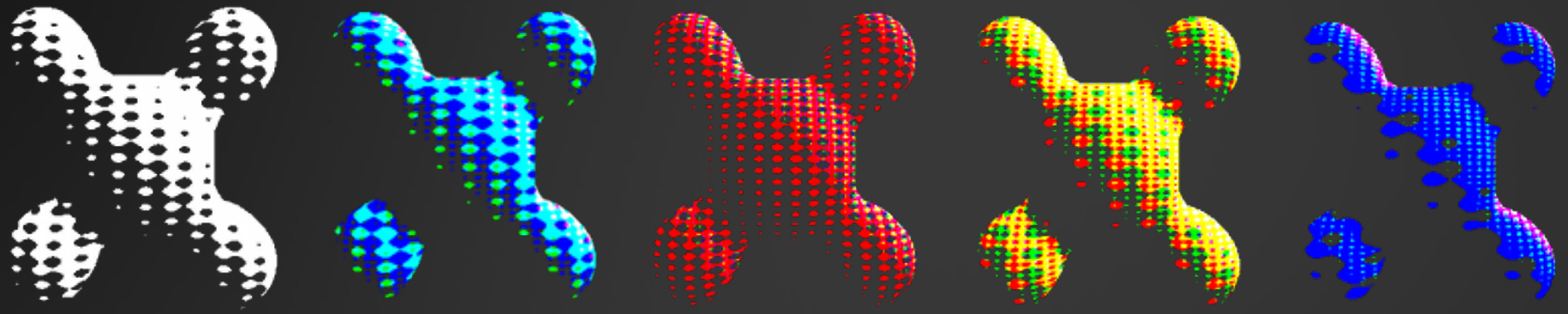
Приведенная реализация позаимствована из коллекции шейдеров на **http://code.google.com/p/glslang-library/**.

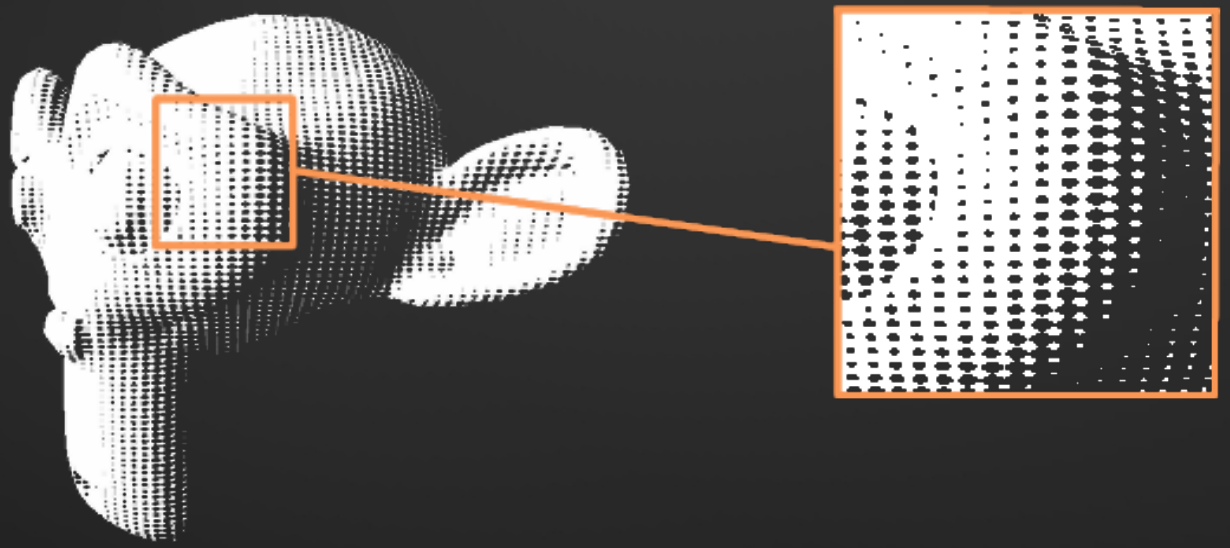
Вершинная программа:

```
varying vec3 N;
varying vec3 P;
varying vec4 S;
varying vec3 L;

void main()
{
  gl_Position = ftransform();
  N = normalize(gl_NormalMatrix*gl_Normal);
  P = gl_Vertex.xyz;
  L = vec3(
      vec4(gl_LightSource[0].position,1)-gl_Position
      );
  S = gl_ProjectionMatrix*gl_Position;
}
```

Фрагментная программа:

```
varying vec3 N;
varying vec3 P;
varying vec4 S;
varying vec3 L;
const vec3 Scale = vec3(0.3, 0.3, 0.3);
const vec3 Offset = vec3(0.5, 0.5, 0.5);
const vec3 Register = vec3(0.0, 0.0, 0.0);
const vec3 Size = vec3(100.0, 100.0, 100.0);

void main()
{
  vec3 l = normalize(L);
  vec3 n = normalize(N);
  vec2 p = S.xy;
  vec2 sr = p*Size.r+Register.r;
  vec2 sg = p*Size.g+Register.g;
  vec2 sb = p*Size.b+Register.b;
  float diffuse = dot(l,n);
  float r = (sin(sr.x)+cos(sr.y))*Scale.r+Offset.r+diffuse;
  float g = (sin(sg.x)+cos(sg.y))*Scale.g+Offset.g+diffuse;
  float b = (sin(sb.x)+cos(sb.y))*Scale.b+Offset.b+diffuse;
  gl_FragColor=vec4(step(0.5,r),step(0.5,g),step(0.5,b),1);
  gl_FragColor.a = 1.0;
}
```





© Полная SOPA!

В октябре 2011 года в США был предложен законопроект по противодействию онлайн-пиратству SOPA (Stop Online Piracy Act), который наделяет владельцев интеллектуальной собственности правом судебным путем требовать блокирования доступа американских граждан к веб-сайтам, связанным с нарушением авторских прав и распространением нелицензионной продукции. Также можно будет останавливать электронные платежи в пользу сайтов, попавших под подозрение. Причем, неважно, где эти сайты заре- гистрированы – в Соединенных Штатах или за их границами. Блокировать доступ к сайтам было предложено посредством системы DNS, а исполнение этой задачи должно было быть возложено... на интернет-провайдеров.

Стоит пояснить: сейчас, согласно закону «Об авторском праве в цифровую эпоху» (Digital Millennium Copyright Act – DMCA), правообладатель, обнаруживший в интернете принадлежащее ему произведение, должен сначала обратиться к администраторам сайта с просьбой удалить контент. И только в случае отказа имеет право обратиться в суд. А ответственность за размещение пиратской продукции несет пользователь, это сделавший. Теперь же весь груз ляжет на плечи владельцев сайтов. И груз этот неподъемный. У одного только Facebook пользователей более 800 миллионов человек – попробуй, успеди за всеми! Получается, что из-за одного человека доступ в целому ресурсу может быть заблокирован...

Против DNS-блокирования категорически возразили крупнейшие интернет-компании – Google, AOL, eBay, Facebook, Google, Twitter и другие. Противники SOPA заявили, что законопроект фактически означает установление цензуры в интернете, что не только противоречит демократии, но и крайне негативно отразится на развитии интернета и ИТ-отрасли в целом. Глава антивирусного разработчика «Лаборатории Касперского» Евгений Касперский также выступил с жесткой критикой SOPA, а его компания в знак протеста против принятия закона покинула ассоциацию правообладателей BSA.

Продвижение закона вызвало целую волну протеста в Сети. Для привлечения внимания к проблеме, 18 января 2012 г. Wikipedia в течение суток ограничила доступ пользователей к англоязычному сегменту ресурса. В акции протеста также приняли участие Mozilla, Reddit, Wordpress, Cheezburger Network, Identi.ca, XDA-developers и openSUSE, которые также временно прекратили свою работу. 18 января стало своеобразным «днем борьбы» с SOPA...

В результате конгресс США принял решение отложить законопроект до достижения консенсуса между его сторонниками и противниками. После всех проведенных дискуссий, авторы SOPA согласились убрать из текста спорные моменты. Ожидается, что измененный вариант SOPA будет готов к рассмотрению только в следующем году.

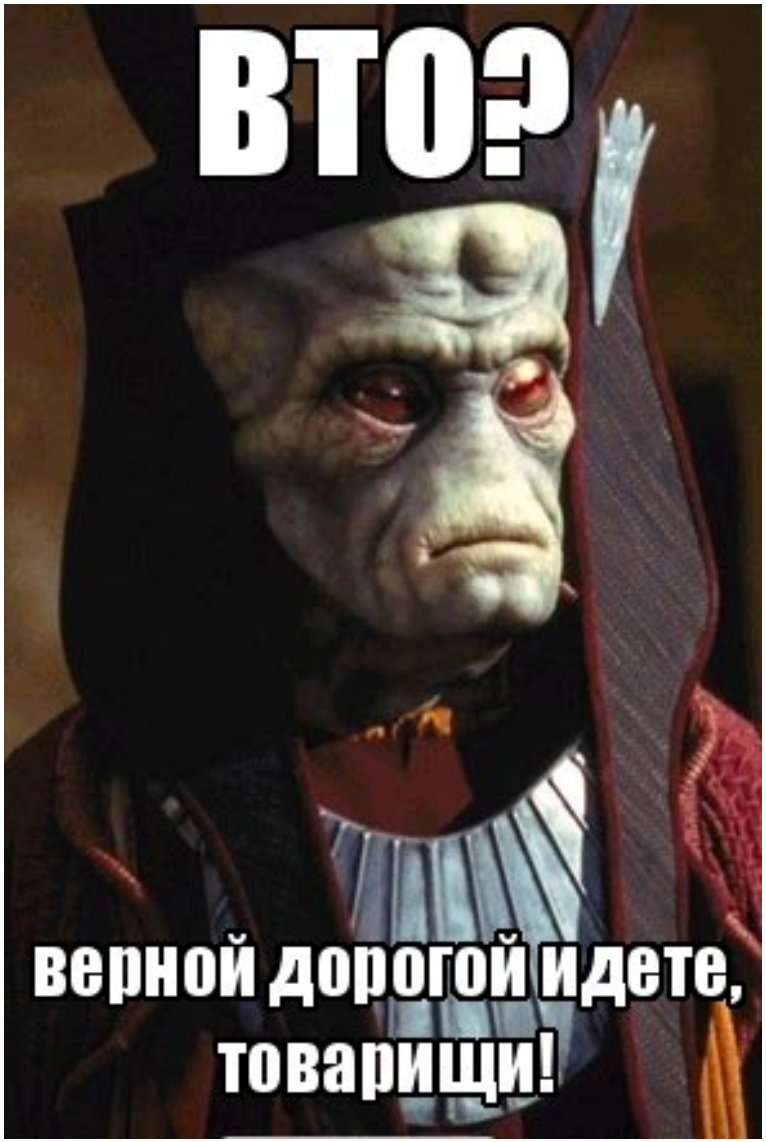
Конечно, возникает закономерный вопрос: как американский закон может коснуться сайтов, зарегистрированных за пределами США? Напрямую – не может. Но в списке главных «пиратских» сайтов американского правительства много русскоязычных ресурсов – почти все они зарегистрированы не в России, принимают платежи с карточек и размещают рекламу через поисковые ресурсы. И именно по ним может быть нанесен первый удар...

Несмотря на явную победу противников SOPA, в настоящее время компаниями-правообладателями осуществляются попытки менее гласного продвижения другого похожего закона PIPA (Protect IP Act), нацеленного на предоставление допол- нительных инструментов для борьбы с распространением нарушающего авторские права контента на сайтах, не подпадающих под юрисдикцию США – путем блокирования финансовых операций с такими сайтами, запрещения установки ссылок на них и блокирования доступа на уровне DNS. В данном случае правообладатели могут добиться того, что, например, «ВКонтакте» будет недоступен на территории Америки...

А тем временем в Токио завершилась первая стадия процедуры принятия международного соглашения «Торговое соглашение по борьбе с контрафакцией» (Anti-Counterfeiting Trade Agreement, ACTA), которое было подписано представителями 22 стран Европейского союза и ратифицировано Еврокомиссией (для окончательного принятия соглашения, оно должно быть ратифицировано 27 участниками Евросоюза).

Соглашение также намерены подписать США, Канада, Япония, Южная Корея и Австралия.

На данный момент опубликовано несколько проектов соглашения ACTA, но юридический анализ его финального соглашения до сих пор недоступен, поэтому пока вся критика базируется на последних черновых вариантах. В частности, отмечается возможное усиление контроля со стороны провайдеров, связанное с требованием к созданию технической системы для идентификация любого абонента – что фактически полностью деанонимизирует пользователей Интернета. «Особое внимание» будет уделено пользователям децентрализованных сетей, таких как BitTorrent, для обязательной идентификации которых «должны быть приняты все надлежащие меры». Дополнительно упоминается право... проводить досмотр носителей информации при пересечении границ и блокировании доступа абонентов к сети после многократных уличений в нелегальном использовании контента.



На фоне шумного противостояния против PIPA/SOPA, первая стадия принятия ACTA прошла на удивление быстро и спокойно – хотя и было отмечено проведение отдельных молодежных акций протеста в Европе. Представители Фонда свободного ПО (FSF) считают, что вполне реально сообща добиться отклонения соглашения Европейским парламентом, если проявить гражданскую активность и участвовать в проходящих в сети акциях протеста.

Кстати, следует обратить внимание на обстоятельство, которое касается граждан России. ACTA является документом, дополняющим действующее соглашение TRIPS («Соглашение по торговым аспектам прав интеллектуальной собственности»), соблюдение которого является обязательным для всех стран-членов ВТО. Как известно, Россия официально вступает в ВТО уже в текущем 2012 году. Последние принимаемые поправки говорят о начале активной работы России по синхронизации своего законодательства с нормами TRIPS, что неизбежно приведет к серьезному ужесточению ответственности за нарушения в области авторских прав...

Так что в России вполне может появиться аналог SOPA. Права граждан на частное использование произведений ограничивают со всех сторон. Принятие «антипиратских поправок» активно лоббируют Российская антипиратская организация (РАПО), Гильдия продюсеров России (ГПР) и Российский союз правообладателей (РСП), который возглавляет Никита Михалков...

Гражданин РФ имеет право на частное воспроизведение и копирование (п. 1 ст. 1273 и 1275 ГК РФ), но ГПР пролоббировала две поправки к Гражданскому кодексу: запрет на частное копирование при публичном показе и запрет делать частные копии с использованием специальной аппаратуры... А РСП продвинул так называемый «налог Михалкова» – осенью 2010 года вступило в силу новое постановление правительства «О вознаграждении за свободное воспроизведение фонограмм и аудиовизуальных произведений в личных целях» (ст. 1245 ГК РФ), согласно которому 1 % от стоимости каждого чистого компакт-диска, мобильного телефона, видеокамеры, ноутбука или винчестера перечисляется на счет РСП. Объем рынка оборудования и носителей оценивается в $ 10-15 млрд. в год, поэтому общая сумма ежегодных сборов за частное копирование составлает порядка $ 150-200 млн. 15 процентов от этих денег пойдут кому-то в карман...

Логично предположить, что новое бремя, в конечном счете, ляжет на потребителя, который будет вынужден покупать технику и диски немного дороже. Получается, правительство фактически вынуждает граждан переплачивать за новый плеер или ноутбук только за то, что они потенциально могут быть использованы для копирования защищенных авторским правом музыки и фильмов!

Поборники авторских прав также пытаются противостоять файлообменным сетям. К счастью, у провайдеров на данный момент нет законного основания отслеживать содержание файлов, которыми обмениваются пользователи, а тем более – ограничивать их передачу. Тайна связи гарантируется Конституцией РФ...

Неоднозначна в нашей стране и судьба свободных лицензий. Начнем с того, что досрочная передача произведения в общественное достояние (Public Domain) по текущему законодательству... вообще невозможна. А работающие за рубежом свободные лицензии оказались не вполне совместимы с российским гражданским правом. Их использование создает высокие риски. Так, в России автор может отозвать лицензию Creative Commons. Это обрушит всю цепочку производных работ – созданных другими людьми модификаций произведения. Возможность отзыва заставляет перед использованием произведения связываться с правообладателем: не передумал ли он? А это противоречит идее простого и легкого использования.

ГПР и РАПО так вообще распространили заявление... об опасности свободных лицензий, потому что «пользователи таких лицензий будут иметь возможность избегать ответственности за нарушение исключительных авторских прав, прикрываясь статусом добросовестного приобретателя»...

Юридически лицензии Creative Commons ничем не отличаются от обычных лицензионных соглашений, которые предусмотрены ст. 1286 Гражданского кодекса, хотя до недавнего времени часто можно было слышать, что свободные лицензии являются совершенно особенными договорами или не являются договорами вовсе.

Поправка к ст. 1233 ГК (п. 6), предложенная в 2011 году, вроде бы ставит все на свои места. Смысл ее состоит в том, что наконец-то правообладателю вроде как стало можно одним воле- изъявлением публично обозначить условия использования своего авторского или смежного права для всех – то есть, неограниченного круга лиц. Собственно, ровно это и делает международно признанная Creative Commons, но Россия как обычно идет своим путем...

Во-первых, вызывает подозрение необходимость указывать срок, в течение которого предоставляется возможность безвозмездно использовать произведение. При отсутствии в заявлении правообладателя указания на срок считается, что он составляет 5 лет. В течение срока действия заявление не может быть отозвано, а предусмотренные в нем условия использования не могут быть изменены. Непонятно, что должно произойти по истечении этого срока, и каким образом он может быть продлен?

Кроме того, для того, чтобы «волеизъявиться», правообладателю нужно это сделать не абы где, а... «путем размещения на официальном сайте федерального органа исполнительной власти по интеллектуальной собственности». Этот орган называется Роспатент. Как утверждается, в этом случае можно будет установить лицо, которое сделает такое заявление – его будет легко привлечь к ответственности в случае недобросовестных действий. Однако этот пункт вызывает недоумение: что именно регистрировать в Роспатенте? И как?

Формально, никакой угрозы свободным лицензиям здесь вроде бы нет, потому что они, с правовой точки зрения, как были принимаемы в виде оферты, так и продолжают. А лицензионный договор о предоставлении права использования произведения не подлежит государственной регистрации, как и само произведение. Но желание некоторых слишком ретивых правоохранителей толковать это как обязательное требование наличия регистрации может возникнуть...

Например, если GPL-программа не будет зарегистрирована на сайте Роспатента, тогда любой может ее модифицировать, скомпилировать и продавать, не открывая код. И к этому «товарищу» нельзя будет предъявить претензии в нарушении GPL, так как нет записи в Роспатенте, что данная программа защищена лицензией GPL. Могут возникнуть и проблемы у пользователей любой GPL-программы. Ведь возникает логичный вопрос – как заставить регистрироваться иностранных разработчиков? Большинство популярного СПО создается многими авторами, разбросанными по всему миру – зарегистрировать такие программы в Роспатенте практически невозможно!

А если у вас нет на руках лицензионного договора, то лицензия на программу должна быть на сайте Роспатента, в противном случае нельзя будет подтвердить легальность программы – пользователей будут считать априори пиратами. Не придется ли доказывать, что «вы не верблюд» и ваш браузер Firefox не является ворованным, если Mozilla не побежит в Роспатент регистрировать лицензию? Или придет прокурор в организацию и скажет, что вы пользуетесь нелицензионным офисным пакетом LibreOffice, потому что нет соответствующей записи правообладателя? Мнение о том, что проверяющие хорошо знают, что такое Linux и GPL, является слабой гарантией того, что компьютеры не будут изъяты для дополнительной проверки. Отсутствие привычно оформленного лицензионного договора – весомый аргумент для контролирующих органов, чтобы вывезти оборудование на экспертизу...

Дело в том, что единственный способ присоединить лицензионный договор к программному продукту описан в п. 3 ст. 1286 ГК РФ: «Заключение лицензионных договоров о предоставлении права использования программы для ЭВМ или базы данных допускается путем заключения каждым пользователем с соответствующим правообладателем договора присоединения, условия которого изложены на приобретаемом экземпляре таких программы или базы данных либо на упаковке этого экземпляра...» Эта статья не дает возможности для легализации ПО, скачанного из Интернета и предоставляемого по лицензии GPL, так как, во-первых, она говорит о «приобретении» экземпляров, чего не происходит при получении СПО через сайт, а, во-вторых, согласно статье, экземпляр произведения – это копия в любой материальной форме, поэтому текст лицензии в электронном виде этому условию не удовлетворяет.

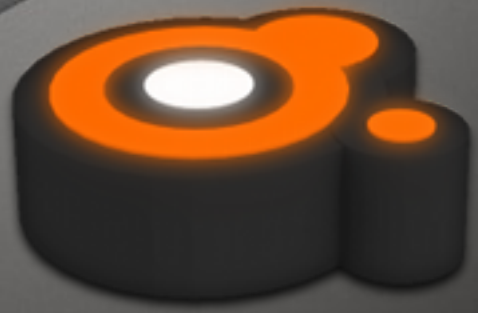
Статья требует, чтобы условия договора были нанесены на материальный носитель или на его упаковку. Говорят, если программы получены из Интернета, можно показать проверяющим сайт, откуда они были загружены. Правда, это самый ненадежный способ подтверждения «легальности»: для полиции все, что распространяется через Сеть, «пиратское» по умолчанию...

Все отлично понимают, что регистрировать свободные программные продукты мало кто будет. Регистрация вроде необязательна, однако ее вводят не случайно, а с целью ограничить легальное использование свободного софта – в этом и состоит главная проблема...

Gecko
gecko0307@gmail.com

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