Независимый электронный журнал о разработке компьютерных игр

№16 • 2011

FPS

Язык D
Двоичное дерево
Сопрограммы

SIGGRAPH 2011
+ 3DWorld CG Awards

Blender
Симуляция огня

Цифровая алхимия
В начале было Слово...

Python 2 или Python 3
Что выбрать?

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

portalgame.net.ru
Игровой портал

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

game.oxnull.net
Разработка игр на конструкторе Game Maker

bigslava.3dn.ru
Сообщество творческих людей

game-creation.tk
Портал создания игр

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

№ 16
'11

В этом выпуске:

MAKE IT HOME
SIGGRAPH2011

В АВГУСТЕ этого года в Ванкувере (Канада) прошла самая известная и самая престижная конференция и выставка компьютерной графики SIGGRAPH 2011. Аббревиатура SIGGRAPH расшифровывается как Special Interest Group on Graphics and Interactive Techniques («Специальная Группа по Вопросам Графических и Интерактивных Технологиий»). Выставка проводится организацией ACM SIGGRAPH с 1974 года в Лос-Анджелесе, Новом Орлеане, Сиэтле, Далласе, Бостоне и других городах США. Ежегодно на нее приезжают тысячи людей со всего мира, а в последние годы к ним прибавилась многомиллионная аудитория зрителей в Интернете.

На SIGGRAPH съезжаются представители крупных компаний, демонстрируя новейшие разработки в области компьютерной графики и анонсируя новые версии выпускаемых ими продуктов. Кроме того, на SIGGRAPH приезжают специалисты из крупных университетов, которые представляют статьи на темы, имеющие отношение к CG. Наконец, это место, где встречаются и проводят мастер-классы лучшие 3D-художники. В рамках SIGGRAPH также проводится фестиваль анимации и еще множество других мероприятий.

По традиции на SIGGRAPH 2011 была представлена новая версия спецификации OpenGL – 4.2. Она включает обновление GLGL до 4.20, а также содержит новые функции, расширяющие доступную функциональность для разработчиков и увеличивающие производительность приложений.

Рабочая станция HP Z800 на выставке одновременно обрабатывала 12 потоков Full HD видео. Это стало возможным благодаря объединению сил компаний Fusion-io, NVidia, Thinkbox Software и Tweak Software, создавшими творческий тандем специально для SIGGRAPH 2011. Разработчики оснастили HP Z800 видеоускорителем NVidia Quadro Plex7000, а также твердотельными накопителями ioDrive Duos. В системе были установлены также модули Fusion ioMemory. Результат предсказуем — рабочая станция без малейших «тормозов» или потерь качества действительно обрабатывала одновременно 12 потоков несжатого FullHD видео, да еще с цветокоррекцией. Твердотельные накопители были объединены в RAID массив с высокой производительностью. ioDrive Duos показала отличный результат – три гигабайта в секунду!

Интересно, что система, подобная этой, многими рассматривается как достойный инструмент для графических дизайнеров – особенно для тех, кто работает с 3DS Max, Maya, Softimage и другими ресурсоемкими приложениями. Представители Thinkbox Software, создавшие набор модулей Krakatoa 2.0 MX для 3ds Max, тоже неплохо отозвались о системе.

Side Effects Software, Inc. объявили, что DreamWorks приобрели неограниченную лицензию на пакет Houdini. Это является продолжением практики использования Houdini студией DreamWorks для создания визуальных эффектов и персонажей в своих анимационных фильмах. Частью соглашения является принятие участи Dreamworks в разработке продукта, что позволит упростить обмен интеллектуальной собственностью между компаниями и согласовать усилия в исследованиях. Это сотрудничество нацелено на привнесение передовых и проверенных в производстве технологий в будущие релизы Houdini.

«Мы гордимся, что DreamWorks Animation выбрали наш продукт как приоритетный инструмент для анимации», – сказал Ким Дэвидсон, президент и исполнительный директор SES. – Это захватывающий момент расширения наших партнерских отношений. Мы надеемся работать бок о бок еще долгие годы.»

Работая совместно, обе компании хотят найти новые пути развития в искусстве и науке создания фильмов. Основанная на узлах процедурная архитектура Houdini позволит DreamWorks внедрить свои технологии в конвеер, чтобы соответствовать растущим с каждым годом запросам анимационных и VFX-проектов.

На SIGGRAPH 2011 был анонсирован выход Houdini 11.1, который станет переходной версией к Houdini 12. Обещают, что особенностями Houdini 12 станут скорость и производительность. Достичь этих целей помогут новый движок для работы с геометрией, а также задействование последних возможностей OpenGL. Будут улучшены модули для просчета жидкостей и имитации ткани, представлена вторая версия модуля Pyro FX 2.0 для создания дыма и огня. По заявлениям SES, Houdini 12 будет самым важнейшим релизом пакета, когда-либо выпущенным.

За последние двадцать лет топовые студии использовали награжденный американской Киноакадемией пакет Houdini для создания визуальных эффектов – как в кино, так и в анимационных фильмах. Houdini наиболее известен своим процедурным рабочим процессом, который поддерживает как творческие, так и технические аспекты компьютерной графики.

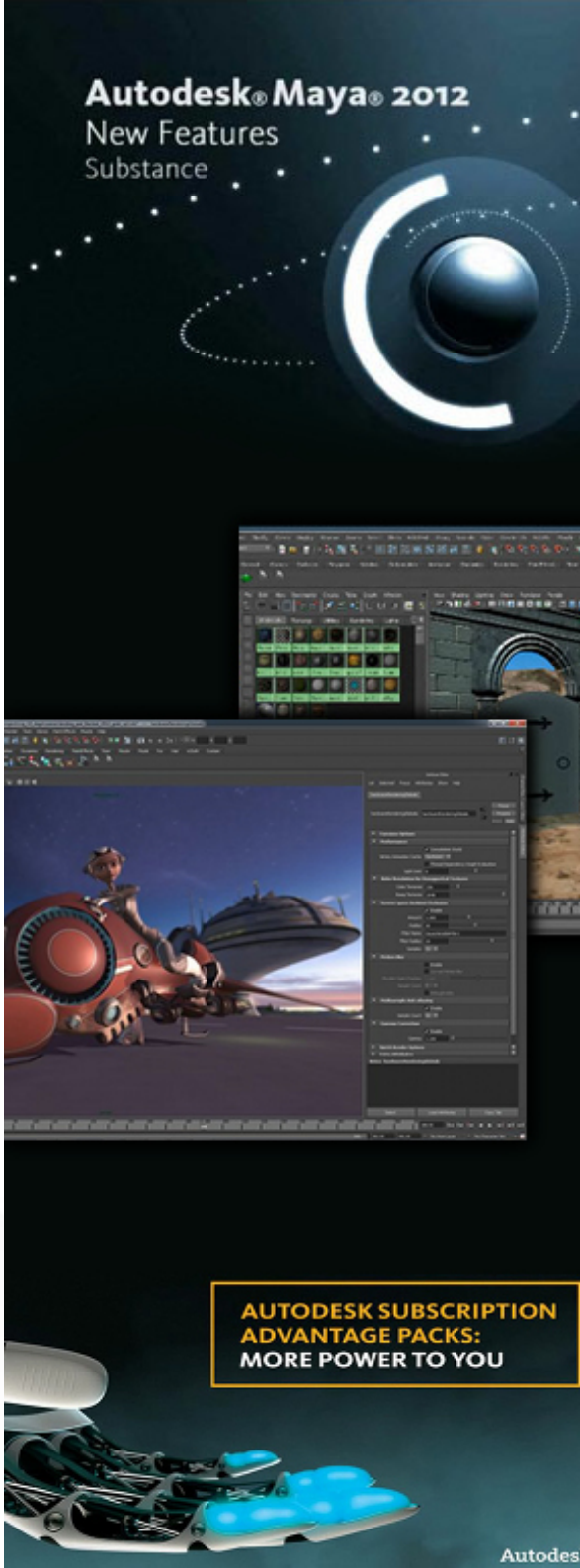
Компания Maxon представила 13-ю версию своего 3D-редактора Cinema 4D. Основные особенности релиза – новая физическая камера, многопроходный OpenEXR, встроенное 3D-стерео, улучшенная работа с внешними ссылками Xrefs.

Компания Smith Micro Software объявила о предстоящем релизе программ Poser 9 и Poser Pro 2012. В новой версии добавлен просчет подповерхностного рассеивания, улучшена производительность визуализации, появилась поддержка Weight Map Rigging. Также добавлен набор инструментов для редактирования Vertex Weight Map. Также стал возможен предварительный просмотр сцен в режиме реального времени.

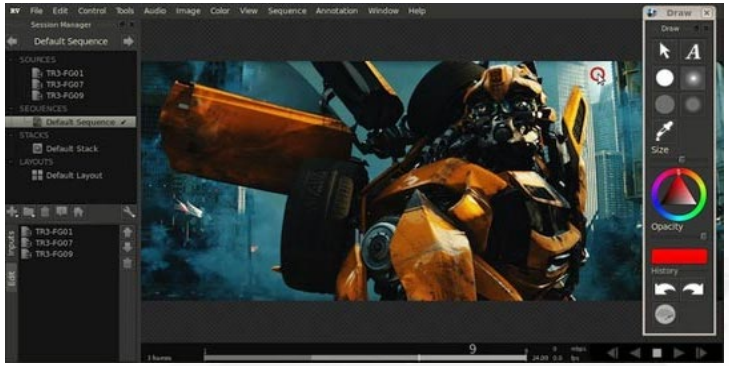
На выставке также представлена новая версия бенчмарка SPECapc (SPEC Application Performance Characterization) для Autodesk Maya 2012. Тест производительности для последней версии 3D-редактора планируется выпустить до конца года. А в следующем году будет обновлено приложение для оценивания производительности графической подсистемы SPECviewperf. Двенадцатая версия SPECviewperf будет поддерживать оба API – OpenGL и Direct3D, а также многие интерактивные приложения. Его можно будет без труда портировать на другие операционные системы, а добавление текстовых файлов станет проще. На сайте SPECapc (http://www.spec.org/) можно найти бенчмарки для более ранних версий Maya, 3DS Max, SolidWorks и других программ.

Компания Autodesk продемонстрировала пакет дополнений Subscription Advantage Pack для Maya 2012. Основная особенность – поддержка OpenCL. Кроме того, интегрирован новый плагин Bullet Physics. Обеспечивается поддержка всех возможностей новейших графических процессоров от AMD. Для справки: SAP – это дополнение, доступное только для пользователей, которые дополнительно приобрели подписку на продукт.

Pixar представила новую версию плагина RenderMan для Maya, который является полностью интегрированным в пакет рендером. Программа отличается от других продуктов линейки Pixar (таких, как RenderMan Pro Server и RenderMan Studio) более низкой стоимостью и сравнительной простотой использования. Плагин сможет порадовать пользователей всеми современными опциями, которые присутствуют в хороших визуализаторах: возможностью просчета глобального освещения, эффекта подповерхностного рассеивания, размытия при движении и т.д. Обещают, что в RenderMan для Maya войдут все последние технологии, представленные в RenderMan Pro Server 16.0. В частности, существенно улучшена технология трассировки, добавлены новые средства для освещения и повторной визуализации, которые использовались в «Истории игрушек 3» и «Тачки 2».



Cebas Visual Technology Inc. продемонстрировала работу новой версии рендера FinalRender 4GPU. Она использует технологию NVidia CUDA с поддержкой GPU для ускорения вычислений. Напомним, что программы Cebas успешно использовались при создании таких фильмов, как «Гарри Поттер и Дары Смерти: Часть I и II», «Трансформеры», «Зеленый фонарь», «2012», «Алиса в стране чудес», «Перси Джексон и Олимпийцы», «Кошмар на улице Вязов» и «Железный человек 2».



Компания Tweak Software продемонстрировала возможности новой версии своего продукта RV – мощного вьювера изображений для VFX-художников и аниматоров. Ключевые возможности программы включают новый «презентационный» режим, позволяющий просматривать материал на одном мониторе, а управлять просмотром – с другого; поддержку стерео 3D TV для HDMI 1.4a и DVI; поддержку JavaScript, ARRI Alexa ARRIRAW и RED RAW, спецификации ACES (Academy Color Encoding Specification) и CDL.

Компания Imagination Technologies объявила о предстоящем выпуске нового релиза движка рендеринга Brazil. В новой версии Brazil был переписан с нуля для увеличения скорости. Теперь пользователи могут редактировать «на лету» все, что есть на сцене, в том числе модели, источники света, материалы и шейдеры. Brazil построен на использовании OpenRL и Imagination's Raytracing API, и может использоваться на любом оборудовании. Карты и материалы, созданные в Brazil RS 2.0, будут корректно открываться в новой версии программы.

Компания Organic Motion представила новую версию системы захвата движения OpenStage. Эта система motion capture интересна тем, что для ее работы не требуются маркеры. Система способна делать точный захват движений актера в реальном времени, и ему для этого не нужно одевать специальный костюм. Немаловажная особенность системы – быстрая настройка.

Кинофестиваль SIGGRAPH – один из немногих, которые объявляют свои призы не в финале, а до самого анимационного шоу. Поэтому победители были известны заранее.

«Лучший фильм»
«The Fantastic Flying Books of Mr. Morris Lessmore» (Уильям Джойс и Брендон Олденбург, Moonbot Studios)

«Спецприз жюри»
«Paths of Hate» (Демиен Нено, Platige Image)

«Лучший студенческий проект»
«Flamingo Pride» (Томер Эшед, The Konrad Wolf Potsdam-Babelsberg Film and Television University)

«Специальное поощрение жюри»
«Escape of the Gingerbread Man» (Тод Польсон, The Monk Studios)
«The Experience of Fliehkraft» (Тилл Новак, FrameboX)

«Специальные поощрения жюри в категории «студенческий проект»
«Dreamgiver» (Тайлер Картер, Brigham Young University)
«Sweater Dog» (Джина Моффит, Ringling College of Art and Design)

По завершении тридцать восьмого SIGGRAPH, оргкомитет подвел итоги. Согласно официальным данным, выставку посетило около 16 тысяч зарегистрированных участников, среди которых ученые, программисты, художники и мультипликаторы. Это рекорд за все время проведения SIGGRAPH в Ванкувере. На выставе и конференции были гости из 74 стран мира. Свою продукцию представили более 155 компаний. Среди них Intel, NVidia, AMD, Autodesk, Google, HP, Pixologic, Khronos Group, Pixar Animation Studios, Cebas, MAXON Computer, NewTek, Next Limit Technologies и другие.

Следующая выставка пройдет в Лос-Анджелесе с 5 по 9 августа 2012 года. Сайт SIGGRAPH 2012 уже работает, «заглянуть в будущее» можно тут: **s2012.siggraph.org**.

Gecko
gecko0307@gmail.com

*За предоставленную информацию благодарим портал **3domen.com***

Награды года в области CG
от журнала 3D World

3D World, один из ведущих журналов в мире 3D-графики, в сентябре вручил целую серию наград за лучшие достижения 2010-11 гг. в области разработки программного и аппаратного обеспечения, а также за творческие успехи в области иллюстративной графики, анимации, спецэффектов и архитектурной визуализации. Выбор жюри во многом оказался довольно любопытным. В частности, Blender Foundation достались сразу две награды – технологическая и творческая...

«Прикладная программа года»

Лучшей новой программой был признан новый пакет виртуальной лепки Sculptris. Изначально это была индивидуальная разработка программиста Томаса Петерсона, которой он занимался в индивидуальном порядке с декабря 2009 г. Впоследствии Петерсон получил работу в компании Pixologic, которая разрабатывает аналогичную по назначению программу ZBrush. В результате между Sculptris и ZBrush была налажена определенная доля совместимости – пректы из первого можно быстро экспортировать во второй.

http://www.pixologic.com/sculptris

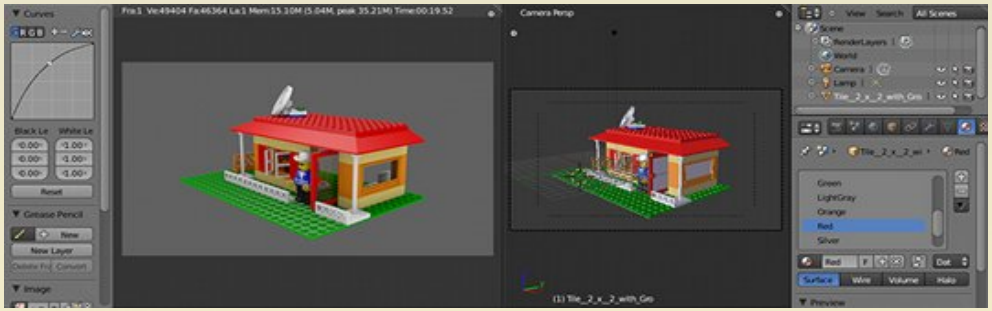
Мы польщены тем, что Sculptris занял первое место в номинации. Спасибо всем проголосовавшим! Pixologic известна как инноватор в области цифрового искусства, и разработка Sculptris согласуется с нашим желанием помочь художникам творить, не ограничивая себя техническими барьерами.

Джимми ГОЛДИНГ
начальник отдела маркетинга Pixologic

«Обновление года»

Лучшим обновлением ПО была по праву признана новая версия Blender 2.5, поднявшая этот пакет на ступень выше и поставив в один ряд с профессиональными коммерческими аналогами. Blender – прекрасный пример удачного подхода к разработке программного обеспечения, истинный шедевр современного свободного ПО.

http://blender.org



Мы получили множество положительных отзывов за этот проект по пересмотру пользовательского интерфейса и инструментов пакета. Это была сложная задача, более трех лет работы – далеко не все бесплатные и открытые проекты могут похвастаться подобным! Официальное признание со стороны 3D World мы воспринимаем как жест благодарности за этот труд.

Тон РОЗЕНДАЛЬ
глава Blender Foundation

«Плагин года»

Звание лучшего плагина досталось физическому движку Lagoa Multiphysics.

http://www.vimeo.com/channels/lagoausers

Мы довольны признанием и поддержкой, которую получил Lagoa Multiphysics. Огромное спасибо всем, кто дал нам возможность внести свой вклад в индустрию!

Тьяго КОСТА
программист Lagoa Technologies

«Полнометражный фильм года»

Победителем в этой номинации стал мультфильм Гора Вербински «Ранго» (ILM/Paramount Pictures), с Джонни Деппом, озвучивающим главного героя – хамелеона по кличке Ранго.

http://www.rangomovie.com

Большое спасибо 3D World за эту награду! «Ранго» стал уникальным опытом для ILM. Этот фильм помог нам изучить все тонкости работы над крупным проектом. Я хочу поздравить всех художников студии и поблагодарить за их вклад.

Тим АЛЕКСАНДЕР
ответственный за визуальные эффекты
Industrial Light & Magic

«Короткометражный фильм года»

Эта награда досталась третьему открытому фильму Blender Foundation – «Синтел» (режиссер Колин Леви), который в очередной раз доказал, что Blender отлично подходит для серьезного кинематографа.

http://www.sintel.org

Наши открытые фильмы всегда были захватывающими экспериментами, попыткой расширить границы для Blender. В случае с «Синтел» авторы, к тому же, получили дополнительный вызов – использовать альфа-версию Blender 2.5, находившуюся в разработке. Технически фильм содержит кое-какие недоработки, но с творческой точки зрения – это весьма выдающаяся работа. Спасибо вам за награду!

Тон РОЗЕНДАЛЬ
глава Blender Foundation,
продюсер «Синтел»

«Программной инновацией года» названа платформа Genesis для пакета DAZ Studio 4. Моделирование человекоподобных фигур – достаточно трудное занятие, и Genesis представляет собой очередное средство экономии усилий во время этого процесса.

«Аппаратной инновацией года» признаны процессоры Intel Core i7, столь полюбившиеся всем «трехмерщикам».

«Лучший игровой трейлер года» – дебютный ролик к Tomb Raider: Turning Point. Трехминутное видео более года создавалось на студии Visual Works, расположенной в Токио и принадлежайщей Square Enix.

Ну, и напоследок: **«Лучшая игровая графика года»** – Crysis 2. Кто-то удивлен?

http://www.3dworldmag.com

Тимур ГАФАРОВ
tgafaroff@gmail.com

Blender

Новости

СВОБОДНЫЙ пакет трехмерного моделирования Blender стремительно развивается и обновляется. Этим летом на SIGGRAPH 2011 глава Blender Foundation Тон Розендаль огласил планы по дальнейшему развитию программы, охватывающие технические и организационные вопросы.

Во-первых, было решено начать работу по подготовке новой версии Blender 2.6. В первую очередь в нее была внесена «перечная» ветка SVN (Pepper branch), которая включает несколько проектов Google Summer of Code 2011. Основные особенности релиза:

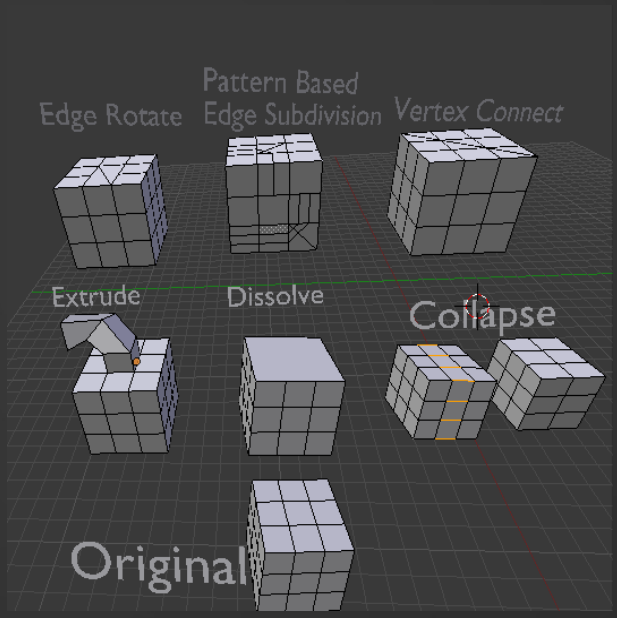
- Поддержка трехмерного звука;
- Интернационализация (поддержка нелатинских алфавитов);
- Улучшенная поддержка формата COLLADA;
- Библиотека поиска пути Recast & Detour для BGE;
- Множество улучшений в редакторе узлов, в системе анимации, инструментах развесовки и т.д.

Вторым знаковым проектом является переписывание системы композитинга, за которое взялся Йерун Баккер. Использующий OpenCL прототип был продемонстрирован прошлой осенью на Blender Conference и вызвал настолько живой отклик, что спустя несколько месяцев Баккер заручился финансовой поддержкой сообщества и приступил к работе, разбив реализацию проекта на несколько этапов. На первом и втором этапах система композитинга в Blender превратилась в мозаичную. Если раньше расссчет узла выполнялся только после того, как был полностью рассчитан предыдущий, то сейчас данные передаются по мере готовности каждого тайла. Само собой, мозаичная система также заметно упрощает балансировку нагрузки. На третьем и четвертом этапах на OpenCL переписываются основные узлы. На последнем, пятом, все прочие.

Важность этого проекта оказалась так велика, что фонд Blender ускорил внедрение системы ежемесячной подписки, при которой пользователи сами выбирают, как много денег они жертвуют проекту. Поступающие в фонд средства распределяются по критически важным проектам, и новая система композитинга – один из них.

В долгосрочной перспективе также находится интеграция Bmesh и других, не менее интересных разработок, в числе которых новая система рендеринга Cycles, инструмент воксельной лепки Unlimited Clay, поддержка физического движка Bullet для оффлайн-рендеринга, новые инструменты для работы с кривыми (Nurbana) и булевыми операциями (Carve-booleans), NPR-рендеринг (Freestyle) и многое другое. Это также предусматривает внесение кода прошлых и нынешних проектов GSoC в основную ветку программы.

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 многоугольниками 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.



Проект по «глубокой» интеграции Bullet, носящий название Bullet Construction Toolkit, ставит целью упрощение использования физики при рендеринге, а также более тесное ее связывание с системой анимации. Напомним, что в текущей версии Blender физика напрямую поддерживается только в рамках игрового движка BGE.

Маркетинговая часть нововведений Blender Foundation подразумевает начало выпуска ежеквартального технического журнала «Blender Proceedings», в котором будет подробно освещаться, что происходит в проекте, какие новые функции добавлены, какие изменения внесены в API и т. д. Ожидается, что финансирование этого проекта будет осуществляться за счет платной подписки на издание.

Кроме того, был анонсирован новый открытый фильм – проект Mango. На данный момент о нем мало что сообщается; известно лишь, что это будет короткометражный фильм в жанре научной фантастики с участием реальных актеров и использованием Blender для создания спецэффектов. Ввиду сложности проекта предполагается сократить длительность фильма с обычных 10 до 5-6 минут. Режиссер картины – Ян Хьюберт, текущей работой которого является Project London – фанастический фильм, снятый с применением Blender и находящийся в настоящий момент на этапе пост-продакшн. По всей видимости, именно это стало решающим фактором при выборе режиссера для проекта Mango. Кстати, многие известные разработчики из сообщества Blender уже выразили заинтересованность в участии в проекте. Среди них – Николас Бишоп, Сергей Шарыбин, Брехт ван Ломмель, Кэмпбелл Бартон и Йерун Баккер.

Наконец, был запланирован запуск профессиональной социальной сети Blender Network – партнерской программы, главная задача которой состоит в упрощении поиска разработчиков для коммерческих разработок на основе Blender. Эта концепция была предложена еще на Blender Conference 2010. Планируется три типа аккаунтов: для учащихся, фрилансеров и профессионалов (последние два – платные, за счет этих средств будут окупаться технические затраты и поддержка). Участники сети (художники, аниматоры, программисты, консультанты, специалисты по спецэффектам) смогут выполнять оплачиваемые задания различной сложности.

«Нет дыма без огня»

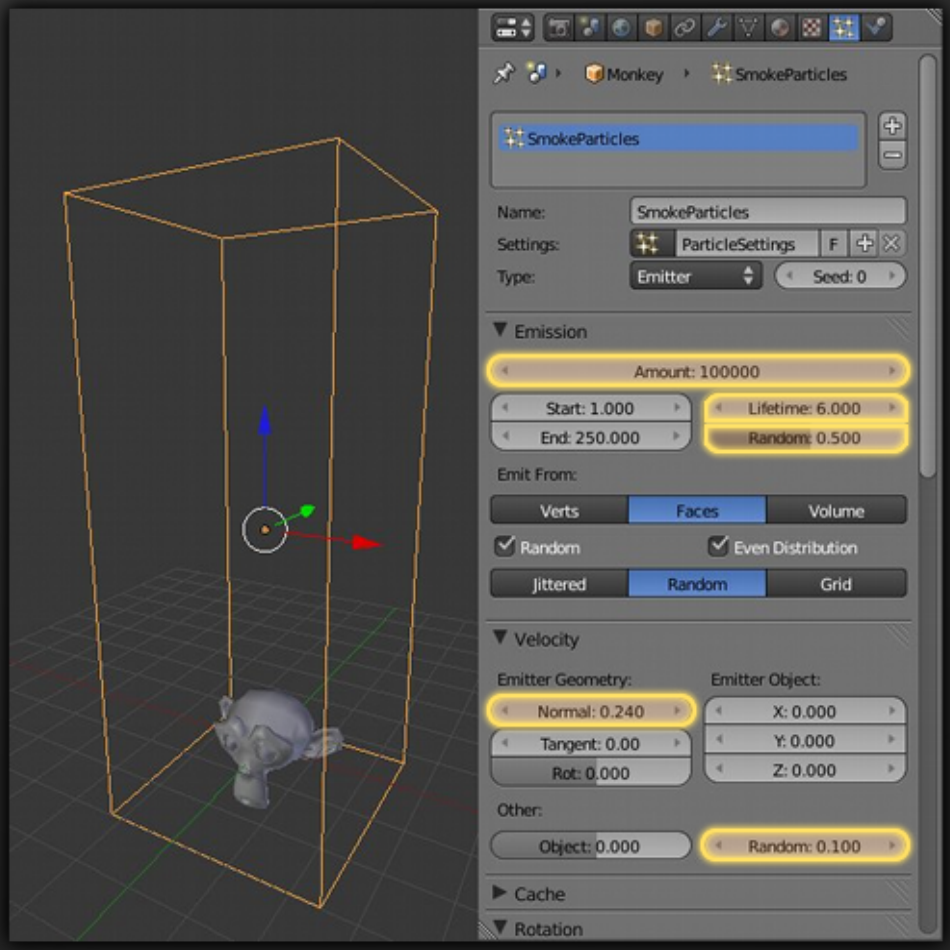
Симуляция огня в Blender

В «FPS» №10 ('10) мы рассмотрели нашумевшую новинку в Blender 2.5 – симуляцию дыма. Эту же систему можно использовать для самых разных эффектов, и не в последнюю очередь – для создания красивого реалистичного огня. Ранее огонь приходилось моделировать только при помощи обычных частиц, без учета физики – это, разумеется, сказывалось на качестве результата не самым лучшим образом...

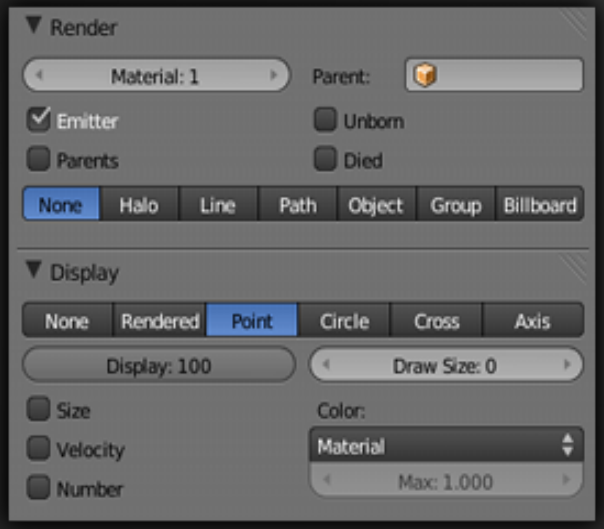
Предполагается, что у вас уже есть базовые знания в области использования симулятора дыма, так что я буду заострять внимание только на моментах, связанных с огнем.

1. В качестве источника огня лучше использовать какой-нибудь сравнительно сложный объект. На первых порах для этой цели сойдет примитив Сьюзанн. Добавьте объект и в настройках физики сделайте его источником дыма (система частиц появится автоматически). Добавьте также на сцену domain-куб.

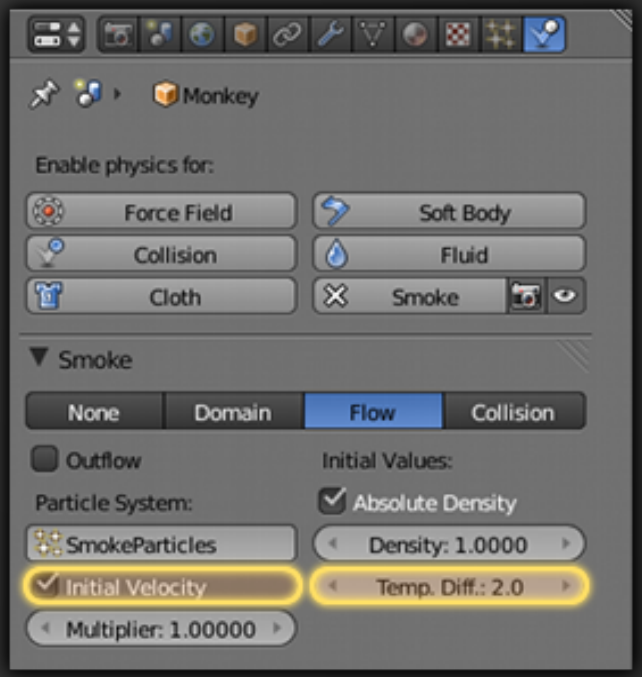
2. Для системы частиц можно использовать следующие настройки: 100000 частиц, продолжительность жизни 6 кадров. Затем настройки ускорения **Velocity**: нужно уменьшить ускорение по нормали (**Normal**) и добавить случайного (**Random**). Лучше не использовать слишком большие значения, так как симулятор дыма в этом отношении весьма чувствителен. Я выставил **Emission -> Random** на 0.5, **Velocity -> Normal** на 0.24 и **Velocity -> Random** на 0.1.



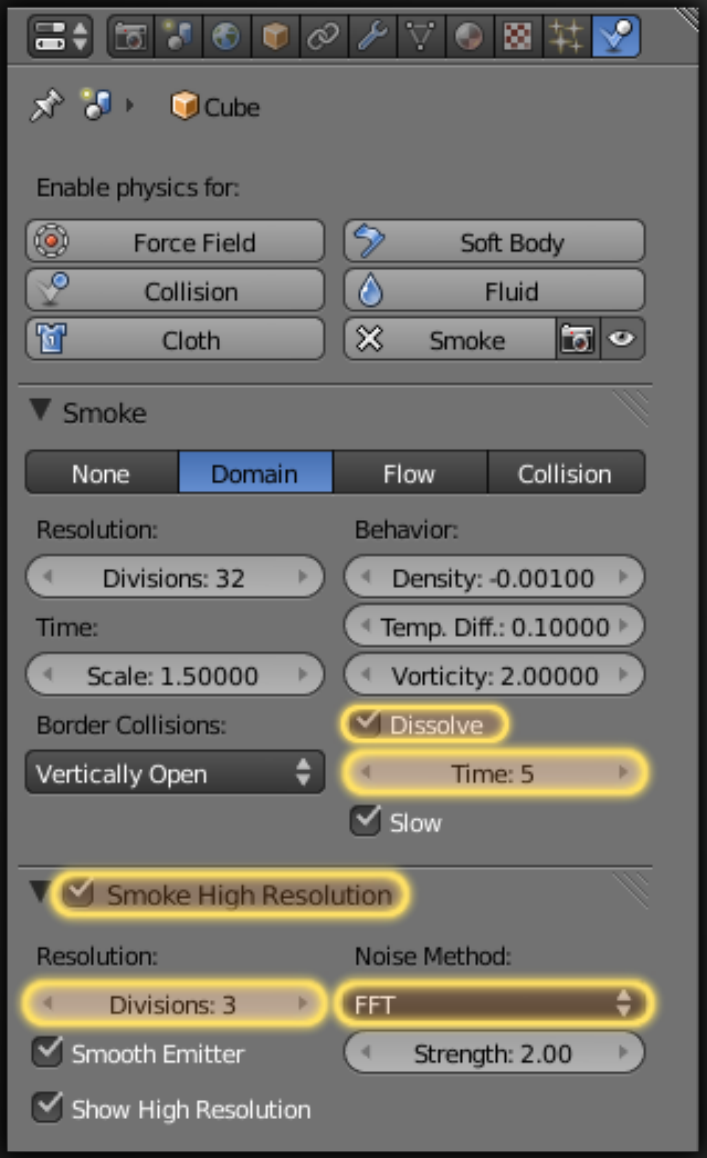
Теперь уберите гравитацию в настройках **Field Weights**, поставьте **Render** на **None** – и **Display** по своему вкусу.



3. Теперь настроим источник дыма. Чтобы получить быстро поднимающийся огонь, можно увеличить **Smoke -> Temp. Diff.** до 2.0. Его можно понизить, чтобы получить эффект замедленности движения (slow motion). Это значение также зависит от размера моделируемого пламени. Включите также **Smoke -> Initial Velocity** (наша система частиц заточена для получения начального ускорения).

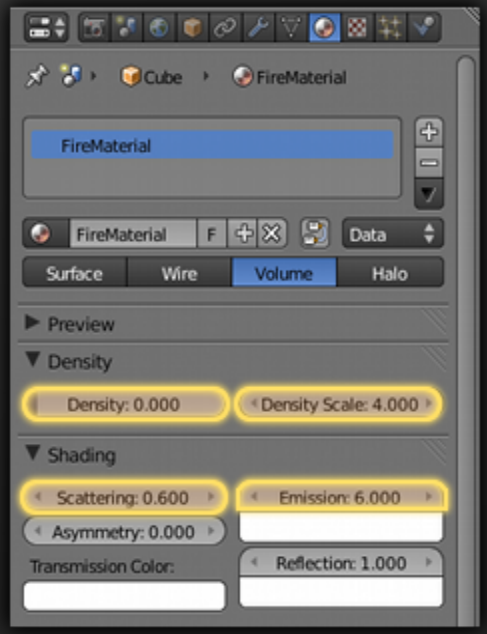


4. Теперь настройки domain-куба. Самое важное здесь – включить опцию **Smoke -> Dissolve**. Это заставит дым медленно рассеиваться, создавая очень похожий на огонь эффект. Я поставил **Smoke -> Time** на 5. Вы можете менять это значение основываясь на том, насколько мал источник огня и насколько высокое нужно пламя. Разрешение симуляции (**Smoke High Resolution -> Resolution**) остается за вами. Для теста, например, сойдет небольшое значение – 3 или 4. Обратите внимание, что разрешение серьезно влияет на внешний вид эффекта. Поэтому постоянное использование высокого разрешения – не лучший вариант. Высокое разрешение создает много маленьких язычков пламени, что порой выглядит не реалистично. Поэтому уменьшайте или увеличивайте разрешение в зависимости от размера огня.

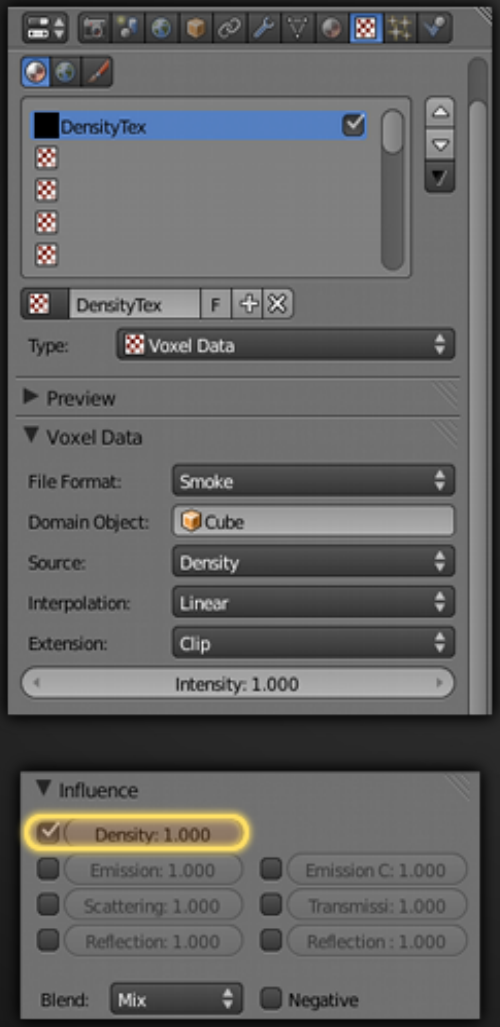


Не менее важно для создания огня переключить метод симуляции на **FFT**. В некоторых сборках FFT почему-то нет (в последней используемой мною версии – 2.60 rc1 – она есть). На рисунке вы можете видеть сравнение между методами **Wavelet** и **FFT**. Во многих случаях они почти идентичны – но FFT немного резче и поэтому лучше для огня.

5. Самое интересное – подобрать правильный материал для огня. Хитрость в том, чтобы использовать значение плотности дыма (density) в качестве значения свечения (emission). Дым исчезает настолько быстро, что не успевает потерять форму – в такой ситуации он очень похож на пламя.

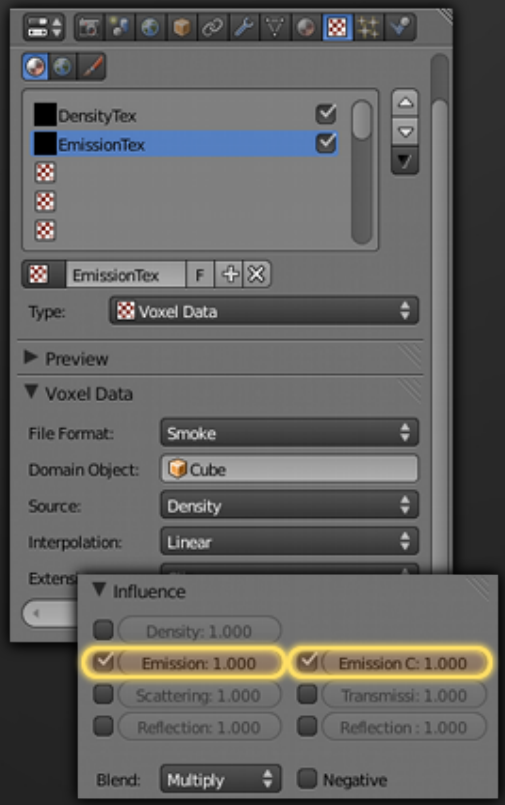


В настройках материала domain-куба я использовал достаточно небольшие значения **Density -> Density Scale** и **Shading -> Scattering** – 4 и 0.6 соответственно. **Density -> Density** выставьте на 0. **Shading -> Emission** установите между 2 и 10, в зависимости от того, насколько яркий огонь вы хотите.

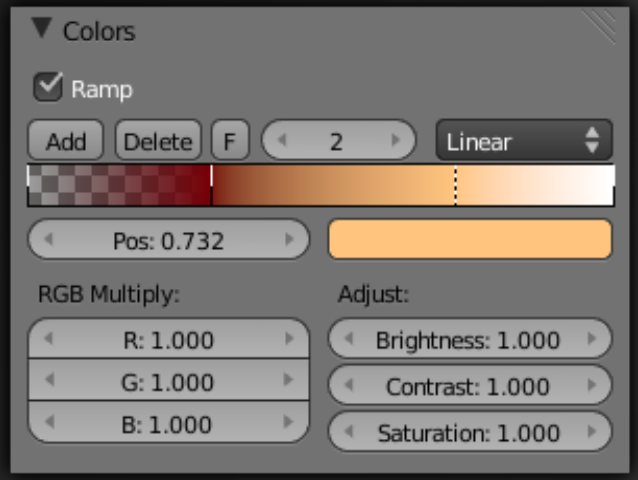


6. Настроим текстуру огня. Добавьте такую же текстуру, какую используют для создания дыма. Она будет влиять только на плотность.

Затем добавьте еще одну текстуру, которая будет контролировать яркость и цвет свечения. Тип текстуры тот же (**Voxel Data** с объектом дыма Cube). На вкладке **Influence** включите только **Emission** и **Emission Color**. Поменяйте **Blend** на **Multiply** – так мы можем уменьшить свечение там, где плотность низкая, и увеличить – где высокая. Таким образом, яркость огня основывается на плотности.

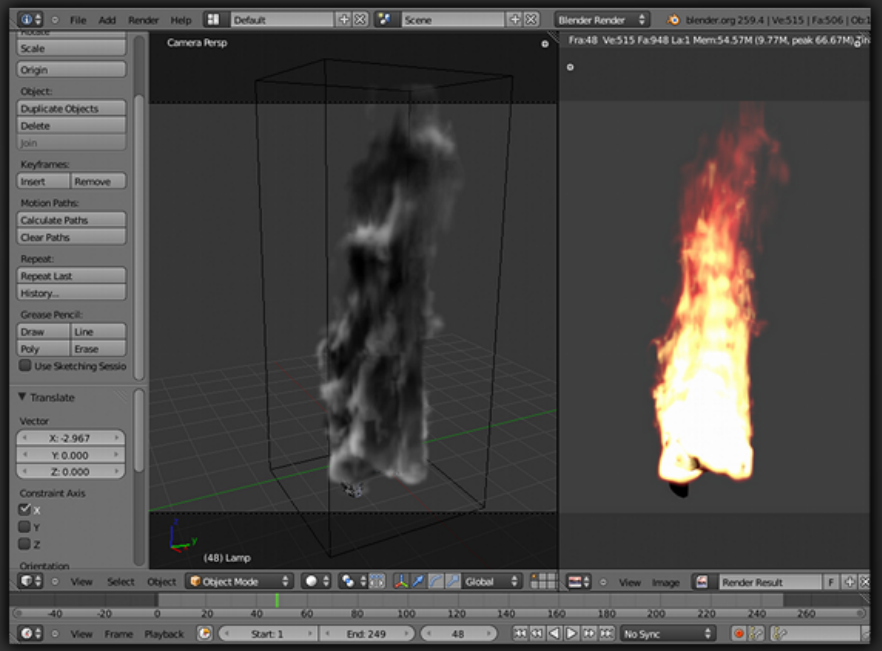


Теперь самое важное. На вкладке **Colors** включите **Ramp**, чтобы использовать градиент для контроля яркости и цвета огня. Выставьте цвета примерно как на скриншоте.



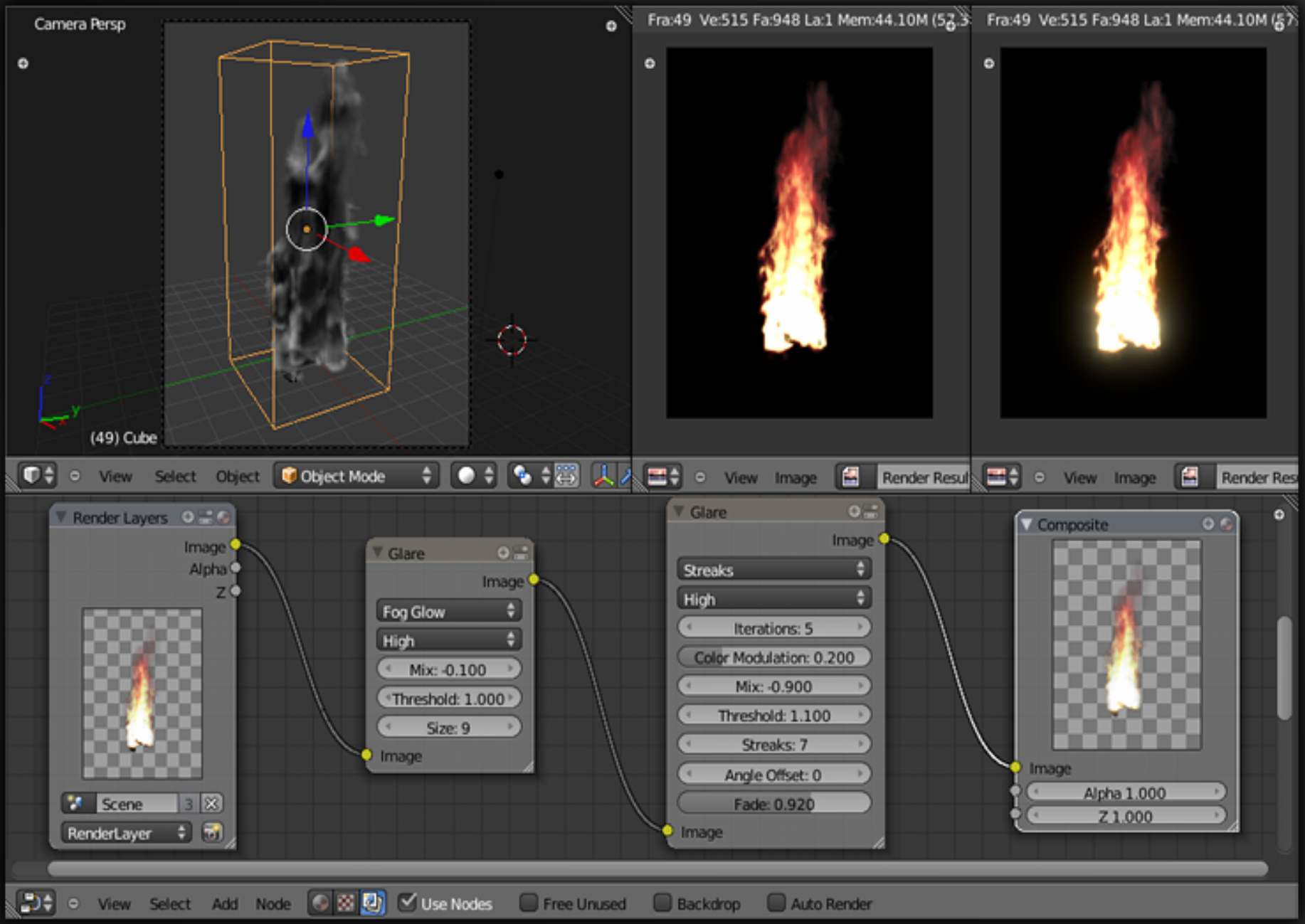
Левая сторона градиента равна нулю плотности, а правая – единице. Там, где плотность высокая, огонь будет яркий и «горячий», а там, где низкая – уже «холоднее». Вы можете передвинуть левую сторону градиента вправо, чтобы добавить дыма. Если вы остановитесь на середине, то все с плотностью ниже 0.5 будет рендериться как обычный дым, без какого-либо свечения.

7. Включите воспроизведение анимации и дождитесь, пока дым поднимется повыше. После этого можно остановить воспроизведение и отрендерить кадр.



8. Для пущей красоты можно добавить пост-обработку. В редакторе узлов включите композитные узлы (**Compositing Nodes**).

Эффектом пост-обработки, который мы добавим, будет простое свечение, которое сделает наш огонь ярче. Я также использовал два узла блеска (**Glare**), чтобы создать сбалансированный, но приятный эффект сияния.


Camera Persp
(49) Cube
Fra:49 Ve:515 Fa:948 La:1 Mem:44.10M
View Select Object Object Mode
View Image Render Resul
View Image Render Res
Render Layers
Image
Alpha
Z
Scene
RenderLayer
Glare
Image
Fog Glow
High
Mix: -0.100
Threshold: 1.000
Size: 9
Image
Glare
Image
Streaks
High
Iterations: 5
Color Modulation: 0.200
Mix: -0.900
Threshold: 1.100
Streaks: 7
Angle Offset: 0
Fade: 0.920
Image
Composite
Image
Alpha 1.000
Z 1.000
View Select Add Node
Use Nodes
Free Unused
Backdrop
Auto Render

Кстати: если вы переключите тип предпросмотра материала на «Куб», то увидите маленькое изображение того, что творится внутри domain. Это может быть полезно при настройке градиента на цветовой рампе. Одно плохо – его нельзя увеличить.

Справа – законченный результат работы. Желаю вам творческих успехов!

Gecko
gecko0307@gmail.com

Оригинал урока:
www.miikahweb.com

Grease Scatter

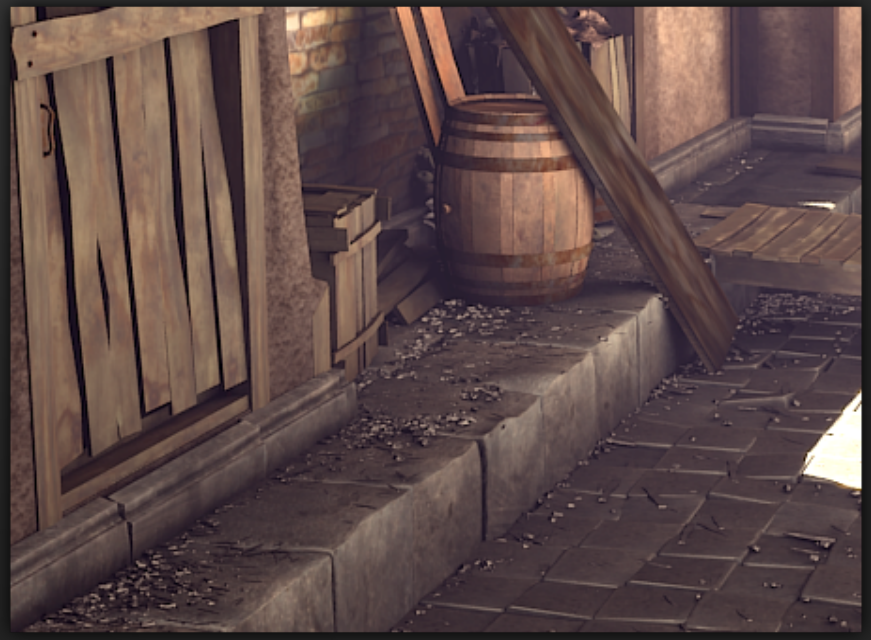
В Blender есть один малоизвестный, но любопытный инструмент – **Grease Pencil** («Восковый карандаш»). Он позволяет свободно рисовать прямо на трехмерной сцене произвольные пометки: стрелки, аннотации, схемы, подсказки и т. д.

Эта возможность дает определенные преимущества для коллективного взаимодействия и планирования – как и в случае с традиционной графикой и анимацией, где для быстрой передачи идей используются черновые эскизы. Очень часто бывает нужно сделать пометки прямо во время работы, а не в отдельном месте (в другой части окна или даже вообще в другом приложении).

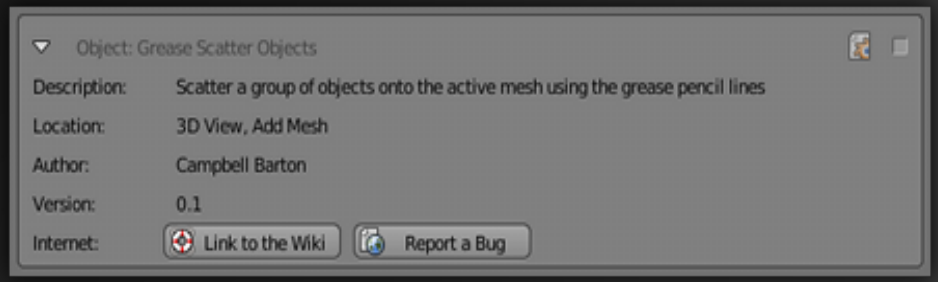
Название «Grease Pencil» было взято в дань уважения восковым мелкам, которыми первые CG-аниматоры пользовались для рисования плановых пометок на своих CRT. Восковый карандаш также может пригодиться как инструмент руководителя для замечаний и правки, а также в качестве инструмента преподавателей – как своеобразная «классная доска». Наброски, сделанные Восковым карандашом, могут быть преобразованы в кривые и даже в трехмерную модель.

Оригинальное применение Восковый карандаш получил в недавно вошедшем в состав Blender дополнении **Grease Scatter**.

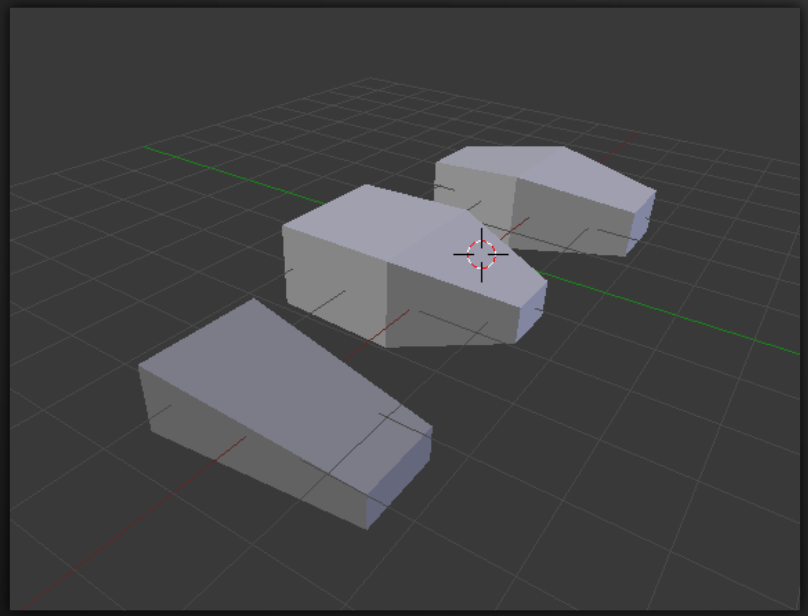
Оно изначально было разработано для работы над фильмом «Синтел». С его помощью можно моделировать разбросанные мелкие предметы – камешки, осколки, зерно, стружку, конфетти и т.д. Такие детали могут значительно повысить реалистичность сцен, особенно тех, которые воспроизводят открытые пространства – ведь редко встретишь такое место на улице, где не было бы всякого мелкого мусора. Это дополнение использует штрихи Grease Pencil для управления «разбрасыванием».



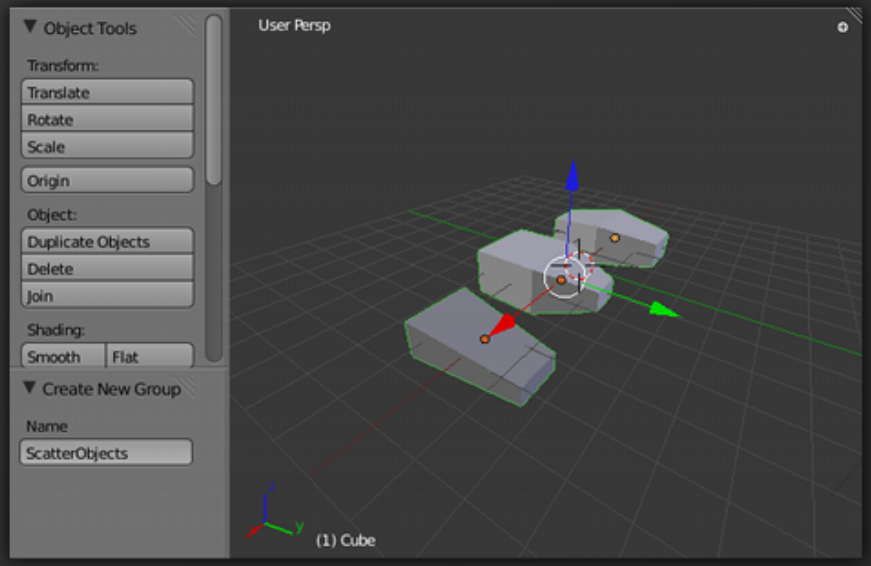
1. Активируйте дополнение:
User Preferences -> Addons -> Object -> Grease Scatter Objects.



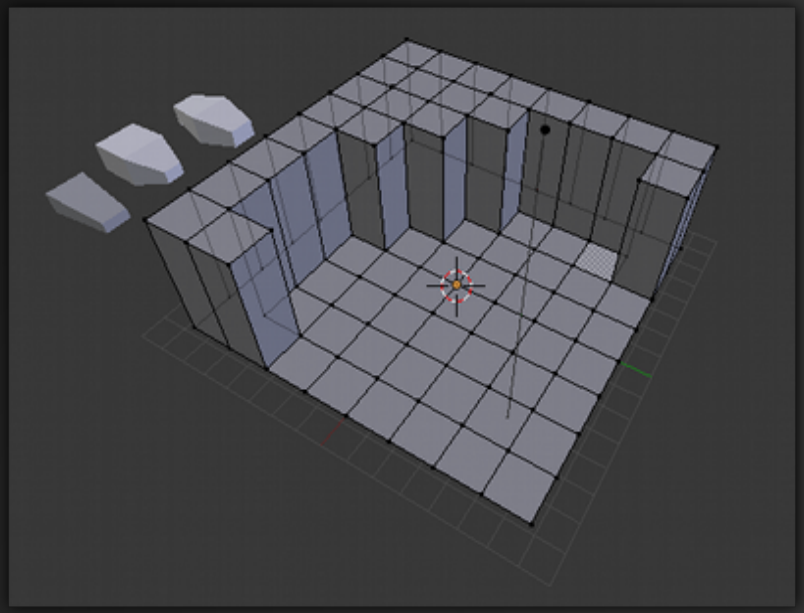
2. Создайте несколько простых по форме объектов, которые будут служить частицами нашего будущего «мусора».



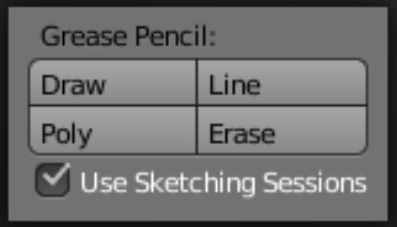
3. Выделите их и нажмите **Ctrl+G**, чтобы сгруппировать. Объекты при этом будут обведены зеленым контуром. На панели инструментов введите имя для группы – это важно, так как Grease Scatter принимает его в качестве параметра.



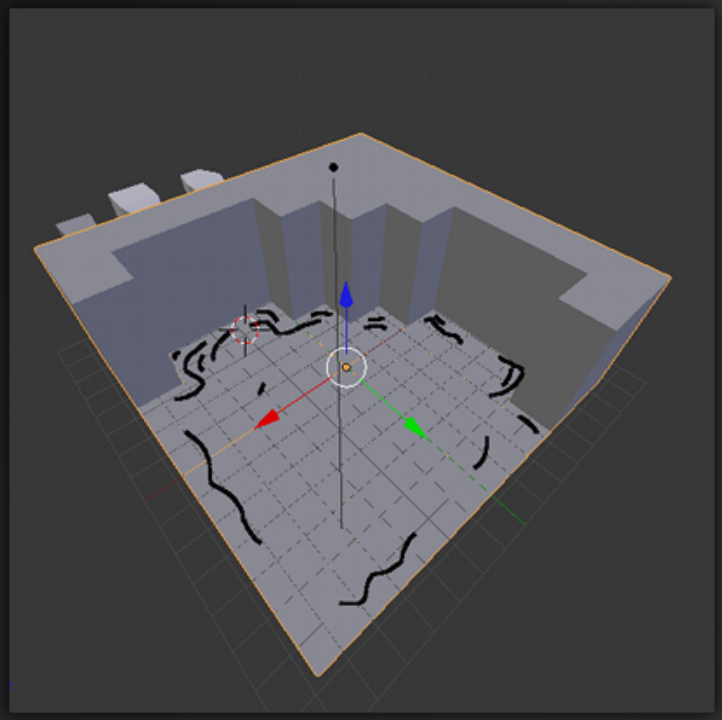
4. Теперь нужна какая-то основа, на которой будут разбросаны объекты. Отодвиньте их в сторону и создайте что-нибудь наподобие фрагмента стены со сложной конфигурацией. Назовем ее условно «архитектурой». Масштаб пока соблюдать необязательно.



5. Нажмите **N** в окне 3D-вида, чтобы вызвать панель свойств. На вкладке **Grease Pencil** создайте новый слой рисования (кнопка **New Layer**). **Drawing Settings** переключите на **Surface**, чтобы штрихи ложились на поверхность меша. На панели инстурментов (под указателем **«Grease Pencil»**) нажмите **Draw** и активируйте режим **Use Sketching Sessions**, чтобы наносить несколько штрихов подряд.

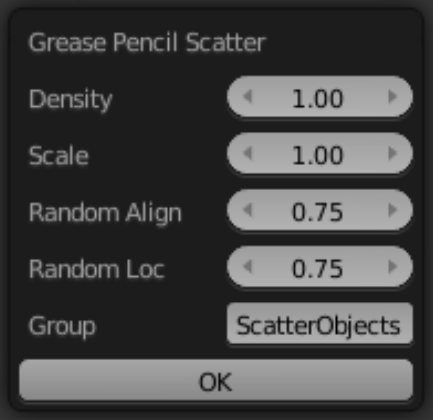


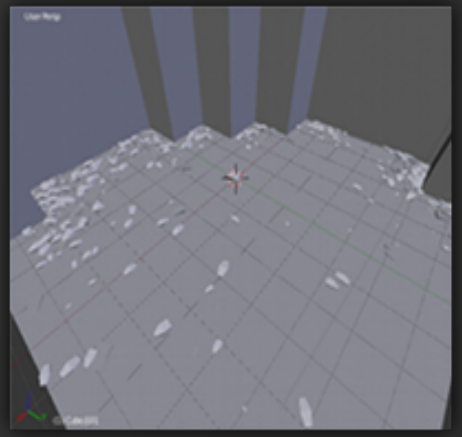
6. Не снимая выделение с «архитектуры», нарисуйте в окне 3D-вида штрихи, которые должны обозначать места разбрасывания. Это, как правило, углы и участки вдоль стен – именно там обычно скапливается мусор. При этом вы можете, как обычно, поворачивать и перемещать точку обзора при помощи мыши. Обратите внимание, что штрихи Воскового карандаша будут автоматически проецироваться на поверхность объекта.



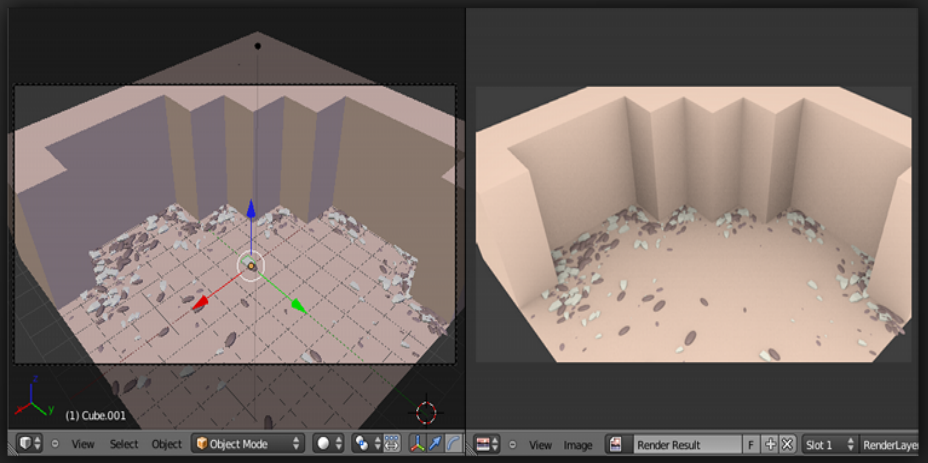
7. Теперь (опять-таки не снимая выделение) нажмите **Ctrl+A** и выберите **Mesh -> Grease Pencil Scatter**. Введите имя группы в поле **Group** и нажмите **OK**.

Теперь можно уменьшить объекты «мусора» до нужного размера – все изменения объектов-источников коснутся их виртуальных копий. Выделять их удобно из менеджера объектов (**Outliner**).





8. Можно изменить материалы объектов-источников, добавить им модификаторы сглаживания и т.д.



Gecko
gecko0307@gmail.com

Cycles. Первые шаги

Многие пользователи Blender наверняка уже знают, что в последнее время активно ведется разработка Cycles – долгосрочного проекта по созданию новой системы рендеринга на замену устаревающему Blender Internal. Основной «козырь» Cycles: поддержка аппаратного ускорения на GPU, которое в идеале позволяет рендерить практически в реальном времени, с минимальными задержками между итерациями при редактировании сцены. По признанию автора Cycles Брехта ван Ломмеля, который ради этого проекта вернулся на постоянную работу в Blender Foundation, новый рендер является промежуточным эволюционным звеном между безкомпромиссным трассировщиком и полностью программируемым движком в стиле RenderMan. В частности, в Cycles есть физически корректная система шейдинга, полностью основанная на узлах.

В настоящее время Cycles использует NVidia CUDA и OpenCL. Архитектура нового движка подразумевает возможность одновременного использования разных устройств – от GPU до сетевых рендер-ферм. Ожидается, что работа по интеграции Cycles в Blender будет вестись до конца этого года, после чего Брехт переключится сначала на оптимизацию скорости работы, а затем на реализацию прочей запланированной функциональности. «Побочным эффектом» этого проекта является реорганизация API рендеринга, которая позволит еще теснее интегрировать внешние подключаемые движки – такие, как LuxRender, Aqsis, Mitsuba и YafaRay.

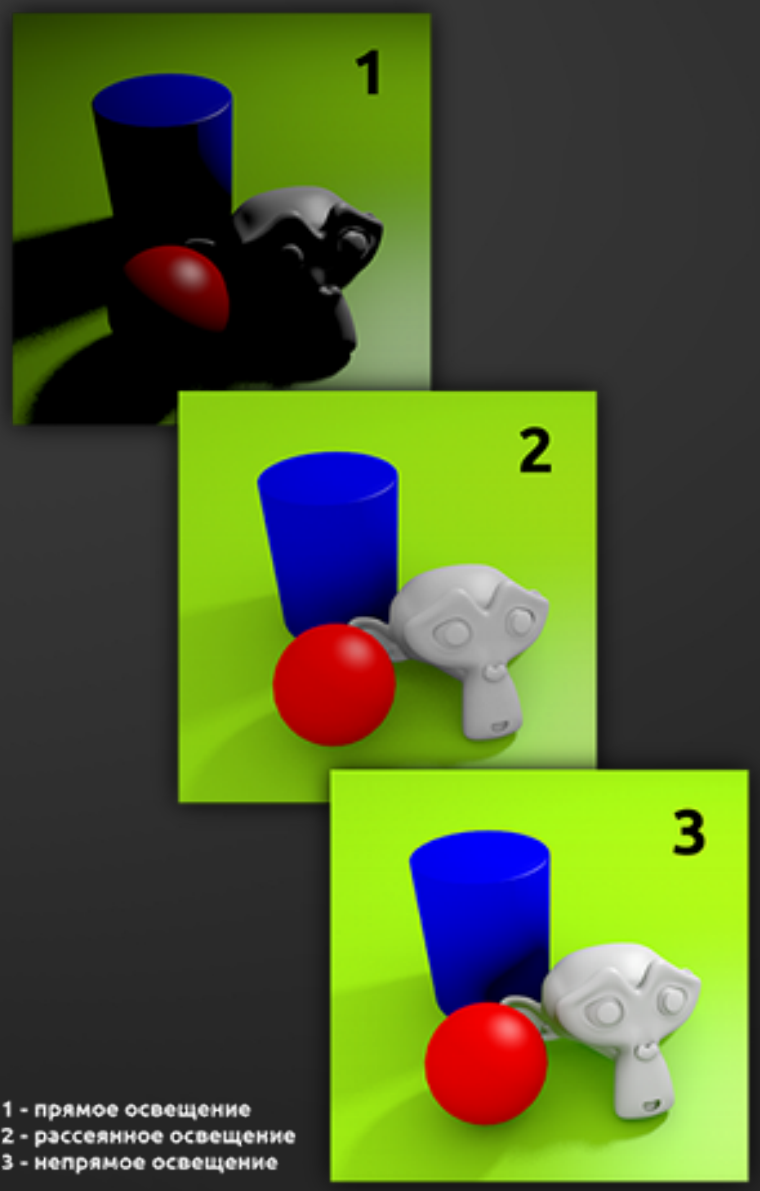
На данный момент Cycles нестабилен и находится на стадии ранней альфа-версии. Однако попробовать его в действии может любой желающий, скачав соответствующую сборку Blender с замечательного сервиса **GraphicAll.org**. Я, к примеру, скачал Blender 2.59.3 r40422 для Linux/i386 (**http://graphicall.org/645**).

Для запуска сборки на моей системе мне пришлось снабдить программу более свежей версией glibc, а также написать такой shell-скрипт:

```
#!/bin/sh
BF_DIST_BIN=`dirname "$0"`
BF_PROGRAM="blender"
LD_LIBRARY_PATH=${BF_DIST_BIN}/lib:${LD_LIBRARY_PATH}

PYTHONPATH=${BF_DIST_BIN}/2.59/python
PYTHONHOME=${BF_DIST_BIN}/2.59/python
BLENDER_SYSTEM_SCRIPTS=${BF_DIST_BIN}/2.59/scripts

export LD_LIBRARY_PATH PYTHONPATH PYTHONHOME
export BLENDER_SYSTEM_SCRIPTS

"$BF_DIST_BIN/lib/ld-linux.so.2" "$BF_DIST_BIN/$BF_PROGRAM"
```

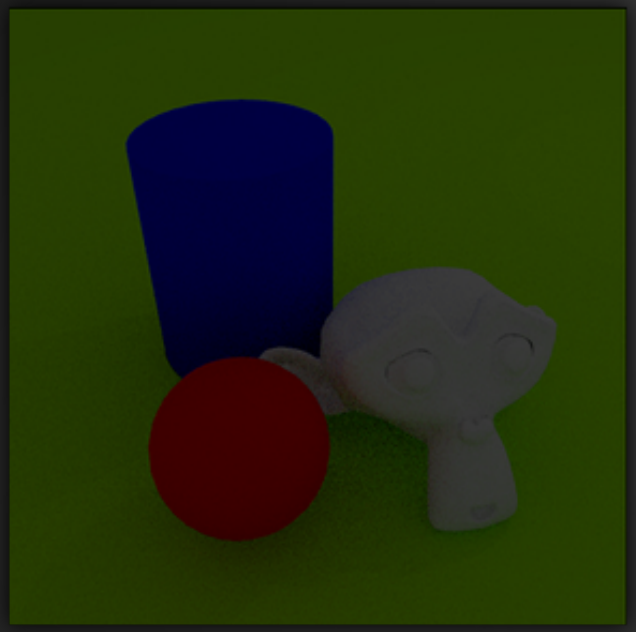
Этот скрипт можно запускать по символьной ссылке или кнопкой запуска с рабочего стола GNOME.

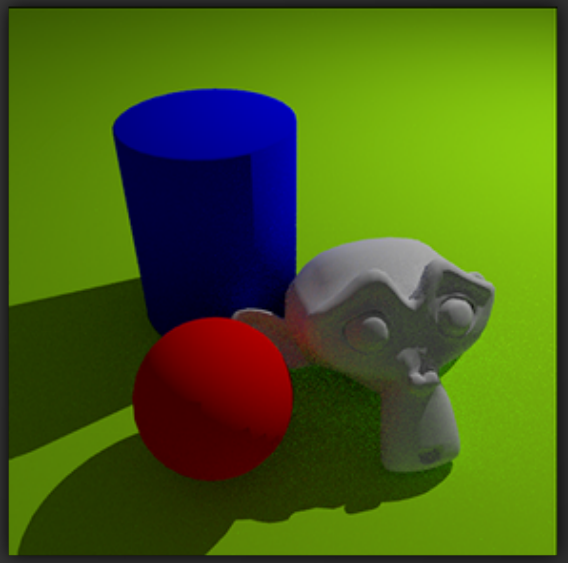
В этой статье мы рассмотрим основные принципы Cycles: глобальное освещение и BSDF. Для тех, кто не в курсе: глобальное освещение (global illumination, GI) – это алгоритмы, которые используются при рендеринге для более реалистичной имитации света. Они учитывают не только прямой свет от источника (direct illumination), но и отраженный свет от различных поверхностей (indirect illumination). В Blender Internal есть несколько «фокусов» для имитации GI (в частности, environment lighting), но они далеки от совершенства. В Blender 2.4x был, правда, движок radiocity, позволявший запекать фотонное проецирование в текстуру, однако он был довольно плохо интегрирован в существующую систему рендеринга – после любых изменений сцены приходилось рассчитывать GI заново, а это требовало дополнительных телодвижений перед тем, как нажать заветную клавишу F12. Любителям фотореализма оставалось использовать YafaRay, LuxRender и другие внешние трассировщики – но и там были свои «подводные камни». Блендеру был нужен встроенный движок GI – такой, как, например, mental ray в 3ds Max. В этой ситуации появление Cycles оказалось как нельзя кстати...

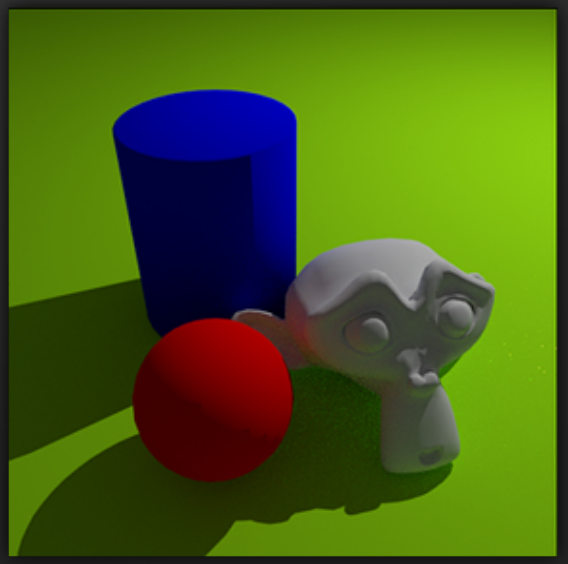


1 - прямое освещение
2 - рассеянное освещение
3 - непрямое освещение

С первого раза Cycles немного непривычен и, что называется, «неинтуитивен». Настройки рендеринга по умолчанию не дают сколько-нибудь качественного результата – поначалу это сбивает с толку.

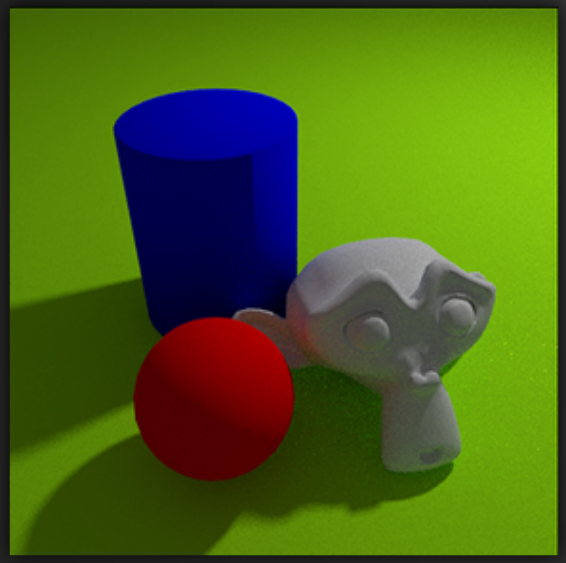






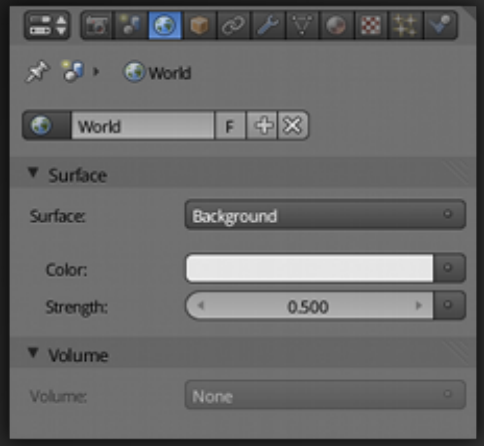
Общая тусклость – оттого, что источник света сцены по умолчанию не работает в Cycles. Выделите его и в настройках **Lamp** включите **Use Nodes** на вкладке **Surface**. Выберите **Emission**, цвет сделайте белым, параметр **Strenght** увеличьте до нескольких сотен (700-900). Качество картинки можно повысить, увеличив число сэмплов на пиксель. В настройках рендеринга (на вкладке **Integrator**) увеличьте **Samples -> Render** до 50. Кроме того, можно использовать пресет **Full Global Illimination**.

Пока вы используете точечный источник света, вы будете получать четкие тени. Для более реалистичного результата следует заменить его объемным. Для этого можно использовать икосферу (Icosphere). Сделайте ее дочерней для лампы и уменьшите до нужного размера. В настройках материала выставьте те же настройки, что и для лампы. Саму лампу можно отключить для рендеринга в Outliner'е.

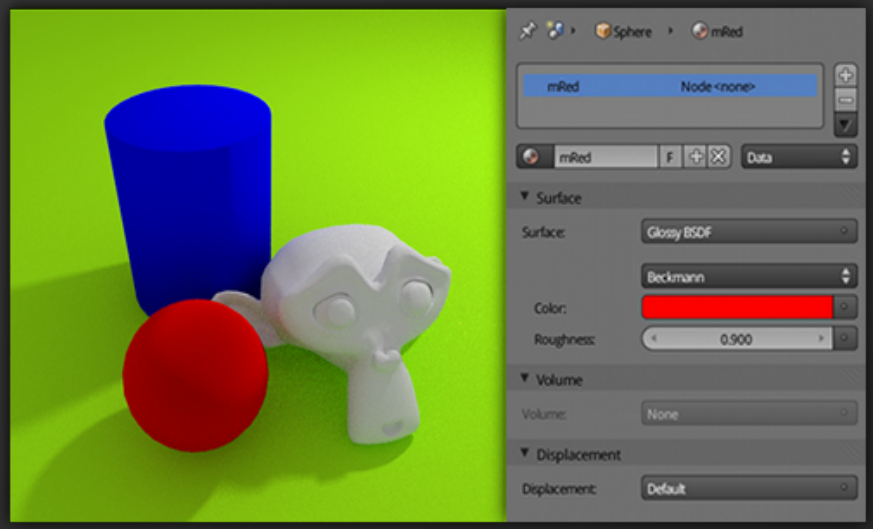


Один точечный источник света создает резкую тень. Объемный источник света создает области полутени и тени. При этом область тени меньше, чем в случае с точечным источником света.

Чтобы сделать изображение в целом посветлее, можно увеличить яркость фонового свечения:



В Cycles есть различные модели освещения для материалов (в терминологии движка – BSDF, Bidirectional scattering distribution function – двунаправленная функция распределения рассеивания). Их пока не так много, но минимальный набор уже имеется. Например, здесь я применил к красной сфере модель Бекмана (Beckmann) – подробнее о ней вы можете узнать в FPS №14 '11.



Конечно, Cycles пока носит экспериментальный характер. Маловато настроек, нет предпросмотра материалов. Но разработка не стоит на месте: «официальную» интеграцию движка обещают уже в Blender 2.61 в декабре месяце. На мой взгляд, есть все основания для оптимизма.

Gecko
gecko0307@gmail.com

Blender

Архитектурное моделирование: интервью с Томасом Крийненом

В последние годы растет популярность Blender как инструмента архитекторов. О реальном применении программы для градостроительного проектирования можно было прочитать уже пять лет назад, а развитие LuxRender и YafaRay сделало Blender конкурентоспособным инструментом архитектурной визуализации. Темпы роста интереса к программе, тем не менее, сдерживаются не только некоторыми ограничениями в самом приложении, но и проблемами с возможностью открывать в Blender файлы проприетарного ПО.

Молодой нидерландский архитектор и программист Томас Крийнен (Thomas Krijnen) начал в этом году проект **IfcOpenShell**, представляющий собой библиотеку для чтения и записи файлов IFC – открытого формата, наряду с gbXML используемого в методологии информационного моделирования зданий (BIM). Эта методология, интерес к которой в индустрии тоже растет, позволяет использовать в рабочих процессах произвольное ПО, что повышает шансы Blender на занятие более прочной позиции в индустрии.

Томас уже взаимодействует с участниками проектов BIMserver и osBIM, но в своем собственном проекте он пока что один. Поэтому несколько недель назад разработчик опубликовал обращение к сообществу, в котором попросил посильно поучаствовать в проекте. Его интересуют не только патчи, но и сообщения об ошибках, тестовые файлы и т.д.

Недавно было опубликовано интервью, в котором Томас подробнее рассказывает о проекте и о своем взгляде на Blender как инструмент архитектора.

Расскажи немного о себе.

Меня зовут Томас Крийнен. Помимо проекта IfcOpenShell я работаю над степенью магистра в техническом университете города Эйндховен, Нидерланды. Мой основной учебный курс объединяет архитектуру и IT.

В этом году ты начал новый проект IfcOpenShell. В Blender уже есть поддержка формата gbXML (предоставляемая, к примеру, проектом BlendME наряду с другими «строительными» дополнениями). Откуда интерес к IFC? Это как-то связано с твоими коммерческими проектами?

В индустрии AEC (Architecture Engineering and Construction) долгое время были распространены закрытые форматы файлов, а способность к взаимодействию между ПО была так себе. Так что у свободного ПО было крайне мало шансов стать инструментарием архитекторов и строителей. Насколько я моуг судить, в какой-то момент индустрия сделала шаг навстречу открытым стандартам, таким как IFC и gbXML (или только собирается сделать, а может быть все же сделала, но еще передумает). Эти стандарты, как и методология BIM (Building Information

Modeling, информационное моделирование зданий), делают возможным модульное использование ПО, что для UNIX-систем является нормой вот уже несколько десятков лет.

Я представляю себе экосистему взаимодействующего модульного софта, где каждое приложение выполняет четко определенную задачу. Например, в ней может быть инновационное приложение X для генеративного дизайна, создающее контуры, которые затем рассматриваются в программе для структурного анализа Y, на основе чего в программе Z рисуется остальная часть здания, которая затем рендерится в программе W, чертится в программе V, хранится и анализируется на предмет ошибок в U, просматривается клиентами в T, ну и т.д. Каждое из них передает данные без потери информации, и пользователь сам выбирает, какое приложение использовать в каждом случае – закрытое или свободное. А еще я думаю, что вот-вот рак на горе свистнет :)

Уже существуют некоторые открытые решения для работы с IFC. Автор блога cad-3d.blogspot.com недавно их кратко рассмотрел. Какое место среди них занимает твой IfcOpenShell, и каким ты видишь его будущее?

Наиболее важным для описанной выше экосистемы я считаю полную поддержку геометрии в IFC, поскольку нам, архитекторам, нравится работать с формами, а в AEC мы имеем дело с весьма реальными объектами. Мне показалось наиболее разумным задействовать для поддержки IFC платформу Open CASCADE, предназначенную для твердотельного параметрического моделирования. Ну а поскольку в Open CASCADE уже есть поддержка STEP, на базе которого построен IFC, реализация большинства геометрических операций IFC не составляет особых сложностей.

Писать собственный код разбора файлов с нуля было, вероятно, не самой удачной идеей, но я надеюсь в конечном счете обеспечить прозрачную поддержку всех форматов IFC (кроме основанного на STEP есть еще и формат на базе XML). Существующие открытые библиотеки для работы с IFC этого пока не могут. IfcOpenShell, правда, пока что тоже – как говорится, оцените иронию. Надеюсь, что IfcOpenShell станет полнофункциональной библиотекой для чтения и записи файлов IFC с акцентом на геометрию и метаданные (вроде свойств материалов).

Судя по журналу SVN, ты сейчас единственный разработчик в проекте. Полагаю, отсюда и обращение к сообществу. У тебя есть что-нибудь вроде плана развития проекта, на который любой потенциальный участник мог бы опереться?

Вы правы насчет количества участников. Тем не менее, я чувствую поддержку как со стороны проекта osBIM, который более официально запустится в октябре этого года, так и со стороны коллег в университете. Плана работы пока нет, но когда я писал этот призыв, то в том числе надеялся промотивировать потенциальных участников к размышлению, как именно можно использовать подобную свободную реализацию IFC. Некоторые интересные идеи перечислены в трекере на SourceForge, но желающие поучаствовать могут их спокойно проигнорировать и предложить что-то свое, вне зависимости от того, хотят ли они использовать IfcOpenShell в закрытом или открытом ПО.

Blender и свободные визуализаторы вроде LuxRender и YafaRay понемногу становятся совершенно полноправными инструментами в профессиональной архитектурной визуализации. Какие преимущества у них ты видишь как архитектор?

Преимуществ немало. Мне нравится честность, с которой разработчики сообщают пользователям не только о сильных сторонах, но и об ограничениях своего ПО. По моему личному опыту, количество ошибок в закрытом и свободном ПО примерно одинаково. Разница же состоит в том, что разработчики СПО склонны открыто сообщать о них пользователям и предлагать обходные пути. Если же вы натыкаетесь на ошибку в закрытом приложении, это все равно что биться головой об стену, поскольку вы предоставлены самому себе...

Есть и чисто технические преимущества, такие как ранняя реализация новых наработок из академической среды (посмотрим, к примеру, на разные недавние новшества в LuxRender). Подобный функционал дает вам конкурентное преимущество. Ну и кроме того, вы сохраняете полный контроль над своей работой, поскольку формат хранения данных открыт и задокументирован. И конечно же, свободное ПО работает на любых платформах.

Наконец, мне как человеку, который никак не решит, архитектором ему стать или программистом, весьма импонирует то, что разработчики относятся к пользователям как к потенциальным участникам проекта и поощряют их активное участие в улучшении софта.

Ты все еще используешь закрытое ПО в своей работе? Если да, то почему?

Иногда я использую 3ds Max и Revit, что объясняется, главным образом, необходимостью открывать присылаемые коллегами .dwg и .rvt. Кроме того (чего уж скрывать) в области AEC свободное ПО не всегда оказывается на одном качественном уровне с коммерческим.

Что на твой взгляд необходимо улучшить в Blender, чтобы сделать его более конкурентоспособным в качестве инструмента для архитектурного проектирования?

На мой лично взгляд, Blender нужно продолжать делать то, что он умеет делать лучше всего. Я не вижу особой необходимости в каких-то глобальных изменениях. Переработанный в 2.5x интерфейс, как мне кажется, сделал неактуальной основную критику в адрес проекта. С другой стороны, не помешало бы больше внимания уделять совместимости: обеспечить максимальную поддержку COLLADA, ну и, может быть, даже IFC.

Насколько пристально ты следишь за развитием 2.5x, а теперь уже и 2.6x? А за совершенствованием GPU-версии LuxRender? Какие новшества для тебя наиболее актуальны?

На самом деле, хотелось бы следить повнимательнее. Просто сейчас у меня есть только быстро перегревающийся лэптоп и 16-ядерный сервер без монитора и GPU. Иными словами, часть новинок оказывается как бы не в тему.

А вот развитие нового Python API в Blender 2.5 меня как раз очень интересует. Собственно, главным образом за возможность писать сценарии на Python я и полюбил Blender, ведь с ним мне не приходится выбирать между искусством и программированием.

В одном из своих проектов ты использовал метасферы (metaballs) для создания естественных, органических форм. Насколько я могу судить, для архитектуры это несколько нетипичное решение. Что из недавних изобретений, может быть, совершенно безумных и экспериментальных, кажется тебе потенциально интересным для архитектуры?

В таком контексте единственная сравнительно новая технология, которую я могу сейчас назвать, достаточно очевидна и вовсе не экстремальна. Мне очень интересно, как скажется на архитектуре быстро растущий тренд CAM (Computer Aided Manufacturing) и личных трехмерных принтеров, особенно в масштабе целых зданий. То есть, я говорю не о печати отдельных блоков на фабрике, а об управляемых компьютером устройствах, слой за слоем печатающих бетоном целые здания.

Все это может привести к смене представлений о том, как личная самоидентификация выражается через жилище, поскольку исчезнет сама причина строительства одинаковых домов.

Интервью взял Александр ПРОКУДИН
Источник: ***http://linuxgraphics.ru***

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

Язык D: новости

D вернулся в top-20 ежемесячного рейтинга TIOBE Programming Community Index. Рейтинг публикуется компанией TIOBE, которая занимается мониторингом и оценкой качества программного обеспечения, в том числе – составляет рейтинг популярности языков программирования исходя из числа просмотров веб-страниц. Подсчет выполняется при помощи поисковых систем Google, Bing и Yahoo, а также других популярных ресурсов, имеющих разделы по разработке ПО – например, Wikipedia, YouTube и Baidu.

Впервые с середины 2009 года в двадцатку Tiobe вернулся язык D, который занял 20-е место. Как полагают в TIOBE, возможная причина возвращения D в «топ» – выход книги Андрея Александреску «Язык программирования D» в июне прошлого года. D вытеснил из двадцатки язык F#, который на короткое время впервые попал в нее в августе.

Номером один в списке Tiobe остается Java, а за ним идут C, C++, C# и PHP. Язык Objective-C добрался до шестого места. C#, в свою очередь, поднялся с шестого на четвертое место, а PHP переместился с четвертого на пятое. В двадцатку также попали Visual Basic, Python, Perl, JavaScript, Ruby, Delphi/Object Pascal, Lua, Lisp, Transact-SQL, Pascal, PL-SQL, Ada и RPG (OS/400). Java сохраняет первое место уже десять лет, за исключением нескольких месяцев, когда двадцатку возглавлял C.

D на пути к включению в состав GCC. Разработчики языка D сообщили об урегулировании вопроса с передачей Фонду свободного ПО (Free Software Foundation) имущественных прав на код фронтэнда компилятора GDC. Передача прав на код является одним из основных условий, требуемых для включения новых фронтэндов в GCC. В дальнейшем остается согласовать некоторые незначительные технические вопросы и обеспечить гарантию, что после включения в состав GCC код будет поддерживаться в актуальном состоянии.

DigitalMars еще в прошлом году заявила о желании включить GDC в состав GCC, но процесс остановился из-за нежелания терять права на код. Предложенный вариант с форком кода GDC и передачей прав только на форк, был отвергнут разработчиками GCC. После чего был предложен еще один вариант: Digital Mars передает права на код FSF – но FSF в ответ тут же предоставляет неограниченную лицензию, позволяющую делать с кодом все, что захочется. Подробности того, как удалось согласовать данный вопрос, не сообщаются. Судя по всему, Фонд СПО согласился на третий вариант, связанный с предоставлением фонду неограниченной лицензии на код.

Будем надеяться, что поддержка со стороны FSF и разработчиков GCC помогут D выйти, наконец, из «андеграунда» и занять достойное место в индустрии.

Значимые релизы за последние два месяца

> В конце октября обновился компилятор DMD для D1 и D2 – до версий 1.071 и 2.056 соответственно. Релиз включает большое количество багфиксов. Компилятор доступен под все основные операционные системы (Win32, Linux, Mac OS X, FreeBSD) для i386 и x86_64.
http://www.digitalmars.com/d/download.html

> Visual D (проект по интеграции D в среду разработки Microsoft Visual Studio) обновился до версии 0.3.28. Исправлено много ошибок, добавлены новые опции настроек и другие мелкие доработки.
http://www.dsource.org/projects/visuald
http://www.microsoft.com/visualstudio/ru-ru

> Вышел DDT (D Development Tools) 0.5.0 – плагин поддержки D для среды разработки Eclipse.
http://code.google.com/a/eclipselabs.org/p/ddt
http://www.eclipse.org

> Разработчики wxD – биндинга к популярной библиотеке wxWidgets – выпустили новую версию библиотеки. wxD 0.16 теперь компилируется LDC 0.9.2 и DMD 2.054, поддерживает wxWidgets 2.9.2 / Cocoa.
http://wxd.sourceforge.net
http://www.wxwidgets.org

> GtkD, тулкит для разработки графических приложений на D с использованием популярной библиотеки GTK+, обновился до версии 1.5. Эта версия полностью поддерживает GTK+ 2.22.x со всеми сопутствующими библиотеками (glib, cairo, pango и т.д.). Обновлен биндинг к Gstreamer, добавлен интерфейс к GtkBuildable. Кроме того, была улучшена работа с памятью.
http://www.dsource.org/projects/gtkd
http://www.gtk.org

> Вышла бета-версия Orange 1.0.0 – свободной библиотеки сериализации (перевода структур данных в последовательность битов). Orange поддерживает все доступные типы в языке D, имеет раздельный сериализатор и архиватор. Библиотека работает на D1 и D2, Tango и Phobos; распространяется по лицензии Boost.
http://dsource.org/projects/orange

> Для D появился парсер языка разметки YAML. Проект D:YAML ставит целью создание полноценной YAML-библиотеки для D2/Phobos. В октябре вышла версия 0.2 этого проекта.
https://github.com/kiith-sa/D-YAML

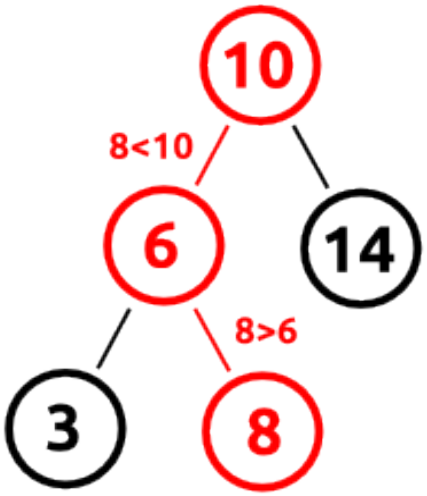
Gecko
gecko0307@gmail.com

Двоичное дерево, хеширование и ассоциативный массив

Язык D, в числе прочего, привлекает множеством встроенных алгоритмов, из которых наиболее часто используются средства работы с массивами. Особенно интересны ассоциативные массивы, которые принимают в качестве индекса любое значение. Можно просто их использовать, не вдаваясь в технические подробности – однако меня всегда интересовало: как они реализуются? В этой статье я привожу простую реализацию ассоциативного массива на основе двоичного дерева.

Двоичное дерево (binary tree), как правило, используется для ускорения поиска значений. Вместо того, чтобы перебирать все возможные варианты, мы приходим к нужному результату «кратчайшим путем». Двоичное дерево представляет собой структуру из пар «ключ-значение», где ключ – некий уникальный номер (необязательно порядковый). Каждая такая пара представляет собой отдельный узел дерева. Узлы могут иметь двух потомков – условно «левого» и «правого». Ключ левого потомка всегда меньше ключа родителя, а ключ правого – больше.

Таким образом, рекурсивный алгоритм поиска значения по ключу заключается в следующем: если заданный ключ меньше ключа текущего узла, опрашивается левый потомок, если больше – правый. Поиск продолжается, пока заданный ключ не будет равен ключу текущего узла, либо пока мы не окажемся «в тупике».



Ход поиска ключа "8"
в двоичном дереве

Базовый интерфейс двоичного дерева поиска состоит из трех операций:

find(key) – поиск узла, в котором хранится пара (key, value).
insert(key, value) – добавление в дерево пары (key, value).
remove(key) – удаление узла с ключом key.

По сути, двоичное дерево поиска — это структура данных, способная хранить таблицу пар (key, value) и поддерживающая три операции: find, insert, remove.

Ниже приведена реализация двоичного дерева поиска в виде шаблона класса:

```
class BST(T)
{
    public this() { }

    protected:
    BST left = null;
    BST right = null;
    int key = 0;
    T value;

    protected this(int k, T v)
    {
        key = k;
        value = v;
    }

    public void insert(int k, T v)
    {
        if (k < key)
        {
            if (left is null) left = new BST(k, v);
            else left.insert(k, v);
        }
        else if (k > key)
        {
            if (right is null) right = new BST(k, v);
            else right.insert(k, v);
        }
        else value = v;
    }

    public BST find(int k)
    {
        if (k < key)
        {
            if (left !is null) return left.find(k);
            else return null;
        }
        else if (k > key)
        {
            if (right !is null) return right.find(k);
            else return null;
        }
        else return this;
    }
}
```

Дерево в его текущем состоянии можно использовать, например, для хранения имен и фамилий по идентификационным номерам:

```
auto bst = new BST!string;
bst.insert(12608, "Dmitry Egorov");
bst.insert(67045, "Ivan Petrov");
assert( bst.find(12608).value == "Dmitry Egorov" );
```

Для удаления узлов можно использовать следующий алгоритм: если у узла нет обоих потомков, удаляем узел и обнуляем ссылку на него у родителя; если у узла всего один потомок, копируем его пару «ключ-значение» в родителя и удаляем. Если у узла есть оба потомка, то находим узел, являющийся крайним левым узлом правого потомка, копируем из него данные и рекурсивно удаляем.

```
private BST findLeftMost()
{
    if (left is null) return this;
    else return left.findLeftMost();
}

public void remove(int k, BST par = null)
{
    if (k < key) {
        if (left !is null) left.remove(k, this);
        else return;
    }
    else if (k > key) {
        if (right !is null) right.remove(k, this);
        else return;
    }
    else {
        if (left !is null && right !is null) {
            auto m = right.findLeftMost();
            key = m.key;
            value = m.value;
            right.remove(key, this);
        }
        else if (this == par.left) {
            par.left = (left !is null)? left : right;
            delete this;
        }
        else if (this == par.right) {
            par.right = (left !is null)? left : right;
            delete this;
        }
    }
}
```

Но как же быть с другими способами индексирования? Для использования в качестве индекса значений нецелочисленных типов применяются специальные функции, преобразующие их в уникальные числа – так называемые хэш-функции.

Хэширование – это преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Результат такого преобразования называют дайджестом (digest). Существует множество алгоритмов хэширования с различными характеристиками (разрядность, вычислительная сложность, криптостойкость и т.п.). Выбор той или иной хэш-функции определяется спецификой решаемой задачи. Простейшими примерами хэш-функций могут служить контрольная сумма и CRC.

```
pure int hash(string key, int tableSize = 5381)
{
    int hashVal = 0;
    for (int x = 0; x < key.length; ++x)
    {
        hashVal ^= (hashVal << 5)+(hashVal >> 2)+key[x];
    }
    return hashVal % tableSize;
}
```

Таким образом, мы можем передавать в методы insert и find контрольную сумму строки – результат функции hash. В связи с этим, не лишним будет перегрузить операторы индексирования, чтобы класс внешне был больше похож на массив. Для этого проще будет объявить новый класс-враппер, наследующий от BST.

```
class AArray(T): BST!(T)
{
    public:

    this() { }

    void opIndexAssign(T v, string i)
    {
        insert(hash(i), v);
    }

    T opIndex(string i)
    {
        auto node = find(hash(i));
        assert (node !is null);
        return node.value;
    }

    T* opIn_r (string i)
    {
        auto node = find(hash(i));
        if (node !is null) return &node.value;
        else return null;
    }

    void remove(string i)
    {
        super.remove(hash(i));
    }
}
```

Пример использования:

```
auto aa = new AArray!string;
aa["sysadmin"] = "Dmitry Egorov";
aa["programmer"] = "Ivan Petrov";

assert (aa["sysadmin"] == "Dmitry Egorov");

if ("programmer" in aa)
    aa.remove("programmer");
assert ("programmer" !in aa);
```

Реализации ассоциативных массивов, основанные на различных деревьях поиска, наиболее популярны. Так, например, в STL контейнер map реализован на основе красно-черного дерева. В Ruby, Tcl и Python используется один из вариантов хэш-таблицы. Существуют и другие варианты.

У каждой реализации есть свои достоинства и недостатки. Важно, чтобы все три операции выполнялись как в среднем, так и в худшем случае за время O (log n), где n – текущее количество хранимых пар.

Тимур ГАФАРОВ
tgafaroff@gmail.com

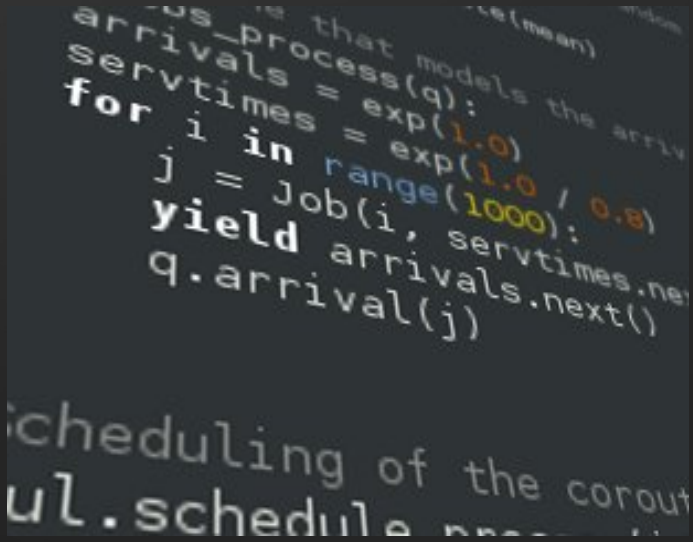
Сопрограммы в языке D

Сопрограмма (coroutine) – компонент программы, обобщающий понятие подпрограммы (функции). Сопрограмма может иметь не одну, а несколько входных точек; ее можно останавливать и продолжать с сохранением ее внутреннего состояния.

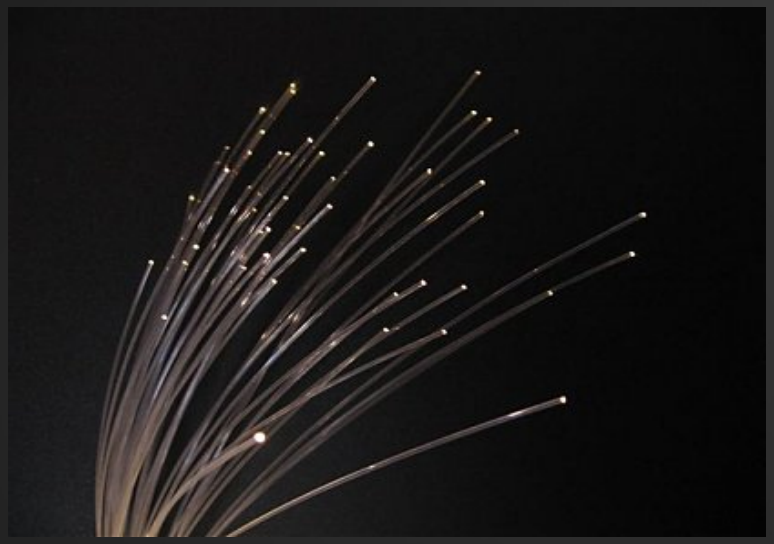
Сопрограммы являются более гибкими и обобщенными, чем подпрограммы, но реже используются на практике. Подпрограмма имеет всегда одну входную точку, сопрограмма имеет стартовую точку входа и размещенные внутри последовательность возвратов и следующих за ними точек входа. Подпрограмма может возвращаться только однажды, сопрограмма может возвращать управление несколько раз. Любую подпрограмму можно реализовать как сопрограмму. По утверждению Дональда Кнута, «подпрограмма является частным случаем сопрограммы».

Изначально применение сопрограмм являлось методикой ассемблера и практиковалось лишь в некоторых высокоуровневых языках (Simula, Modula-2). Большинство современных популярных языков программирования, включая C и его производные, не имеют прямой поддержки сопрограмм в языке или стандартной библиотеке. Это обусловлено, большей частью, требованиями к стековой реализации подпрограмм. Из популярных языков со встроенной поддержкой сопрограмм можно отметить C#, Python, Ruby, Perl 6, Io, Go, Lua, Modula-2, Simula, Tcl.

Сопрограммы полезны для реализации модели акторов. Это математическая модель параллельных вычислений, которая трактует понятие «актор» как универсальный примитив численного расчета: в ответ на сообщения, которые он получает, актор может принимать локальные решения, создавать новых акторов, посылать свои сообщения, а также устанавливать, как следует реагировать на последующие сообщения. Изначально эта концепция использовалась как основа для понимания исчисления процессов и как теоретическая база для ряда практических реализаций параллельных систем.



Во многих языках сопрограммы реализуются в виде волокон (fiber). Волокно – это «легковесный» вариант потока (thread). Подобно потокам, волокна используют общее адресное пространство. Однако волокна используют кооперативную многозадачность, в то время как потоки – вытесняющую: потоки в основном зависят от планировщика задач, который приостанавливает одну и дает выполняться другой; волокна же передают управление другим волокнам.



По сути, волокна описывают те же концепции, что и сопрограммы. Разница, если она есть, состоит в том, что сопрограммы – это конструкции языкового уровня, а волокна – системного. Взаимное переключение между волокнами выглядит как безадресный переход между внутренними участками кода сопрограмм. Волокна могут рассматриваться как реализация сопрограмм, или же в качестве основы, на которой реализуются сопрограммы.

В языке D волокна поддерживаются в Phobos – как дополнительный «бонус» в виде класса Fiber в core.thread. Возврат из сопрограммы осуществляется при помощи статического метода Fiber.yield.

```
import core.thread;
import std.stdio;

void hello()
{
    writeln("Hello, world!");
    Fiber.yield();

    writeln("How are you?");
    Fiber.yield();

    writeln("Bye!");
    Fiber.yield();
}

void main()
{
    Fiber fiber = new Fiber(&hello);
    fiber.call();
    fiber.call();
    fiber.call();
}
```

Подобно потокам, волокна могут быть созданы как через композицию, так и через наследование.

```
class MyFiber: Fiber
{
    this()
    {
        super(&run);
    }

    void run()
    {
        writeln("Hello, world!");
        Fiber.yield();

        writeln("How are you?");
        Fiber.yield();

        writeln("Bye!");
        Fiber.yield();
    }
}

void main()
{
    auto fiber = new MyFiber();

    fiber.call();
    fiber.call();
    fiber.call();
}
```

После полного завершения, волокно необходимо сбросить, чтобы использовать его повторно. Это можно сделать методом reset.

Текущее состояние волокна (HOLD, EXEC или TERM – «приостановлено», «выполняется» или «завершено») можно узнать из поля State.

```
while (fiber.state != Fiber.State.TERM)
{
    if (fiber.state == Fiber.State.HOLD)
        fiber.call();
}

fiber.reset();
```

Волокнам требуется меньшая поддержка со стороны операционной системы, чем для потоков. Если в программе используется много однотипных потоков, то целесообразно создавать волокна, так как они значительно экономят ресурсы.

Тимур ГАФАРОВ
tgafaroff@gmail.com

Python 2 или Python 3?

Многим известно, что в последние годы активно разрабатывается третья версия замечательного языка программирования Python. Многие продукты, находящиеся в авангарде свободного ПО (например, Blender), уже перешли на нее. Но немало софта все еще пишется под Python 2.x. Да и как быть с громадной базой кода, накопленной за годы для старой ветки языка? Начинающим знакомство с Python можно предложить несколько советов, которые помогут им определиться с выбором версии языка.

Если кратко: Python 2 – это статус-кво. Язык стабилен, насколько это возможно, и после релиза 2.7 никаких кардинальных изменений и нововведений в нем уже не предвидится. Поддержка 2.x, по всей видимости, будет сохраняться еще несколько лет, поскольку перевод уже существующих программ на новую версию Python требует значительных трудозатрат, и не всякий производитель станет этим заниматься. Однако нет никаких причин не использовать Python 3 для абсолютно новых проектов, написанных с нуля (как, например, в случае с Blender 2.5).

Итак, вам рекомендуется выбрать Python 2, если вы:

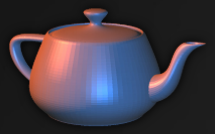
- работаете в среде, которую не можете контролировать полностью (например, в проприетарном приложении или сетевой инфраструктуре). В этом случае вам и не предоставляется свобода выбора – придется писать на той версии языка, какая есть.

- используете специфическую библиотеку или инструментарий, написанный на Python 2.x и не поддерживающий третью версию языка (к примеру, PyGTK).

- планируете изучение языка по книге или справочной документации. Таких материалов очень много для Python 2.x и пока маловато для Python 3.x. Точно такая же ситуация и в онлайн-сообществах Python, где поддержка второй ветки языка, как правило, гораздо лучше.

При этом необходимо помнить, что многие новые возможности Python 3.x были обратно портированы на Python 2.x. Как следствие, число вещей, которые вы можете сделать в Python 3.x и не можете в старых версиях, не так уж и велико. Хорошо написанный код для 2.x будет мало чем отличаться от такового для 3.x. Если писать код с учетом эволюции языка, не используя deprecated-особенности, портировать его с одной версии на другую будет гораздо проще. Тем более, что для этой цели существуют специальные утилиты – например 2to3, которая поставляется в комплекте с Python 3.

Конечно, портирование кода – идеальный вариант. Но если это невозможно, наиболее разумным решением будет, как ни странно, вовсе отказаться от старых библиотек. Необходимо понимать, что legacy-код – это якорь, тормозящий движение и развитие проектов. Старайтесь писать такие программы, которые можно, при необходимости, легко переписать на любом языке.



Модели освещения

Коэффициент Френеля

Для получения реалистичных отражений (равно как и преломлений) необходимо учитывать тот факт, что доля отраженной и преломленной световой энергии не является постоянной величиной – она зависит от угла падения.

Например, если смотреть на воду под большим углом (сверху), то можно увидеть дно. А если смотреть под малым углом (например, с берега озера на горизонт), то дна уже не видно – водная поверхность будет отражать небо. В компьютерной графике при моделировании отражающих материалов этот переход, как правило, описывается коэффициентом Френеля (Fresnel).

Огюстен Жан Френель (1788-1827) – французский физик, один из создателей волновой теории света. Переоткрыв 1815 г. принцип интерференции, Френель дополнил известный принцип Гюйгенса, введя понятие о когерентности световых волн и их интерференции (принцип Гюйгенса-Френеля). На основе этих двух принципов он разработал теорию дифракции света. Также изобрел несколько интерференционных приборов – зеркало Френеля, бипризму Френеля, линзу Френеля.

Коэффициент Френеля определяет соотношение отраженной и преломленной энергии на границе двух сред с разными показателями преломления. Количество отраженной энергии выражается следующей формулой:

$$R(\theta) = \frac{1}{2} \cdot \left(\frac{g-c}{g+c}\right)^2 \cdot \left(1 + \left(\frac{c(g+c) - (\eta_i/\eta_t)^2}{c(g-c) - (\eta_i/\eta_t)^2}\right)^2\right)$$

$$c = \cos\theta \cdot (\eta_i/\eta_t)$$

$$g = \sqrt{1 + c^2 - (\eta_i/\eta_t)^2}$$

где η_i, η_t – показатели преломления сред, θ – угол между нормалью и падающим лучом.

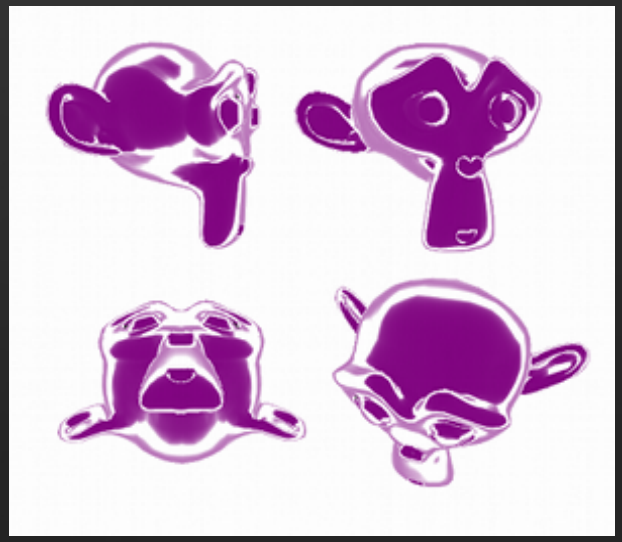
Показатель преломления вещества равен отношению фазовых скоростей света (электромагнитных волн) в вакууме и в данном веществе. Он зависит от свойств вещества и длины волны излучения. Для некоторых веществ показатель преломления достаточно сильно меняется при изменении частоты электромагнитных волн от низких частот до оптических и далее. По умолчанию обычно имеется в виду оптический диапазон.

Существуют оптически анизотропные вещества, в которых показатель преломления зависит еще и от направления и поляризации света. Это, например, все кристаллы с достаточно низкой симметрией кристаллической решетки, а также вещества, подвергнутые механической деформации.

Показатели преломления различных веществ

Воздух	1.0002926	Кварц	1.544
Вода	1.332986	Топаз	1.63
Стекло	1.52	Лед	1.31
Кремний	4.010	Сахар	1.56
Алмаз	2.419	Этиловый спирт	1.36
Ацетон	1.36	Мед	1.484
Бензол	1.5	Молоко	1.35
Глицерин	1.473	Пиво	1.345
Водка	1.363	Сахарный сироп	1.49
Скипидар	1.472	Масло	1.535
Нейлон	1.53	Роговица глаза	1.38
Обсидиан	1.50	Пластмасса	1.460-1.55
Хрусталь	2.000	Оргстекло	1.488
Соль	1.516	Слюда	1.56-1.60

Точная формула Френеля обычно не используется ввиду ее вычислительной сложности. Есть несколько ее приближений, часто применяемых для вычислений в реальном времени – например, аппроксимация Шлика. В приведенной ниже реализации используется «честная» формула с постоянным значением (η_i/η_t).



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;

const float etha = 0.66; // 1.0003/1.52
const float ethasqr = etha * etha;

void main (void)
{
    vec4 refractColor = gl_FrontMaterial.diffuse;
    vec4 reflectColor = gl_FrontMaterial.specular;

    vec3 V = normalize(vec3(V_eye));
    vec3 L = normalize(vec3(L_eye));
    vec3 N = normalize(vec3(N_eye));

    float NV = dot(N,V);

    float c = NV * etha;
    float g = sqrt(1.0 + c * c - ethasqr);
    float gc = (g - c)/(g + c);
    float f = (c*(g-c) - ethasqr) / (c*(g+c) - ethasqr);
    float R = 0.5 * gc * gc * (1 + f * f);

    gl_FragColor = mix(refractColor, reflectColor, R);
    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.*

Gecko
gecko0307@gmail.com

Цифровая алхимия

В начале было Слово...

Алхимия

Что такое алхимия? Для современного человека – это своеобразная «мать» химии, средневековая наука о превращении металлов в золото. Однако в общем смысле алхимия – это философское учение о взаимосвязи Материи и Духа, о трансформации материального и духовного. Целью алхимиков является осуществление качественных изменений внутри одушевленного или неодушевленного объекта, его перерождение и переход на новый уровень бытия.

Родиной алхимии считается Древний Египет. Алхимики вели начало своей науки от Гермеса Трисмегиста (он же египетский бог Тот), и поэтому алхимическое искусство также называлось герметическим. Свои сосуды они запечатывали печатью с изображением Гермеса (отсюда выражение «герметически закрытый»).

Философы, желавшие передать потомкам основы своего учения и плоды своих трудов, всячески хранили себя от вульгаризации своего искусства, чтобы им не могли злоупотреблять непосвященные. Поэтому алхимические трактаты изобилуют загадками, символами и иносказаниями...

Огонь, вода, воздух, земля

Вся алхимия базируется на теории четырех элементов. В основе ее лежит платоновский принцип единой материи, которая, принимая различные формы, создает многообразие живой и неживой природы. Из этой первичной материи произошли четыре элемента (или стихии): Огонь, Вода, Воздух и Земля. Эти элементы соответствуют четырем агрегатным состояниям вещества: земля – твердое вещество; вода – жидкость; воздух – газ; огонь – раскаленный газ, плазма.

Позднее был добавлен пятый элемент – квинтэссенция (от лат. quinta essentia – «пятая сущность»), эфир, или anima mundi («душа мира») – тонкая божественная стихия, пронизывающая всю Вселенную.

Сера, соль, ртуть

Tria Prima алхимиков. Соль – универсальная субстанция, ртуть – вездесущий дух, сера – флегма, бальзам жизни. Идея о единстве серы, ртути и соли происходит из теории о подобии микрокосма и макрокосма. Человек в ней рассматривается как мир в миниатюре, как отражение Космоса со всеми присущими тому качествами. Отсюда и значение элементов: Соль – Тело, Ртуть – Дух, Сера – Жизнь. Таким образом, Космос и Человек состоят из одних и тех же элементов.

Философский камень

Практическая задача алхимии – превращение (трансмутация) неблагородных металлов в благородные. Эта идея базируется на представлениях греческой философии о том, что материальный мир состоит из одного или нескольких «первоэлементов», которые при определенных условиях могут переходить друг в друга. Вещество, способное к трансмутации, носит название философского камня. Различают два камня: один – белый, превращающий металлы в серебро («белая тинктура»); другой – красный, превращающий их в золото («красная тинктура»)...

Алхимия как игра

Алхимия – очень обширная область, имеющая пересечения с астрологией, магией, эзотерикой. Будучи хорошей «пищей для ума», в наши дни она получила оригинальное воплощение в виде компьютерных игр.

Идея игры на основе алхимических принципов существует уже достаточно давно. Одной из первых была Alchemy для DOS и Windows 9x, цель которой заключается в создании целого мира на основе четырех элементов – Воды, Земли, Воздуха и Огня. Комбинируя их, игрок создает новые субстанции и переходит на более высокие ступени развития. Вся прелесть заключается в том, что таких комбинаций необычайно много, и их число растет по мере прохождения. Развиваться можно в разные области: химию, физику, флору, фауну. Можно развивать технологии – ядерную энергию, химическую промышленность и т.д.

У игры есть современные римейки. Один из самых известных – отечественная «Алхимия» (**http://alchemygame.ru**). Это браузерная игра на JavaScript, которая изначально имела текстово-табличный интерфейс (впоследствии появились и графические варианты). Появившись в 2010 г., игра наделала шума, вызвала много эмоций и обсуждений. Было предложено несколько вариантов развития геймплея. После волны обсуждения появилось несколько полных клонов и версий с улучшениями – с увеличеснным количеством элементов и с более удобным интерфейсом. Стало ясно, что игра перспективная, и будет развиваться.

В базовой версии «Алхимии» предлагается открыть все возможные элементы (их количество варьируется от версии к версии) путем соединения – реагирования. Так, соединив Воздух и Огонь получаем Энергию, соединив Воду и Землю – Болото. Если соединить Болото и Энергию – получим Жизнь, затем соединяем ее снова с Водой – получаем Водоросли...

Таким образом совершается эволюционный цикл, от бактерий до «венца творения» – Человека, который оказывается способен изменять окружающий мир, создавая орудия труда, материалы и машины. Можно также создавать различные химикаты, нематериальные субстанции и даже мифические существа (Дракон, Голем, Вампир, Призрак).

Логика в игре не совсем прямая. То есть, реакция – это не всегда смешивание объектов в прямом смысле, а скорее смешивание их свойств. Это вносит в игровой процесс элемент хаоса и угадывания. Как следствие – в «Алхимии» немало юмористических моментов. Например:

Человек + Человек = Секс

Человек + Шерсть = Обезьяна

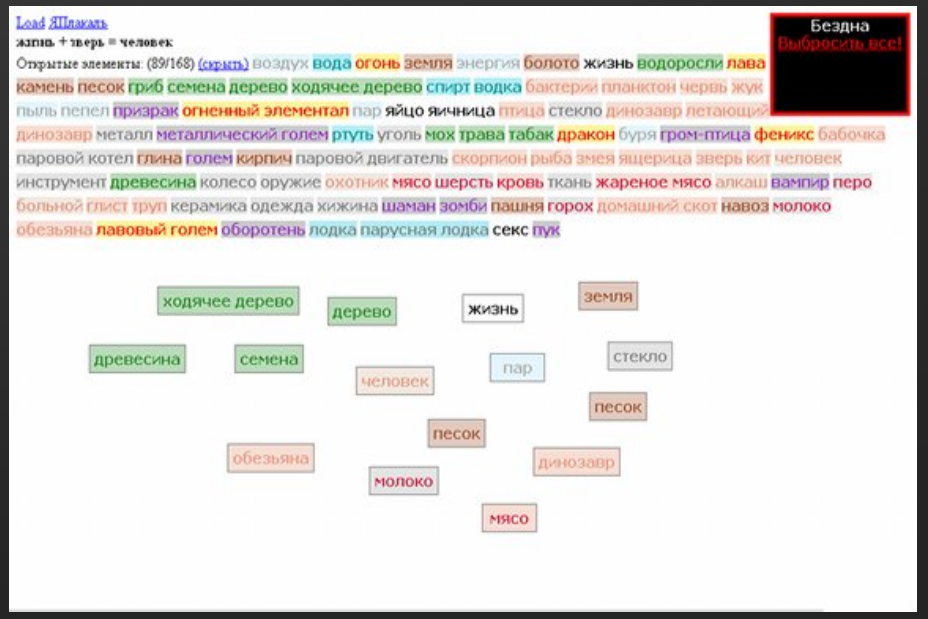
Человек + Гриб = Шаман

Человек + Горох = Пук

Человек + Трава = Растаман
Рыба + Стекло = Аквариум
Растаман + Аквариум = БГ

Множество других интересных рецептов можно найти в Интернете. Есть в игре и философский камень, который превращает любые другие элементы в золото. Как получить его – оставляю на ваше усмотрение.

Открыв новый элемент, вы получаете доступ к нему через меню. Удалить элементы можно перетащив их в специально предусмотренную Бездну. В последних версиях игры можно соединять три и более элементов: просто выделите мышкой несколько штук – и они попробуют среагировать. В «Алхимии» предусмотрено сохранение/загрузка на сервере. Можно сохранить игру на одном компьютере, а загрузить на другом: поиграв на работе, вам не придется дома начинать все заново. Есть подсказки: вы можете видеть, что можно получить из открытых элементов – и знать, к чему стремиться, а не перебирать все комбинации. Наконец, есть список рецептов для записи «формул» удачных реакций.



Самое мнетерсное: есть модкит для тех, кто хочет создать свою «Алхимию»! Любой желающий может либо создать с нуля свой набор реакций, либо добавить что-то к существующим. Фанатами уже было создано огромное количество модов, текстовых и графических, на самые разные темы – химия, физика, философия, магия, кулинария, садоводство, животноводство, детективы, мистика... На сайте игры также имеется возможность задать вопрос по игре, внести свое предложение или новую идею.

Стоит добавить, что «Алхимия» не только здорово затягивает и отлично помогает убить время – эта игра развивает память и творческое мышление.

Официальный сайт «Алхимии»:
http://alchemygame.ru

Базовая версия игры (мод №1):
http://alchemygame.ru/mods/1/play

«Алхимия» с картинками (мод «Чародей»):
http://alchemygame.ru/mods/1402/play

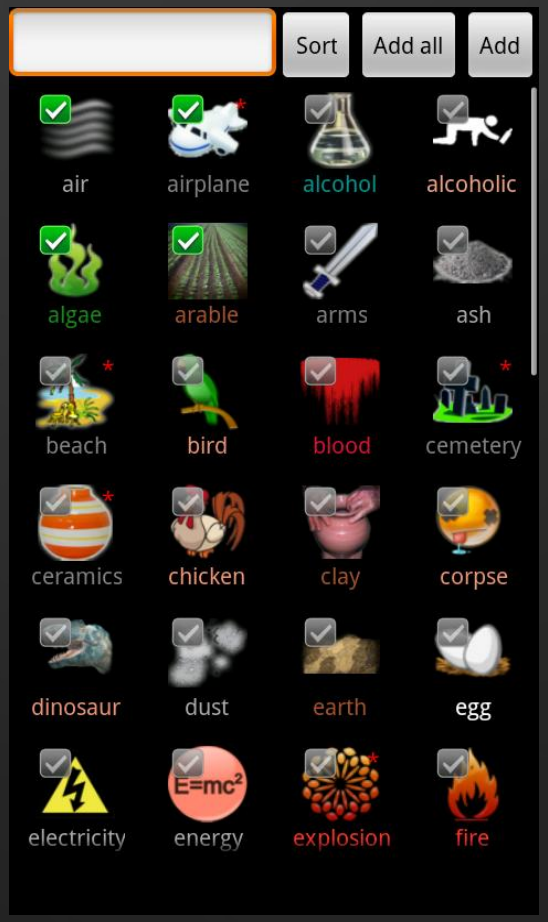
Приложение ВКонтакте :
http://vkontakte.ru/app1912160

Приложение для Google Chrome:
https://chrome.google.com/webstore/detail/inhhkggcjkbodpgnjehhckiihdgpgbkj

Для Android:
Скачать можно с сайта автора: **http://0xff.me/alchemy.apk**,
или на Android Market:
https://market.android.com/details?id=me.zed_0xff.android.alchemy

Для iPhone:
Скачать можно с сайта автора: **http://avaloid.com/doodle-god**,
или на App Store:
http://itunes.apple.com/us/app/doodle-god/id376374689?mt=8

Классическая Alchemy для DOS:
http://homepages.bicos.de/steinruecken/alchemy



Glibc hell: ищем выход

glibc (GNU C Library) – стандартная библиотека языка C, которая обеспечивает системные вызовы и основные функции, такие как printf, open, malloc и т.д. Она является одним из ключевых компонентов операционной системы GNU/Linux и используется практически всеми динамически скомпонованными программами.

Иногда необходимо, например, на системе с glibc 2.8 запустить программу, собранную с glibc 2.11. Если просто ввести в консоли ./<имя программы>, вы получите грозное сообщение «version 'GLIBC_2.11' not found». Такая проблема, конечно, не стоит для пользователей, которые устанавливают пакеты исключительно из официального репозитория своего дистрибутива. Однако и они не застрахованы: во-первых, версия дистрибутива может банально устареть, и мейнтейнер прекратит ее поддержку – пакеты просто будет неоткуда брать.

В этом случае обычно советуют установить новую версию системы, но в некоторых случаях это не представляется возможным. Например, если вы не являетесь владельцем компьютера, на котором работаете, и не имеете полномочий обновлять/переустанавливать ОС.

Тогда предлагают самый «правильный» выход: собрать программу из исходников, под вашу версию glibc. Но как быть, если программа проприетарная, и исходники недоступны? Или доступны, но сборка сопряжена с громадными трудностями и затратами времени (представьте себе, например: скомпилировать OpenOffice.org!) – не говоря уже о том, что это подчас очень непростой процесс, требующий определенных знаний всех тонкостей GNU-утилит, autotools, GCC и т.д.

Сборка программы практически никогда не ограничивается простым «sudo ./configure && make && make install» – здесь и решение зависимостей вручную, и проблемы с совместимостью разных версий библиотек, и много других нюансов. Наконец, программа может быть написана не на традиционном C, а на любом из десятков современных языков – так что придется сначала устанавливать нужный компилятор, а это вообще отдельная история... Обычный пользователь все это явно не осилит.

Казалось бы – вот она, темная сторона Linux и свободного ПО. Постоянно меняющийся и обновляющийся «зоопарк» дистрибутивов и версий зачастую работает против вас. В Windows большинство пользователей никогда не видели никаких исходников и не знают даже слов «компилятор» и «репозиторий» – они привыкли ставить софт в готовом бинарном виде, при помощи инсталлятора с диска или с сайта разработчика. Все годами работают на старой XP или даже Win2k, устанавливают современные версии программ – и у всех все работает (иногда с грехом пополам, но ведь работает!). Думаю, что не погрешу против истины, предположив, что именно из-за этого некоторые разочаровываются в Linux и возвращаются обратно на Windows. Печально? Отнюдь. Как всегда, выход есть!

Секрет в том, что программу надо вызывать особым образом. Сначала надо где-то взять собственно glibc 2.11. Можно, например, поискать в репозитории для более свежей версии дистрибутива (а можно и из исходников собрать). Ее файлы необходимо поместить в какой-нибудь каталог – например, в папку lib в вашей домашней директории. Тогда запускать требующую glibc 2.11 программу придется так:

```
$ LD_LIBRARY_PATH=~/lib ~/lib/ld-linux.so.2 ./myprogram
```

Сначала устанавливается переменная LD_LIBRARY_PATH – мы говорим, что библиотеки нужно искать в папке ~/lib (если вы поместили их в другое место, поменяйте путь). Затем запускается динамический загрузчик ld-linux.so.2, который загружает программу myprogram вместе с необходимыми ей библиотеками.

Технический смысл здесь следующий. При обычном запуске любой программы система незаметно для пользователя запускает /lib/ld-linux.so.2, загружающий все нужные библиотеки. Можно это делать самостоятельно, но обычно этого не требуется - система и так знает, что запускать его надо из /lib (точнее, это указано в самом исполняемом файле). Но если мы используем нестандартную (то есть, отличную от установленной в /lib) версию glibc, мы должны использовать соответствующую версию ld-linux.so.2. Для этого и приходится указывать путь к нему вручную.

Описанный метод лично мне помог запустить на моей устаревающей системе несколько свежих игр и сборок Blender с сайта GraphicAll.org. Напоследок привожу shell-скрипт, который позволяет запускать такие программы из меню или с рабочего стола (просто отредактируйте его под себя, поместите в папку с программой и создайте кнопку запуска, ссылающуюся на этот скрипт):

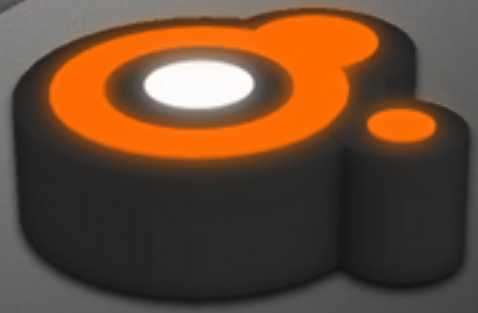
```
#!/bin/sh
BF_DIST_BIN=`dirname "$0"`
BF_PROGRAM="myprogram"
LD_LIBRARY_PATH=${BF_DIST_BIN}/lib:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH
"$BF_DIST_BIN/lib/ld-linux.so.2" "$BF_DIST_BIN/$BF_PROGRAM"
```

Примечание: скрипт написан с учетом того, что новая версия glibc находится в папке lib в директории с программой.

Тимур ГАФАРОВ
tgafaroff@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