29
2014

FPS

BLENDER: живопись полигонами

История языка D

GScript

Культовые игры: социальные симуляторы

Вся правда о творческом рейдерстве

Приставочное пиратство

+ многое другое!

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

FPS
№29

FPS - бесплатный, свободно распространяемый электронный журнал, посвященный различным видам цифрового творчества. Тематика FPS охватывает разработку игр и игровых движков, обработку изображений и звука, уроки по трехмерному моделированию и ретуши фотографий, а также свежие новости из мира цифровых технологий.

FPS - это журнал для программистов, художников, моделлеров, линуксоидов, энтузиастов движения СПО, хакеров и просто творческих людей, ищущих и умеющих находить в этом мире качественную «пищу для ума».

Мы ратуем за свободу слова и свободный обмен информацией - журнал распространяется на условиях лицензии Creative Commons (CC-BY-NC-SA). Мы рады любому сотрудничеству, приветствуем любые новые идеи и приглашаем в наш авторский коллектив всех желающих!

Журнал издается в Казани с января 2008 г. и в данный момент выходит раз в два-три месяца.

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

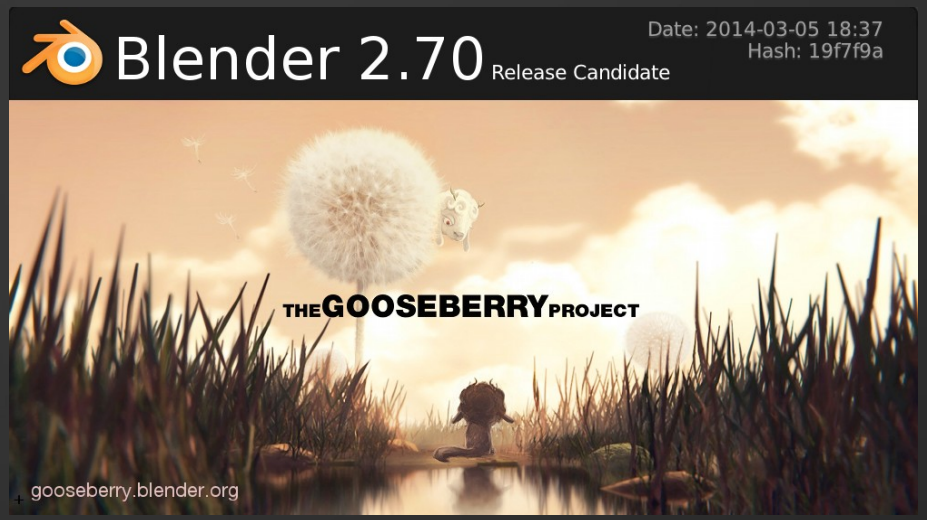
По вопросам сотрудничества обращайтесь по адресу:
gecko0307@gmail.com

Читайте в номере:

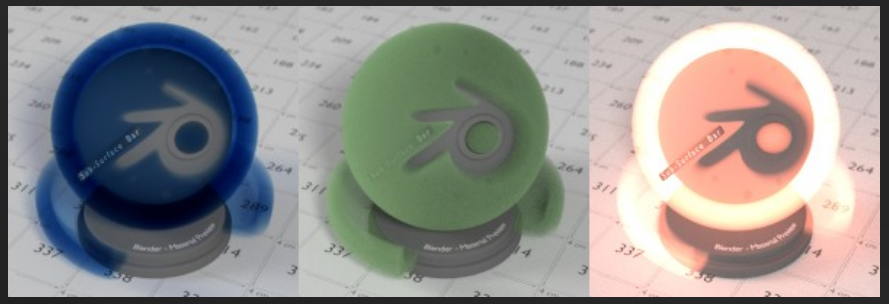
Blender Новости

Состоялся релиз Blender 2.70. В этой версии дебютировала начальная поддержка объемных материалов в Cycles, которая позволит в будущем реалистично рендерить такие эффекты, как огонь, дым и туман. Есть поддержка объемного поглощения света (absorption), рассеивания (scatter) и излучения (emission). К сожалению, пока невозможно использовать воксельные данные из симулятора дыма. Кроме того, объемный рендеринг невозможен на GPU.

В системе отслеживания движений решена проблема ошибок солвера во время эпизодических потерь маркеров: теперь каждому маркеру можно назначить вес, который определяет его влияние на общее решение системы. Изменение веса проблемных маркеров может значительно улучшить качество отслеживания. Также трекер теперь использует новую систему распознавания характеристик.



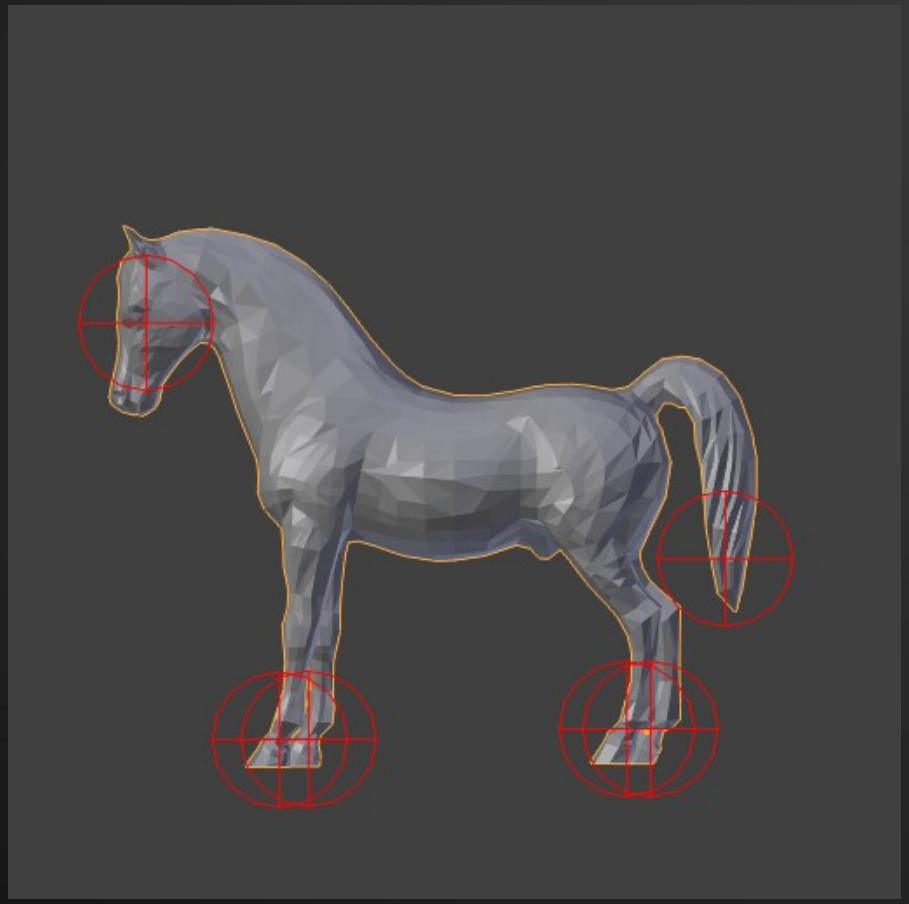
Значительные изменения коснулись пользовательского интерфейса: левая панель (Toolshelf) теперь использует вертикальные вкладки для организации кнопок-инструментов. Можно создавать новые вкладки при помощи Python-скриптов. Кроме того, можно синхронно изменять спаренные поля ввода чисел: кликните на первое поле, а затем потяните курсор вертикально.

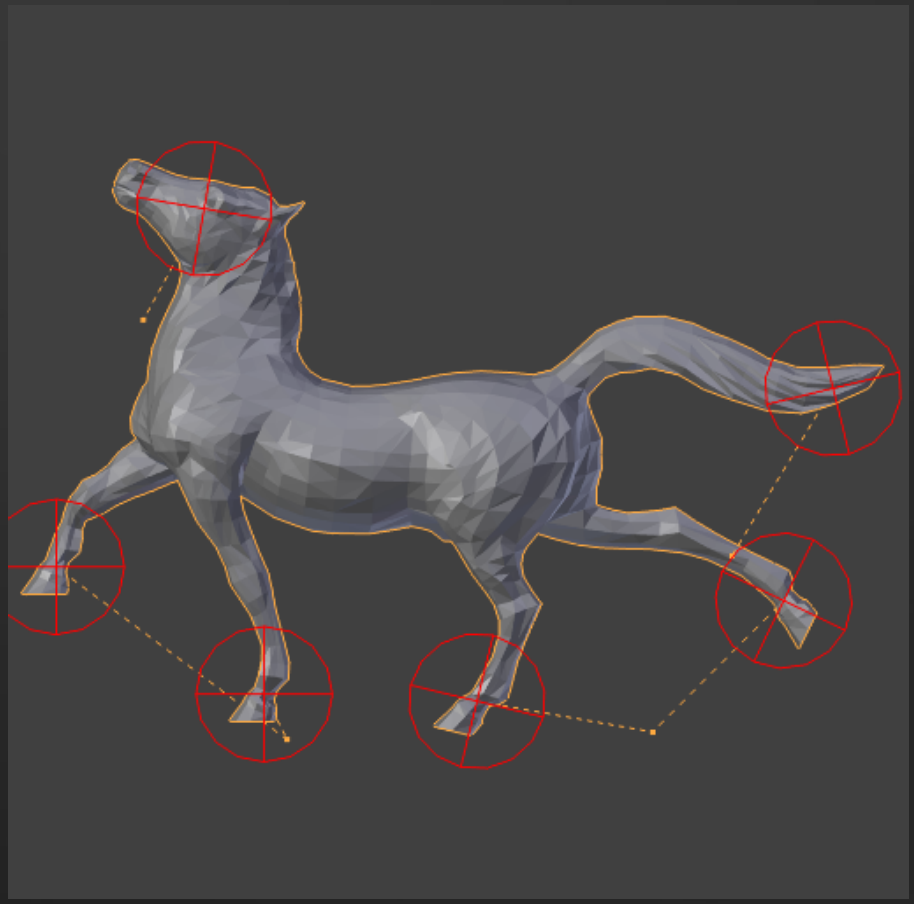


Добавлен новый модификатор деформации методом Лапласа (Laplacian Deform), позволяющий деформировать меш с сохранением геометрических деталей поверхности. Также добавлен каркасный модификатор (Wireframe), позволяющий преобразовать полигональную сетку в каркасное представление.

В игровом движке появилась поддержка LOD - раздельных уровней детализации для моделей. Проведена реорганизация API рендер-движка Freestyle. Из новых встроенных дополнений – интеграция Sketchfab (веб-платформы для публикации 3D-контента) и Node Wrangler – аддон, облегчающий работу в редакторе узлов.

Скачать Blender 2.70 для Windows, Linux и Mac OS X можно здесь: http://www.blender.org/download



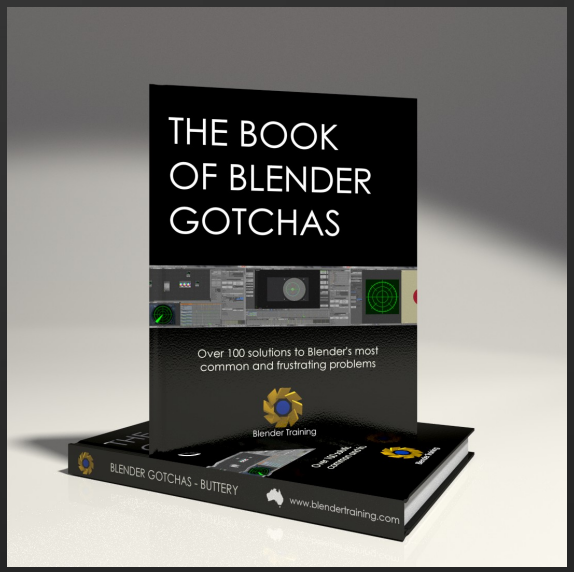


Вышла Gaminandes – игра по мотивам короткометражки «Caminandes: Gran Dillama»: игрок управляет героем фильма, ламой по имени Питти. В качестве движка используется GamePlay3D, графическая часть выполнена в Blender и GIMP. Есть версии для Android и Blackberry, как с рекламой (бесплатно), так и без ($1.99). Планируется также порт на Windows.

Sheep it! – бесплатная распределенная рендер-ферма для пользователей Blender. Загруженный проект разбивается на множество небольших подзадач, которые распределяются между участниками системы. Схема распределения задач проста: вы получаете кредиты за предоставление сервису своих вычислительных мощностей, и чем больше у вас кредитов, тем выше приоритет при рендеринге ваших проектов. За работу по рендерингу вашего проекта вы «платите» кредитами. При этом для «зарабатывания» кредитов даже необязательна установка Blender: вам нужна только виртуальная машина Java, на которой будет запущено клиентское приложение.

Sheep it!
Render Farm

Увидела свет электронная книга Эндрю Баттери «The Book of Blender Gotchas», которая раскрывает решения 100 самых распространенных проблем и вопросов, касающихся работы в Blender – так называемые «gotchas». Приобрести книгу можно здесь: https://blendertraining.com/products.

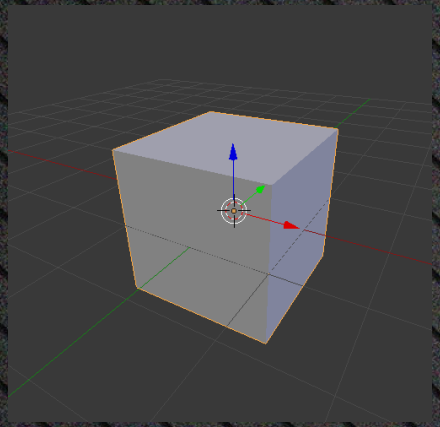


Журнал «FPS» отслеживает все самые свежие новости из мира Blender, моделирования, анимации и рендеринга! В следующем номере ждите очередную подборку новостей. Оставайтесь с нами и держите руку на пульсе последних событий!

Живопись полигонами

Новая техника NPR

В развитии трехмерной графики и анимации одной из важнейших современных тенденций является **NPR** – нефотореалистичный рендеринг. Мы уже писали о различных методах стилизации графики (FPS №19 '12 «Freestyle: первое знакомство», FPS №18 '12 «Эффект типографской печати на GLSL») – все они имеют дело с постобработкой. Однако мало кто знает, что существует несложная техника воссоздания акварельной или масляной живописи без использования явной постобработки, основанная на простом смещении полигонов.

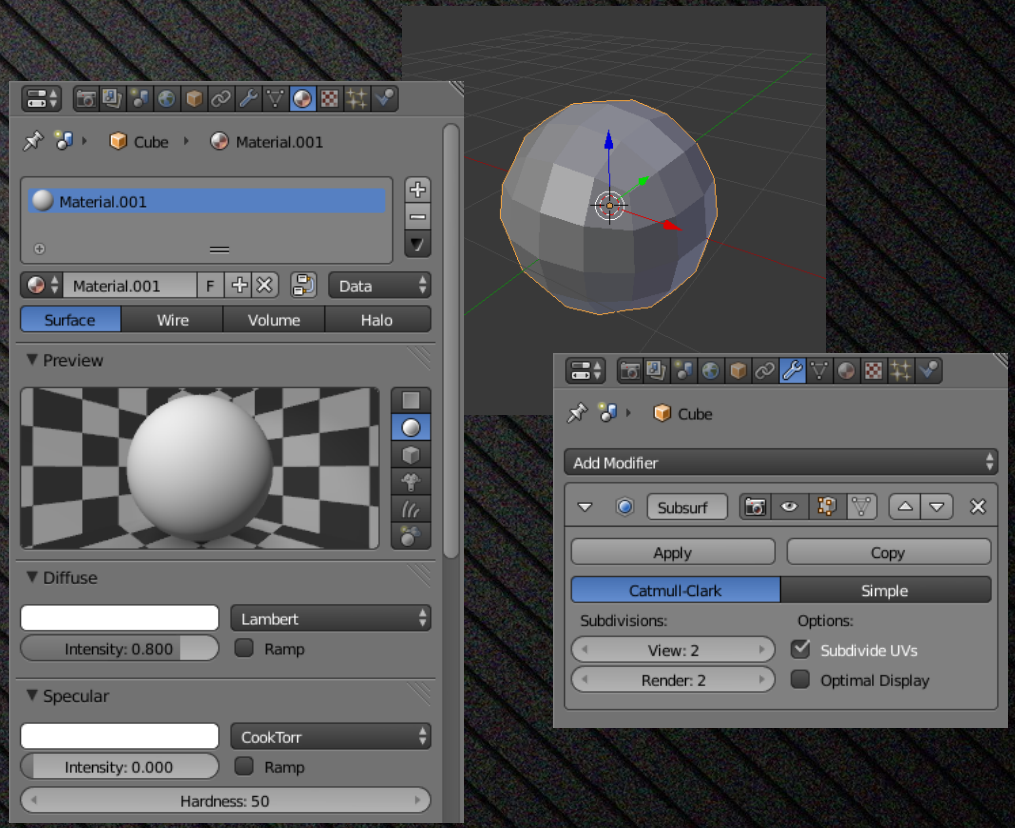


Эта техника получила название «закрашивание полигонами» (**Painting with Polygons**) и впервые была представлена на SIGGRAPH 2009 Айзеком Боткиным, автором книги «Outside Hollywood». Ее основная идея проста: если отрендерить несколько изображений сцены со слегка деформированной геометрией, а затем смешать эти изображения при помощи размытия при движении (в пределах одного кадра), то в результате получится эффект полупрозрачных мазков.

Со стилистической точки зрения, конечно, далеко не всякий мотив можно обыграть в этой технике: вряд ли она будет уместна в каком-нибудь футуристическом аниме, например. Предлагаю опробовать ее в одной из подходящих областей – в рендеринге неба и облаков.

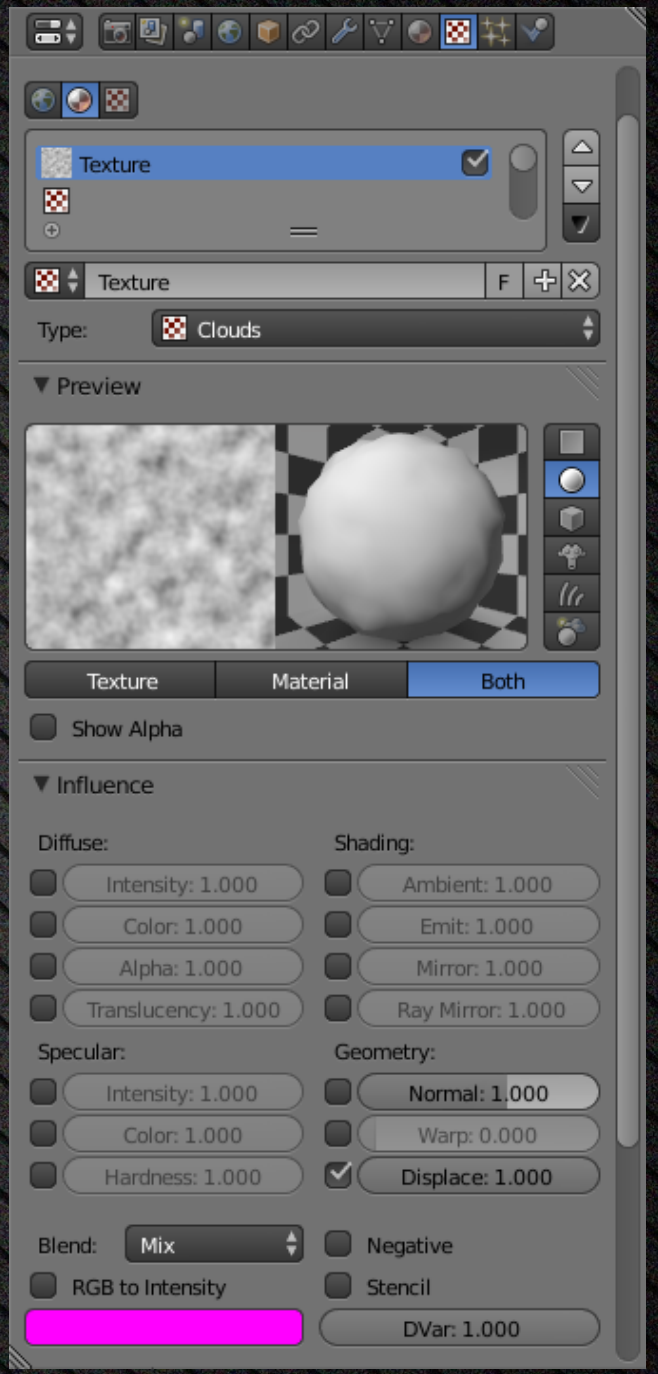
1. Создайте куб, примените к нему модификатор **Subdivision Surface**. Это будет базовый объект, от которого можно отталкиваться при создании облаков произвольной формы.

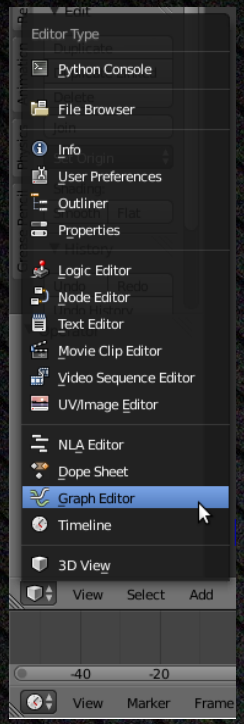
2. Создайте облаку новый материал. Уменьшите интенсивность бликовой компоненты до нуля.



3. Создайте для материала новую процедурную текстуру типа **Clouds**. На вкладке **Influence** уберите галочку напротив **Color** и поставьте напротив **Displace**. Коэффициент **Displace** выставьте равным 1. Этим мы задаем смещение вершин геометрии облака на основе процедурной карты высот.

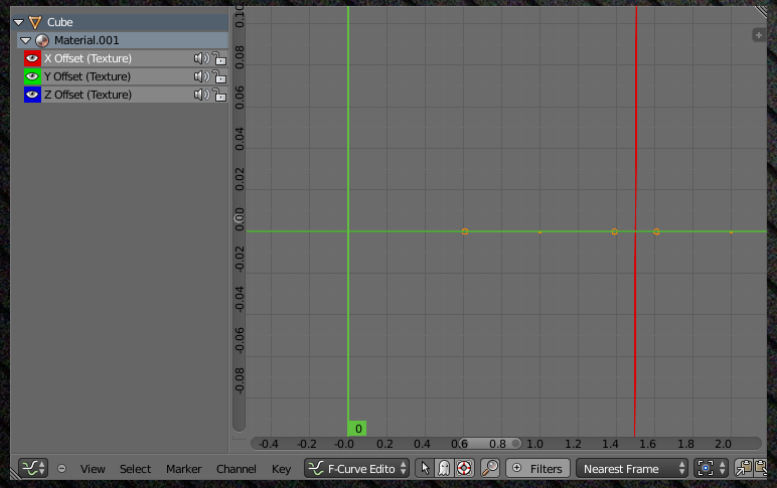
4. На вкладке **Mapping** выставьте X-координату **Offset** равной -10. Перед этим убедитесь, что текущий кадр на временной шкале – первый. Теперь наведите курсор на поле X и нажмите клавишу I, чтобы записать это значение как ключевой кадр анимации. Поле при этом станет желтым.

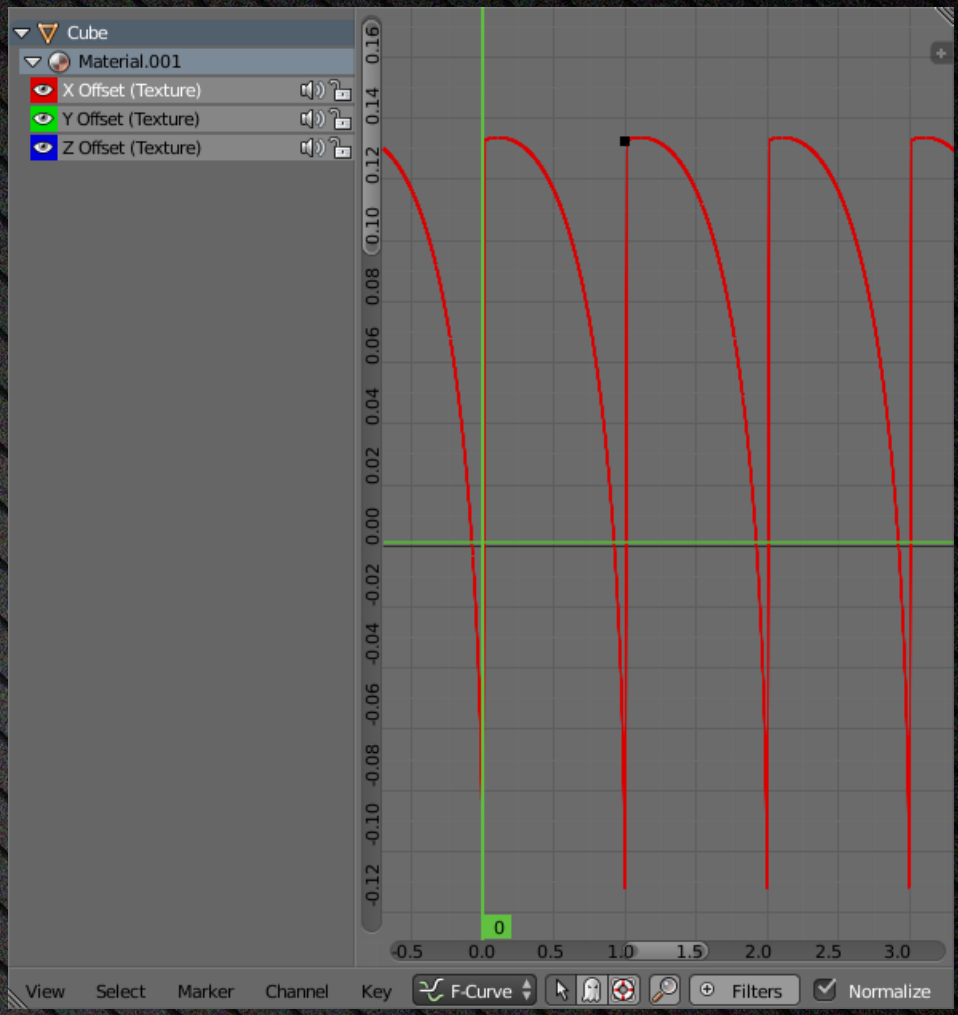




5. Теперь перейдите на один кадр вперед (клавиша-стрелка «Вправо») и измените значение X на 10. Снова запишите ключевой кадр. Смысл сделанного в том, что мы задаем два разных состояния смещения, между которыми будем интерполировать изображение.

6. Вернитесь на первый кадр и перейдите в редактор графов (Graph Editor). Выберите канал для X-координаты на панели слева. При помощи колесика мыши увеличьте маштаб графа. Вы должны увидеть примерно такую картинку, как на скриншоте.

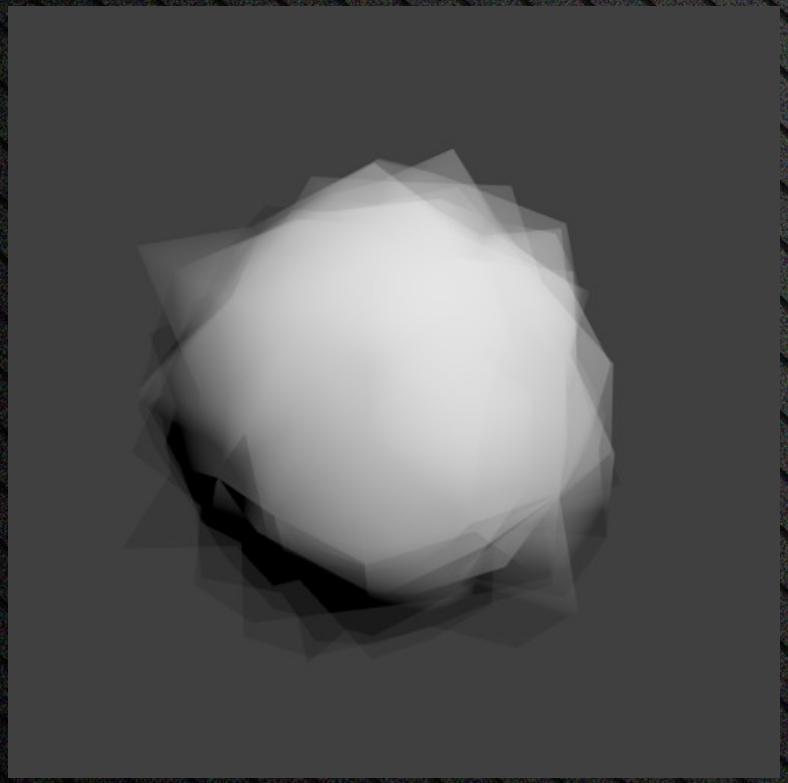




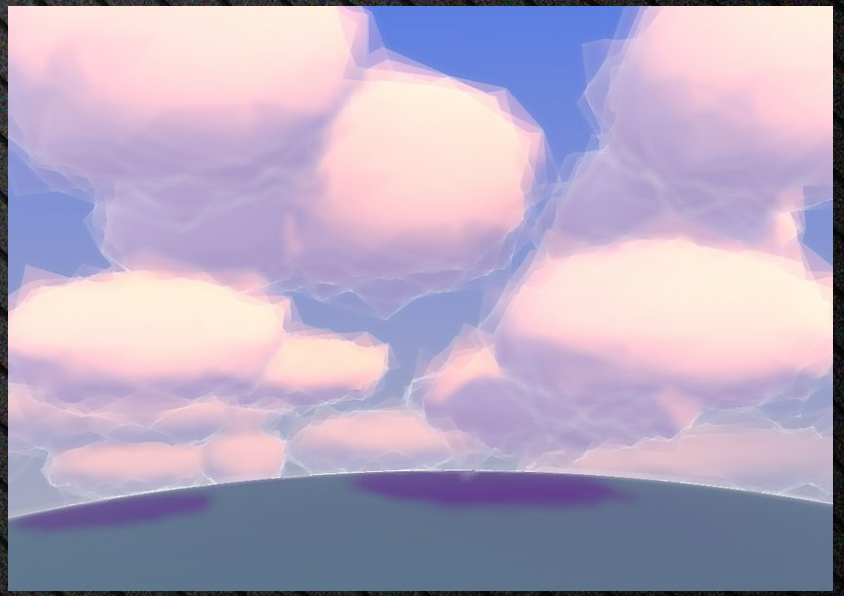
7. Путем редактирования контрольных точек графа, сформируйте такую же кривую, как на скриншоте. Затем выберите в меню Channel → Extrapolation Mode → Make Cyclic (F-Modifier).

8. В настройках рендеринга поставьте галочку на вкладке Sampled Motion Blur. Количество **Motion Samples** выставьте равным 5, а параметр Shutter – 0.2.

9. Если все сделано правильно, то, отрендерив сцену, вы должны увидеть что-то, похожее на пушистое облако.



10. Теперь можно настроить материал, освещение и параметры окружения, чтобы облако выглядело более реалистично.



Тимур Гафаров

Обзор дополнений Blender

Выпуск 8

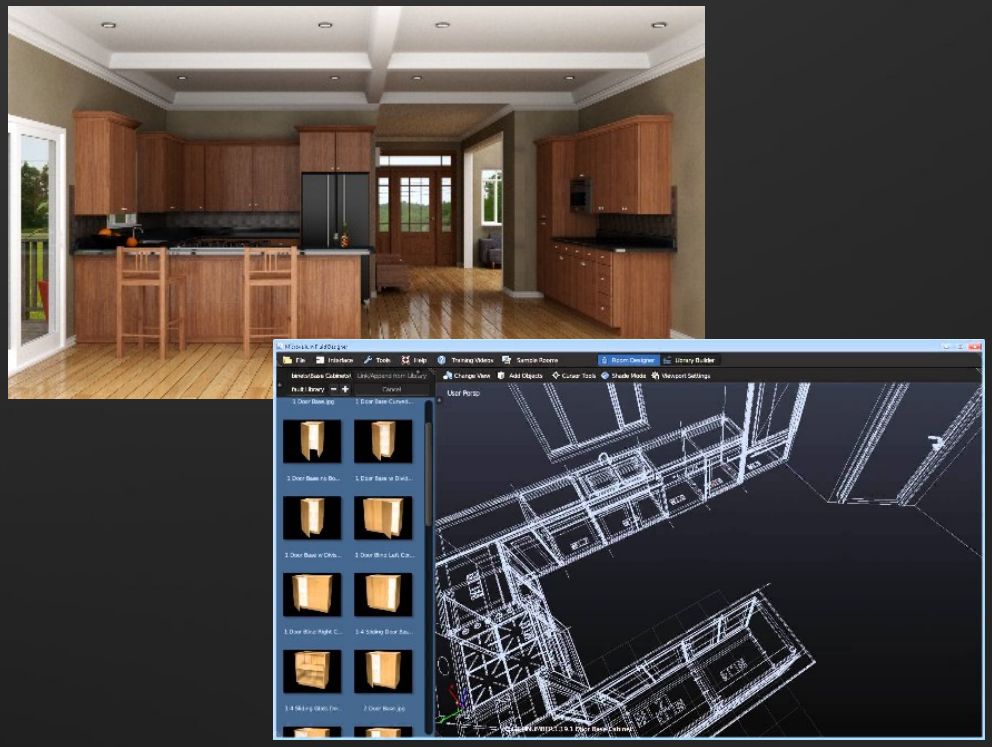
Благодаря удобному и мощному API для языка Python, Blender поддается практически неограниченному расширению. Наш журнал отслеживает выход новых полезных дополнений для Blender, которые могут заинтересовать пользователей, использующих программу в качестве инструмента для разработки игр или создания игрового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

Если вы разрабатываете собственное дополнение или просто нашли в Интернете чей-то интересный проект, будем очень рады, если вы напишете нам об этом и поделитесь ссылкой. Пишите на gecko0307@gmail.com, либо в наше сообщество: http://gplus.to/fps_community

В этом выпуске «Обзора дополнений» мы продолжаем начатый в FPS №28 '14 разговор о специализированных аддонах, которые будут полезны левелдизайнерам, архитекторам и дизайнерам интерьеров.

Fluid Designer

Несмотря на название, аддон не имеет никакого отношения к симулятору жидкостей :) На самом деле, это готовящаяся к выходу интегрированная среда (CAD-система) для дизайнеров интерьеров, основанная на Blender. Fluid Designer включает менеджер параметрических объектов, которыми представлены предметы мебели и детали интерьера – в комплекте с продуктом будет поставляться большая база готовых объектов. Как обещают разработчики, Fluid Designer будет доступен под GNU GPL.



Разработчик: Microvellum

Сайт проекта

BookGen

Еще одно отличное дополнение для дизайнеров интерьеров: BookGen позволяет процедурно генерировать ряды книг, стоящих на полке. Можно контролировать ширину, высоту и глубину ряда, толщину томов и другие параметры. Можно также сгенерировать книги случайных размеров. Дополнение автоматически формирует UV-развертку генеруемых мешей.

Разработчик: Оливер Вайссбарт

http://oweissbarth.de/software/book-gen-blender-addon

Randomiser

Дополнение Randomiser реализует эффект скрэмблинга для объектов текста – то есть, подстановку случайных символов в строке фиксированной длины. Что самое интересное – результат можно анимировать и получить эффект «дешифровки»: постепенного подбора правильных букв из постоянно меняющегося хаоса символов. Естественно, необходимо при этом использовать моноширинную гарнитуру.

Разработчик: Бен Симондс

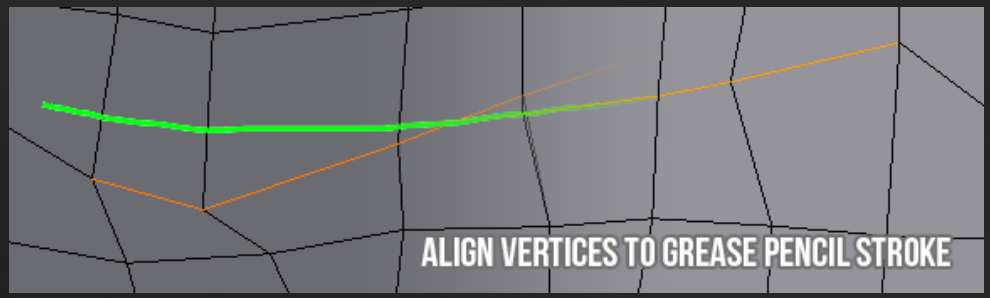
http://bensimonds.com/2014/04/02/randomiser-add-on

Align to Pencil

Это дополнение полезно при моделировании: оно позволяет выровнять выделенные вершины меша по последнему штриху Grease Pencil, что удобно для быстрого исправления топологии.

Разработчик: Бьёрнар Фрёйс

Ссылка на скачивание



Язык D

Новости «с Марса»

свежие релизы и обновления

LDC + iOS

Разработчикам компилятора LDC удалось сделать возможным написание приложений на D для iOS. Опубликован форк компилятора, способный собирать работающие приложения для этой ОС. Пока проект находится в статусе альфа и имеет много недочетов, однако, учитывая заинтересованность публики в языке, способном покрыть все мобильные платформы, у него большое будущее.

https://github.com/smolt/ldc

D/Objective-C

Разработчики D/Objective-C – расширения D, которое позволяет работать с объектами Objective-C – объявили о полной поддержке 64-битных систем.

https://github.com/jacob-carlborg/dmd/tree/d-objc

Flint от Facebook

Через некоторое время после новости о том, что Facebook использует D в своей инфраструктуре, компания порадовала сообщество языка открытым проектом, выпустив в свободный доступ исходники Flint – анализатора исходного кода на C++, написанного на D.

https://github.com/facebook/flint

YLink – новый мультиформатный линкер

Поддержка 64-битных систем компиляторами DMD и LDC породила потребность в универсальном линкере, совместимом с форматами COFF и OMF. На сегодняшний день ситуация с линкером под Windows удручающая: 32-битная версия DMD выдает объектные файлы формата OMF и для компоновки использует старый как мир линкер Digital Mars – Optlink, написанный на ассемблере, изначально еще под DOS. 64-битные же генерируют объектники формата COFF и требуют проприетарный линкер из MS Visual Studio. Под Linux, к слову, такой проблемы никогда не существовало, так как там используется формат ELF и, соответственно, GNU-линкер ld.

И в такой подходящий момент появился проект YLink по созданию нового линкера для D с нуля. Он включает поддержку COFF и OMF и доступен по лицензии Boost. Правда, сейчас проект находится в экспериментальной стадии – еще не реализованы многие возможности, в том числе отладочная информация и поддержка DLL. Но перспектива у YLink определенно есть – будем надеяться, что когда-нибудь он придет на замену устаревшему Optlink, и ужасно раздражающие спонтанные ошибки компоновки можно будет забыть, как страшный сон.

https://github.com/yebblies/ylink

Dvorm

Проект по созданию ORM для D. ORM (Объектно-реляционное отображение) – это технология программирования, которая связывает базы данных с концепциями ООП, создавая виртуальную объектную БД.

https://github.com/rikkimax/Dvorm

DerelictOrg

Майк Паркер (aka aldacron), разработчик Derelict, объявил о прекращении поддержки ветки Derelict3: актуальной веткой проекта теперь считается DerelictOrg. Нумерации мажорных версий также не планируется.

Напомним, Derelict – это коллекция кроссплатформенных динамических привязок для мультимедийных библиотек с C-интерфейсом. В настоящий момент проект DerelictOrg включает привязки к OpenGL 3.x/4.x, SDL2, SFML2, ALURE, OpenAL, ODE, FreeImage, ASSIMP3, FreeType, GLFW3, DevIL, OGG/Vorbis, PhysFS, PQ и Lua. В Интернете также существует множество «неофициальных» привязок, использующих API Derelict – процесс создания привязки очень прост, ее может создать любой желающий. Derelict – незаменимый инструмент для разработки игр на D, многократно облегчающий взаимодействие игры со сторонними библиотеками.

https://github.com/DerelictOrg

dlib 0.3

Состоялся релиз коллекции библиотек dlib 0.3. Из нововведений этой версии стоит отметить абстрактные потоки ввода/вывода (dlib.core.stream), независимые от Phobos, а также интерфейс файловой системы (dlib.filesystem) с готовыми реализациями для POSIX и Windows. Также проведено большое количество улучшений в dlib.image, добавлена поддержка HDRI, распараллеливание обработки изображений и форматы TGA и BMP. Кроме того, проведен рефакторинг dlib.math.matrix: элементы матриц теперь располагаются по столбцам, а не по строкам.

Напомним, dlib – это коллекция библиотек общего назначения, пригодных для создания игровых движков, графических редакторов и конвейеров, а также для других мультимедийных и вычислительных задач. dlib полностью самодостаточна и не имеет внешних зависимостей.

https://github.com/gecko0307/dlib
https://github.com/gecko0307/dlib/releases/tag/v0.3.0
https://github.com/gecko0307/dlib/wiki

GScript

GScript – минималистичный скриптовый язык для D. Это императивный язык с динамической типизацией, поддержкой данных типа Float и String, а также гетерогенных массивов. Реализована модульная система, рекурсивные функции, передача аргументов по значению и по ссылке, цикл foreach, UFCS, проверка типов, строки с полной поддержкой Юникода (UTF-8). Есть небольшая стандартная библиотека, а также возможность расширять язык собственными функциями на D. В будущем планируется поддержка ООП.

Кодогенератор и виртуальная машина к языку пока находятся на стадии прототипа – реализация полностью рабочая, но далека от оптимальной.

Исходники проекта распространяются по лицензии Boost.

https://github.com/gecko0307/gscript

GladeD

Эта библиотека пригодится тем, кто много работает с GtkD – она генерирует D-классы из файлов Glade, содержащих описание интерфейса в XML-представлении. Это удобно тем, что не нужно тащить с приложением Glade-файл – вся информация о интерфейсе будет встроена в программу.

https://github.com/burner/gladeD

Cerealed 0.4.1

Библиотека Cerealed обновилась до версии 0.4.1. Этот релиз включает поддержку указателей, диапазонов (InputRange и OutputRange), а также другие мелкие улучшения и багфиксы. Напомним, Cerealed – это библиотека бинарной сериализации, альтернатива таким решениям, как Orange и std.serialization.

https://github.com/atilaneves/cerealed

DAuth

Анонсирована DAuth – система аутентификации для D, основанная на соленом хэше. Библиотека сама по себе не включает криптографические алгоритмы – вместо этого она способна использовать любые совместимые с интерфейсами Phobos хеш-функции и генераторы случайных чисел, так что надежность и криптостойкость системы, в конечном счете, засисит от реализации в проекте.

https://github.com/Abscissa/DAuth

Vibe.d 0.7.19

Веб-фреймворк Vibe.d обновился до версии 0.7.19. Данная версия влючает валидатор сертификата SSL, улучшения компилятора шаблонов Diet и клиента Redis, а также поддержку последней версии DMD.

http://vibed.org

Бета-версии DCD 0.3.0 и DScanner 0.1.0

Доступна бета-версия сервера автозаполнения кода DCD 0.3.0. В данном релизе используется реализация AST, лексический анализатор и парсер из проекта DScanner.

Напомним, DCD – это демон автодополнения, совместимый практически со всеми IDE и текстовыми редакторами, где есть поддержка плагинов или скриптов. На данный момент DCD работает с Textadept, Kate/KDevelop, Vim, Emacs и Zeus.

https://github.com/Hackerpilot/DCD/tree/0.3.0-beta1
https://github.com/Hackerpilot/Dscanner/tree/0.1.0-beta1

Lumen + KDE

Разработчики Lumen сообщили, что проект официально войдет в состав окружения KDE 4.13.

Напомним, Lumen – это плагин автозаполнения для компонента KTextEditor, работающий в KDE-шных IDE и текстовых редакторах (таких, как Kate и KDevelop). Плагин основан на сервере автозаполнения DCD, обеспечивает поддержку автодополнения кода, семантического анализа и отладки (есть интеграция с GDB).

Плагодаря этой связке становится возможным использовать KDevelop для разработки приложений на D.

https://github.com/Dav1dde/lumen

DDT 0.10.0

Вышла новая версия DDT 0.10.0. Релиза включает поддержку DUB, а также многоличленные улучшения пользовательского интерфейса.

Напомним, DDT (D Development Tools) – это плагин поддержки D для среды разработки Eclipse.

https://github.com/bruno-medeiros/DDT/releases/tag/Release_0.10.0
http://code.google.com/p/ddt
http://www.eclipse.org

Mono-D 1.8

Mono-D – плагин для среды MonoDevelop/Xamarin Studio, предоставляющий поддержку языка D в этой IDE – обновился до версии 1.8. Этот релиз носит исправляющий характер и включает, в основном, внутренний рефакторинг. Из значимых нововведений стоит отметить выборочную подсветку кода: например, блоки else для version (Windows) не подсвечиваются под Windows, так как не игнорируются компилятором.

http://mono-d.alexanderbothe.com

GScript: просто скриптовый язык

Парадоксально: языков программирования существует великое множество, но их почему-то всегда не хватает. Всегда найдется предметная область, в которой существующие языки плохо применимы, и программисты придумывают еще один, чтобы компенсировать это досадное упущение. Или бывает так, что в старых языках становится трудно выразить какие-то только что изобретенные идеи и концепции – и создается новый язык, в котором эти концепции «встроены» изначально.

Поэтому я никогда не понимал тех, кто считает, что языков уже достаточно, и что создавать новые – это «велосипедостроение». Языков никогда не бывает достаточно, потому что нет ни одного языка, который бы покрывал абсолютно все задачи и потребности.

Такая ситуация возникла и у меня, когда понадобился скриптовый язык, который можно было бы использовать в D. Поскольку для этого реализация языка должна иметь C-интерфейс, выбор был невелик:

1. **Lua** или **Python.** Я уже имел опыт использования Lua в C++, поэтому этот вариант казался самым очевидным. Для Lua существует динамическая D-привязка через Derelict, а также враппер LuaD, обеспечивающий тесную интеграцию языка с системой типов D. Выглядело это все достаточно привлекательно, но, во-первых, LuaD не имеет поддержки последних версий Lua, а, в во-вторых, у меня свое собственное видение того, как должна работать скриптовая система.

Программист должен иметь возможность загружать много отдельных скриптов из различных файлов, и вызывать из них те или иные функции. Скрипты должны быть многоразовыми, а не просто «загрузил скрипт, выполнил, завершил работу». К сожалению, Lua не очень соответствует этим идеям, как и все остальные существующие скриптовые языки: все они тянут одеяло на себя, претендуя на роль основного средства вычислений в программе. Разработчики Python и вовсе не приветствуют встраивание языка в приложение: они рекомендуют писать на Python все приложение целиком.

2. **JewelScript**. Это минималистичный язык из семейства фигурных скобок со статической типизацией, чем-то напоминающий D и C++. Как и в случае с Lua и Python, тут тоже имеется C-шный API, так что мне почти сразу удалось привязать JewelScript к D, и поначалу все казалось идеальным. Однако API этот оказался переусложненным и запутанным – элементарно добавить в язык новую встроенную функцию без построения для этого отдельного класса тут невозможно, а добавление класса, в свою очередь – целая эпопея... Упростить и автоматизировать этот процесс при помощи каких-то «умных» шаблонов мне так и не удалось.

Тут прояснилось еще одно требование к скриптовой системе: удобный API, позволяющий взаимодействовать с D. Трудно ожидать такого от C-библиотек, при взаимодействии с которыми для передачи указателя на функцию нужно предварительно оборачивать ее в extern(C) – в таких условиях можно забыть о замыканиях и ООП.

Этому требованию полностью может удовлетворить только язык, сам написанный на D. Но, к сожалению, на тот момент (и на момент написания данной статьи) полноценных скриптовых языков на D2 не существовало (для D1 были языки Monster и Croc – автор последнего упорно не желал переписывать свой проект на D2).

Также мне попадалась реализация JavaScript, написанная на D2, но заброшенная и неподдерживаемая – последними версиями DMD ее уже не скомпилировать. Поэтому не оставалось ничего другого, как создать свой собственный «идеальный язык» с нуля. Так и появился GScript.

Я сторонник C-подобных языков (C, C++, C#, D, Java, JavaScript), поэтому решил реализовать нечто похожее.

Язык реализует императивную, процедурную и модульную парадигмы. Модуль – это файл, в котором определены функции, которые пока являются главными структурными элементами языка. Глобальные переменные языком не поддерживаются – я считаю использование глобальных переменных дурным тоном. При крайней необходимости, аналогичную функциональность можно легко реализовать при помощи функций-примитивов (например, путем взаимодействия с БД или хэш-таблицей).

Вот пример программы «Hello, World» на GScript:

```
func main()
{
    writeln("Hello, World!");
}
```

Типизация – динамическая: тип переменной зависит от присвоенного ей значения. Переменные необходимо объявлять при помощи ключевого слова «var». Вот пример использования локальной переменной:

```
func main()
{
    var x = 100;
    writeln(x);
    x = "hello";
    writeln(x);
}
```

Язык поддерживает следующие внутренние типы: Float (32-битное число с плавающей запятой), String (строка в кодировке UTF-8), Array (массив). Все массивы – гетерогенные: их элементами могут быть данные любых типов, включая другие массивы различной длины.

```
func main()
{
    var x = [1, 4.5, "hello", [5, 6, 7]];
    writeln(x[2]);
}
```

Работе с массивами в языке уделено особое внимание. Например, есть синтаксический сахар для упрощения индексирования многомерных массивов (т.е., массивов, элементами которого являются другие массивы):

```
func main()
{
    var x = [
        [5, 6],
        [2, 7]
    ];
    writeln(x[0][1]);
    writeln(x[0, 1]);
}
```

В данном случае x[0, 1] - то же самое, что x[0][1].

Следуя принципам минимализма, я старался изначально не переусложнять дизайн языка: к примеру, динамическая типизация упростила виртуальную машину – и, вместе с этим, позволила языку бесшовно взаимодействовать со встроенными типами D. Массивы GScript – это обычные массивы D, которые создаются менеджером памяти D и подконтрольны сборщику мусора D. И, чтобы их создавать и ими управлять, не нужно пользоваться какими-то особыми средствами – конкатенация массивов в GScript делается конкатенацией массивов в D, нет никаких лишних уровней абстракции.

```
func main()
{
    var x = [1, 2];
    x ~= 3;
    x ~= [0, 6]
}
```

Функции могут принимать аргументы и возвращать значения.

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

Строго говоря, функция всегда возвращает значение, даже если в ней не указан оператор return: в данном случае компилятор автоматически вставляет в конец функции возврат значения по умолчанию (Null). Но возвращаемое значение можно не использовать, вызывая функцию как процедуру.

По умолчанию аргументы передаются по значению, но можно использовать передачу по ссылке. Что характерно, передача по ссылке возможна в любую функцию, так как ссылочный тип (ref) указывается при конкретном вызове функции, а не при ее объявлении:

```
func sqr(x)
{
    x *= x;
    return x;
}

func main()
{
    var n = 10;
    sqr(ref n);
    writeln(n); // n будет равно 100
}
```

Сама функция про ссылку ничего не знает и знать не обязана (это один из принципов языка – функция не должна ничего знать о контексте, в котором она вызывается). Вы можете вызвать sqr с передачей аргумента по значению, и ничего особенного не произойдет:

```
func main()
{
    var n = sqr(10);
    writeln(n);
}
```

Эта особенность языка сугубо экспериментальная: я отдаю себе отчет в том, что такое до меня нигде реализовано не было, и в каких-то ситуациях такой подход может повлечь проблемы. Но с такими ситуациями я пока не встречался, а выгоды от использования ref при вызове могут быть весьма велики.

GScript пока не включает ООП, но в нем поддерживается синтаксис вызова функции как метода – для этого используется оператор «:». Именно так работает функция-примитив length, возвращающая длину массива:

```
func main()
{
    var arr = [1, 2, 3]
    writeln(arr:length);
    writeln(length(arr));
}
```

Здесь length(arr) и arr:length – одно и то же.

Кстати, к длине массива в индексе предоставляет доступ оператор «$», как в D:

```
func main()
{
    var arr = [1, 2, 3]
    arr[$-1] = 0;
}
```

В комплекте с языком идет небольшая, но растущая стандартная библиотека, в которой имеются функции манипуляции массивами:

```
import std.array;

func main()

{
    var arr = [7, 2, 10, 3, 5, 1];
    arr:reverse(); // инверсия массива
    arr:sort(); // сортировка массива

   /*
    * Функция flatten создает из
    * многомерного массива одномерный.
    */

    var arr2 = flatten(
        [2, 3, [0, 1], [[3, 6], [2]]]);
    assert(arr2 == [2, 3, 0, 1, 3, 6, 2]);
/*
    * Функция table возвращает
    * многомерный массив
    * с заданными размерностями
    * (по сути, матрицу или таблицу).
*/

    var arr3 = table(2, 2);
    assert(arr3 == [[0, 0], [0, 0]]);
}
```

Строки GScript полностью поддерживают Юникод и кодируются в UTF-8. Вы можете обращаться к строке поэлементно, как к массиву (при этом символы длиной более 1 байта обрабатываются корректно). Функция length по отношению строкам также возвращает не фактическую длину байтового массива, которым внутренне представлена строка в системе типов, а именно длину строки в символах.

```
func main()
{
    var str = "abcçйцъ";
    assert(x[6] == "ъ");
    assert(str:length == 7);
}
```

Строки, формально, массивами не являются, но их точно так же можно конкатенировать и совершать над ними другие характерные для массивов операции. Единственная особенность: как и в D, все строки неизменяемы – модифицировать произвольный символ в строке невозможно. С точки зрения реализации, это затруднительно хотя бы из-за неодинакового размера символов в UTF-8.

Императивная часть языка полностью завершена, но на этом работа не закончена: в данный момент идет разработка стандартной библиотеки, а также дизайн будущей объектной системы. Так что в ближайщем будущем ждите много новых материалов по GScript на страницах журнала!

Тимур Гафаров

Наши проекты

Cook

Программа автоматизации сборки проектов на языке D. В отличие от аналогичных инструментов (Make, CMake, Scons, Jam, DSSS и др.), Cook не требует конфигурационного файла: всю информацию о проекте она получает самостоятельно, сканируя модули (файлы *.d). При этом программа отслеживает прямые и обратные зависимости между модулями: если модуль был изменен, необходимо скомпилировать заново не только его, но и все модули, которые от него зависят (это важно, если был изменен внешний интерфейс модуля: объявления классов, семантика шаблонов и т.д.). Для этого Cook производит лексический анализ модулей - но не всех, а только тех, которые были изменены со времени последнего анализа. Данные анализа кэшируются в файл для повторного использования (кэш автоматически обновляется при пересборке). Cook работает в Windows и Linux.

http://github.com/gecko0307/cook
http://github.com/gecko0307/cook2

dlib

Коллекция библиотек «на все случаи жизни» для D, которая может быть использована в игровых движках и других мультимедийных приложениях. Написана на D2 с использованием Phobos, не имеет никаких других внешних зависимостей. Разработка dlib пока находится на ранней стадии - API нестабилен и может измениться в любой момент, если появится возможность улучшить общую архитектуру.

http://github.com/gecko0307/dlib

Уолтер Брайт: История D

Недавно D вновь вошел в двадцатку самых популярных языков программирования в рейтинге TIOBE. Не в последнюю очередь это связано с Facebook и ее проектами на D, но мы сейчас не будем рассуждать о том, насколько язык востребован в бизнесе, а вместо этого оглянемся назад и совершим небольшую ретроспективу истории D.

Создатель D – Уолтер Брайт, американский специалист по разработке компиляторов с многолетним стажем. Он известен тем, что разработал один из первых оптимизирующих компиляторов Си – Datalight C (впоследствии – Zortech C), а также первый нативный компилятор C++, генерирующий машинный код (Zortec C++, впоследствии – Symantec C++, ныне – DigitalMars C++). Кроме того, Брайт является автором классической игры Empire (1977 г.) – прародителя таких культовых стратегий, как Civilization, Global Conquest и Xconq.

Не так давно Уолтер Брайт опубликовал статью в «Dr. Dobb's», в которой поделился своими воспоминаниями о ранних этапах своей карьеры программиста – о том, что предшествовало созданию D.

«В начале 80-х я в составе команды программистов разрабатывал ПО под MS-DOS. Мы использовали C, так как это был единственный рабочий высокоуровневый язык для PC. Реализации других языков были невероятно отвратительны. Впрочем, качество компиляторов C тоже оставляло желать лучшего – но, по крайней мере, ими можно было пользоваться. Они генерировали ужасный код – оптимизация как таковая отсутствовала. И у меня появилась идея, что я мог бы написать компилятор получше...»

Впоследствии для IBM PC появилось множество различных компиляторов C – Брайт в свое время насчитал их около 30 штук. Он стал искать области с конкурентным преимуществом и наткнулся на книгу Бьярна Страуструпа по C++.

Разработка C++ началась к конце 70-х – изначально это был эдакий «Си с классами». Он так и назывался – C with Classes. Страуструп считал, что классические ООП-языки вроде Simula содержали весьма интересные для бизнеса идеи, но были слишком неудобны на практике. Основываясь на своей кандидатской диссертации, он решил дополнить C объектными возможностями Simula. В 1983 году язык был переименован в C++, и помимо собственно ООП, в нем было реализовано множество новых концепций – виртуальные функции, перегрузка операторов, ссылки, константы, проверка типов и т.д.

«Я подумал: «Эй, а ведь стоит только добавить несколько новых ключевых слов, и через пару месяцев в моем компиляторе будет поддержка C++!» Если бы я тогда знал, во что я ввязываюсь, я бы, наверное, передумал...

Я проработал над компилятором C++ более 15 лет. И, конечно, за это время у меня не могли не появиться идеи по улучшению языка. Да, существуют различные расширения C и C++, но они не получили сколько-нибудь широкого распространения. И это объяснимо: всем нужен язык, строго следующий стандарту. Но, к сожалению, внесение чего-то нового в стандарты языков – это длительный и трудоемкий бюрократический процесс. А я терпеть не привык...

В 1999 году я вышел на пенсию и шесть месяцев ничего не делал, только смотрел телевизор – чуть с ума не сошел. Мне захотелось снова вернуться к работе. Мне надоело терпеть недостатки существующих языков, а тут как раз подвернулось время, чтобы начать что-то делать в этом направлении. Сталкиваясь с подобной задачей, я всегда вспоминаю фразу гнома Гимли из «Властелина колец»: «Верная смерть. Никаких шансов на успех. Так чего же мы ждем?!»

Действительно, почему бы и нет? Так и началась история нового языка, который известен как D...

Первый релиз D в Интернете состоялся в декабре 2001 года. Оказалось, что я не единственный в своих требованиях к идеальному языку программирования: у меня появились сотрудники со всего мира. Именно это я считаю главной особенностью интернет-революции: вы можете успешно работать с людьми, даже не будучи знакомым с ними лично. Таким образом, существование D было бы невозможно без Интернета – как иначе разрозненные энтузиасты могли бы объединиться вместе?..»

Брайт признается, что предпочитает работать в одиночку: это даже неоднократно подчеркивалось в его служебных характеристиках. Именно эта особенность, прямо или косвенно, послужила причиной появления двух стандартных библиотек D – Tango и Phobos.

На самом деле, «официальная» стандартная библиотека у языка всегда была одна - Phobos. Но в начале «нулевых» она находилась в очень сыром состоянии - фактически, ее разрабатывал всего один человек, сам Уолтер Брайт. Однако он был большей частью занят улучшением компилятора, и физически не мог успевать доводить библиотеку до совершенства.

Поэтому группа энтузиастов из сообщества языка взяла на себя задачу написать альтернативную библиотеку, которая содержала бы все необходимые программистам функции. Так появилась Tango.

Она была куда лучше, чем тогдашняя Phobos – и в скором времени стала де-факто главной стандартной библиотекой D1. Однако в то время она не была совместима с рантаймом Phobos, и поэтому две библиотеки нельзя было использовать одновременно – приходилось выбрать что-то одно. Это породило множество препятствий для распространения языка. Digital Mars официально не поддерживала Tango, ее необходимо было устанавливать самостоятельно.

И, тем не менее, многие перспективные проекты, включая библиотеки для разработчиков, использовали именно Tango, и полностью игнорировать ее было нельзя. А библиотеки, которые «из коробки» работали и с Tango, и с Phobos, можно было пересчитать по пальцам. Это был просто чудовищный антипиар для D...

Ситуация начала меняться к лучшему, когда к проекту присоединился знаменитый программист Андрей Александреску. Он довел Phobos до ума, используя новейшие парадигмы и собственные исследования в области компьютерных алгоритмов. В это время как раз шла разработка второй ветки языка, и Phobos стала первой и единственной стандартной библиотекой для D2. Она остается таковой по сей день.

В то же время, Tango впоследствии была портирована на D2 и теперь она совместима с Phobos – при желании обе библиотеки можно использовать одновременно. К Phobos и Tango можно относиться как, например, к STL и Boost в C++.

Первая версия D2 увидела свет в 2007 году. С тех пор несколько лет вокруг D не утихал злорадный смех: как можно надеяться на какое-то широкое признание, если у языка есть две параллельно существующие несовместимые версии? При этом все как-то умалчивают о том, что, вообще-то, у Python этих версий целая куча (2.5, 2.6, 2.7, 3.x), и долгое время любой пользователь Linux имел у себя в системе по три, а то и по четыре установленных версии языка. В этой связи претензии к D с его двумя четко разграниченными версиями выглядят просто смешно.

В 2010 году произошло еще одно важное событие в истории D: состоялся выход книги Андрея Александреску «Язык программирования D», сразу ставшей абсолютным бестселлером. Многие считают ее одной из лучших книг по программированию вообще. С этого момента принято считать D2 стабилизировавшимся – сейчас это основная и, по сути, единственная версия D. Разделения на D1 и D2 уже не существует.

Сейчас, впрочем, ходят слухи о D3: в новой версии языка планируется поддержка AST-макросов, которые позволят расширять синтаксис и вручную добавлять в язык новые возможности.

Игровые новости из мира Linux

Весна этого года ознаменовалась серьезными анонсами: несколько гигантов игрового middleware выпустили версии своих движков для Linux.

Первой была компания Crytek: в середине марта она объявила о выходе Linux-порта популярного игрового движка CryEngine, на основе которого построены такие игры, как Crysis, Far Cry, Warface и Ryse: Son of Rome. В рамках конференции Game Developers Conference была продемонстрирована новая версия движка CryEngine, которая, кроме нативной поддержки платформы Linux, также включает принципиальной новую систему рендеринга, использующую метод шейдинга на основе физических процессов (Physically Based Shading).

Вслед за этим, Epic Games выпустила обновление своего движка Unreal Engine 4.1, примечательное реализацией поддержки Linux – и, в частности, основанной на Debian GNU/Linux операционной системы SteamOS.

Кроме того, компания с недавних пор предоставляет доступ к исходному коду UE4 всем желающим. Актуальный код движка был размещен на GitHub – правда, репозиторий не публичный и доступен только авторизованным пользователям, которые являются подписчиками специального сервиса. Цена вопроса – всего $ 19 в месяц: вместо того, чтобы целиком лицензировать движок, вы платите только за временный полный доступ ко всем компонентам продукта, включая среду Unreal Editor, примеры контента, шаблоны готовых игр и полные исходные тексты движка на языке C++.

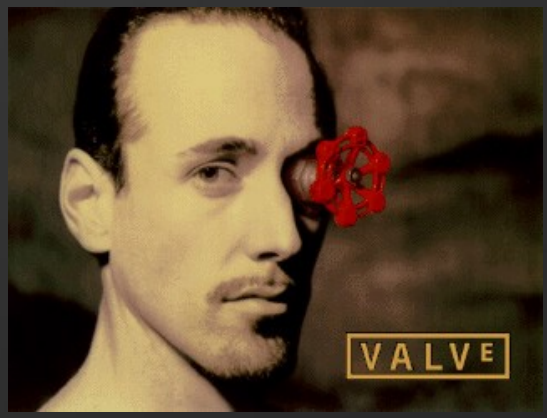
Код движка может использоваться в любых коммерческих проектах, но с разработчиков данных проектов взимаются отчисления в размере 5% от дохода, полученного от продажи игры. Сбор отчислений с доходов от продажи делает движок Unreal Engine 4 интересным решением не только для крупных игровых проектов, но и для небольших стартапов и авторов бесплатных игр.

Движок доступен для платформ Windows, Linux, Mac OS X, iOS и Android. Также Epic Games совместно с проектом Mozilla работают над созданием HTML5-варианта движка UE4, позволяющего создавать на нем браузерные 3D-игры, без использования внешних плагинов: код игры и движка на C++ при помощи компилятора Emscripten преобразуется в JavaScript с расширениями статической типизации Asm.js, а для вывода графики используется WebGL.

Что еще портировано на Linux? Недавно было объявлено о выпуске Linux-версии System Shock 2 – игры в стиле киберпанк, разработанной студиями Irrational Games и Looking Glass Studios. Она была выпущена 7 июля 1999 года и является сиквелом к игре 1994 года System Shock. Игра доступна через клиент Steam для платформы Linux.

Кроме того, стало известно о подготовке нативных портов для Linux пошаговой стратегии Civilization: Beyond Earth и космического симулятора Star Citizen. Civilization: Beyond Earth выйдет во второй половине 2014 года одновременно для Linux, OS X и Windows. Релиз Star Citizen ожидается в 2015 году.

Компания Valve, между тем, продолжает радовать линуксоидов новыми проектами: недавно она открыла исходники ToGL – прослойки для трансляции вызовов Direct3D в OpenGL. ToGL нацелен на упрощение портирования для Linux и OS X игр, изначально созданных для платформ Windows и Xbox: разработка прослойки велась в процессе портирования игры Dota 2. Код ToGL написан на C++, экспортирован непосредственно из дерева исходных текстов Dota 2 и открыт под лицензией ToGL Code License, которая аналогична лицензии MIT и допускает свободное использование кода в сторонних проектах.



В настоящий момент ToGL ограничен возможностью трансляции в OpenGL вызовов Direct3D 9.0. В состав входит работающий на уровне байткода транслятор языка описания шейдеров HLSL в GLSL. Обеспечена частичная поддержка Shader Model 3 – например, уже поддерживаются множественные цели рендеринга (Multiple Render Targets), но пока не доступны средства извлечения текстур в вершинный шейдер (VTF).

Следом за ToGL, были опубликованы исходные тексты проекта VoGL, в рамках которого развивается система отладки и трассировки работы с OpenGL. В частности, VoGL позволяет перехватить и сохранить поток операций OpenGL, повторно проиграть его, сохранить состояние в произвольной позиции, измерить производительность.

В состав входит подборка утилит и библиотек, а также графический интерфейс VoglEditor, написанный на Qt. В настоящее время обеспечена полная поддержка OpenGL 3.3, в ближайших планах – поддержка OpenGL 4.x. Код проекта открыт под лицензией MIT.

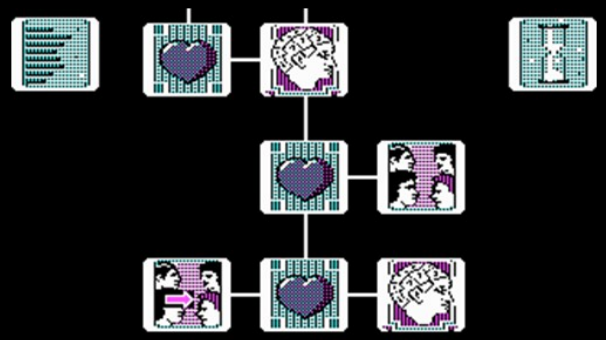
Культовые игры

Социальные симуляторы: от Alter Ego до Sims 4

Социальные симуляторы были всегда популярны у геймеров во всем мире – ведь игры этого жанра позволяют прожить еще одну жизнь, сделать то, что ты не можешь сделать в реальной жизни. Многие в них пытаются спрятаться от реальности и реализовать все свои мечты и фантазии. Основа этих симуляторов – это взаимодействие со средой, с другими игроками, общение и переживание различных жизненных ситуаций.

Много лет социальные симуляторы занимали определенную нишу на рынке видеоигр, с годами спрос на них только растет. С появлением игры-социальной сети **Second Life,** они стали еще больше развиваться: люди находят отдушину в игре, закрываясь от внешнего мира и обитая в виртуальном.

Но не будем о грустном – разберемся в истории социальных симуляторов. **Alter Ego** была разработана в далеком 1986 году Питером Фаваро: она моделирует жизнь человека, начиная от младенческого возраста. Жизнь в этой игре – дерево, где на ветвях находятся различные жизненные ситуации, которые игрок должен решить. Решив какую-то бытовую задачу, игрок переходит к следующей, так и проходит его жизнь. Что интересно, было создано две версии игры: для мужчин, и для женщин.



После возникновения такого явления как «симулятор жизни», жанр стал активно развиваться. Японские разработчики тоже придумали свои симуляторы типа **Harvest Moon**, американцы – **Jones in the Fast Lane.** Первая игра представляет собой симуляцию жизни фермера. Появившись в 1996 году, игра продолжала развиваться, появлялись новые версии. Jones in the Fast Lane выпущена в 1990-м: это игра, в которой вам предстояло играть одним человеком, достигнуть за время игры определенного уровня образования, количества денег и счастья.

Шло активное развитие сети Интернет – с ней стали появляться и многопользовательские игры. Самой знаменитой из них стала **Second Life.** Она была разработана в 2003 году американской компании Linden Lab. С тех пор уже 11 лет игра не теряет свой популярности: в созданном разработчиками виртуальном пространстве обитают множество игроков.

Конкретной цели игра не имеет – геймер создает своего персонажа и передвигается по локациям, общаясь с жителями игрового мира и создавая свои артефакты.

Игрок сам выбирает внешний вид своего персонажа. Как правило, аватар получается внешне далеким от внешности своего владельца, что породило много шуток. Персонаж может быть как фэнтезийным – например, иметь крылья, так и более антропоморфным. Вы легко можете стать пышногрудой блондинкой, даже если в реальности являетесь пожилым лысеющим мужчиной – и наслаждаться вниманием к вашей персоне со стороны других игроков.

Стоит отметить, что на данный момент игра уже пережила свой пик популярности – однако в игровом мире до сих пор можно встретить огромное количество игроков. Все довольно дружелюбные и приятные в общении люди.

Ну а теперь переходим к гиганту игровой индустрии – Electronic Arts – и к созданной им игре **Sims**. В 2000 году Уилл Райт разработал игру, ставшей самым продаваемым симулятором жизни по всему миру. После выхода первой части игра продолжала развиваться: в 2004 году появилась уже вторая часть – произошел переход от псевдотрехмерного пространства в настоящее трехмерное. Игра стала объемной и более натуральной, виртуальные люди – симы – более живыми, у них появились особые желания и жизненные цели. К Sims и **Sims 2** также было выпущено множество дополнений, симулирующих, например, жизнь в университете и жизнь домашних питомцев.

Игра за 4 года сделала огромный шаг вперед. Однако разработчики решили не останавливаться на достигнутом, и за 5 лет была создана новая версия игры – **Sims 3**. Игрушечные человечки сейчас не живут только в одном своем доме – локация расширилась до пределов города: они могут свободно перемещаться по нему, заходить в гости и гулять в парках.

За 14 лет существования игр серии **Sims** она стала культовой, у нее множество поклонников и почитателей. Осенью 2014 года планируется выпуск уже четвертой части игры – от прежней версии она будет отличаться новым геймплеем, обновленным редактором создания персонажа и другими фичами, которых не было раньше.

О пользе или вреде социальных симуляторов говорить сложно – многие исследования показали, что данная игра благоприятно влияет на психику детей, ведь она помогает понимать особенности социальных связей, научиться правильно общаться, следить за собой, за порядком и чистотой в доме. Ребенок учится не только познавать окружающий мир, но и менять его в лучшую сторону.

Но монета всегда имеет две стороны. Игра может стать не только приятным времяпрепровождением или развивающим занятием для детей – в ней может скрываться и опасность: как бы не променять реальный мир на виртуальный – ведь в нем все проще и доступнее!..

Наталия Чумакова

Grand Theft Console

Приставочное пиратство

С тех пор, как появились первые коммерческие игры для домашних компьютерных платформ, появились и игровые пираты, стремящиеся найти способ дешевого копирования игр для продажи нелицензионных копий.

В музыкальной и киноиндустрии тогда господствовали аналоговые форматы, которые было невозможно скопировать без ухудшения качества, а эпоха CD и дешевых компьютерных рекордеров еще не настала. Неудивительно, что самые первые средства DRM появились именно в индустрии программного обеспечения – ведь цифровые технологии изначально предполагают возможность копирования информации без ущерба для носителя.

В 80-х игровой рынок был сосредоточен на Amiga, ZX Spectrum и Commodore 64. Программы для этих платформ загружались с обычных аудиокассет, и это делало их абсолютно незащищенными от несанкционированного копирования: скопировать их мог любой, у кого был двухкассетный магнитофон.

Большинство издателей не беспокоилось по этому поводу: себестоимость игр была достаточно низкой. Но были и исключения. Одним из ранних вариантов защиты игр от копирования служили длинные списки кодов, прилагаемые к лицензионным версиям. Такие списки обычно печатались на бумаге темного цвета таким же темным, но имеющим другой цветовой оттенок шрифтом – эти листы было невозможно скопировать на черно-белом ксероксе, который просто «не видел» различий в оттенках. Зато человек без проблем мог их прочитать.

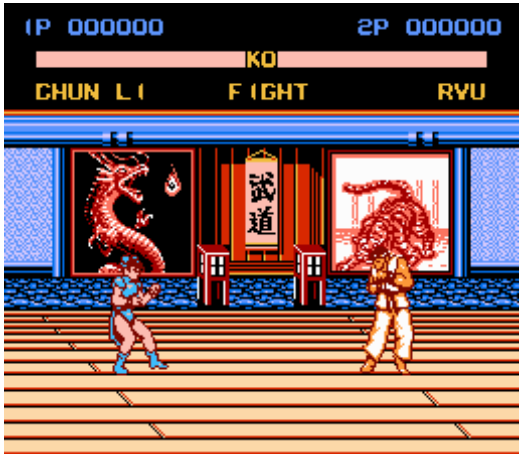
Появление ПЗУ-картриджей – и, в частности, культовой консоли Nintendo Entertainment System (NES) – усложнило пиратам жизнь. В отличие от компьютеров с унифицированными интерфейсами и сменными накопителями, картриджевые приставки сами по себе представляли собой готовую систему защиты от копирования.

Как правило, в них использовались картриджи оригинального дизайна, которые невозможно было скопировать без специального производственного оборудования. Для обычных пользователей такие ограничения были практически непреодолимы – но, разумеется, профессиональных пиратов это не останавливало.

Существовали также большие сложности с выпуском оригинальных игр: Nintendo одной из первых начала применять практику лицензирования прав на создание игр сторонним разработчикам – для этого необходимо было заключать договора и платить взносы. NES была снабжена специальной системой CIC (Checking Integrated Circuit), которая позволяла проводить аппаратную аутентификацию – картриджи без CIC-чипа попросту не запускались. Однако пираты и здесь нашли выход: чтобы продавать пиратские игры, нужно просто создать пиратскую консоль!

Появились аппаратные клоны приставок от известных производителей: все вы, наверное, знакомы с клоном NES – Dendy, которая была собрана в Китае и распространялась в России компанией Steepler. Всего же клонов NES существует бесчисленное множество. Позже пираты также стали орудовать и на других приставках – Genesis, SNES, GameBoy. «Неофициальные» игры и консоли распространялись, как правило, в тех странах, до которых Nintendo в то время не добралась.

Одной из крупнейших пиратских компаний в мире NES была Cony Soft. Компания была основана в начале 90-х и прекратила существование в 1998 году, успев выпустить более 40 игр и сменив свое название аж 10 раз – она также известна как Yoko, Whirlwind Manu, Super Game, Hummer Team и т.д.



Она занималась созданием нелицензионных игр типа Street Fighter и Mortal Kombat и портированием игр с более мощных приставок (Boogerman, Tazmania). Фактически, все игры серии MK и Street Fighter на NES создавала одна и та же команда! Cony также известна своими качественными портами, которые они создавали под лейблом Super Game: вспомните такие игры, как Boogerman, Aladdin, Lion King – почти в точности воссозданные с оригиналов на SMD и SNES. Также Cony принадлежат все движки Street Fighter и MK, на основе которых и сейчас, наверно, что-то создается.

Каждое название этой компании – отдельная отрасль их деятельности. Например Yoko и Hosekn (США) занимались в основном портами Mortal Kombat, Super Game – портами 16 битных игр, Whirlwind Manu – выпуском чужих игр, Wai Xing – созданием игр для китайского рынка.

Еще одна известная пиратская контора – Sachen. Она появилась в 1988 году в Тайвани и тогда еще называлась Joy Van. Позже она вместе с Color Dreams, Bunch Games и др. объединилась в единую компанию Panesian (Hacker international). Впоследствии некий Л.С.Тчаквоский купил все права на игры Sachen. История этой компании закончилась через 15 лет – это ознаменовало закрытие их официального сайта.

Sachen также известна как JY Company или JV (сокращение от Joy Van) – логотип JV стоял на всех их картриджах. Компания создала больше 30 оригинальных игр, эксклюзивный клон NES – Q-Boy, множество эксклюзивных аксесуаров и имела даже собственную сеть магазинов.

Было множество других, менее известных групп пиратов. Например, Active Enterprises известны тем, что создали картридж Action 52, который содержал 18 оригинальных игр.

Пираты были везде – в Китае, в России, США, Канаде, Тайвани, Бразилии, даже в Австралии. Больше всего их было на востоке (Китай, Тайвань, Гонк-Конг и т.д). Стоит упомянуть об одной из очень значемых для истории игровой индустрии пиратской конторе – Waixing Computer Science & Technology Co, Ltd. Она создавала для NES, наряду с собственными разработками, порты различных RPG и стратегий: на ен счету такие порты, как Heroes of Might and Magic и Bio Hazard.

Также существовала очень похожая команда Shenzhen Nanjing Technology Co, Ltd, известная своим качественым портом Final Fantasy VII, о котором даже написали на сайте, посвященном RPG – http://finalhearts.ru.

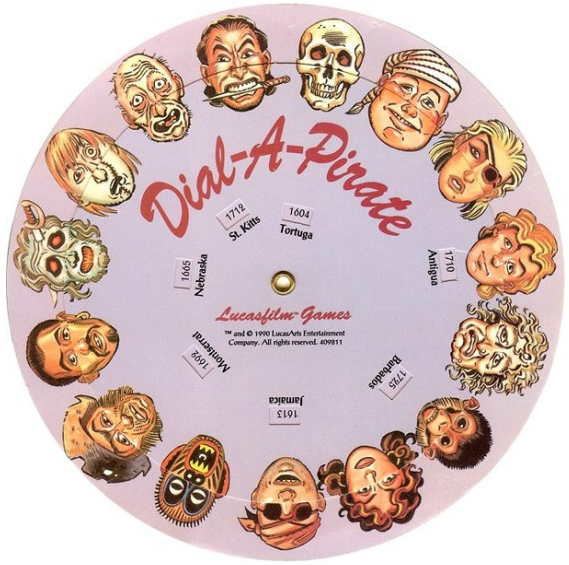
К началу 90-х получили распространение всевозможные устройства для «резервного копирования» игровых картриджей, в которых использовались самые обычные компьютерные дискеты. Легальность таких устройств была весьма сомнительна – та же Nintendo отчаянно судилась с их производителями. Но вместе с тем копирование оригинальных картриджей стало доступно даже простым пользователям.

Поэтому разработчики игр стали применять более высокотехнологичный способ, внося в программный код функцию проверки спецификаций «железа», на котором запускается их приложение. Если обнаруживался, к примеру, какой-то нестандартный накопитель, игра либо могла вовсе не запускаться, либо в ее ходе возникали какие-то неприятные сбои вроде отказа сохраняться.

Компания EarthBound, один из крупнейших производителей игр для SNES, реализовала еще более коварный способ защиты. Первоначально при запуске пиратская копия должна была работать как обычно, но количество врагов, с которыми надо было сразиться, при этом стремительно увеличивалось, делая игру практически непроходимой. Однако если игрок все-таки умудрялся пройти ее почти до самого конца, то перед схваткой с главным боссом игра зависала, а все данные сохранений удалялись.

Но если у видеоприставок существовал хотя бы какой-то базовый способ защиты от пиратства в виде проприетарных картриджей, то у игр для ПК такового не было никогда. В результате в ход шли всевозможные ухищрения. Например, дискеты со сбойными секторами, которые невозможно было скопировать «в лоб». Но это одновременно значило, что пользователь лишался возможности сохранить резервную копию игры, а, учитывая уровень надежности дискет, на которых они поставлялись, это было более чем актуально.

Весьма оригинальный способ защиты от копирования использовала студия LucasFilm Games в игре The Secret of Monkey Island. В комплект поставки входит вращающийся диск Dial-a-Pirate с изображениями пиратов и прорезанными окошечками, в которых можно было увидеть разные числовые коды. В процессе игры на экране появлялось лицо пирата и место, где он был повешен, а игрок должен был найти это лицо, вращая диск. После этого в соответствующем окошечке был виден код, который требовалось ввести для прохождения.



В других играх часто использовались всевозможные физические головоломки и прочие занятные мелочи, которые можно было «взять в руки». Впрочем, такие ухищрения довольно быстро стали бессмысленными, поскольку игроки стали выкладывать коды и способы решения задач в форумы и на специализированные сайты в Интернете. В дальнейшем они были полностью вытеснены привычными «серийниками», которые вводятся при установке игры.

Защита с использованием серийных номеров неразрывно связана с эпохой CD, когда игры и другие программы распространялись преимущественно на компакт-дисках. На коробке с диском печатался серийный номер, подходящий только к данной копии программы.

С повсеместным распространением Интернета очевидным недостатком этого метода стала проблема распространения образов дисков и серийных номеров к ним по сети. Кроме того, взломщики стали создавать кейгены – генераторы серийных номеров. Поэтому сейчас данный метод используется уже не так часто и только в совокупности с другими способами защиты.

Одним из более эффективных вариантов защиты в эпоху Интернета стала асимметричная криптография – она также широко применяется в сфере защиты пользовательских данных и шифрования пересылаемых сообщений. Генерируется два ключа: открытый и закрытый. В программе зашит открытый ключ – он служит для проверки серийных номеров. Закрытый ключ есть только у разработчика: с его помощью можно сгенерировать серийный номер.

Подобрать закрытый ключ и создать кейген в таком случае практически невозможно. Данная защита взламывается только созданием пропатченной версии программы с подменой публичного ключа – но и это не всегда простая задача.

Фархад Гафаров

«Содержимое заблокировано...»

Вся правда о творческом рейдерстве

Независимые авторы, создающие и публикующие свои произведения в Интернете в свободном доступе (к которым мы, редакция «FPS», также причисляем себя) привыкли чувствовать себя в определенной безопасности. Мы не пираты, мы не нарушаем ничьи авторские права, мы сами решаем, как распоряжаться своими работами. Возможно, этим мы кого-то раздражаем, но до недавнего времени корпорации, мечтающие прибрать к рукам все и вся, практически ничего не могли с этим поделать.

Но сейчас эта наша естественная и неотъемлемая свобода все чаще оказывается под угрозой. Blender Foundation, куратор разработки Blender и открытых фильмов, создаваемых с помощью этого пакета, недавно столкнулась с нелепой и абсурдной ситуацией: свободный анимационый фильм Sintel, размещенный на официальном канале организации, в начале апреля... был заблокирован администрацией YouTube. В выводимом сообщении утверждается, что видео содержит контент, принадлежащий компании Sony Pictures Movies & Shows – она потребовала закрыть доступ к ролику из-за нарушения ее авторских прав!

Оказалось, что кадры из Sintel использовались в наборе демонстраций, поставляемом в комплекте с новыми 4K-проекторами и телевизорами Sony – естественно, никаких претензий на них компания предъявлять не имеет права. Фильм и весь связанный с ним медиаконтент созданы студией Blender Institute и распространяются по лицензии Creative Commons BY. Каким же образом, в таком случае, видео смогли заблокировать?

Дело в том, что в YouTube используется автоматизированная система выявления фактов нарушения авторских прав – так называемый Content ID.

Правообладатель помещает в базу данных сервиса хэш защищенного авторским правом контента. Когда какой-либо пользователь загружает видео, из него также извлекается хэш, который сопоставляется с содержимым базы. В случае совпадения, система заявляет права на это видео в интересах владельца контента и применяет выбранные им санкции – видео может быть заблокировано (по всему миру или в отдельных странах) или монетизировано в пользу правообладателя.

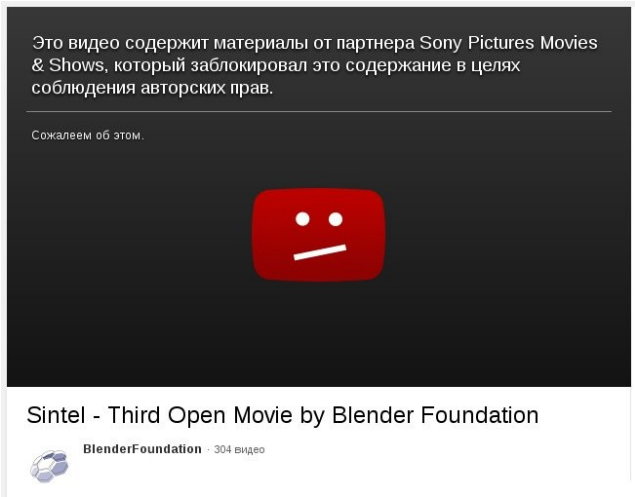
Преимущество такой системы – отсутствие необходимости вовлекать в работу по сортировке контента живых людей-модераторов. Именно это и делает ее «бездушной машиной»: от стратегической чистки YouTube от пиратских видеороликов страдают не только простые пользователи, но и, не в последнюю очередь, независимые индивидуальные авторы.

Вот вам красочный пример. Несколько лет назад NASA разместило на YouTube трансляцию высадки марсохода Curiosity. Буквально через несколько минут этот ролик, являющийся общественным достоянием, был заблокирован... по претензии новостного агентства Scripps Local News. И такое у NASA происходит с завидной регулярностью – примерно раз в месяц.

– I get flagged videos of birds flying around that don't even have audio
– Well duh, Rovio owns your birds.

Дело, конечно, не ограничивается только американским космическим агенством. «Камнем преткновения» часто оказывается... пение птиц и шум ветра. Пользователи говорят, что Content ID ведет себя неадекватно: определяет как объект авторского права «белый шум», лай собаки, записи классической музыки – и размещает в пользу «правообладателей» рекламные ролики, полностью игнорируя принципы добросовестного использования.

Иными словами, копирасты (иначе на назовешь) попросту наживаются на простых пользователях, которые в данной ситуации оказываются совершенно беззащитны. Система блокировки на YouTube очень сильно смещена в пользу заявителей прав – жалобы на действия заявителей бессмысленны: в ответ пользователи получают уведомление, что правообладатели рассмотрели заявки и все так же уверены, что в них содержится защищенный материал.



Любой мошенник может указать, что видео принадлежит ему, и получит деньги с рекламы. Российский офшор Netcom Partners с удовольствием пользуется этой ситуацией ради собственной выгоды и буквально захватывает чужие видеоролики. Некоторые компании - к примеру, Universal Music Group – могут блокировать видеоролики даже без рассмотрения заявок.

Заблокированные якобы за нарушение авторских прав видеообзоры игр стали, фактически, притчей во языцех. А когда один российский разработчик, создатель римейка «Battle City», выложил трейлер игры на YouTube, этот ролик вскоре заблокировали по жалобе телеканала ТНТ – у них в «Comedy Club» был выпуск с пародией на «Танчики»…

О ситуации, аналогичной истории с Sintel, поведал один из пользователей Blender, который вместе с другом снял короткометражный анимационный фильм и разместил его на YouTube с целью заработка от рекламы. Количество просмотров росло, и дела у ребят шли достаточно успешно, пока внезапно некая компания AudioFish не заявила о своих правах на саундтрек. Эта претензия выглядела абсурдом – музыка к фильму была специально записана музыкантом, и никому более принадлежать не могла! Однако система действует по идиотскому принципу «Сначала стреляй, потом разбирайся»: разбирательство продлилось несколько месяцев, за это время авторы потеряли свой траффик на пике популярности – весь доход, естественно, отошел к AudioFish…

Если копирасты называют пиратство воровством, то как назвать эти их действия? Это же настоящий грабеж! И тот факт, что Google как владелец YouTube повторствует всему этому, говорит далеко не в ее пользу. Мы, авторы, не против борьбы с пиратством и защиты авторских прав, но если это переходит границы разумного и начинает выполнять, фактически, обратную функцию – то стоит ли овчинка выделки?

Задумайтесь: ведь настоящие пираты-грабители – это не те, кто раздает бесплатно, а те, кто отнимает и блокирует! Ведь те, кого принято называть интернет-пиратами, не претендуют на авторство произведений – они не пытаются наживаться на чужом контенте, они просто отстаивают право пользователя на неограниченное копирование информации. И любой мало-мальски разумный автор должен понимать, что в при современном развитии технологий препятствовать копированию не просто невозможно, но и просто глупо – чем больше людей скачают ваше произведение, тем популярнее оно станет.

Здесь же мы имеем дело с воровством в самом прямом понимании этого слова. Если само пиратство – преступление довольно сомнительное, то борьба с пиратством стала удобным прикрытием для преступления откровенного! И закон здесь, по сути, на стороне крупных компаний, а права индивидуального автора не защищены ничем и не интересуют никого…

Все это отчасти напоминает печально известные совтверные патенты в США: если вы независимо создадите реализацию какого -нибудь алгоритма или идеи, вас может засудить владелец патента на эту идею, даже если вы об этом патенте слыхом не слыхивали. И самый яркий пример уродства этой системы – «патентные тролли», компании-патентодержатели, которые ничего не производят, а зарабатывают только на высуженных патентных отичслениях. К счастью, патентов на идеи в искусстве пока не существует, но кто знает, что может прийти в голову американским законодателям: не станут ли завтра крупные медиахолдинги и кинокомпании монополистами, скажем, на жанр фэнтези?..

А что касается Google, то ее двуличие видно невооруженным глазом: с одной стороны она поддерживает OpenSource, великодушно использует свой патентный портфель для помощи Android-вендорам, а с другой – подминает под себя Интернет, агрессивно подавляя конкурентов – и, порой, просто отдельных безобидных разработчиков.

Сначала Google удалила из официального каталога Android все приложения для блокировки рекламы – по чисто формальному поводу «неавторизированное нарушение нормального функционирования других приложений и сервисов». То есть, пользователь обязан лицезреть рекламу и уже как бы не имеет права технически контролировать собственный девайс!

Затем война с «антирекламщиками» началась уже на десктопах: Google начала удалять из магазина соответствующие дополнения для Chrome. Сейчас уже говорят о скором прекращении поддержки установки в Chrome дополнений из сторонних источников – в стабильных выпусках браузера для Windows дополнения можно будет установить только из каталога-магазина Chrome App Store. Все ранее установленные из внешних источников дополнения, которые не представлены в App Store, также будут заблокированы.

Это ужесточение, впрочем, не касается версий для Linux и Mac OS X – да и под Windows пользователь сможет переключить браузер в режим для разработчиков и, как прежде, устанавливать любые дополнения «на свой страх и риск». Но кто из обычных пользователей станет это делать? Скорее всего, они просто не будут даже пытаться устанавливать какие-то сторонние дополнения.

Делается это под предлогом борьбы с вредоносным ПО – но нетрудно понять, что на деле это очередной способ несправедливой конкурентной борьбы. Остается только надеяться, что разумные люди бойкотируют Chrome и перейдут на Firefox и другие альтернативные браузеры, уважающие свободу своих пользователей.

Каждый год мы читаем громкие лозунги и красочные новости о GSoC, о том, как Google помогает разработчикам СПО, выплачивает гранты и премии. Но истинное отношение корпорации к хакерам и независимым разработчикам выражается в другом, и знают о нем немногие.

В конце прошлого года из каталога Google Play был удален инсталлятор известной альтернативной Android -прошивки CyanogenMod – в качестве причины называется «призыв пользователей к нарушению гарантии на устройства, что запрещено в условиях использования сервиса Play Store».

При этом сам по себе инсталлятор, выполняющийся на платформе Android, только показывает пользователю, как включить доступ к устройству с использованием утилиты ADB – непостредственно перепрошивка выполняется другой программой, запускаемой под управлением Windows. Представители Google согласились, что приложение CyanogenMod Installer безобидно и не представляет угрозы, но сам факт стимулирования разрыва условий гарантийного обслуживания не позволяет приложению оставаться в каталоге.

Назовем вещи своими именами: Google называет Android свободным ПО, но при этом всячески препятствует и борется с теми, кто хочет этой свободой воспользоваться! А причина ясна – если пользователи будут заботиться о своей свободе и принимать соответствующие меры, то возникнет угроза для бизнеса. Следовательно, не свобода ей важна, а все те же деньги – если вы не способствуете обогащению Google, прямо или косвенно, вы ей неинтересны и даже вредны.

И эта ситуация сейчас повсеместна, хотя большинство ее не замечает. Компании, желающие контролировать экономику и общественное сознание, поощряют нейтральность и безучастность.

Мол, разрабатывайте очередной клон Flappy Bird – мы не против, мы даже поможем вам на этом заработать. Но как только вы осознаете, где правда, а где ложь, как только захотите что-то в этом мире изменить, хотя бы на чуть-чуть восстановить справедливость – как к вам оборачиваются спиной!

И OpenSource для них – лишь средство рекламы: модное словцо, при помощи которого можно привлекать сотрудников и клиентов. Взгляните на ситуацию с видеоредактором Lightworks: года три назад все новостные ленты и тематические сайты по свободному ПО пестрили сообщениями о том, что владельцы профессионального инструмента голливудского уровня собираются сделать его открытым.

Сейчас уже 2014 год, а воз и ныне там. Но шумиха сделала свое дело: Lightworks успешно разрекламирован, все кинулись скачивать урезанную бесплатную версию – а то и покупать лицензию. А обещание открыть исходники как-то тихо и незаметно забылось.

Есть большая разница между открытым проектом и открытым продуктом. Просто открыть исходники – еще недостаточно, чтобы обеспечить свободу пользователей. Но свобода пользователей – последнее, о чем задумываются корпорации. Они, как правило, не отличаются дружественностью к сообществу, не любят принимать сторонние патчи и вообще редко рассматривают какие-либо предложения по улучшению продукта. И примеры напрашиваются сами собой: Oracle и OpenOffice.org, HeroineVirtual и Cinelerra, Canonical с их лицензионными претензиями, Google с ее ридонли-моделью разработки Android, сплошная тивоизация в самих Android-устройствах и так далее.

Все это лишний раз говорит нам о том, что не стоит тешить себя иллюзиями – бизнесу важна лишь прибыль: о какой-либо свободе, справедливости и общечеловеческих ценностях он не заботится и заботиться не будет. И, когда вы в следующий раз будете выкладывать куда-то свою информацию (фактически, доверять ее корпорациям) помните об этом: ведь им нужны просто деньги – а не вы, не ваше творчество и не ваши идеи...

Тимур Гафаров

Уважаемые читатели!

Наш журнал регулярно выходит на протяжении почти 6 лет – с января 2008 года. Все эти годы он оставался бесплатным изданием, предлагая публике эксклюзивный контент с минимумом рекламы. Мы всегда работали на совесть – не ради денег, а на благо наших читателей. «FPS» был и остается проектом энтузиастов и полностью независимым изданием – мы не защищаем интересы корпораций или политиков, мы пишем о том, что считаем нужным и важным. Мы стоим за свободу слова и творчества, за обмен информацией и знаниями: все материалы журнала можно беспрепятственно копировать, распространять и использовать в любых производных работах.

И мы надеемся, что так будет продолжаться и дальше. Но на создание новых номеров у авторов уходит достаточно много сил и времени, которые никак материально не компенсируются. Поэтому, если вам нравится журнал, и вы хотели бы, чтобы он жил, развивался, становился больше и качественнее, просим поддержать его электронной валютой – при помощи Bitcoin, WebMoney, PayPal или Яндекс.Денег, любой суммой на ваше усмотрение. Для нас важен любой, даже маленький вклад!

Наш WMR-кошелек: **R120156543694**

Адреса Bitcoin:
16PSGbj5foeqMN8isdoyiKvWYGM9V5idFk
16XaSt1U5eXWG7EAkuEMpFE6M6fPia5o4F
1PdNHTL5nJsZJGXyNJ4c5xPW3eApEoB9pQ
1MFauzBqewUN8MWPy7DqGeLiJayxCqGBCL
1MSPN7TXuTPbuGjT1AQxex3ECjRgPhpFEA
1NB1xXoyJ71beEPpmmKioWMaon2uCigCeQ

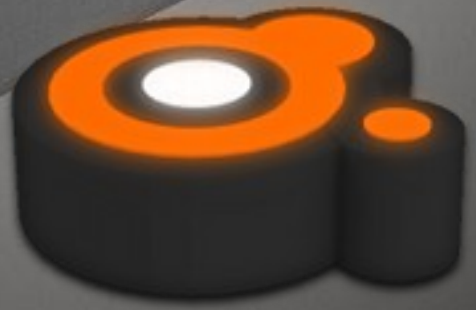
Адрес PayPal:
gecko0307@gmail.com

Яндекс.Деньги:
410012052560079

Заранее благодарны!
Редакция

Это все!

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



http://fps-magazine.blogspot.ru