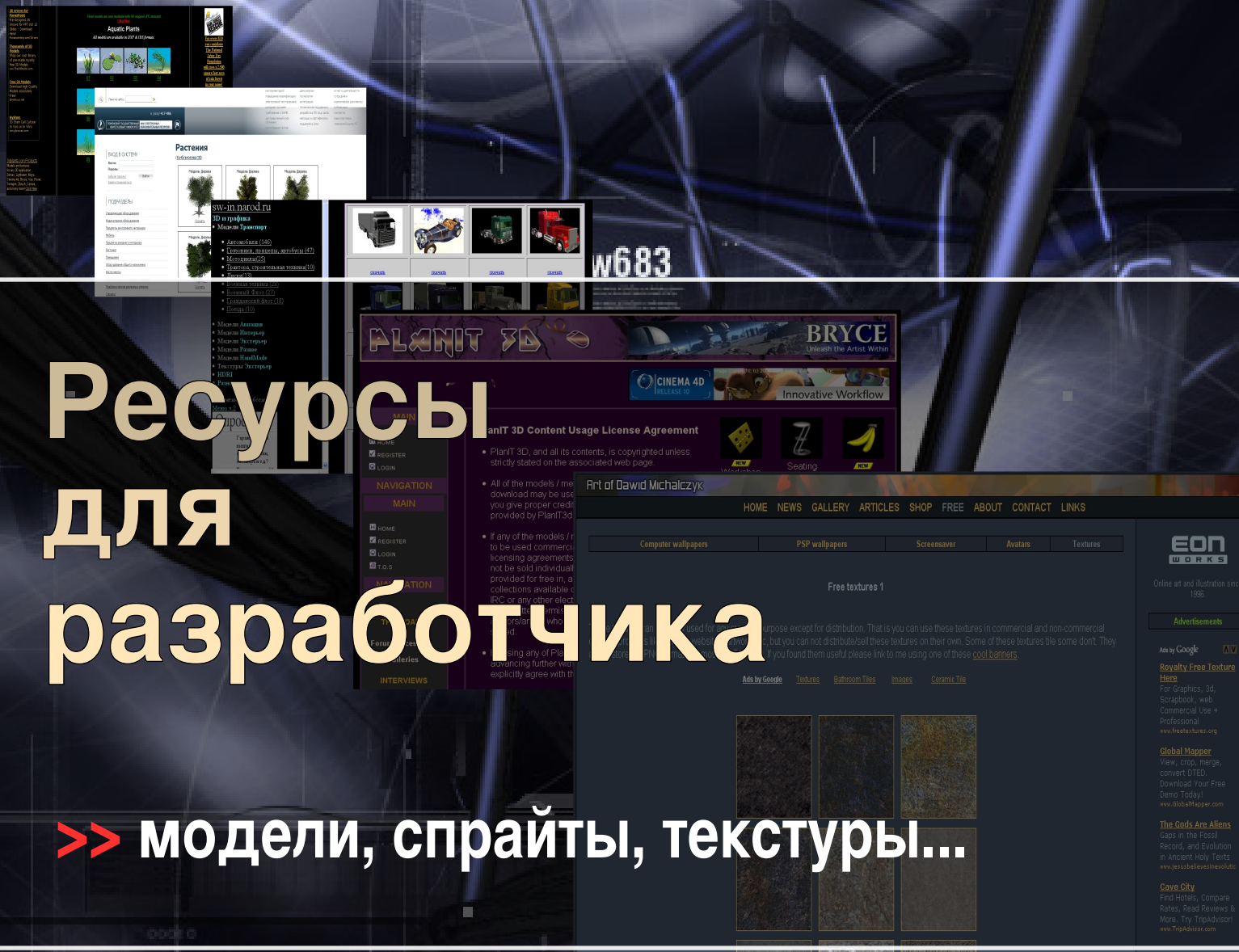


# LINUX GAMES TECHNOLOGIES



## Ресурсы для разработчика

>> модели, спрайты, текстуры...

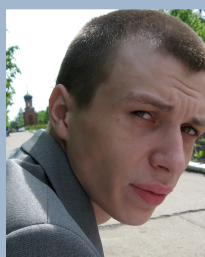
- + OpenGL и SDL
- + Моделирование растительности
- + Интернационализация проекта
- + PAS и Novashell

Новая рубрика

>>>> Игротека <<<<<

В погоне за временем  
по трассам Racer.

## Наши авторы:

Владислав  
ВолодинКирилл  
БаженовАндрей  
ЯкушевАндрей  
Кондратьев**Редакция:**Андрей Прахов, Светлана  
Капцова, Виктор Сухоруков**Вёрстка:**

Вячеслав Резник

Для вёрстки журнала  
использовались инструменты  
Open Sources: Scribus, Open  
Office, Gimp.Перепечатка любых материалов журнала  
возможна только с разрешения редакции.Редакция: [info@lingametech.com](mailto:info@lingametech.com)

# Дорогие друзья!

Итак, свершилось! Мы всё же преодолели возникшие сложности и выпустили второй номер журнала. Правда, с небольшим опозданием. Конечно, прошло уже почти два месяца с момента выхода первого номера, но, если помните, на его создание мы потратили свыше 4 месяцев. Сократить более чем вдвое время на разработку — это небольшое, но всё же достижение, а впереди планка ежемесячных выпусков. Думаю, мы всё же достигнем её.

В этом выпуске мы постарались учесть просьбы высказанные читателями в отношении вёрстки журнала. Изменился подход к размещению материала, появилось навигационное меню, полезные интернетовские адреса получили активные ссылки, но главное, конечно, — это содержимое.

В качестве темы номера был выбран обзор полезных сайтов с контентом для разработчика и, это не случайно. На наш взгляд, именно создание контента отнимает львиную долю времени отведенного на разработку. Надеемся, подборка ресурсов окажется вам полезной.

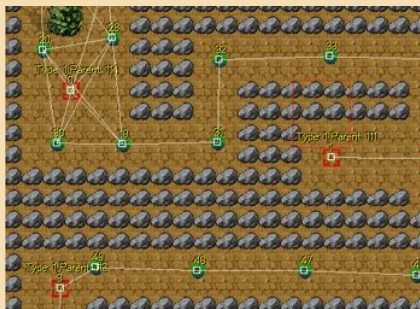
В этом номере дебютирует новая рубрика «Игротека», посвящённая обзору игр не с точки зрения разработчика, а глазами простого пользователя. Было много споров о целесообразности подобной рубрики в казалась бы индустриальном журнале. Высказанные пожелания читателей на форумах и наша уверенность в том, что невозможно сделать хорошую игру без образца для подражания, решили исход спора. Но основной ответ должны дать Вы — наши дорогие читатели. Пишите письма в редакцию, оставляете сообщения в комментариях на сайте — ваше мнение будет решающим.

С уважением,  
Андрей Прахов

# Содержание

## Обзоры

### Novashell



Необычная среда для разработки двухмерных игр, созданная всего двумя разработчиками.

стр. 6

### Platinum Arts Sandbox



Детский конструктор или больше?

стр. 9

## Новости

Quake Live появится и для Linux, уверяет Джон Кармак.

Владельцы приставки PS3 Slim не смогут использовать Linux.

Игры для военных. Симуляция боевых действий под управлением Linux.

Проект NoN близок к завершению. Игра выйдет одновременно для трёх популярных систем.

стр. 4

## Тема номера

# Сокровищница Интернета

Ресурсы для разработчика: 3D модели, текстуры, спрайты

стр. 12

## Учебники

### Кодовый блок

Игра для Gnome: Интернационализация.

Серия уроков, часть 2.  
Научим программу разным языкам.

стр. 25

SDL & OpenGL: Первые шаги в создании своего движка.

Серия уроков, часть 1

стр. 331

### Графика

Практикум BGE: Поработаем садовником.

Создание растений и деревьев для игрового движка Blender.

стр. 44

## Игротека



### В погоне за временем

Racer - автосимулятор с физической моделью поведения автомобиля, обладает симпатичной графикой. Погоняем?

стр. 22



# Новости

## Quake Live для Linux



В феврале этого года появилась бета-версия онлайн игры Quake Live. Тогда же Джон Кармак, основатель студии id Software, на пресс-конференции в Далласе заявил, что «реализация поддержки Маков и Linux-машин находится в числе наивысших приоритетов». Действительно, полгода спустя 19 августа, id Software сообщила о выпуске специальной версии игры под nix-системы.

Напомним, что Quake Live является онлайн версией своего знаменитого предшественника и работает исключительно в браузере. До недавнего времени игра была бесплатной, а компания предоставила пользователям пару сотен игровых серверов. Однако используемая модель внутриигровой рекламы, по словам Кармака, оказалась крайне малоэффективной. Компания продумывает варианты возврата денег вложенных в проект. Возможно будут введены премиальные аккаунты с некоторым дополнительным функционалом, но это нововведение будет исключительно опциональным.

## Владельцы приставки PS3 Slim не смогут использовать Linux



В своё время, Интернет облетела новость, что на Play Station 2 вполне можно установить и использовать операционную систему GNU/Linux. Причем, компания Sony активно способствовала этому почину пользователей. Появилась даже специально адаптированная версия Linux для этой приставки. Тем более, на фоне такого подъёма выглядит странным заявление представителей компании, что новейшая приставка PS3 Slim аппаратно будет закрыта для установки иных операционных систем.



**Новости**

Джон Коллер (John Koller), директор по маркетингу SCEA пояснил, что этим способом они пытаются добиться стандартизации поддерживаемых ОС. Не менее странным является решение исключить поддержки игр для PS2 в новой версии приставки.

**Игры для военных**

Пресс-центр МВД республики Узбекистан заявил, что в Ташкентском высшем военно-техническом училище был создан Специализированный центр моделирования и симуляции. СЦМС предназначен для проведения учений и практических занятий с применением компьютерной симуляции. Министерство надеется, что данное нововведение существенно повысит уровень образования

выпускников училища. СЦМС основан на компьютерном парке, где установлена система GNU/Linux. При этом используется компьютерная программа Janus, «представляющая собой интерактивную, стохастическую модель наземных боевых действий высокого уровня с точной цветной графикой», - по словам представителей МВД.

**Проект HoN близок к релизу**

Независимый разработчик S2 Games объявил о скором завершении работы над проектом «Heroes of Newerth». Игра HoN основана на вселенной «Savage-Newerth». В целом HoN похожа на знаменитый мод Defense of the Ancients игры Warcraft 3 со значительно переработанной графикой и геймплеем. В настоящее время игра находится в режиме закрытого бета-тестирования и будет доступна к концу этого года. Одновременно с выходом

Windows-версии появятся сборки для Mac и GNU/Linux.

LGT

# Игровой конструктор Novashell

*Основанный на популярной библиотеки ClanLib,  
кроссплатформенный и бесплатный игровой конструктор. Смотрим?*

## От редакции:

Создание игры – дело сложное, поглощающее много времени. Неудивительно, что разработчики стараются любыми способами сократить стоимость и время производства, путем использования соответствующего инструментария. Давайте познакомимся с необычной средой разработки двухмерных игр Novashell.

## Первый взгляд

Проект
Игровой конструктор 2D
<b>Жанр:</b> Adventure, RPG
<b>ОС:</b> GNU/Linux, Windows, Os X
<b>Язык прогр.:</b> C++
<b>Скрипты:</b> есть (Lua)
<b>Звук:</b> есть (OpenAL)
<b>Сеть:</b> нет
<b>Лицензия:</b> Zlib

На фоне бурного развития трехмерных игр как-то в стороне, обособленно, существуют двухмерные. Кажется, что 2D начисто проигрывает красочным, объемным картинкам 3D. В действительности, направление казуальных игр (а именно они чаще всего являются двухмерными) по массовости продаж и количеству хитов затмевают стандартные жанры. Почему это происходит – тема для отдельной статьи. Вкратце скажу лишь, что казуальные игры позволяют играть во время работы, имеют малый размер и небольшую стоимость. А вот аудитория казуалок, как правило, люди среднего возраста скучающие в офисе.

Если у всех на слуху имеются коммерческие или свободные 3D движки, то двухмерные можно сосчитать по пальцам. Как правило, берётся подходящая библиотека (SDL, ClanLib...) и на основе её создается игра. Но есть и специализированные конструкторы. Среди коммерческих можно вспомнить знаменитый Torque Builder компании GarageGames, а среди бесплатных хорошим выбором может оказаться рассматриваемый Novashell.

Проект создавался, как внутренний продукт компании для создания собственных игр. Разработчики поступили достаточно мудро, открыв конструктор для свободного пользования, что позволит им иметь неплохую базу невольных тестеров. Все остальные продукты распространяются на коммер-



>> Чета Робинсон – создатели Novashell

ческой основе.

Novashell представляет собой фреймворк, имеющий в качестве базы известную кроссплатформенную библиотеку ClanLib. Особенностью конструктора является механизм запуска игр, который состоит из двух частей – исполняемого бинарного блока и собственно игры, упакованной в архив zip. Подобно другим фреймворкам, Novashell объединяет в себе запускаемую среду и режим редактирования. Давайте пробежимся по основным характеристикам конструктора:

- Наличие физики на основе движка Box2D;
- Поддержка Lua (основного языка для скриптования);
- Параллельный скроллинг без ограничения на количество слоев;
- Система частиц;
- Поддержка звука с некоторыми эффектами: crossfading, эхо, позиционирование.

## Конструктор в деле

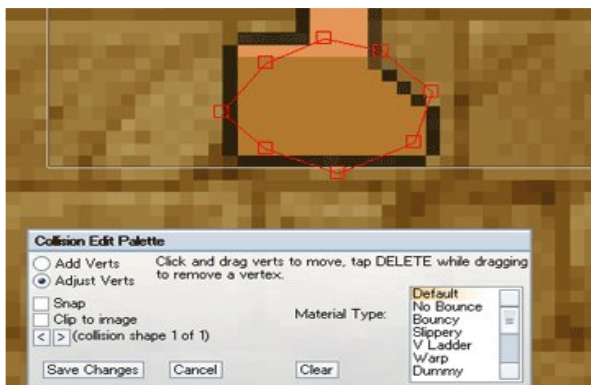
Первое же знакомство с конструктором обескураживает – абсолютно ничего непонятно. К счастью, дистрибутив идет с большим количеством тестовых игр, а на сайте имеется хорошая четкая документация.

Основные операции по созданию карт производятся во встроенном редакторе. Как правило, все действия можно совершать, не покидая программу. Конструктор позволяет заливать имеющуюся карту или выделенное пространство одной и той же текстурой. Очень приятной особенностью является возможность вырезания отмеченного куска картинки и использования его, как отдельного спрайта или текстуры.

Сам процесс первоначальной расстановки объектов на сцене достаточно прост. Можно перемещать, масштабировать до немыслимых размеров, вращать элементы с помощью горячих клавиш и мыши. Также каждый объект



имеет свойства, которые содержит специальное диалоговое окно при выборе соответствующего пункта плавающего меню. Здесь можно указать буквально все: от размеров объекта до связанного с ним скрипта. Кстати, конструктор не имеет встроенного текстового редактора, для написания скриптов используется сторонняя программа. По умолчанию, для Linux выбирается gedit, а для Windows – стандартный блокнот. Впрочем, привязки легко поменять в файле настроек.



>> Несколько щелчков мышью и физика настроена

Так же незатейливо выглядит настройка физики объектов. Для спрайтов имеется специальный редактор, позволяющий указывать зоны области определения collision. Наметив контур, к примеру, спрайта камня и подкрутив некоторые параметры, можно заставить его вполне реалистично катиться с горки.

Для настройки движения объекта можно использовать два способа: программирование с помощью скрипта или графического указания траектории движения. Второй вариант наиболее оптимален, в случае, если спрайт необходимо перемещать по кольцевой траектории. Расставляются ключевые точки, указывается направление движения, скорость и вот уже объект заскользил по карте. Кстати, несложно подстроить углы поворота спрайта при изменении направления движения. Программа максимально облегчает этот труд.



>> Вот так схематично отображается маршрут движения

#### А Ваше мнение?

Если Вы умеете работать с этим приложением и Вам есть что рассказать, то, возможно, именно Ваш опыт станет полезным для читателей. Пишите на адрес редакции: [info@lingametechnologies.com](mailto:info@lingametechnologies.com)

## Вывод

Novashell – это мощная среда разработки казуальных игр. Несмотря на текущий статус «альфа», она вполне работоспособна и надежна. Единственным минусом является сложный и запутанный интерфейс. Однако наличие хорошей документации сводит на нет этот недостаток. Словом, стоит попробовать программу в деле.

LGT

# Platinum Arts Sandbox

*Позиционируется для детей, но можно ли использовать его для «взрослых» проектов?*

**От редакции:**

«Песочница» - именно так можно перевести название этого уникального конструктора. По задумке авторов Platinum Arts Sandbox (PAS) предназначен для прототипирования игровых проектов и использования его в качестве учебного инструмента в школьных заведениях. В соответствии с этой идеей программа выглядит очень простой, но только на первый взгляд...

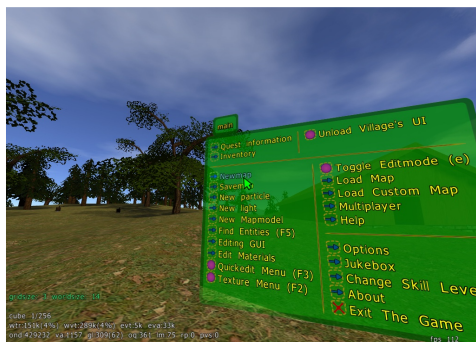
## Несерьезное название с серьезной начинкой

Проект
Игровой конструктор 3D
Жанр: FPS
ОС: GNU/Linux, Windows, Os X
Скрипты: есть (CubaScript)
Звук: есть
Сеть: есть
Лицензия: free
Разработчик: Platinum Arts LLC

Что нужно школьнику для счастья, который уже пресытился компьютерными играми? Правильно, дать ему возможность создания собственных. Именно этим руководствовались разработчики, выпуская в свет PAS. На сайте проекта имеется обращение к школьным заведениям использовать конструктор в качестве учебного инструмента для обучения детей программированию. Однако нам, людям уже взрослыми достаточно опытным, гораздо более интересен вопрос в области его профессионального применения. Смотрим на страницу разработчиков и видим фразу: «Platinum Arts Sandbox will be easy enough for kids to use but also powerful enough for full game projects.» [[«PAS легок в использовании для детей и в тоже время мощный инструмент для полноценных игровых проектов»](#), - прим. ред.]. Давайте посмотрим поближе...

В основе конструктора лежит известный движок Cube 2 с вполне приличными графическими характеристиками. Поддержка шейдеров, динамического освещения, «гладкая» анимация – список достаточно солидный. Благодаря усилиям разработчиков, программа поставляется с набором эффектов, моделей, звуков, некоторого подобия AI. Режим конструктора и игры являются совмещенными. Пара нажатых клавиш и вы можете редактировать сцену. Причем интересно то, что любые манипуляции с кодом и миром возможны не выходя из программы.

При первом запуске вы попадете на демонстрационный игровой уровень.



>> PAS умеет работать с трехмерными меню

Так как с конструктором идут уже несколько десятков разноплановых карт, то перебрав их все, можно немного ознакомиться с возможностями движка. По умолчанию PAS ориентирован на создание шутеров от первого лица. Немного погоняв демо, можно убедиться в корректной работе камеры, физики и всего-того, что является отличительными признаками FPS. Однако по уверениям разработчиков, данный конструктор можно использовать и для создания игр других жанров. В качестве примера в пакете программы имеется заготовка для игр-скроллеров (а' ля Марио).

## А как в работе?

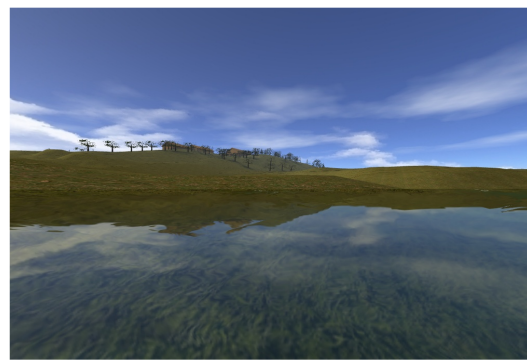
Вволю погонявшись по пустынным локациям, мы попробовали программу в её основной функции - конструктора. Для начала редактирования достаточно выбрать соответствующую команду в главном меню или нажать горячую клавишу. Но, стоп! Раз разговор зашел о меню, давайте его и изучим. Главная особенность меню заключается в том, что оно совмещает в себе типичное игровое меню и меню разработчика. Как уже говорилось, любые манипуляции с миром конструктора возможны лишь через него и некоторый набор горячих клавиш. Забудьте о привычном оконном интерфейсе со множеством кнопочек! Все что вы пожелаете сотворить с имеющейся картой, вы сделаете играючи, причем в буквальном смысле. Хотите поставить дерево? Извольте двигаться к нужному месту, как игрок в шутере и устанавливайте модель. Появилось желание изменить геометрию ландшафта? Взлетите в небо и игровым курсором отметьте нужную зону. Действительно, процесс настройки карты выглядит очень привлекательно, но только для детей. Игра создается во время игры и эта возможность - лакомый кусочек для юного поколения. Однако, если рассматривать этот подход с точки зрения профессионализма, то такой путь чрезвычайно непродуктивен.

Оставим в покое интерфейс и вплотную рассмотрим возможности конструктора. PAS имеет в своем составе готовый набор моделей, текстур, эффектов частиц и даже шейдерные заготовки природных явлений. Естественно, вы можете расширить этот список путём простого копирования контента в соответствующие папки и настройки некоторых файлов. Принцип использования заготовок по аналогии чрезвычайно прост. Указываете место на кар-



те, выбираете соответствующую функцию из меню и вот уже на пустой равнине появилось озеро с кристальной водой. Кстати, предлагаемые шейдерные эффекты выглядят очень впечатляюще. Вы можете использовать поверхности воды и лавы, стекло, зеркальные отражения. Мало того, при создании воды программа автоматически предоставляет эффекты погружения. Скажем проще. Войдя в такую воду, вы увидите всплески, услышите соответствующий звуковой шум, а окунувшись, сможете наблюдать потрясающий вид подводного мира.

Не менее просто создать сам ландшафт. Программа имеет некоторый набор возможностей для геометрического искажения высот и заливки выбранных мест карты текстурами. Вот только опять-таки приходится приспосабливаться к игровому интерфейсу конструктора и запоминать некоторое количество управляющих клавиш. Кстати, многие функции доступны только с клавиатуры и не имеют аналогов в стандартном меню. Перечень клавиш можно найти в текстовом файле в корневом каталоге конструктора.



>> Вода выглядит просто замечательно

Кроме того PAS предоставляет стандартный набор эффектов окружения: skybox, дождь, снег. По умолчанию окружение получается очень даже приличным. Светит солнце, голубеет небо, медленно плывут облака. Естественно, некоторые параметры окружения, включая сами текстуры, доступны для изменения.

## Вывод

Как и любой инструмент PAS имеет свои сильные и слабые стороны. Достаточно продвинутая визуализация, скриптовый язык CubeScript, поддержка популярных форматов моделей (включая известный md3), parallax mapping и карты нормалей – всё это позволяет говорить о пригодности конструктора к созданию средних по сложности игр. Но вот концепция игрового конструирования ставит жирный минус этой в целом замечательной программе.

LGT

### А Ваше мнение?

Если Вы умеете работать с этим приложением и Вам есть что рассказать, то, возможно, именно Ваш опыт станет полезным для читателей. Пишите на адрес редакции: [info@lingamotech.com](mailto:info@lingamotech.com)



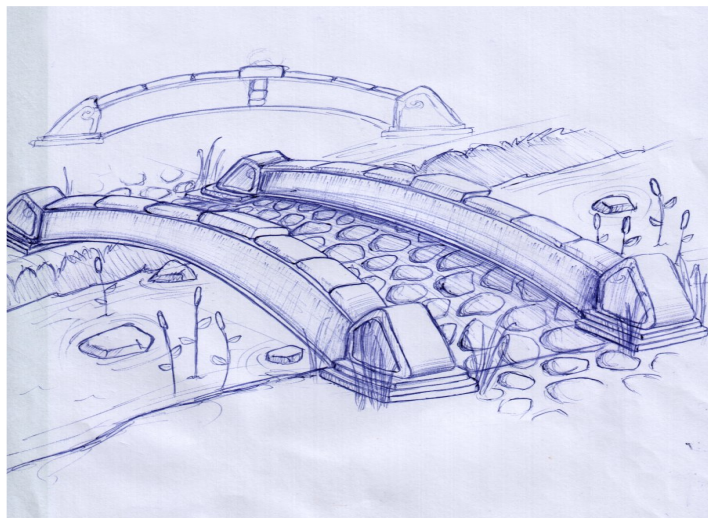
## От редакции

*Хорошо, когда команда разработчиков включает в себя всех необходимых специалистов для создания игры. Но такое бывает редко, даже продвинутым студиям с солидным бюджетом приходится обращаться «на сторону» за дополнительной помощью. Что же говорить об инди-разработчиках или молодой команде энтузиастов!*

*Чаще всего разработчики встречаются с дефицитом моделеров, текстурищиков и звуковиков. Значительное время работы над проектом можно сократить, если воспользоваться услугами фрилансеров или скачать необходимый контент на специализированных ресурсах в Интернете. Именно последний пункт и попал в поле зрения нашего обозревателя.*

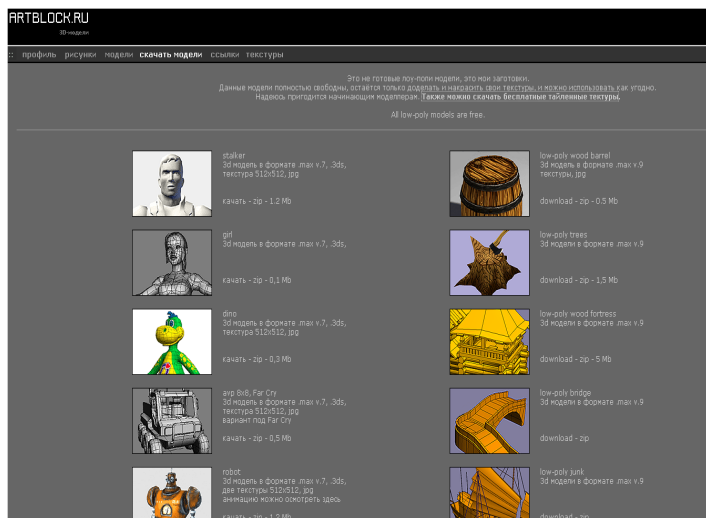
## 3D МОДЕЛИ

Создание отдельной модели объекта относительной сложности может занять от одного до нескольких дней работы моделера. Конечно, многое зависит от профессионализма исполнителя и общего уровня организованности процесса. По правилам, моделер подключается к проекту после заверше-



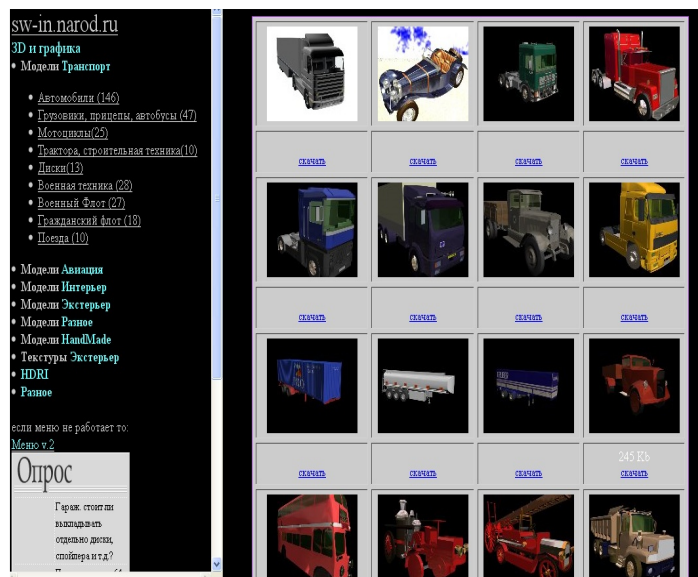
>> Типичный скетч для последующего моделирования

ния работы художника над скетчем объекта. В свою очередь, художник должен подготовить рисунок будущей модели в трёх ракурсах. Но это при идеальном стечении обстоятельств. На деле моделер начинает работу, исходя из имеющегося дизайн-документа. В итоге, с учетом 3 ключевых персонажей, двух десятков NPS и сотни вспомогательных объектов, процесс для одного исполнителя может затянуться на пару долгих лет. Спасти положение может только всемогущий Интернет.



## Artblock.ru

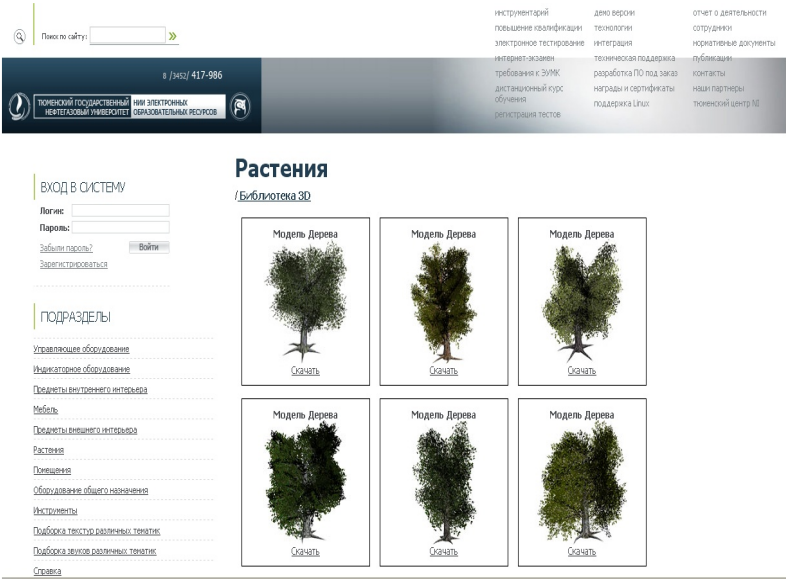
Начнем обзор с сайта [Artblock.ru](http://Artblock.ru). Его создатель Андрей Макрушин – профессиональный 3D-моделер, участвовавший в нескольких игровых проектах. Этот ресурс полностью посвящен низкополигональным моделям. Некоторые объекты уже имеют встроенные текстуры. К сожалению, на сайте представлены модели только в двух форматах: 3DS и разных версиях 3DS MAX. Впрочем, Blender, как основной инструмент моделирования в GNU/Linux, умеет импортировать формат 3DS. Набор весьма разноплановый, как говорится от А до Я, начиная от персонажей и заканчивая объектами ландшафта. Все представленные модели являются свободными для скачивания и могут использоваться в игровых проектах, в том числе коммерческих.



## Sw-in.narod.ru

Следующий сайт на очереди – [sw-in.narod.ru](http://sw-in.narod.ru). Несмотря на своё неординарное название, сайт имеет просто очень богатую подборку моделей на все случаи жизни. Объекты отсортированы по категориям: транспорт, авиация, интерьер, экстерьер и другие. Для просмотра контента используются графические миниатюры, щелчком по которым можно рассмотреть их детально. Все модели позиционируются, как бесплатные. При этом регистрация для скачивания с сайта не является приоритетом. Однако, представленные модели являются высокополигональными, что требует их обработку для использования в играх. Форматы самые разные от .max до .obj. Почти все имеют качественные текстуры.

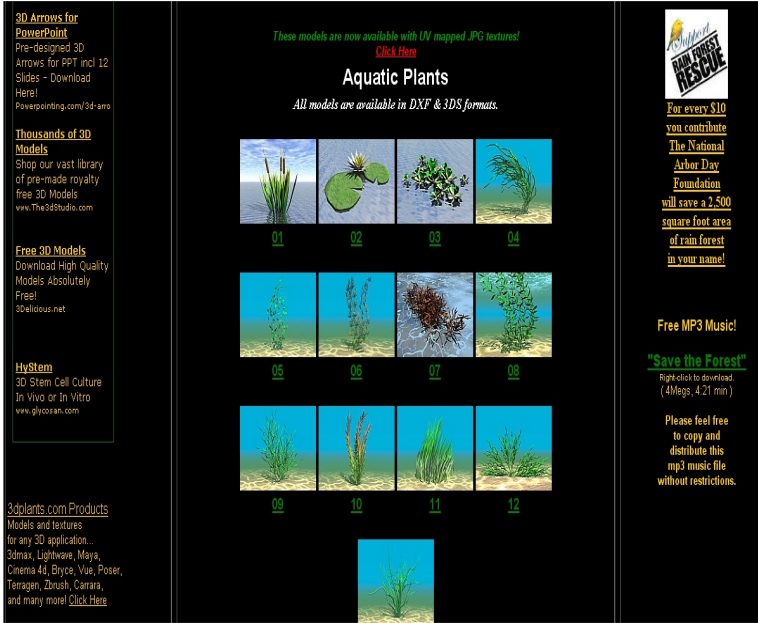




Cde.tsogu.ru

Официальный сайт Тюменского государственного нефтегазового университета. Его основная задача – помощь студентам при выполнении лабораторных работ с использованием 3D объектов. Контент отсортирован по категориям. Особенно интересно выглядит набор моделей деревьев и растений, выполненных в виде низкополигональных каркасов с наложенной текстурой более мелких объектов (листьев). Как и на предыдущих сайтах модели абсолютно бес-

платны и открыты для скачивания без регистрации. Есть только одно большое «Но». Львиная доля моделей хранится в формате max. Поэтому придётся потрудиться, прежде чем использовать их в своих задумках.



3dplants.com

Продолжаем рассматривать тематические сайты. На очереди 3dplants.com. Ресурс посвящён исключительно моделям и текстурам, которые связаны с растительным миром. Богатый набор мелких растений и крупных кустарников. Присутствует неплохая подборка текстур одной тематики. Модели представлены сразу в нескольких форматах, как правило, 3DS всегда есть в наличии. Приятно, что для скачивания не нужна регистрация и что еще приятнее, не мешает контекстная реклама. Па-

ра щелчков и файл уже у вас.

3d-animation.com.ar

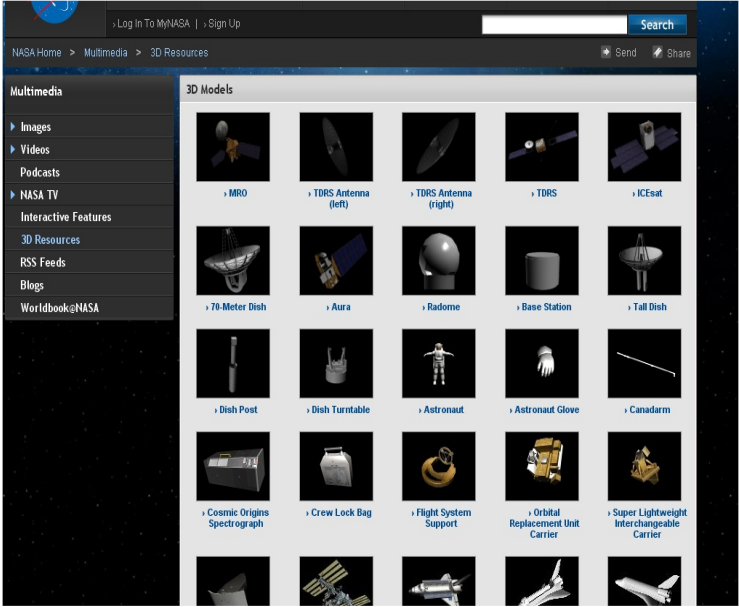
Несмотря на своё громкое название, ресурс имеет только статичные модели. Набор достаточно небольшой. Представлено несколько категорий (персонажи, оружие, машины...) до 10 моделей в каждой. Однако, и этот сайт можно рекомендовать разработчикам в силу того, что на нём имеются только низкополигональные модели. Очень удобной выглядит информация о количестве полигонов для каждого объекта. Все модели представлены в формате 3DS. Для

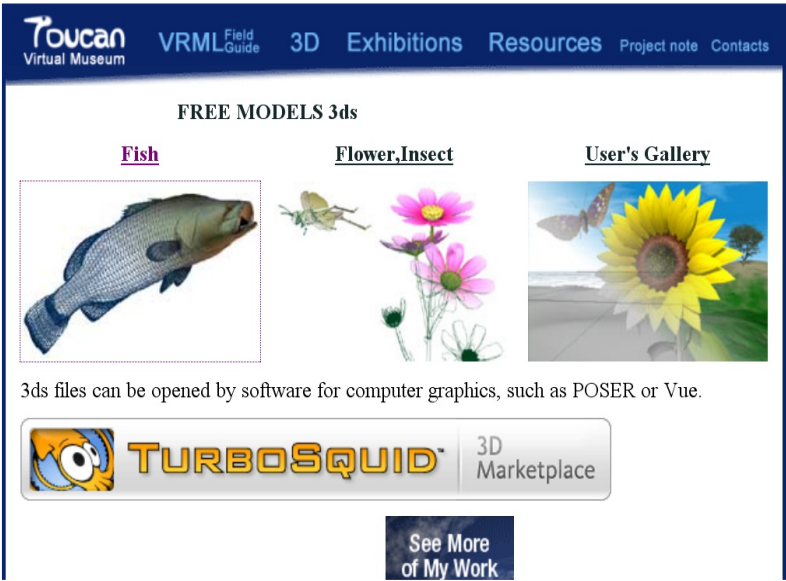
скачивания регистрация не нужна.



Nasa.gov

Официальный сайт организации NASA. Помимо специфической информации, сайт имеет подборку моделей космического характера, а также текстуры. Объекты «свалены» в один каталог, но наличие миниатюр и исчерпывающая информация о них сводит на нет этот беспорядок. Предпочитаемый формат – 3DS. В основном объекты представлены без текстур. В краткой аннотации указывается: формат, автор, количество полигонов (а также вершин и ребёр). Скачивание бесплатное и без регистрации.





Всё по восточному лаконично: щёлкаете по картинке и начинается загрузка. Все модели текстурированные.

Toucan.web.infoseek.co.jp

За столь поразительно длинным названием скрывается японский сайт с очень неплохой подборкой моделей рыб, растений и насекомых. Не пугайтесь, в отличие от основного контента сайта, который явно не рассчитан на европейца, раздел моделей использует английский язык. Вы выбираете в диалоговом окне одну из 3-х категорий, после чего открывается окно с миниатюрами. Какая-либо информация об объектах отсутствует.



Формат файлов: 3DS и max.

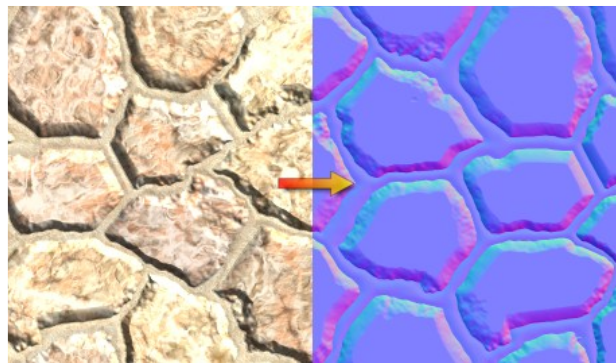
Planit3d.com

Красочный дизайн ресурса скрывает за собой несколько тысяч высококачественных моделей. Как и везде, контент отсортирован по тематическим категориям. Выбор чрезвычайно богатый: от техники до растений и животных. Щелчком по нужному наименованию открывается окно с миниатюрами моделей. К сожалению, никакой дополнительной информации об объекте, кроме названия, не предоставляется. Все модели достаточно детализированные и текстурированные. Регистрация для ска-



## ТЕКСТУРЫ

В народе говорят, «По одежке встречают...». Так и с играми. Ведь в первую очередь пользователи обращают внимание на графическую составляющую. И пусть игра хоть трижды имеет сногшибательный геймплей, непроработанная графика сведёт на нет многолетние труды разработчиков. Наверное,



>> Пример типичной normal map

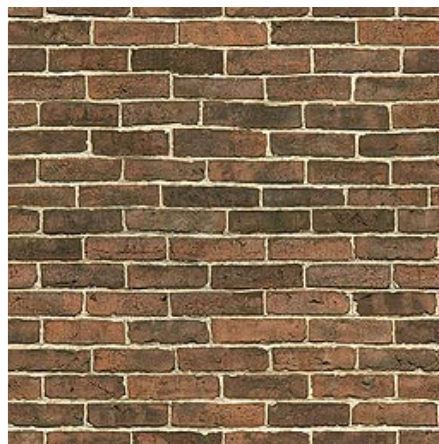
приоритетное значение в графике имеют не модели, а используемые текстуры. Какой угловатой модель бы не была, правильно подобранная текстура способна скрасить «огрехи» 3D. Существуют специальные технологии наложения текстур, имитирующих высокополигональные объекты. Речь идёт о разновидностях бампинга. Создание текстур дело нелёгкое, в следующих выпусках журнала мы обязательно осветим этот вопрос. Важно знать, что текстуры должны следовать следующим правилам и

рекомендациям:

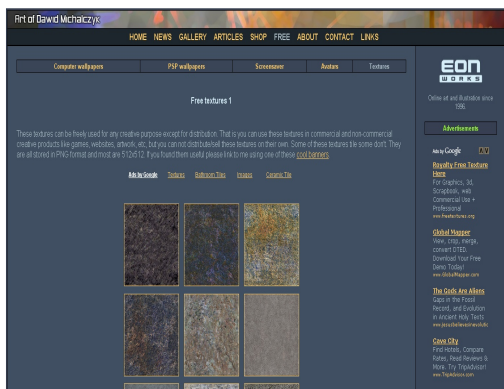
- предпочтительный формат Targa (tga) или PNG. Почему не распространённый JPEG? Во-первых эти форматы имеют хороший баланс между качеством и сжатием, во-вторых они поддерживают канал прозрачности (альфа-канал). К сожалению, JPEG этих свойств лишён;

- Разрешение должно быть кратным. К примеру: 512x512, 1024x1024 и т.д. Это обуславливается спецификой организации памяти 3D-ускорителей.

Кроме того, различают тайловые текстуры и UV-текстуры. Последние используются для «одевания» персонажа и в основном рисуются специально для конкретной модели. Тайловые представляют собой так называемые бесшовные текстуры, то есть такие, когда две соединённые вместе, образуют единый рисунок. Именно «бесшовные» мы и будем искать на просторах бескрайнего Интернета.

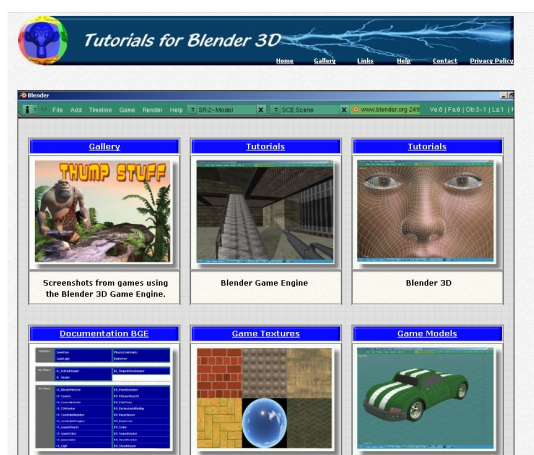


>> Бесшовная текстура



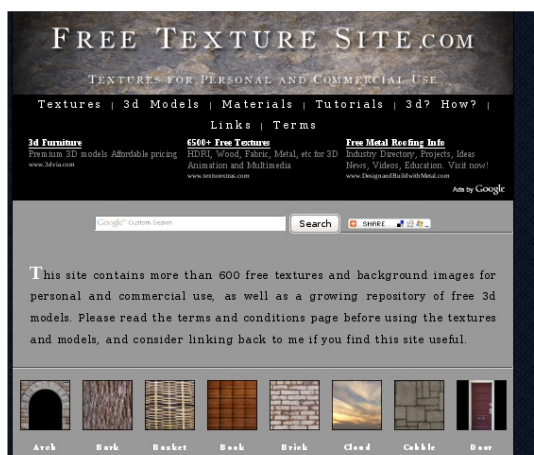
## Art.eonworks.com

Персональный сайт «Eon Works» арт-студии Дэвида Михальчик (Dawid Michalczyk). Среди всего прочего имеется небольшая коллекция тайловых текстур размером 1024x1024 и 512x512. Здесь можно найти поверхности камня, железа, некоторых видов пластика. Формат файлов PNG. Для скачивания регистрация не нужна.



## Tutorialsforblender3d.com

Сайт может быть интересным для разработчиков использующих игровой движок BGE (Blender), но не только. Здесь есть уроки, документация по BGE, некоторый набор моделей в формате blend и текстуры. Особое внимание уделите коллекции текстур. Это подборка качественных текстур, в основном строительных материалов. Самое главное, имеются готовые к употреблению рельефные карты (normal map). Большинство рельефных карт сопровождаются примером в формате blend (файл проекта редактора Blender). И последнее, есть готовые бесшовные текстуры для создания skybox и skydome.

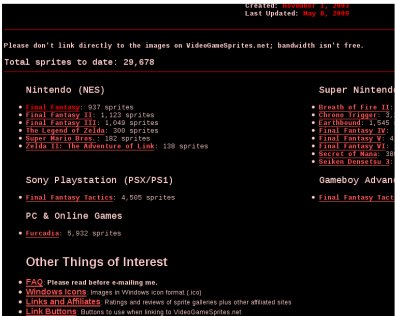


## Dougturner.net

Отличный ресурс содержащий большое количество текстур, как для развёртки UV, так и бесшовные. Именно на нём нам удалось найти такие полезные и редкие текстуры как для заливки моделей предметов обихода, деталей экстерьера. Весь контент предлагаемый сайтом, предназначен исключительно для использования в играх. Единственный минус — это хранение в формате JPEG. Печально, что даже те текстуры, где предполагается наличие альфа-канала, хранятся в этом формате. Впрочем, зная GIMP, вы с лёгкостью решите эту проблему.

# СПРАЙТЫ

Засилье 3D всё же не убило столь милые и красочные «плоские» игрушки. Мало того, направление казуалок с каждым годом крепнет и крепнет. Да и начинать карьеру игродела стоит с относительно несложных в исполнении игр. Итак, встречайте хит-парад ресурсов, посвящённых этому виду графики.



## Videogamesprites.net

Здесь вы сможете найти спрайты и фоны из популярных игр разных игровых систем, таких как NES, PSX, SNES, GBA. Файлы структурированы по играм, в которых они использовались. Регистрация на этом сайте не требуется.



## Sdb.drshnaps.com

Ещё более мощная коллекция двухмерных спрайтов. При первом взгляде на меню сайта поражает обилие названий игровых приставок, которые выступают тут в качестве основных каталогов. Даже если вам не требуются картинки 2D — загляните на него ради простого интереса. Для разработчика казуалок и двухмерных игр этот ресурс может представлять реальную ценность. Кстати, коллекции тематических спрайтов, как правило, хранятся в одном файле (формат PNG).

## Reinerstileset.4players.de

Этот ресурс хранит модели и спрайты из свободных игр. Да, да, помимо простого 2D вы сможете найти здесь хорошую подборку трёхмерных низкополигональных моделей. Что же касается спрайтов, то в основном они являются псевдо-трёхмерными (изометрическими). Всего имеется восемь глобальных категорий (2D): люди, животные, строения, окружение, транспорт, монстры и другие. На наш взгляд, этот ресурс можно смело назвать лучшим в этой области.



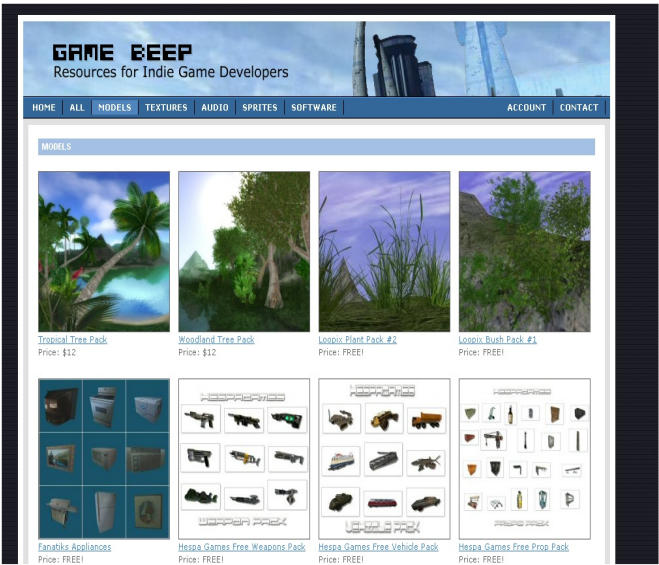


Bonus

Завершают нашу прогулку по просторам интернета ещё несколько сайтов. По некоторым причинам они вынесены за пределы рассмотренных категорий, но тем не менее представляют собой безусловно полезные для разработчика ресурсы.

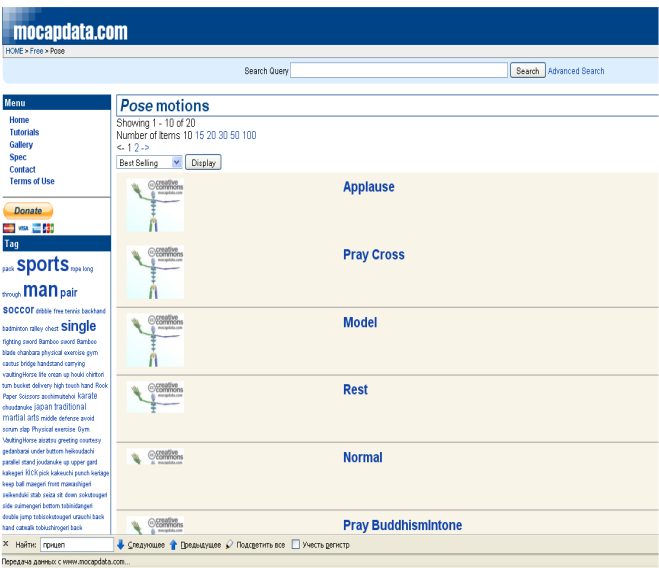
Gamebeep.com

Название сайта красноречиво гласит: «Ресурсы для инди-разработчиков». Здесь коллекции 3D моделей, двухмерных спрайтов, простых текстур и даже звуков. Но, внимание! Это коммерческий проект и большинство предложений платные. Правда, есть некоторое количество контента из категории «free». Качество продуктов на достаточно высоком уровне, как платных, так и бесплатных. Все объекты разделены по категориям и по сути представляют собой пакеты, характерные для определённого жанра. Цены не кусаются. В пределах 10-15 долларов.



Mocapdata.com

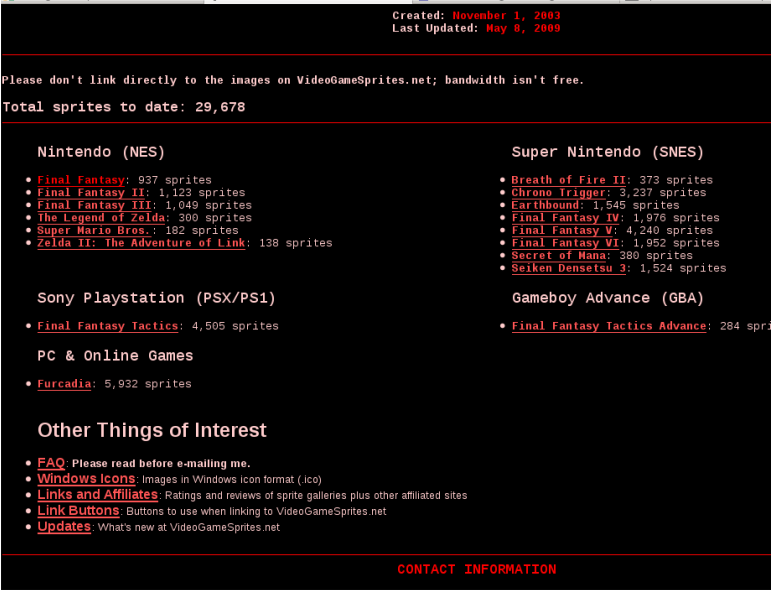
Вы слышали о технологии Motion Capture? Вкратце, это студийная запись видеоизображения актёра с нескольких камер с последующей обработкой для использования в программах трёхмерного моделирования. В итоге анимация персонажа получается максимально реалистичной. Сайт «Mocapdata» является хранилищем уже обработанных сцен. Имеются как коммерческие, так и бесплатные образцы. Всего в разделе «free» насчитывается более 4000 заготовок.



Для каждой анимации предлагается несколько форматов. К счастью, попу-



лярный редактор Blender умеет работать с некоторыми из них (форматы C3D и BVH). Для скачивания или покупки контента обязательно проходить регистрацию.



**Freegamearts.tuxfamily.org**  
Неудобный, неказистый, но содержащий немалое количество контента для разработчиков игр. Заметьте, что администрация сайта ориентируется только на пользователей unix-систем. В связи с этим, вы встретите на нём только те форматы файлов, которые могут быть прочитаны в GNU/Linux.

Итак, здесь вы сможете найти модели, анимацию, текстуры, звуки, некоторый инструментарий и многое другое. Но чтобы выбрать то, что необходимо, придётся положиться на свою интуицию. К примеру заглядываем в раздел текстур и видим перечень доступных в алфавитном порядке. Лишь только по названию файла можно угадать о его содержимом. Иконок, графических миниатюр и всё в таком роде вы не встретите ни на одной странице сайта. Естественно напрочь отсутствует элементарное описание.

Мы не стали бы рекомендовать этот ресурс пользователям со слабыми нервами и медленным соединением. И всё же, если вы готовы потерять немало времени на поиск нужной картинки или модели, не забудьте заглянуть на него!

Вам известны другие полезные ресурсы? Напишите нам и, возможно, они принесут пользу читателям журнала. Адрес редакции: [info@lingametechnologies.com](mailto:info@lingametechnologies.com)

LGT



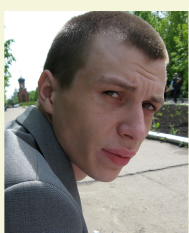
# В погоне за временем

## От редакции:

«Racer» является бесплатным кросс-платформенным автосимулятором. Физика, используемая в игре, создает неповторимые ощущения графического реализма. В Racer'e представлено множество трекков и моделей автомашин. Созданы широкие возможности для настройки игры. Именно поэтому Андрей Якушев

– администратор сайта об играх под Unix (GNU/Linux, Mac OS и FreeBSD, и другие Unix подобные ОС) [Unixgames.ru](http://Unixgames.ru) решил прокатиться на автомобилях Racer'a не в погоне за игровым временем, а с целью осмотреться по сторонам. Удалась ли прогулка по скоростной трассе?

## Автор



**Андрей Якушев**

Работает инженером в сфере информационных технологий. Главное в его профессии компьютеры и программирование. Является администратором сайта [Unixgames.ru](http://Unixgames.ru) - игры для Unix (Linux игры, Mac OS и FreeBSD). Увлекается автосимуляторами, музыкой и, немножко, моддингом

## Назад в прошлое

С игрой «Racer» я познакомился почти семь лет назад, случайно обнаружив дистрибутив на прилагаемом к игровому журналу диске (причем без всякого описания и упоминания в содержании). С удовольствием, отметив то, что игра является автосимулятором (всегда тепло относился к играм автомобильной тематики), я подключил свой руль от Thrustmaster и приступил к знакомству с игрой.

Первое, что поразило - это огромное для моей старенькой машинки количество FPS, что, впрочем, не изменилось и сейчас. Игра не требовательна к ресурсам компьютера. Второе - это огромное количество модов созданных поклонниками игры: более двухсот автомобилей и сотни трасс доступных на просторах паутины. Среди них встречались настоящие шедевры 3D-моделирования, способные поспорить по качеству исполнения с моделями из какого-нибудь Race Driver или Need For Speed. И, наконец, третье, это опции самой игры. Настроить можно было абсолютно все! Интерфейс, звуки, тонкие графические настройки, пове-



>> С любовью сделанная модель



>> Графическая составляющая выглядит весьма привлекательно

дение автомобилей, количество и положение камер на трассе, управление. Параметров несколько сотен! Это рай для человека, который любит все подгонять под свой вкус. Такого количества настроек не было ни в одной игре. Да и есть ли сейчас?!

## Наше время

Загрузив с официального сайта последние версии игры для всех доступных платформ (к слову, версию для MacOS X, толком протестировать не удалось, так как игра категорически отказывалась определять и

геймпад и руль), я с удовольствием отметил, что за время прошедшее с момента первого знакомства, в игре намного улучшилась физическая модель автомобиля. Она значительно подросла в графическом плане, а главное стала намного стабильней. Сейчас «Racer» - это автосимулятор, который и «едет» хорошо и выглядит приятно.

### Требования

#### ОС:

GNU/Linux,  
Windows, Os X

#### Железо:

Pentium III или  
AMD с  
частотой не  
менее 800МГц,  
видеокарта с  
поддержкой  
OpenGL; либо  
совместимый  
Mac.

**Сеть:** есть

**Разработчик:**  
Ruud van Gaal

#### Сайт:

[www.racer.nl](http://www.racer.nl)

Игровой процесс прост, впрочем, как и в любом другом автосимуляторе. Выбираем трассу, машину и вперед, наматывать круги. (рис. 5) Можно создать свой сервер в локальной сети и погонять вместе с товарищами. А если окажется, что у них нет руля или геймпада, то можно управлять автомобилем и с помощью мыши.

Искусственный интеллект в игре, к сожалению, практически не развит. Возможность соревноваться с компьютером есть, но, складывается впечатление, что боты не знают о других передачах, кроме первой. И при малейшем соприкосновении с другим автомобилем могут забыть про дорогу, уехав в сторону противоположную движению.

Графика ничем выдающимся не блещет. Новомодные спецэффекты, к сожалению, доступны только для Windows-версии. Пользователям GNU/Linux и Mac приходится терпеть графику уровня 2003-2005 года.



>> Немного пустынно, но на скорости не до этого



## Интересно

В июле 2009 года выпущена бета-версия игры Racer 0.8.3. В отличии от предыдущих версий, которые были основаны на физическом движке ODE, обновленная игра порадует игроков более точной имитацией физики движения объектов за счет движка Newton.

Но, я считаю, что в играх подобного жанра роскошная графика не столь важна. Ведь, когда сосредотачиваешься на быстром и точном прохождении круга, нет времени смотреть по сторонам.

Про звук ничего плохого сказать не могу. Рев двигателей приятно ласкает слух, а большего и не надо...

## Взгляд в будущее

Радует то, что разработчик Ruud van Gaal не забрасывает свой проект и регулярно обновляет игру, вводя новые фишки. Регулярно выходят аддоны в виде новых автомобилей и трасс. Вокруг игры выросло несколько сообществ во всем мире, в том числе и в России ([www.racer.cwx.ru](http://www.racer.cwx.ru)).



>> Камеру можно заставить смотреть и сверху

Так что «Racer» - это игра из разряда must have для любителей автомобильных гонок. К тому же конкурентов в среде Linux у нее практически нет, а размер дистрибутива не заставляет беспокоиться за дисковое пространство.

LGT

## У Вас есть любимая игра?

Расскажите о ней и, возможно, круг её почитателей значительно расширится. Пишите письма на адрес: [info@lingametechn.com](mailto:info@lingametechn.com)





# Интернационализация

## Научим программу разным языкам

### От редакции:

Всегда приятно, если интересная программа или игра умеет «разговаривать» на языке понятном для пользователя. Считается, что добавить поддержку иных языков не представляет особых сложностей. И все же есть определенные правила, следовать которым необходимо для корректной работы программы. Давайте познакомимся с таковыми для графической среды Gnome.

### Сначала разберемся

#### Автор



**Владислав  
Володин**

Имеет два высших образования. Увлекается книгами и программированием, в особенности для GNU/Linux с GTK+. Особое хобби — игра Dungeons and Dragons с друзьями по выходным. Ведет свой блог [vest-one.blogspot.com](http://vest-one.blogspot.com).

Сначала следует пояснить, что такое «Интернационализация» и чем она отличается от знакомого определения «Локализация». Основное отличие заключается в том, что приложение, поддерживающее интернационализацию физически представляет собой один исполняемый файл (с сопутствующими ему файлами с данными), которое равно запускается в операционной системе и отображает локализованную информацию в зависимости от настроек ОС [см. соответствующую информацию на сайте <http://ru.wikipedia.org/wiki/Интернационализация>, — прим. ред.]. Локализованное приложение отображает только тот язык, для которого оно было создано. Так, например, в программах Windows можно встретить фразы «Скачать русскую версию». Так вот, это были локализованные приложения.

### Переходим к делу

Наше приложение пока отображает только английскую локаль [см. статью «Autotools с нуля» в LGT от 07/2009, — прим. ред.]. Попробуем встроить в него интернационализацию управляемую библиотекой Glib. Для начала нам понадобятся пара пакетов: `intltool` и `gettext`. Они помогут избавиться от некоторой рутинной работы.

Подправим наш исходный код, и доведём его до следующего состояния (жирным шрифтом отмечены изменения в файле):

```

#include <config.h>
#include <stdio.h>
#include <glib/gi18n.h> – поддержка макросов gettext

int main(int argc, char** argv)
{
    setlocale (LC_ALL, ""); // устанавливаем текущую локаль, информацию
    берём с переменных окружения (это можно опустить в GUI программиро-
    вании)
    bindtextdomain (GETTEXT_PACKAGE, QUOD_LOCALEDIR); // устанавливаем
    каталог, где лежат файлы для разных языков
    bind_textdomain_codeset (GETTEXT_PACKAGE, "UTF-8"); // устанавлива-
    ем кодировку для вывода символов (благо Linux легко поддерживает
    Unicode)
    textdomain (GETTEXT_PACKAGE); // установка домена по умолчанию
    printf(_("Hello World!\n")); // отмечаем текст, который надо пере-
    вести с помощью макроса _()
    return 0;
}

```

После просмотра соответствующей документации становится ясно, что сам Glib локализован с помощью механизма gettext, поэтому он принят стандартом де-факто. Существуют разные макросы для выделения текста, требующего интернационализации. Например, для строк, к которым не может быть применён вызов gettext нужно применять макрос `N_()`. Дополнительную информацию возможно найти в документации: Glib, HOW-TO к intltool, в справке «Internationalising GNOME Applications». Стоит отметить, что информации много, но она весьма разрозненна.

Следующим этапом идёт создание каталога `PO`, где будут находиться переводы. Только перед этим необходимо подправить наши шаблоны:

```

~/gnome-quod$ mkdir po

configure.ac
# ===== initialization =====
AC_INIT([Gnome Quod],
[0.4.0],
[vest@mail.com],
[gnome-quod])

AC_CONFIG_SRCDIR([src/main.c])
AC_CONFIG_HEADERS([config.h])
AM_INIT_AUTOMAKE([-Wall -Werror dist-bzip2])
AC_DEFINE([NDEBUG], [], [Disable debugging information])

# ===== basic compiler settings =====
AC_PROG_CC
AM_PROG_CC_C_O
AC_HEADER_STDC

# ===== internationalization i18n =====
IT_PROG_INTLTOOL([0.40.4]) – требование библиотеки intltool не ниже

```

0.40.4

**GETTEXT\_PACKAGE=gnome-quod** — название пакета выносим в отдельную переменную

**AC\_DEFINE\_UNQUOTED([GETTEXT\_PACKAGE], ["\$GETTEXT\_PACKAGE"],** — которую делаем в качестве [The domain to use with gettext]) — директивы для препроцессора (см. config.h)

**AM\_GLIB\_GNU\_GETTEXT** — не совсем понятно, но скорей всего включает поддержку GETTEXT у Glib (или через)

**AC\_SUBST(GETTEXT\_PACKAGE)** — добавляет опции в компилятор для включения gettext

**QUOD\_LOCALEDIR=[\${datadir}/locale]** — указываем, где будут находиться локализации

**AC\_SUBST(QUOD\_LOCALEDIR)** — запоминаем этот путь и добавляем его в опции компилятора

**PKG\_CHECK\_MODULES([GLIB],[glib-2.0 >= 2.18])** — проверяем, что у нас стоит Glib нужной версии

```
# ===== generate files =====
```

```
AC_CONFIG_FILES([
```

```
  Makefile
```

```
  src/Makefile
```

```
  po/Makefile.in — выводим ещё один шаблон Makefile'a в новом каталоге po
```

```
])
```

```
AC_OUTPUT
```

В HOW-TO указывается, что вместо **AM\_GLIB\_GNU\_GETTEXT** можно использовать **AM\_GNU\_GETTEXT([external])**, а сам autoreconf предлагает воспользоваться **AM\_GNU\_GETTEXT\_VERSION**. Так, с первым вариантом я получил следующее предупреждение:

```
...
AC_DEFINE_UNQUOTED([GETTEXT_PACKAGE], ["$GETTEXT_PACKAGE"],
[The domain to use with gettext])
AM_GNU_GETTEXT_VERSION
...
```

```
$ autoreconf
```

```
...
autoreconf: configure.ac: AM_GNU_GETTEXT_VERSION is used, but not
AM_GNU_GETTEXT
```

А с другим вариантом:

```
autoreconf: configure.ac: AM_GNU_GETTEXT is used, but not
AM_GNU_GETTEXT_VERSION
```

Обратите внимание, что пакет ещё пока не до конца сконфигурирован и это просто демонстрация. В вашем случае ошибки могут быть несколько иными.

#### Важно!

В исходных кодах можно использовать как TAB-символы, так и пробелы. То же самое касается и шаблона configure.ac. Однако, в файле Makefile.am допускается использование только TAB.

Теперь давайте подправим другие файлы:

```
// Makefile.am
SUBDIRS = src po — добавляем лишний каталог
@INTLTOOL_DESKTOP_RULE@ — этот макрос позволяет intltool опреде-
лить, где какие файлы требуют перевода (в частности, в корневой ди-
ректории может находиться будущий ярлык программы).

INTLTOOL_FILES = \
intltool-extract.in \
intltool-merge.in \
intltool-update.in

EXTRA_DIST = $(INTLTOOL_FILES)

DISTCLEANFILES = \

intltool-extract \
intltool-merge \
intltool-update
# здесь указывается, что следующие шаблоны будут распространяться в
дистрибутиве без обработки. В случае очистки генерированные файлы
необходимо удалять вручную.

# эти строки я встречал не всегда. Они отвечают за очистку кеша, со-
здаваемого intltool
clean-local:
rm -f po/.intltool-merge-cache

// src/Makefile.am
bin_PROGRAMS = quod
quod_SOURCES = main.c

AM_CFLAGS = \
-I$(top_srcdir) \ — путь с заголовками уровнем выше (обычно для
config.h)
-DQUOD_LOCALEDIR="\${QUOD_LOCALEDIR}" \ — здесь лежат наши локали
$(GLIB_CFLAGS) — флаги для компиляции с Glib

quod_CFLAGS = \ — задаём флаги для конкретного исполняемого файла
(quod)
$(AM_CFLAGS)

quod_LDADD = \ — тоже самое, но для линковщика
$(INTLLIBS) \ — библиотеки intltool
$(GLIB_LIBS) — библиотеки Glib
```

В предыдущей статье описывалось использование макроса `AM_PROG_CC_C_O`. Именно, благодаря ему, указывались нужные флаги для конкретного бинарного файла. Теперь следует дописать небольшие файлы в каталог `PO`, и затем приступить к запуску скриптов. Процедура создания файлов немного автоматизирована:



## Кодовый блок

## Игра для Gnome

```
~/gnome-quod$ intltoolize - ЭТОТ скрипт создаст файл
po/Makefile.in.in
```

```
// вручную создаём po/LINGUAS
# please keep this list sorted alphabetically
#
ru — по алфавиту указываем какие языки будут присутствовать

// вручную создаём po/POTFILES.in
# List of source files containing translatable strings.
# Please keep this file sorted alphabetically.
src/main.c — здесь описываем все файлы, где нужен перевод (даже
файлы с ярлыками для рабочего стола)
```

Проверяем:

```
~/gnome-quod/po$ ls -l
```

итого 8

```
-rw-r--r-- 1 vest vest 51 2009-07-25 14:37 LINGUAS
lrwxrwxrwx 1 vest vest 34 2009-07-25 14:36 Makefile.in.in ->
/usr/share/intltool/Makefile.in.in
-rw-r--r-- 1 vest vest 114 2009-07-25 14:38 POTFILES.in
```

На самом деле Makefile.in.in это всего ~/gnome-quod/po\$ msginit --  
locale=ruнавсего ссылка. Далее попытаемся создать pot-файл, содержащий  
строки, нуждающиеся в переводе:

```
~/gnome-quod/po$ intltool-update --pot — выполнять это надо внутри
каталога po
```

```
~/gnome-quod/po$ cat gnome-quod.pot
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE
package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-07-25 14:40+0400\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: ../src/main.c:12
#, c-format
```

```
msgid "Hello World!\n"
msgstr ""
```

Вот так выглядит шаблон, созданный скриптом. Сам pot-файл не нужен, так как генерируется автоматически. То, что требуется от вас — это сделать копию файла с именем того языка, который будет содержать нужную локаль (в нашем случае ru.po). Сам файл хорошо документирован, но следует обратить внимание на то, что *CHARSET* необходимо исправить в *UTF-8*. Есть, конечно, другой способ создать шаблон нужной локализации средствами gettext (но про UTF-8 строки не следует забывать. Помните, наши изменения в исходном файле main.c):

```
~/gnome-quod/po$ msginit --locale=ru
```

Новый каталог сообщений должен содержать адрес электронной почты, чтобы пользователи могли присылать свои замечания по поводу переводов, а также чтобы сопроводители программ могли связаться с вами в том случае, если возникнут непредвиденные технические проблемы.

```
Which is your email address?
1 Vest@home.tula.net
...
9 vest@vest-desktop
```

```
Please choose the number, or enter your email address.
9 — я ввёл это просто так (на самом деле введите либо номер, либо
свой реальный адрес)
A translation team for your language (ru) does not exist yet.
If you want to create a new translation team for ru, please visit
http://www.iro.umontreal.ca/contrib/po/HTML/teams.html
http://www.iro.umontreal.ca/contrib/po/HTML/leaders.html
http://www.iro.umontreal.ca/contrib/po/HTML/index.html
```

**Создано ru.po. — ВОТ ЭТО САМОЕ ГЛАВНОЕ.**

```
~/gnome-quod/po$ cat ru.po
# Russian translations for Gnome Quod package.
# Copyright (C) 2009 THE Gnome Quod'S COPYRIGHT HOLDER
# This file is distributed under the same license as the Gnome Quod
package.
# Vladislav Volodin <vest@vest-desktop>, 2009.
#
msgid ""
msgstr ""
"Project-Id-Version: Gnome Quod 0.4.0\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2009-03-04 19:08+0300\n"
"PO-Revision-Date: 2009-03-04 19:18+0300\n"
"Last-Translator: Vladislav Volodin <vest@vest-desktop>\n"
"Language-Team: Russian\n"
```



## Кодовый блок

## Игра для Gnome

```
-I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include -I.. -g -O2 -o
quod quod-main.o -lglib-2.0
```

```
...
~/gnome-quod$ quod
Привет Мир!
$
```

Вот и все, можно поздравить вас с первым «интернациональным» приложением. Хотя вы можете попробовать создать пакет с помощью команды *make dist*.

Не забывайте обновлять *POTFILES.in*, когда добавляете новые исходные файлы в каталог *SRC*. Команда *intltool-update ru* поможет обновить файлы с переводом, если в них были добавлены новые строки. При этом сама предупредит о числе переведённых или не переведённых строк. Вот так выглядит вывод немного измененного кода:

```
// main.c
printf(_("Hello World!!\n")); — лишний восклицательный знак

~/gnome-quod$ cd po
~/gnome-quod/po$ intltool-update ru
... завершено.
0 переведенных сообщений, 1 неточный перевод.

~/gnome-quod/po$ cat ru.po

#: ../src/main.c:13
#, fuzzy, c-format — здесь команда исправила, что наше исходной со-
общение теперь другое. После исправления это слово можно стереть
msgid "Hello World!!\n" — ведь появился восклицательный знак
msgstr "Привет Мир!!\n" — добавим его здесь
```

LGT

## В следующий раз:

Как видите, работать с консолью можно, но не совсем удобно. В следующий раз мы доработаем проект так, чтобы он использовал библиотеку *GTK+* и настроим *IDE* для удобства управления исходным кодом на примере *Code::Blocks* (<http://codeblocks.org/>).



# SDL & OpenGL

## Первые шаги в создании своего движка

### От редакции:

Современная игра представляет собой сложный механизм, где в едином сплаве трудятся детали графики, физики, звука... Всё это хозяйство принято называть одним лишь словом — «движок». По определенным причинам многие команды разработчиков предпочитают иметь дело со сторонними движками, всецело посвящая рабочее время своему проекту. Тем не менее, понятие принципов работы игровых движков или даже создание собственного может дать неоценимый опыт своему развитию. И кто его знает, возможно именно Ваш движок сумеет потеснить известные бренды с их пьедесталов!

### Постановка задачи

#### Автор



**Кирилл  
Баженов**

Любит  
программирова-  
ть под Linux.  
Увлекается  
трёхмерной  
графикой,  
играми и ...  
OpenGL.

Любая игра имеет в своём составе центральный программный код, который называют ядром или игровым движком. На его совести лежит обработка скриптов, графики, сети, физики, интерфейса и многого другого. Понятное дело, что проще бывает взять уже готовый движок и использовать его для своей игры. Но мы же не такие?

Итак, в течение нескольких уроков мы с вами познакомимся и научимся работать с милой парочкой SDL и OpenGL, так как именно они позволяют выжать из имеющегося железа все соки, не «заглядывая ему под капот».

### Приятно познакомиться, Mr... Lib

*SDL, Simple DirectMedia Layer*, — свободная, кроссплатформенная, мультимедийная библиотека, разработанная неким Sam Lantinga для своих внутренних нужд еще в далеком 1997 году. Однако концепция и реализация кода были столь удачны, что SDL полюбилась многим разработчикам как коммерческих, так и свободных игр. В отношении GNU/Linux эта библиотека является едва ли не самым лучшим выбором для создания ядра игры. И вот почему.

Для вызова функций библиотеки можно использовать свой любимый язык программирования, такие как: C, C++, D, Ada, Erlang, Java, Lua, Pascal, Perl, PHP, Pike, Python, Ruby и некоторые другие. Так как библиотека

кроссплатформенная, то написанный код с успехом можно портировать на иную платформу, список которых тоже впечатляющий. Но главное не это. SDL без особого труда позволяет работать со звуком, графикой, сетью и специфическими устройствами ввода. И хотя библиотека изначально писалась для работы с двумерной графикой, она умеет работать с 3D через API OpenGL. Более подробную информацию Вы можете найти на официальном сайте [libsdl.org](http://libsdl.org).

## Начальные шаги

Первое, что нам нужно — установить SDL. Можно скачать с сайта последнюю версию, и по традиции «./configure && make sudo make install», а можно применить более простой вариант — установка через менеджер пакетов своего дистрибутива.

Итак, вы установили SDL, можно начинать работу. Использовать мы будем язык программирования C++. Почему не C? Потом мы будем писать очень удобные классы для рисования шрифтов, проигрывания музыки, видео и так далее. C++ в этом плане намного удобнее.

Ну давайте тогда напишем первый исходник!

Для начала необходимо подключить заголовочные файлы SDL и OpenGL:

```
#include <SDL/SDL.h>
#include <GL/gl.h>
#include <GL/glu.h>
```

Мы будем использовать потоки, следовательно:

```
#include <iostream>
using namespace std;
```

Далее следует описать прототипы некоторых используемых функций. В данном случае — подготовка кадра к рендеру и обработка самого кадра. Почему нужна подготовка кадра к рендерингу? Подобное разделение позволит более наглядно объяснить процесс на примере и более логично разделить понятия графики и игровой механики:

```
void prepareFrame();
void renderFrame();
```

Также нам потребуются прототипы функций изменения проекции (подробности ниже):

```
void setPerspective();  
void setOrtho();
```

Следующим этапом будут данные таймера. Таймер — очень важный элемент, так как на его плечи ложатся правильные расчёты скорости рендеринга и игровой механики. Таймер должен регулировать скорость смены кадров, чтобы игра работала одинаково на разных компьютерах. В SDL таймер реализуется через регистрацию функции- «колбэка».

Данная функция будет вызываться один раз в заданный промежуток времени.

Рассмотрим некоторые параметры. *Interval* — передаётся в функцию SDL, и её надо вернуть, а так не используется. Второй параметр, по документации, добавлен в целях совместимости с разными ОС. Нам он также не критичен. Следует описать глобальную переменную *myTimerCallbackParam* (из тех же соображений):

```
Uint32 myTimerCallback(Uint32 interval, void*);  
void* myTimerCallbackParam;
```

Следующая переменная — счётчик FPS, то есть частоты кадров в секунду:

```
int fps = 0;
```

Для контроля за временем нужны две переменные: время, когда был нарисован текущий кадр (*currentTime*) и время предыдущего кадра (*lastTime*). Отняв *currentTime* из *lastTime* получим время, затраченное на рисование кадра:

```
float lastTime = 0.0f;  
float currentTime = 0.0f;
```

Попробуем для начала нарисовать простейший треугольник и повернуть его по осям координат. Используем для этого следующую переменную:

```
float triangleRot = 0.0f;
```

Теперь самое интересное — реализация функции *main()*:

```
int main(int argc, char** argv)  
{
```

В самом начале идёт инициализация SDL. Функция *SDL\_Init()* принимает флажок, служащий подсказкой инициализации, и возвращает -1 в случае ошибки.

Доступные параметры:

SDL\_INIT\_TIMER – инициализация таймера  
 SDL\_INIT\_VIDEO – инициализация графического режима окна  
 SDL\_INIT\_AUDIO – инициализация аудио подсистемы  
 SDL\_INIT\_CDROM – для работы с CD  
 SDL\_INIT\_JOYSTICK – для работы с джойстиком  
 SDL\_INIT\_NOPARACHUTE – инициализация обработчика сигналов, то есть, если программа падает, например получив SIGSEGV, то SDL очистит за собой память корректно.  
 SDL\_INIT\_EVENTTHREAD – инициализация многопоточности.  
 SDL\_INIT\_EVERYTHING – инициализация всего вышеперечисленного.

Для самого начала нам понадобится только видео подсистема и таймер, так что не будем пока что останавливаться на других.

```

if (SDL_Init(SDL_INIT_VIDEO|
             SDL_INIT_TIMER) < 0)
{
    cout << "Cannot init SDL: "
          << SDL_GetError() << endl;
    exit(1);
}

```

Успешная инициализация говорит о том, что ресурсы выделены и уже находятся в работе. Следовательно, их необходимо освободить при завершении работы программы. Возникает вопрос, а как же узнать, когда завершится работа?

Есть очень изящный подход к решению этой проблемы — функция *atexit()*. Она регистрирует функцию, которую следует вызывать перед выходом из программы.

```
atexit(SDL_Quit);
```

Далее следует установка начальных атрибутов контекста OpenGL: двойная буферизация, цветовая модель RGB16 (R5G6B5)

```

SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, true);
SDL_GL_SetAttribute(SDL_GL_RED_SIZE, 5);
SDL_GL_SetAttribute(SDL_GL_GREEN_SIZE, 6);
SDL_GL_SetAttribute(SDL_GL_BLUE_SIZE, 5);

```

А вот следующий кусок кода устанавливает видеорежим. Первые два параметра — размеры окна, третий — глубина цвета. Последний параметр являет собой флажок инициализации окна.

Тут возможно много вариантов, но нам нужен *SDL\_OPENGL* (или *SDL\_OPENGL | SDL\_FULLSCREEN* для полноэкранного режима). Функция возвращает указатель на *SDL\_Surface*, или *NULL* в случае ошибки:

```

if (SDL_SetVideoMode(800, 600, 16, SDL_OPENGL) == NULL)
{
    cout << "Cannot set video mode: "

```



```
<< SDL_GetError() << endl;
exit(1);
}
```

SDL также даёт доступ к некоторым параметрам окна. В данном случае давайте установим иконку окна и заголовок. *SDL\_WM\_SetCaption* принимает два параметра: первый — заголовок окна, второй — заголовок на панели задач. *SDL\_WM\_SetIcon* принимает указатель на картинку (*SDL\_Surface*), и маску картинки. Если мы передадим NULL то форма иконки будет либо квадратной, либо соответствовать альфа каналу картинки (в следующей статье мы познакомимся с использованием маски). *SDL\_LoadBMP* служит для загрузки картинки формата BMP:

```
SDL_WM_SetCaption("My Cool Linux Game!", "game");
SDL_WM_SetIcon(SDL_LoadBMP("icon.bmp"), NULL);
```

## OpenGL

Вот и все, теперь можно браться и за OpenGL. Для начала мы настроим базовую функциональность: цвет очищения экрана и некоторые базовые параметры (рассматривать подробности пока не будем).

```
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glClearDepth(1.0f);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LESS);
glShadeModel(GL_SMOOTH);
```

Далее настроим перспективную проекцию:

```
setPerspective();
```

Нам нужно получить текущее (системное) время. Для этого воспользуемся функцией *SDL\_GetTicks()*:

```
currentTime = SDL_GetTicks();
```

Функция *SDL\_AddTimer* добавляет таймер в программу. Первый параметр — частота срабатывания таймера в миллисекундах, то есть частота вызова функции *myTimerCallback*. Второй параметр — указатель на функцию-«колбэк» (то есть на нашу). Третий — в целях совместимости. Обратите внимание, что значение 1000 было подобрано эмпирическим путём. Попробуйте поэкспериментировать с ним.

```
SDL_AddTimer(1000, myTimerCallback, myTimerCallbackParam);
```

И вот мы подошли к реализации главного цикла. Здесь происходит обработка событий, обработка механики игры и собственно само рисование .

Для начала нужны две переменные: `done` будет служить сигналом для выхода из игры;

```
event — буфер событий SDL.
bool done = false;
SDL_Event event;
```

Собственно, главный цикл:

```
while (!done)
{
```

Нам потребуется вложенный цикл для обработки очереди событий. Для буферизации события из очереди есть функция `SDL_PollEvent`. Она принимает указатель (ссылку) на буфер события и записывает туда нужную информацию. Возвращает `TRUE`, если в очереди есть ещё события, иначе — `FALSE`.

```
while (SDL_PollEvent(&event))
{
    switch (event.type)
    {
```

При закрытии окна генерируется событие `SDL_QUIT` (не путайте с функцией `SDL_Quit()`). Чтобы толерантно отнестись к пользователю, нужно завершить работу программы (то есть прервать цикл). Остальные события пока игнорируем:

```
        case SDL_QUIT:
            done = true;
            break;
        default:
            break;
    }
}
```

Далее идёт процесс рендеринга: сначала готовим кадр, потом его рисуем (в задний буфер).

```
prepareFrame();
renderFrame();
```

Чтоб отобразить нарисованное на экране, следует поменять местами передний и задний буфер:

```
SDL_GL_SwapBuffers();
```

Один кадр нарисован — увеличим счётчик кадров на 1.

```
    fps++;
}
```

Всё, закончили!

```
    return 0;
}
```

Теперь займёмся нашими прототипами.

Функция *prepareFrame()* готовит кадр к рендерингу. У нас это будет расчёт time-based движения треугольника):

```
void prepareFrame()
{
    float elapsedTime = (currentTime - lastTime) / 1000;
    triangleRot = (triangleRot >= 360.0f) ?
    0.0f : triangleRot + elapsedTime * 0.1f;
}
```

Теперь приступаем к функции, которая рисует кадр. Сначала очищаем экран, потом настраиваем перспективную проекцию и рисуем треугольник. Но для начала надо усвоить понятие матрицы OpenGL.

Матрица - это математический объект, записываемый в виде прямоугольной таблицы чисел (или элементов кольца) и допускающий алгебраические операции (сложение, вычитание, умножение и др.) между ним и другими подобными объектами. В OpenGL матрицы бывают нескольких видов, мы пока рассмотрим только *MODELVIEW*. Эти матрицы содержат в себе трансформацию относительно текущей матрицы (*MODELVIEW* или другой).

Основные функции для работы с матрицами:

- *glLoadIdentity()* - загружает единичную матрицу. Необходимо для инициализации начала координат;
- *glPushMatrix()* - запоминает состояние текущей матрицы. Необходимо начать рисовать объект со своей системой координат;
- *glPopMatrix()* - освобождает стек текущей матрицы. Необходимо чтобы последующие объекты не рисовались в системе координат текущего объекта;
- *glMatrixMode()* - сложная для понимания функция. В общем, она преобразует текущую матрицу для специфических целей. Например: *glMatrixMode(GL\_PROJECTION)* — преобразование для настройки проекции, *glMatrixMode(GL\_MODELVIEW)* — для рисования и т.д. Позже мы рассмотрим текстурные матрицы и многие другие. Обратите внимание, что после *glMatrixMode* следует вызвать функцию *glLoadIdentity()*.

Трансформация объекта:

- `glTranslatef()` - переносит объект в указанные координаты;
- `glRotatef()` - вращение объекта. Имеет несколько параметров: угол поворота и коэффициенты поворота по осям. Таким образом запись `glRotatef(90.0f, 1.0f, 0.0f, 0.0f)` повернёт объект на 90 градусов по оси X. В этом случае удобнее использовать кватернионы, но, пожалуй, мы сейчас обойдемся без них;
- `glBegin()` - заносит в стек тип примитива и начинает рисовать объект. Рисуются следующие вершины, заданные с помощью `glVertex` (см. ниже).

Возможные флаги:

```
GL_POINTS — рисовать точки;  
GL_LINES — рисовать линии;  
GL_LINE_LOOP — рисовать замкнутую линию;  
GL_LINE_STRIP — рисовать ломаную линию;  
GL_TRIANGLES — рисовать треугольники;  
GL_TRIANGLE_STRIP — полосы треугольников (подробнее в следующих  
статьях);  
GL_TRIANGLE_FAN — веер из треугольников (подробнее в следующих  
статьях);  
GL_QUADS — квадраты;  
GL_QUAD_STRIP — полосы из квадратов (подробнее в следующих  
статьях);  
GL_POLYGON — многоугольник.
```

`glVertex` — интересная функция. Есть несколько разновидностей. Общий вид:

`glVertex[количество аргументов][тип аргументов]`. Например: `glVertex3f` задаёт вершину в виде трёх координат типа `float`, `glVertex2f` — двух координат типа `float`. Имеется набор предустановленных суффиксов:

- `d` — тип `Double`(с плавающей точкой, двойная точность(64 знака после запятой));
- `f` — тип `Float`(с плавающей точкой, 32 знака после запятой);
- `i` — тип `Integer` (беззнаковое целое, 8 байт);
- `s` — тип `Short`(знаковое, 2 байта).

`glColor` — Задаёт цвет для окраски вершины. Эта функция также имеет несколько разновидностей, аналогично `glVertex`. В качестве параметров



принимается яркость каждого цвета (RGB+альфа). Цвета смешиваются и вершина окрашивается в результирующий цвет. Например, значение (1 1 1 0) означает черный цвет с отключенным альфа-каналом. Имеющиеся суффиксы:

- b — тип Byte(знаковое, 1 байт)
- d — тип Double(с плавающей точкой, двойная точность)
- f — тип Float(с плавающей точкой)
- i — тип Integer(знаковое целое)
- s — тип Short(знаковое, 2 байта)
- ub — тип Unsigned Byte(беззнаковое, 1 байт)
- ui — тип Unsigned Int(беззнаковое целое, 8 байт)
- us — тип Unsigned Short(беззнаковое, 2 байта)

*glEnd()* - заканчивает рисование объекта.

Итого...

```
void renderFrame()
{
    glClear(GL_COLOR_BUFFER_BIT |
    GL_DEPTH_BUFFER_BIT);
    setPerspective();
    glPushMatrix();
    glTranslatef(0.0f, 0.0f, -5.0f);
    // Так, конечно, фигуры никто не крутит,
    // но выглядит интересно!
    glRotatef(triangleRot, 1.0f, 1.0f, 1.0f);
    glBegin(GL_TRIANGLES);
    glColor3f(1.0f, 0.0f, 0.0f); glVertex3f(0.0f, 1.0f, 0.0f);
    glColor3f(0.0f, 1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f); glVertex3f(1.0f, -1.0f, 0.0f);
    glEnd();
    glPopMatrix();
}
```

После рендеринга 3D-графики, обычно рисуется двухмерная (к примеру, интерфейс):

```
    setOrtho();
    // Рисуем двухмерную графику...
}
```

Теперь давайте напишем «колбэк» для таймера. Наша функция будет

вызываться раз в 1000 миллисекунд. Здесь производительность очень важна, поэтому мы только обновим время и FPS.

```
Uint32 myTimerCallback(Uint32 interval, void*)
{
    cout << "FPS = "
        << fps << endl;
    // Когда наша «секунда» прошла,
    // нужно сбросить счётчик FPS.
    fps = 0;
    // Тут стандартные расчёты времени...
    lastTime = currentTime;
    currentTime = SDL_GetTicks();
    // И вернуть interval, как требует SDL
    return interval;
}
```

Теперь настало время поговорить о проекциях. Проекции в OpenGL бывают двух видов: перспективная и ортографическая. Главная особенность перспективной проекции — объекты уменьшаются при увеличении расстояния между ними и точкой обзора. Ортографическая проекция применяется при рисовании 2D графики, например игровых меню. Главная особенность — объекты не уменьшаются при отдалении, но могут пропасть при достижении расстояния отсечения.

Функция для расчёта и применения перспективной проекции:

```
void setPerspective()
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
```

Для начала установим поле просмотра. Поле просмотра — квадратная область экрана, в котором будет происходить рисование. Первые два параметра — координаты левой верхней точки квадрата, второй — координаты правой нижней точки (в пикселях). Очень удобная функция, позволяет реализовывать несколько камер (вид сверху, вид сбоку и т.д.), широко применяется в гоночных играх и многопользовательских не сетевых играх. Установим поле просмотра в целое окно, то есть используем в качестве вторых координат размеры окна.

```
glViewport(0, 0, 800, 600);
```

Следующая функция рассчитывает перспективную проекцию. Первый параметр — поле обзора (в градусах), второй — коэффициент соотношения сторон, обычно равен размеру поля просмотра по X разделенного на размер поля просмотра по Y (не путать с размерами окна). Третий параметр

определяет ближнее расстояние от точки обзора для отсечения объектов. Четвёртый — аналогично, только он определяет дальнее расстояние:

```
gluPerspective(60.0f, GLfloat(800/600), 0.1f, 3000.0f);
```

Ну и вернём исходные параметры.

```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();
```

Когда рисуем 3d-графику, нужно включать сортировку объектов по глубине.

```
glEnable(GL_DEPTH_TEST);  
}
```

Расчёт ортогографической проекции ничуть не сложнее:

```
void setOrtho()  
{  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    glViewport(0, 0, 800, 600);  
    // Второй и третий параметры  
    // это размеры поля просмотра  
    glOrtho(0, 800, 600, 0, 0, 1);  
    glMatrixMode(GL_MODELVIEW);  
    glLoadIdentity();  
}
```

При рисовании 2D-графики сортировка по глубине не нужна.

```
glDisable(GL_DEPTH_TEST);  
}
```

И последнее, нам нужно скомпилировать исходники! Для этого (надеюсь что у Вас установлен gcc 4.2+ и g++) мы воспользуемся следующей командой в консоли:

```
$ g++ -o lesson0 lesson0.cpp -lGL -lGLU -lSDL
```

В соответствующем IDE создайте новый проект, и в опциях линкера (компоновщика) добавьте ключи -lGL -lGLU -lSDL, или добавьте в опциях следующие разделяемые библиотеки: GL GLU SDL. Вот и всё!

LGT

### В следующий раз:

*Мы займемся прорисовкой трёхмерных примитивов и научимся работать с текстурами.*

# Поработаем садовником

*Создание растений и деревьев для игрового движка Blender*

## От редакции:

В прошлом выпуске журнала мы рассказывали об одном из способов создания ландшафта для игрового движка Blender. Несмотря на полученную качественную картинку, ей для выразительности не доставало мелких деталей. О том, как за несколько шагов можно посадить целую плантацию цветов и вырастить рощу деревьев нам поведаёт Андрей Кондратьев, не понаслышке знакомый с этим искусством.

### Автор



**Андрей Кондратьев**

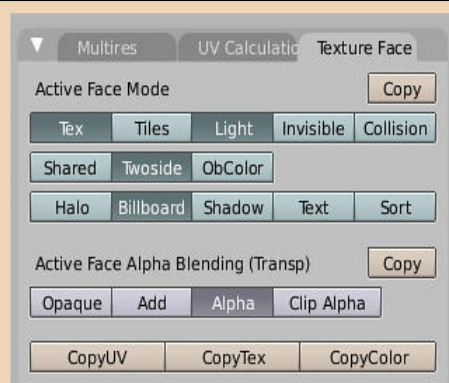
Работает конструктором в области машиностроения. С ранних лет жизненным девизом являлась фраза «Я создаю!». В детстве любил рисовать, лепить фигурки из пластилина, возиться с радиотехникой. Не большой любитель компьютерных игр, но при виде их стабильно возникает желание создать нечто подобное.

## Начнем с малого

Существует несколько способов создания растительности для BGE, конечно всё зависит от поставленной задачи и жанра игры. В первую очередь рассмотрим создание травы и мелкого кустарника.

Следует сразу отказаться от попытки трёхмерного моделирования этих деталей, так как большой массив объектов негативно скажется на производительности игры. Для вывода подобных объектов традиционно используется механизм Billboard. Всё очень просто. Billboard - это двухмерный спрайт, который всегда повернут «лицом» к игроку, вне зависимости от его положения и точки зрения камеры. В нашем случае достаточно использовать обычный примитив Plane с натянутой текстурой. Где взять текстуру? Можно скачать из интернета или вручную нарисовать в том же Gimp. Важно запомнить, что полученное изображение куста должно находиться на прозрачном фоне. Итак, приступим...

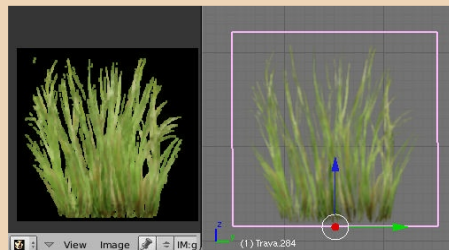
### Шаг 1. Создаем куст



Создаём плоскость (**ADD** → **Mesh** → **Plane**). Переходим в режим EditMode (<TAB>). Нажимаем клавишу <U> для вызова меню **UV Calculation**, затем выбираем пункт Unwrap. Нажимаем <F9> и на вкладке **Texture Face** нажимаем кнопку **BillBoard**. Полигон плоскости необходимо расположить так, чтобы центр mesh-объекта был посередине нижней грани плоскости.



## Шаг 2. Накладываем текстуру



Переключаем режим 3D-окна в Textured. Делим окно на две части. Одному из окон устанавливаем тип UV/Image Editor. В меню данного объекта выбираем пункт **Image -> Open** для загрузки заготовленной текстуры. Вот и всё, кустарник готов.

## Ветер с моря дул...

После того как зелёные холмики игровых просторов зарастут травой, вам захочется как-то оживить пейзаж, добавив колыхание кустов при порывах ветра. Есть несколько способов анимации растительности: использование физики мягких тел, анимация текстуры или ручная подгонка с помощью арматуры. Рассмотрим последний вариант, так как он достаточно прост в исполнении.

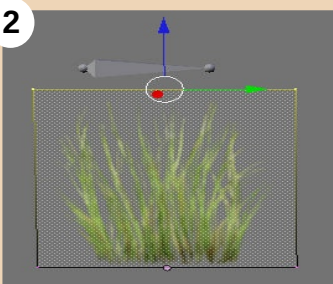
### Внедрение скелета

1



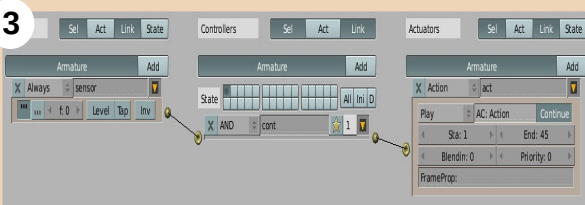
Создаем арматуру (для однополигональных объектов достаточно будет одного звена). Привязываем её к «билборду» (необходимо выделить объект, затем арматуру и нажать **<Ctrl>+<P> -> Armature -> Name Groups**).

2



Выделяем объект и переходим в режим **Edit Mode** (**<TAB>**). Отмечаем две верхние точки. Нажимаем **<F9>** для перехода на панель редактирования. На вкладке **Link and Materials** нажимаем клавишу **Assign**.

3



Выходим из режима редактирования (**<TAB>**), выделяем арматуру и переключаемся в режим **Pose Mode**. Фиксируем начальную позу: **<I> -> Loc**, переходим на 20 кадр и немного

перемещаем кость вдоль необходимой оси. Повторное нажатие **<I> -> Loc** зафиксирует позу. Переключаемся на 45 кадр, арматура переходит в исходную позицию и снова фиксируется тем же способом. Возвращаемся в режим объекта.

# LINUX GAMES TECHNOLOGIES

№2 (сентябрь 2009)

## Дорогие читатели!

*Если у Вас есть интересная тема или Вы являетесь разработчиком игрового проекта - присоединяйтесь к нашим авторам. Ваш личный опыт может помочь новичкам вступить на сложный, но интересный путь Linux Games Technologies.*

***Давайте вместе делать журнал!***

Перепечатка любых материалов журнала возможна только с разрешения редакции.

Редакция: [info@lingametech.com](mailto:info@lingametech.com)