



LINUX GAMES TECHNOLOGIES

Игровой

фейерверк

А также...

- **OpenGL+SDL**
- **F.P.C.**
- **Локализация**
- **Моддинг**



С Новым Годом!

Здравствуйте, друзья!

Осталось совсем немного времени до вступления в 2010 год. Уходящий год был достаточно тяжёлым, как для нашей команды, так и для всего мира в целом. Конечно, хотелось бы, чтобы старый год унёс с собой все накопившиеся проблемы, а новый начался с чистой страницы. В некотором роде исполнение этого желания зависит от всех нас. Примите самые сердечные поздравления от команды LGT!

*Пусть Новый Год стучится к Вам,
И счастьем дом наполнится.
И всё, о чём мечтали Вы,
Пусть в этот год исполнится!*

Впереди длинные новогодние каникулы. Неужели даже в эти праздничные дни заниматься рутинными делами? Нет и ещё раз нет! Поэтому пятый выпуск журнала особенный. Здесь вы не увидите уже привычные рубрики, такие как: «новости», «игроотека», «обзоры»... Точнее они все объединены в один большой раздел под кодовым названием «Игровой фейерверк». Как вы догадались, разговор пойдёт об играх. Последние месяцы года всегда бывают насыщены всевозможными релизами. Не отстает от иных платформ и наш горячо любимый Linux. В редакцию журнала поступило немало пресс-релизов новых проектов. Мы просмотрели и отобрали самые горячие игры, с которыми можно интересно провести свободное время. Тот же, кто не желает оставлять в покое свои собственные проекты, найдёт в журнале продолжения популярных уроков.

Ещё раз с Новым годом и хороших выходных!

С уважением,
Андрей Прахов

Редакция:

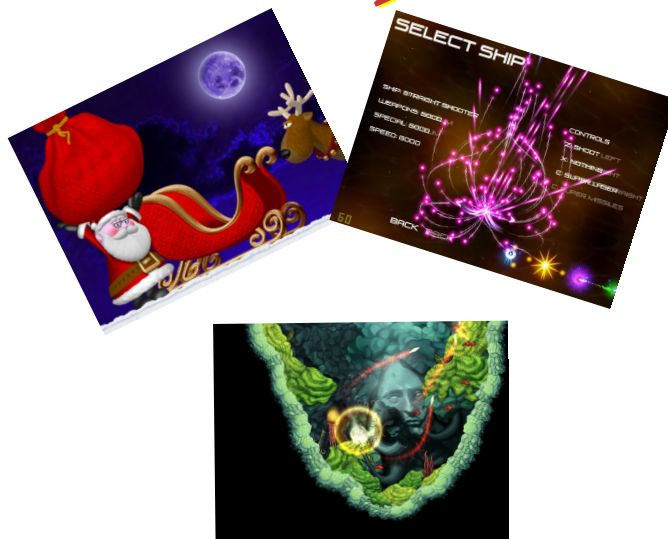
Андрей Прахов, Светлана Капцова,
Анастасия Клячева, Вячеслав Резник

Для вёрстки журнала использовались инструменты Open Sources: Scribus, Open Office, Gimp.

info@lingametechnology.com

В этом номере...

Игровой фейерверк



Локализация

Веснот будет наш!

стр. 9

Переводчики Wesnoth до нас потрудились очень серьёзно. Они решили не только задачу-минимум (локализовали интерфейс), но и перевели двадцать с лишним кампаний, в общей сложности на 25 сценариев и 4 Мб текста. Вышло примерно 2 Мб русского текста, причём довольно сложного стилистически – это два-три толстых романа!

Кодовый блок

SDL & OpenGL

стр. 12

Урок №4. Не секрет, что для вывода удачной графики используются различные расширения OpenGL. Что же такое эти расширения и для чего они нужны?

Игры на F.P.C.

стр. 16

Урок №2. Разбираемся с загрузкой текстур и пробуем камеру в действии.

Моддинг

Ремонт Sauerbraten

стр. 18

Урок №3. Работаем с анимационными моделями

Christmas Eve Crisis

Жанр: 2D-аркада
ОС: Linux, Windows, Mac
Сеть: нет
Звук: есть
Стоимость: бесплатно
Разработчик: [InterAction](#)
[Скачать](#)

Уходящий год стал сложным для каждого из нас. Мировой финансовый кризис откликнулся даже в тех сферах нашей жизнедеятельности, к которым не имеет прямого отношения. Ответом на всеобщую депрессию, наверное, станут предстоящие праздники. Ведь как Новый год встретишь, так его и проведешь. Именно поэтому к встрече нового 2010 многие будут готовиться особенно тщательно. А какой праздник без рождественского чуда!



>> Сначала всё шло как обычно...

Добрая сказка про Санта-Клауса уже переписывалась и пересказывалась миллионы раз. Но такой интерпретации лапландской легенды, пожалуй, еще не было!

Каждый ребенок знает, что в новогоднюю ночь зимний волшебник раскладывает под чудесными елочками подарки, но не каждый, даже взрослый человек, может представить насколько трудно исполнять желания!

Предыстория игры «Christmas Eve Crisis» повествует о том, как добрый дедушка Мороз, а точнее его американский собрат Санта-Клаус, по дороге с Северного Полюса теряет все подарки. На протяжении 10 уровней нам приходится помогать старичку разыскивать их по всему свету. Это простая детская игра, но прохождение всех препятствий заинтересует и их

родителей. Рождественское музыкальное сопровождение настраивает на праздничный лад. Яркая и совершенно оригинальная графика в классических цветах Нового года заставляет поверить в чудо и окунуться в его атмосферу.



>> Графика в игре поразительно яркая и красочная

Бедный зимний волшебник! На какие ухищрения ему приходится идти, чтобы исправить ошибку! Герой игры летает по воздуху, перепрыгивает через ядовитые растения, враждебных мышей, ежей и сов,



>> Каждый подарок должен быть принесён строго по адресу



>> Добраться до скрытой зоны не всегда просто



>> Вода не мешает Санте перемещаться, но вот подарок портится

он опускается на дно ледяного водоема и прыгает по деревьям. Не смотря на почтенный возраст, дедушка ловок и быстр. И все это лишь для того, что бы вовремя доставить заветные подарки!

У игроков есть возможность сравнивать свои результаты с достижениями других по сети, благодаря Интернет - таблице рекордов. Системные требования проекта просты и непритязательны: 600MHz CPU, 256MB RAM, OpenGL ускоритель, SDL 1.2.

Особенностью игры может считаться Secret areas - возможность на определенных этапах прохождения уровня открывать новые секретные районы. Это подогревает интерес игрока, в принципе, как все тайное и неизведанное! Главная задача Санты найти подарки, но по ходу легенды он набирает очки, собирая елочные шары, сладости, колокольчики и стаканы с молоком - всего таких бонусов 10 различных типов.

Минусами игры является отсутствие русскоязычной локализации. Хотя в этой игре подсказки только мешают. Немного раздражает отсутствие сохранения посреди игры. Это возможно сделать лишь в специально отведенных местах, но их очень мало. Именно поэтому, если вдруг придется выйти обратно в систему, уровень нужно будет начинать заново.

Несмотря на то, что сама игра довольно проста, прохождение некоторых препятствий может затянуться на часы. Хотя именно этот фактор заставляет вырабатывать то самое дедморозовское терпение, без которого вряд ли была бы возможна и сама новогодняя сказка!

В общем, праздник рядом, а эта игра лишний раз напоминает об этом!

LGT



Irukandji

SHIP: STRAIGHT SHOOTER

WEAPONS: GOOD IM

Жанр: 2D-аркада
ОС: Linux, Windows, Mac
Сеть: нет
Звук: есть
Стоимость: от 1\$
Разработчик: Charlie's Games
[Скачать демо](#)

Когда-то, давным-давно, когда компьютеры были слабенькими, а рынки забиты картриджами для Dendy, существовала простенькая, но увлекательная игрушка жанра аркадного скроллера. Сюжет её был незамысловат: передвигаясь на корабле в космосе, нужно было уничтожать вражеские объекты, которые очень смахивали на мух. С каждым этапом задача усложнялась за счёт увеличения количества пришельцев и проводимых ими маневров. В дальнейшем появилось немало игр-подражателей.



>> Мир 8-бит. Galaga: Demons of Death

Irukandji — это игра из той же серии. Казалось бы, зачем рассказывать о ней, если подобных проектов, как говорится, «лопаты гребя»? Любой мало-мальски подкованный программист сможет собрать нечто подобное буквально за пару дней. А нет, рассказать о Irukandji стоит хотя бы потому, что она безумна красива.

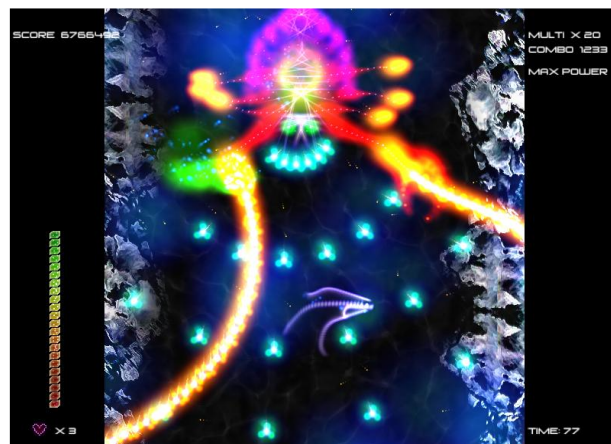
Начнём, пожалуй, всё же с игрового процесса. Действие происходит в подводном мире, где роль монстров играют всевозможные глубоководные твари. Вы управляете неким кораблем, который умеет не только шустро плавать в пределах игрового экрана, но и поливать огнём любой движущийся объект. К слову сказать, модификаций судёнышек насчитывается аж шесть штук, причём с уникальным набором оружия. Управление в игре несколько отличается от привычной раскладки для такого жанра. Однако клавиши по умолчанию очень удобны и не вызывают же-

лания изменить их. Если всё же подобное желание возникло, то настройки игры позволяют это сделать.

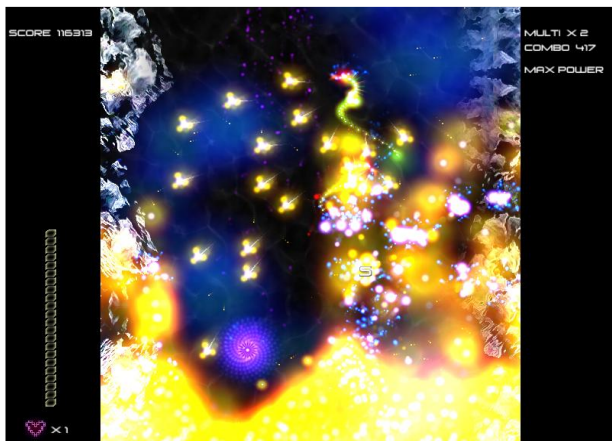


>> Даже служебные экраны Irukandji поражают своей красочностью

Итак, основное управление движением корабля возложено на курсорные клавиши. Направление огня можно корректировать клавишами <Z> и <X>. Кстати, не имеется привязки линии огня к движению корабля. К примеру, можно двигаться вправо, одновременно поливая надвигающихся монстров с левой стороны. Это относится к одиночному оружию, когда зона поражения ограничивается вектором огня. Но есть ещё оружие массового поражения,



>> В верхней части игрового экрана не взрыв, а всего лишь один из обитателей глубин



>> Да-а, от такого удара никто не устоит

которое позволяет уничтожить противника практически на всём поле. Вот только использовать его постоянно не удастся. Разработчики поступили весьма дальновидно, установив предел «раскалённости» для любого типа оружия. Если для стандартного варианта предупреждающая полоска заполняется достаточно медленно, то использование массового огня приводит к стремительному росту температуры. Впрочем, достаточно пару секунд переждать, как индикатор опять обнулится. Вот только времени для отдыха, как правило, не имеется. Игра очень и очень динамичная. Уже с первого уровня игрока втягивают в массовые разборки с окружающим миром.

Противник в игре не отличается особым интеллектом, а скорее пытается задавить количеством. Всё бы ничего, но со временем он обзаводится различным видом оружия, начиная с примитивных «бомбочек» и заканчивая своего рода орудиями залпового огня. Если помножить мощь огня на количество од-

новременно нападающих монстров, то получится дикая фантазмагория, где уцелеть очень проблематично. К счастью, герой не только обладает внушительным набором «жизней», но и может подбирать вываливающиеся из подбитого противника. Одновременно игра подкидывает и некоторые бонусы, которые позволяют модифицировать оружие «на ходу», что, с лавинообразно увеличивающейся массой противника, очень и очень кстати. Но и тут разработчики не поленились испортить нервы игрокам, позволив бонусам буквально удирать от страдающего под ударами кораблика. Иногда приходится гоняться за особо ценным, а потому увёртливым бонусом, по всему экрану, одновременно отбиваясь от противника.

Уже за одну только эту интерпретацию знакомой игровой механики можно было бы поставить Irukandji «отлично», но ведь мы ещё не коснулись визуальной составляющей! Графика в игре не просто хороша, а божественна! Разработчики не только сумели передать сказочный мир глубоководного царства, но и наделили каждый объект уникальной процедурной анимацией. Плавные изгибы форм, потрясающие по красоте залпы огня, неоновые светящиеся цвета не оставят никого равнодушным. В пиковые моменты экран буквально взрывается фантастическим заревом. И что интересно — цветовая насыщенность совсем не портит общее мнение об игре, а наоборот вызывает желание вновь и вновь наслаждаться буйством красок. К сожалению, никакие скриншоты не могут передать красоту игры. Советуем, даже если вы небольшой ценитель подобных игр — скачайте и просто полюбуйтесь на мастерство разработчиков. Право дело, у них есть чему поучиться!

LGT

SCORES

TIME SPENT: 01:22:26

SHIP: SUN SPREAD

HIGHEST SCORE: 40705641

AVERAGE SCORE: 16068204

COMPLETED GAME: YES!

PERFECT GAME: YES!

WORLD'S BEST

1	MIKEA	62131436
2	MIKEA	51970327
3	MIKEA	48711503
4	MR PERFECT	46493904
5	MIKEA	41658720
6	MIKEA	41188780
7	INSICERE_DAVE	40705641
8	MIKEA	37364704
9	INSICERE_DAVE	33922812
10	MIKEA	32114661
11	LAM MYSTERY_X	32075383
12	MIKEA	29155243
13	MIKEA	27802411
14	MIKEA	26975274
15	MIKEA	18995597
16	FLOBERT	18092309
17	MIKEA	17979835
18	MIKEA	15824337
19	MIKEA	14097250
20	MR PERFECT	13626452
21	MIKEA	13472128
22	MIKEA	7351653
23	MIKEA	7193973
24	INSICERE_DAVE	4789215
25	MIKEA	934301
26	INSICERE_DAVE	654853
27	MINKSTER	410
28	TULIOP	65
29	CHARLIE	10
30	CHARLIE	10
31	CHARLIE	10
32	CHARLIE	10
33	CHARLIE	10
34	CHARLIE	10
35	CHARLIE	10
36	CHARLIE	10
37	CHARLIE	10
38	CHARLIE	10
39	CHARLIE	10
40	CHARLIE	10

Aquaria

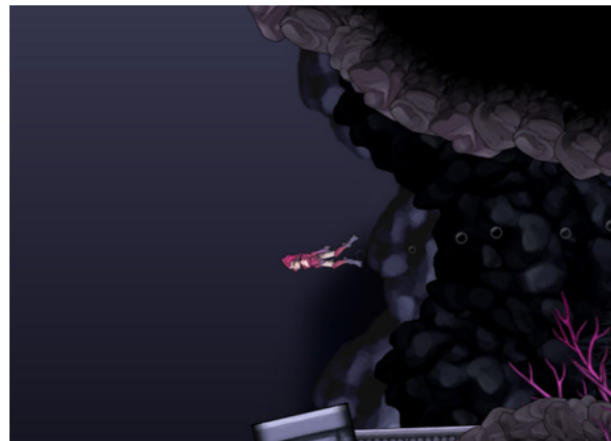
Жанр: 2D-аркада
ОС: Linux, Windows, Mac
Сеть: нет
Звук: есть
Стоимость: бесплатно (бета)
Разработчик: Bit Blot
[Скачать бета-версию](#)

Так уж получилось, что следующей на очереди для обзора стоит ещё одна подводная игра Aquaria. В отличие от своей предшественницы она представляет собой аркаду с качественной мультипликационной прорисовкой. Действительно, играя в неё, ощущаешь себя как в сказке, но давайте обо всём по порядку.

Aquaria — игра, созданная всего двумя людьми, представляет собой миниатюрный шедевр. Концепция, графика, звуковое оформление — всё выполнено на высшем уровне. В своё время она получила немало наград и, надо заметить, вполне заслуженно. Однако, Aquaria — это не совсем новая игра. Она появилось на всеобщее обозрение в 2007 году для систем Windows и Mac OS. Лишь пару месяцев назад было объявлено об усиленной работе над портированием игры для GNU/Linux и в декабре появилась доступная для скачивания beta-версия. Естественно, мы не могли пройти мимо этой новости и скачали сборку под Linux.

Проблем с установкой игры никаких не было. К своему удивлению мы даже обнаружили ярлык Aquaria в меню KDE, что является редкостью для программ установленных из сторонних бинарных пакетов. Однако это было только начало...

При первом запуске игра предлагает просмотреть начальную видеозаставку, вводящую в курс событий.



>> Вот она маленькая героиня большой игры

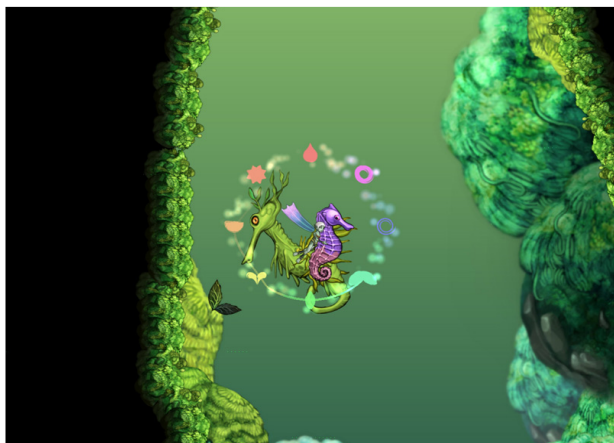
Качественная музыка, великолепная озвучка ролика сразу вызывает положительные эмоции и желание непременно поиграть. В заставке рассказывается о несчастной маленькой русалке (а может быть и подводного эльфа, так как у русалок нет ног), которая непонятно как потеряла свою память и вынуждена скитаться в глубинах океана. Собственно, главной задачей игрока является помощь героине в осознании самой себя. В поисках утраченной памяти ей придётся обыскивать самые потаенные уголки океана, воевать с враждебными глубинными жителями и даже немного колдовать.



>> Даже экран меню проработан очень тщательно



>> Мир Aquaria очень яркий и красивый



>> Магия музыки и цвета в действии

Кстати, именно реализация механизма магии является главной фишкой этой игры. Для создания простейшего файерболла героине придётся... нет, нет, не махать волшебной палочкой, а пропеть несколько нот. Предлагается круговое меню с набором звуков, где путём выбора необходимых пунктов, создаётся музыкальная последовательность «включающая» то или иное заклинание. Заметьте, что это нужно делать максимально быстро, так как противники не дремлют и активно вас атакуют. Конечно, с непривычки достаточно сложно чисто пропеть заклинание, но поверьте, что по окончании этой игры вы будете обладать замечательной музыкальной памятью.

Как уже было сказано, звуковая часть Aquaria проработана очень тщательно. Приятная ненавязчивая музыка, органное звучание магии, голосовая озвучка кат-сцен оставляет неизгладимое впечатление и является одной из сильнейших сторон этой игры. Впрочем, и графика не подкачала.

Все уровни Aquaria очень качественно прорисованы, дизайнер не поспешил на обилие мелких деталей. Пусть графика в игре чисто двухмерная, но её уровень говорит о безграничной любви создателей к своему детищу.

Подводный мир Aquaria очень велик и разнообразен. Почти каждая локация является уникальной: в одной резвятся мелкие рыбки и медузы, во второй грозно щёлкают клешнями крабы, а где-то вы можете повстречать монстров, способных целиком заглотнуть героиню. Вот и приходится бедной русалке (эльфийке?) усиленно работать лапами или петь заклинания. Кстати, пение необходимо не только для вызова магических сил, но и служит своего рода ключиком для некоторых потаённых мест.

Не стоит думать, что вы можете беззаботно болтаться в теплых водах океана, подпевая проплывающим рыбешкам. Игра буквально подталкивает к совершению активных действий. Во-первых, героиня должна питаться. Как правило, добыть еду не представляет особых сложностей, но вот приготовить её... Кроме того, неожиданно появляются своего рода гиды, которые вынуждают отправляться в страшные, тёмные глубины не только океана, но и своей памяти. А если бурные приключения надоедают, то ничто не мешает заняться украшением своего жилища всякими безделушками, которые в обилие встречаются по дороге.

К слову, Aquaria очень интересная и должна найти своих почитателей в рядах линуксоидов. Единственный минус, который мы бы хотели отдельно отметить — это чрезвычайно быстро пролетающие сюжетные тексты в игре. Если вы не знаток английского, то разобраться будет достаточно сложно. Может когда-нибудь мы дождёмся качественной локализации этой замечательной игрушки.

LGT



Веснот будет наш!

Особенности локализации игры

От редакции

«Battle for Wesnoth» - игра знакомая любому линуксоиду. Качественный перевод позволяет наслаждаться ею по настоящему. О том, с какими трудностями пришлось столкнуться локализаторам, поведают координатор переводчиков Виктор Сергиенко.

Как это начиналось

"Битву за Веснот" я увидел в 2007 году и полюбил сразу. Мне нравятся стратегические игры, а эта особенно хороша: простые правила (упрощённый вариант варгеймов), удобный и красивый интерфейс (пиксель-арт в игре сказочно прекрасен. Особенно хороши новые анимированные дрейки в версии 1.7), и при этом — широчайшее многообразие манёвров.

На момент, когда я заинтересовался переводом, все первоначальные переводчики покинули проект. Возможно не смогли поддерживать работу над проектом, а может просто потеряли к нему интерес.

В ноябре 2008 года я написал координатору локализаций, и он предложил мне этим заняться. У меня уже был небольшой опыт переводчика и редактора, поэтому решил, что потяну.

Переводчики Wesnoth до нас потрудились очень серьёзно. Они решили не только задачу-минимум (локализовали интерфейс), но и перевели двадцать с лишним кампаний, в общей сложности на 25 сценариев и 4 Мб текста. Вышло примерно 2 Мб русского текста, причём довольно сложного стилистически — это два-три толстых романа!

Каким он был...

Перевод страдает от того, что IT-шники, как ранее физики, полагают, что могут формальной логикой исчислить всё, в том числе "алгеброй гармонию проверить". Они действительно могут, но для этого им нужно довольно много учиться.

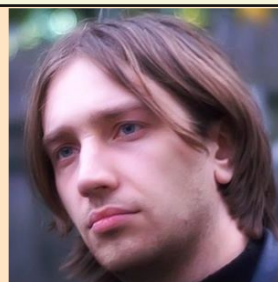
Большая часть наших контрибуторов — это работники IT, и не все интересовались филологией. Поэтому перевод очень сильно страдает (дорогие переводчики, не обижайтесь! Каким бы талантливым не был человек, без обучения он никуда не уедет. Моцарт стал известным композитором к 18 годам, хотя начал писать музыку в 10 лет, а учиться музыке — в четыре). Поэтому в тексте перевода встречались перлы вроде "...если враг имеет своего бойца рядом с этим бойцом...".

О технической части

Локализация "Веснота" построена на классическом инструменте GNU — gettext. Технически, переводчику нужно в текстовом файле (формата "po") напротив каждой английской фразы написать соответствующую русскую. Есть специальные редакторы и кое-какие тонкости, но они не обязательны. В случае ошибок, я сам выправляю испорченный файл.

Так же мы пользуемся несколькими страничками на [вики](#). Храним там розданную работу ("занятые" файлы), рекомендации и по-

Автор



Виктор Сергиенко

Программист, по образованию — математик. Любит музыку, поэзию, большой теннис, компьютерные миры (необязательно игровые), и просто жить.

В проекте «Battle for Wesnoth» является координатором перевода и самопровозглашённым редактором.



лезные ссылки, а главное – словарь перевода. Текста в Весноте – мегабайты, история и сюжеты очень богатые, так что словарь имён и названий весьма обширный.

С другой стороны, никаких технических сложностей с переводом Wesnoth нет. Авторы игры уже много лет поддерживают множество локализаций. Никаких звуков, а тем более видео, переводить не надо; в звуках игры есть только рёв, ойки, стоны и прочие шумы. Поэтому переводим мы только чистый текст.

Ещё есть некоторое количество картинок – логотип игры и карты с подписями. Основа карты одна, но в каждой кампании могут быть разные подписи. Без них игра работает так же, но с английскими подписями к картам кампаний.

Проблемы перевода

В первую очередь – это объём текста и согласование названий, а также прочих терминов. Больше всего копий сломали вокруг имён отдельных существ. В игре есть классы, которые буквально пере-

водятся, как: "Эльфийский боец", "Гномий боец", "Дрейковский боец". Естественно, ни одно из таких составных названий не помещается в интерфейс игры, потому что английские слова раза в полтора короче. Поэтому мы решили переводить каждое название только одним словом.

Пришлось долго и мучительно придумывать более-менее подходящие синонимы, передающие идею каждого бойца. Сейчас у нас только для солдата (fighter) есть "ополченцы", "солдаты", "воители"; а для прочих бойцов – ещё примерно пятьдесят разных слов. Есть даже "геолог"!

Приходилось проверять – подходит ли перевод интерфейса по контексту, а также помещается ли он в отведенное место? И главное, не ошиблись ли в переводе. К примеру, "Clear" – это "чистый", "ясный", "очистить", "сбросить" или "прояснить"?

К сожалению, сам я редко себе позволяю ещё и протестировать переведённые строки; очень многие видны только в специальных режимах игры. Перевожу с наде-

ждой на лучшее – на наш локализованный "авось".

Кроме того, свободные приложения иногда крайне медлительны – например, довольно заметные баги в Gnome, X.org и даже ядре Linux, бывает, живут годами. Так, библиотека SDL под Windows до сих пор не поддерживает unicode в названии окна приложения; так что если в игре включить русский язык, то заголовок окна будет таким: "????? ?? ??????". А ведь игра должна идти не только на *nix!

Наши успехи

Основная работа была сделана до меня. Самым важным достижением считаю то, что мы довели тексты до более или менее литературного вида, и сейчас наши переводчики пользуются рекомендациями из книги Норы Галь [\[советская переводчица с английского и французского, литературный критик, теоретик перевода, редактор, — прим. ред.\]](#). У нас почти нет фраз вроде: "состояние обследования выполнения распоряжения". Или калькированных оборотов со словами "имеет" и "является".

Все кампании главной ветки переведены полностью. Большую



часть этих переводов уже литературно переработали.

Что в итоге...

Интерфейс, редактор и основные кампании большей частью переведены; сейчас обновляем переводы для скорого стабильного релиза 1.8. Тем не менее, кое-что остаётся. В первую очередь необходимо:

- Обновить переводы и перевести новые строки из 1.8 (практически готово, спасибо всем!);
- Продолжать править стиль;
- Перепроверить, одинаково ли переведены имена и названия;
- Перевести карты основных кампаний и логотип.

Во вторую очередь:

- Продолжить переводить лучшие из скачиваемых аддонов (user-made кампаний). Механизм для этого есть;
- Заменить не очень удачные переводы некоторых слов на более подходящие (самые некрасивые мы уже побороли);
- Перевести руководство к игре (в нём используется специальный синтаксис страниц map, и, честно говоря, вряд ли кто-то его читает; гораздо лучше читать секции вики по типу: "Advanced tactics" и "How to play...").
- Протестировать весь UI на правильность перевода.

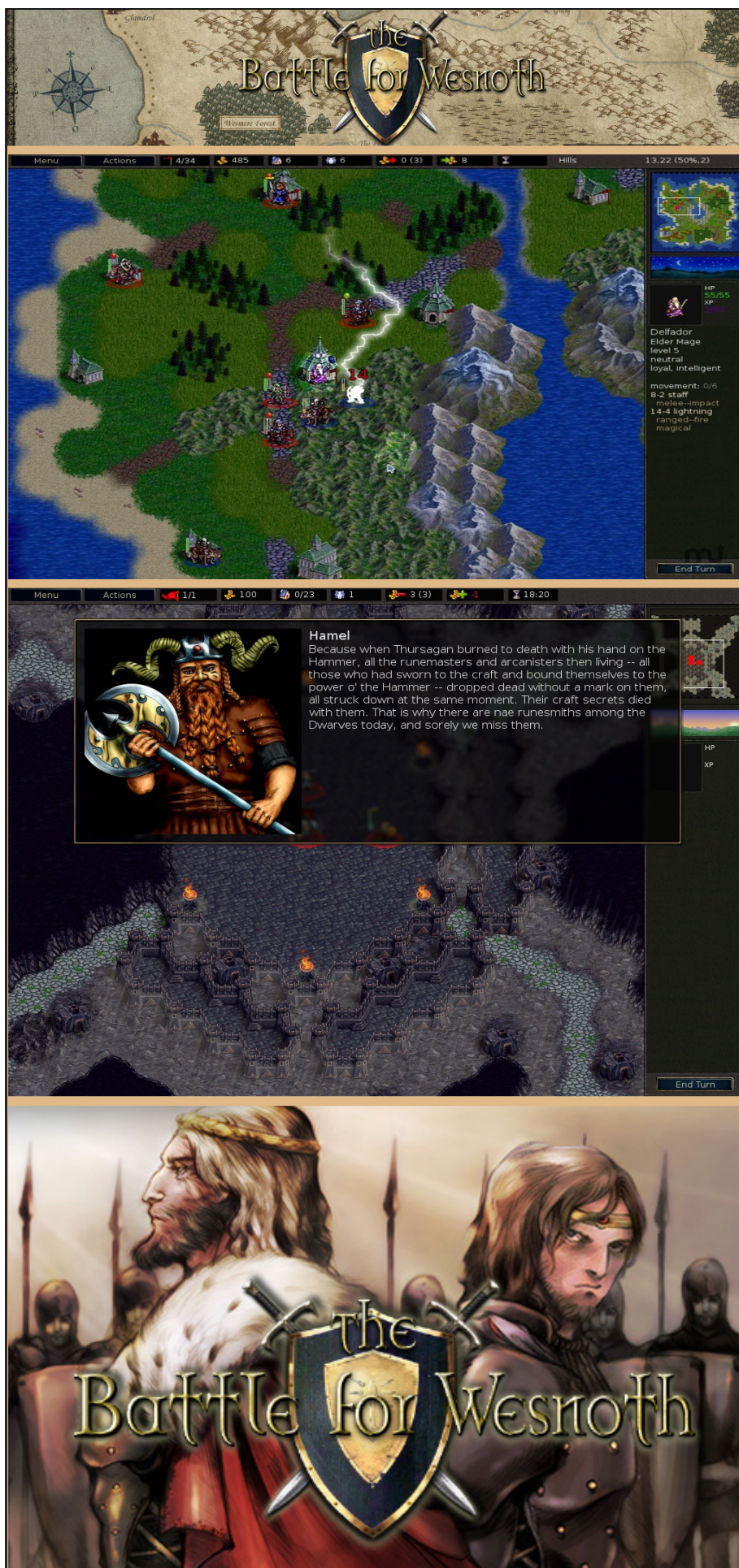
Для всего этого нужны рабочие руки, и, что важнее всего, головы.

Большое спасибо всем переводчикам. Работы сделано много, и хорошей.

Наверное, лучшей похвалой нашей работе были слова профессионального переводчика: "[...] одна из очень немногих игр, в которые я предпочитаю играть в переводе, а не на языке оригинала".

Скоро выходит версия 1.8. Наслаждайтесь!

LGT





SDL & OpenGL

Работаем с расширениями OpenGL

Автор



Кирилл Баженов

Любит программировать под Linux. Увлекается трёхмерной графикой, играми и ...OpenGL.

От редакции

Не секрет, что для вывода удачной графики используются различные расширения OpenGL. Что же такое эти расширения и для чего они нужны? Расширение — это некоторая возможность видеокарты, не предусмотренная стандартным API на момент его создания. Таким образом, механизм расширений позволяет использовать новые возможности современных видеокарт практически сразу после их появления, не дожидаясь переработки API. К слову, многие расширения потом включались в стандарт.

Расширения OpenGL

А как же пользоваться расширениями? Очень просто: инициализировать их. Принцип простой: в заголовочном файле «glext.h» находятся все имеющиеся расширения. Вернее не расширения, а указатели на функции и некоторые константы. Для того, чтобы инициализировать расширение (или часть его) надо получить указатели на эти функции. Описание прото-типа функции выглядит примерно так:

```
GLAPI void APIENTRY glGlobalAlphaFactorbSUN (GLbyte);
```

Или так:

```
typedef void (APIENTRY PFNGLGLOBALALPHAFACTORBSUNPROC)
(GLbyte factor);
```

Разберём всё по порядку. Первый вариант включает-ся, если определить макрос **GL_GLEXT_PROTOTYPES**. Он представляет собой описание прототипа функции. GLAPI и APIENTRY пугаться не стоит, - это всего лишь APIENTRY* (под Windows), и extern. То есть в случае с Linux код выглядит следующим образом:

```
extern void glGlobalAlphaFactorbSUN (GLbyte);
```

Теперь второй вариант, если не определять **GL_GLEXT_PROTOTYPES**. Описание функции немного страшное на первый взгляд. Но волноваться не стоит, так как это всего лишь описание типа данных указателя на функцию. Что такое этот PFN...? Очень просто - тип данных. PFN расшифровывается, как PROC FUNCTION ADDRESS, то есть адрес этой функции. Далее идёт имя функции большими буквами, и заканчивается словом PROC. Просто, не так ли? Для объявления функции расширения нужно написать следующее:

```
extern PFNGLGLOBALALPHAFACTORBSUNPROC
glGlobalAlphaFactorbSUN;
```

С этим разобрались. Теперь, как же с ней работать? Нужно нашему указателю присвоить правильный адрес памяти, содержащий нашу функцию. Для этого существуют специальные функции в разных операционных системах, но мы рассмотрим только средства SDL. Чтобы получить указатель, надо вызвать **SDL_GL_GetProcAddress()**. Принимает в качестве параметра имя функции. То есть, чтобы получить указатель на **glGlobalAlphaFactorbSUN** мы воспользуемся следующим кодом:

```
glGlobalAlphaFactorbSUN =
(PFNGLGLOBALALPHAFACTORBSUNPROC)SDL_GL_GetProcAddress("gl
BindBufferARB");
```

Как видите, ничего сложного. Давайте рассмотрим проверку расширения на доступность. Для получения списка доступных расширений можно использовать **glGetString** с параметром **GL_EXTENSIONS**. Функция возвратит список всех расширений в виде строки, разделённых пробелами. Чтобы найти строку в списке воспользуемся следующим несложным кодом:

```
bool _is_ext_supported(const char* ext, const char*
extlist)
{
    const char* start = extlist;
    const char* ptr;
    while ((ptr = strstr(start, ext)) != NULL)
    {
        const char* end = ptr + strlen(ext);
        if (isspace(*end) || *end == '\0')
            return true;
        start = end;
    }
    return false;
}
```



```
bool isExtSupported(const char* ext)
{
    const char* extlist = (const
char*)glGetString(GL_EXTENSIONS);
    return _is_ext_supported(ext, extlist);
}
```

Первая функция проверяет наличие расширения в списке, а вторая и так понятна. Для более удобного управления расширениями мы введём дополнительную библиотеку. Итак, файл «Extension.h»:

```
#ifndef _EXTENSION_WRAPPER_H_
#define _EXTENSION_WRAPPER_H_

// Включаем необходимые файлы
#include <GL/gl.h>
//define GL_GLEXT_PROTOTYPES
#include <GL/glext.h>

extern "C" {

    extern bool isExtSupported(const char* ext);

    ...
    Расширения...
    ...
}
#endif
```

Первое расширение, которое мы рассмотрим, называется **GL_ARB_vertex_buffer_object**. **ARB** в названии расширения означает, что данное расширение разработано Советом по архитектуре OpenGL (Architecture Review Board). Соответственно, префиксы NV, SUN, ATI и т.д. означают, что расширение разработано компаниями nVidia, Sun, ATI и др. **GL_ARB_vertex_buffer object**, или **VBO** (вершинные буферы) позволяет записать некоторые данные в память видеокарты напрямую и обрабатывать их прямо оттуда. Подобный подход позволяет избежать трат при копировании данных из оперативной памяти при использовании традиционных массивов вершин, и даёт в среднем до 200% производительности, в зависимости от количества вершин, типа видеокарты. В состав расширения входят несколько функций, минимальный набор которых будет рассмотрен ниже. Определяются они следующим образом:

```
extern bool isExtSupported(const char* ext);

// GL_ARB_vertex_buffer_object
extern PFNGLBINDBUFFERARBPROC glBindBufferARB;
extern PFNGLDELETEBUFFERSARBPROC glDeleteBuffersARB;
extern PFNGLGENBUFFERSARBPROC glGenBuffersARB;
extern PFNGLISBUFFERARBPROC glIsBufferARB;
extern PFNGLBUFFERDATAARBPROC glBufferDataARB;
extern PFNGLBUFFERSUBDATAARBPROC glBufferSubDataARB;
extern PFNGLGETBUFFERSUBDATAARBPROC
glGetBufferSubDataARB;
extern PFNGLMAPBUFFERARBPROC glMapBufferARB;
extern PFNGLUNMAPBUFFERARBPROC glUnmapBufferARB;
extern PFNGLGETBUFFERPARAMETERIVARBPROC
glGetBufferParameterivARB;
extern PFNGLGETBUFFERPOINTERVARBPROC
glGetBufferPointervARB;
```

Для инициализации расширения мы введём новую функцию:

```
extern bool initGL_ARB_vertex_buffer_object();
```

Теперь рассмотрим файл «Extension.cpp»:

```
#include <Extension.h>
#include <string.h>
#include <ctype.h>
#include <SDL/SDL_video.h>
```

```
#define GET_PROC_ADDRESS(X) SDL_GL_GetProcAddress(X)

extern "C" {

    static bool _is_ext_supported(const char* ext, const
char* extlist)
    {
        ... Это было описано выше ...
    }

    bool isExtSupported(const char* ext)
    {
        const char* extlist = (const
char*)glGetString(GL_EXTENSIONS);
        return _is_ext_supported(ext, extlist);
    }
}
```

Мы будем отделять одно расширение от другого следующим образом: сначала определение функций, потом функция инициализации. И вот наше первое расширение:

```
PFNGLBINDBUFFERARBPROC glBindBufferARB = 0;
PFNGLDELETEBUFFERSARBPROC glDeleteBuffersARB = 0;
PFNGLGENBUFFERSARBPROC glGenBuffersARB = 0;
PFNGLISBUFFERARBPROC glIsBufferARB = 0;
PFNGLBUFFERDATAARBPROC glBufferDataARB = 0;
PFNGLBUFFERSUBDATAARBPROC glBufferSubDataARB = 0;
PFNGLGETBUFFERSUBDATAARBPROC glGetBufferSubDataARB =
0;
PFNGLMAPBUFFERARBPROC glMapBufferARB = 0;
PFNGLUNMAPBUFFERARBPROC glUnmapBufferARB = 0;
PFNGLGETBUFFERPARAMETERIVARBPROC
glGetBufferParameterivARB = 0;
PFNGLGETBUFFERPOINTERVARBPROC glGetBufferPointervARB
= 0;

bool initGL_ARB_vertex_buffer_object()
{
    glBindBufferARB =
(PFNGLBINDBUFFERARBPROC)GET_PROC_ADDRESS("glBindBufferARB"
);
    glDeleteBuffersARB =
(PFNGLDELETEBUFFERSARBPROC)GET_PROC_ADDRESS("glDeleteBuffs
ersARB");
    glGenBuffersARB =
(PFNGLGENBUFFERSARBPROC)GET_PROC_ADDRESS("glGenBuffersARB"
);
    glIsBufferARB =
(PFNGLISBUFFERARBPROC)GET_PROC_ADDRESS("glIsBufferARB");
    glBufferDataARB =
(PFNGLBUFFERDATAARBPROC)GET_PROC_ADDRESS("glBufferDataARB"
);
    glBufferSubDataARB =
(PFNGLBUFFERSUBDATAARBPROC)GET_PROC_ADDRESS("glBufferSubDa
taARB");
    glGetBufferSubDataARB =
(PFNGLGETBUFFERSUBDATAARBPROC)GET_PROC_ADDRESS("glGetBuffs
erSubDataARB");
    glMapBufferARB =
(PFNGLMAPBUFFERARBPROC)GET_PROC_ADDRESS("glMapBufferARB");
    glUnmapBufferARB =
(PFNGLUNMAPBUFFERARBPROC)GET_PROC_ADDRESS("glUnmapBufferAR
B");
    glGetBufferParameterivARB =
(PFNGLGETBUFFERPARAMETERIVARBPROC)GET_PROC_ADDRESS("glGetB
ufferParameterivARB");
    glGetBufferPointervARB =
(PFNGLGETBUFFERPOINTERVARBPROC)GET_PROC_ADDRESS("glGetBuff
erPointervARB");

    return
isExtSupported(«GL_ARB_vertex_buffer_object»);
}
```

Не так уж и страшно, как могло показаться. Теперь давайте научимся использовать VBO.

Для начала рассмотрим создание буферного объекта:

```
glGenBuffersARB(1, &vertexBuffer);
```

Как видите, создание буфера аналогично созданию текстуры.

Следующее, что нам предстоит усвоить, это привязка и загрузка данных в буфер. Что такое привязка? OpenGL использует только один активный буфер на цель. Целью может быть вершинный и индексный буфер (как вы уже поняли, в вершинный буфер записываются вершины, а в индексный — индексы). Задание активного буфера называется привязкой. Чтобы привязать буфер существует функция **glBindBufferARB**. Она принимает активную цель и имя буфера (имя аналогично имени текстуры). Пример:

```
glBindBufferARB(GL_ARRAY_BUFFER_ARB, vertexBuffer); //
// Для вершинного буфера
glBindBufferARB(GL_ELEMENT_ARRAY_BUFFER_ARB,
indexBuffer); // Для индексного буфера
```

Далее нам надо загрузить данные в буферный объект. Для этого есть процедура **glBufferDataARB**. Ещё один пример:

```
glBufferDataARB(GL_ARRAY_BUFFER_ARB, sizeof(GLfloat) *
NUM_SPHERE_VERTICES * 3, sphereVertexArray,
GL_STATIC_DRAW_ARB);
```

Рассмотрим её параметры:

Первый параметр — это целевой буфер, второй — количество элементов, которые нужно скопировать в буфер (аналогично тексту), третий — сам массив вершин. Четвёртый параметр — это подсказка OpenGL где размещать данные. Подробно мы на ней останавливаться не будем, скажу лишь что **GL_STATIC_DRAW_ARB** используется для рисования для неанимированной, часто используемой геометрии, **GL_DYNAMIC_DRAW_ARB** — данные, возможно, потребуется обновить 1-2 раза и их надо поместить туда, где это будет не слишком болезненным. **GL_STREAM_DRAW_ARB** используем для анимированной геометрии.

Рассмотрим простенький пример, который визуализирует сферу из множества маленьких точек. Для начала давайте напишем функцию вычисления обратного квадратного корня, так как стандартная библиотека C++ такого не имеет:

```
#define IEEE_1_0 0x3f800000
inline float rsqrt(const float x)
{
    unsigned tmp = (unsigned(IEEE_1_0 << 1) + IEEE_1_0 -
*(unsigned*)&x) >> 1;
    float y = *(float*)&tmp;
    return y * (1.47f - 0.47f * x * y * y);
}
// Теперь нам понадобится небольшая структура, обозначающая
// сферу и список всех сфер:
struct Sphere
{
    // Position
    float x, y, z,
    // Color
    r, g, b;
};
list<Sphere*> spheres;
```

А также функция для загрузки сфер из файла:

```
#define NUM_SPHERE_VERTICES 90000
void genSphere()
{
    for (size_t i = 0; i < NUM_SPHERE_VERTICES; i++)
    {
        // Генерация сферы
```

```
GLfloat r1, r2, r3, scaleFactor;
r1 = (GLfloat)(rand() - (RAND_MAX / 2));
r2 = (GLfloat)(rand() - (RAND_MAX / 2));
r3 = (GLfloat)(rand() - (RAND_MAX / 2));

scaleFactor = rsqrt(r1*r1 + r2*r2 + r3*r3);
sphereVertexArray[(i * 3)] = r1 * scaleFactor;
sphereVertexArray[(i * 3) + 1] = r2 *
scaleFactor;
sphereVertexArray[(i * 3) + 2] = r3 *
scaleFactor;
}

// Создание буферов
glGenBuffersARB(1, &sphereVertexBuffer);
glBindBufferARB(GL_ARRAY_BUFFER_ARB,
sphereVertexBuffer);
glBufferDataARB(GL_ARRAY_BUFFER_ARB, sizeof(GLfloat)
* NUM_SPHERE_VERTICES * 3,
sphereVertexArray, GL_STATIC_DRAW_ARB);

// Привязка нулевого буфера (хороший тон)
glBindBufferARB(GL_ARRAY_BUFFER_ARB, 0);

// Загрузка сфер
ifstream inf("spheres");
if (!inf.is_open())
{
    cout << "Data file NOT found!"
    << endl;
    exit(1);
}
while (!inf.eof())
{
    Sphere* sphere = new Sphere;
    inf >> sphere->x >> sphere->y >> sphere->z
    >> sphere->r >> sphere->g >> sphere->b;
    //
    spheres.push_back(sphere);
}
inf.close();
}
```

Рисование из буферного объекта почти не отличается от обычных массивов вершин, за исключением того, что нужно привязать необходимый буфер и установить указатели массивов в ноль. В этом случае OpenGL будет брать данные прямо из буфера:

```
glBindBufferARB(GL_ARRAY_BUFFER_ARB,
sphereVertexBuffer);
glNormalPointer(GL_FLOAT, 0, 0);
glVertexPointer(3, GL_FLOAT, 0, 0);
glDrawArrays(GL_POINTS, 0, NUM_SPHERE_VERTICES);
```

Как видите, ничего сложного. Теперь мы нарисуем 14 сфер по 90000 вершин в каждой:

```
void drawSphere()
{
    if (useVBO)
    {
        glBindBufferARB(GL_ARRAY_BUFFER_ARB,
sphereVertexBuffer);
        glNormalPointer(GL_FLOAT, 0, 0);
        glVertexPointer(3, GL_FLOAT, 0, 0);
    }
    else
    {
        glNormalPointer(GL_FLOAT, 0, sphereVertexArray);
        glVertexPointer(3, GL_FLOAT, 0,
sphereVertexArray);
    }

    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_NORMAL_ARRAY);

    glDrawArrays(GL_POINTS, 0, NUM_SPHERE_VERTICES);

    glBindBufferARB(GL_ARRAY_BUFFER_ARB, 0);

    glDisableClientState(GL_NORMAL_ARRAY);
    glDisableClientState(GL_VERTEX_ARRAY);
}
```


И в функции `renderFrame` добавьте следующий код:

```
glPushMatrix();

for (list<Sphere*>::iterator it = spheres.begin();
     it != spheres.end(); it++)
{
    glPushMatrix();

    glTranslatef((*it)->x, (*it)->y, (*it)->z);

    glColor3f((*it)->r, (*it)->g, (*it)->b);
    drawSphere();
    glColor3f(1, 1, 1);

    glPopMatrix();
}

glPopMatrix();
```

В примере к статье клавиша TAB включает или выключает использование VBO, что позволит вам почувствовать разницу.

Анизотропная фильтрация

Анизотропная фильтрация текстур — одно из самых легких в использовании расширений. Что это такое? Это ещё один метод улучшения качества текстуры на наклонных поверхностях относительно камеры. В отличие от мип-текстурирования, анизотропия даёт менее размытую картинку. Её минус — достаточно сложные вычисления, то есть большие затраты при рендеринге. Принцип действия: сначала выбираем мип-текстуру, которая соответствует разрешению поперёк направления обзора. Берут несколько текстелей вдоль направления обзора (в 2х — 2, 4х — 4 и так далее) и усредняют их цвета. Но чтобы использовать её в OpenGL нам надо две вещи:

- Получить максимальный уровень анизотропной фильтрации;
- Применить его к текстуре.

Реализация первого пункта до безобразия проста:

```
GLuint filter;
glGetFloatv(GL_MAX_TEXTURE_MAX_ANISOTROPY_EXT,
            &filter);
cout << "Anisotropy: " << filter << "x" << endl;
```

Таким образом мы получаем максимально возможный уровень фильтрации. Данный код следует добавить в функцию загрузки текстур, предварительно проверив наличие расширения `GL_EXT_texture_filter_anisotropic`.

```
...
glEnable(GL_TEXTURE_2D);
glGenTextures(1, &texture);
glTexParameterf(GL_TEXTURE_2D,
                GL_TEXTURE_MAX_ANISOTROPY_EXT, filter);
...
```

И всё! В итоге мы получаем гораздо лучшую картинку.

Mesh & SceneNode

Для лучшей организации сцены давайте введём понятия `Mesh` и `SceneNode`. `Mesh` — это только лишь хранилище данных про 3D-объект. Он обычно содержит в себе имена буферов с данными и иногда указатели на данные. Примерная реализация `Mesh`:

```
class Mesh
{
public:

    GLuint vertexBuffer;
    GLuint indexBuffer;

    GLfloat* vertexArray;
    GLuint vertexCount;
    GLuint* indexArray;
    GLuint indexCount;
};
```

`SceneNode` — это логический элемент управления сценой. Обычно это обёртка над мешем, содержащая ещё и свои координаты в пространстве, список дочерних `Node` (размещаются относительно родительского), и так далее. Примерная реализация:

```
class SceneNode
{
public:
    Mesh* mesh;
    Vector3 pos;

    SceneNode(Mesh* m);
    ~SceneNode();

    void setPosition(const Vector3& v);

    void draw();
};
```

Как видите, нод содержит указатель на свой меш, что позволяет экономить память и использовать многократно одни и те же данные. Рисование нода также не должно быть сложным: просто привязываем буферы меша и рендерим по вершинам. Примерная реализация:

```
void SceneNode::draw()
{
    glPushMatrix();
    glTranslatef(pos.x, pos.y, pos.z);
    ... Рисование из буферов ...
    glPopMatrix();
}
```

LGT

В следующий раз...

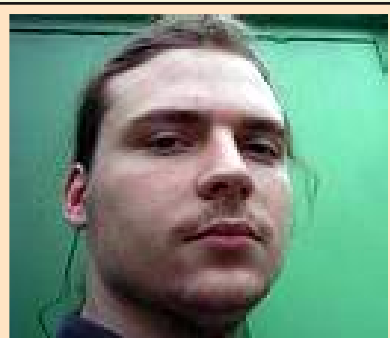
Детально разберёмся с `Mesh` и `SceneNode`, а также рассмотрим основные математические классы и способы работы с ними.



Игры на F.R.C.

Работаем с текстурами и камерой

Автор



Валентин Судаков

Слегка чокнутый человек с высоким уровнем знания игрового программирования и других вещей. Любитель красивых, умных, добрых девушек и OpenGL с Pascal'евскими языками

От редакции

Наверное, любоваться на детище своих рук можно до бесконечности. Правда, если это пустая форма от первого урока, то эйфория пройдет достаточно быстро. В этот раз, Валентин Судаков подскажет, как развеять скуку, путём добавления нескольких строчек кода.

Начнём с текстур

Для начала определимся с перечнем того, что должно быть у нас в этом модуле:

- 1. Загрузка самих текстур в формате Targa;
- 2. Удобная платформа для добавления работы с любыми другими видами текстур.

И так, начинаем...

Раз мы пока работаем с одним форматом Targa, то модуль у нас получится маленький, но уже в нём мы оставим возможность доработки и возможности добавления новых форматов.

Назовём наш модуль "Textures", так как он будет отвечать за работу с текстурами:

```
unit Textures;  
interface  
uses  
    GL, GLU;  
  
    Type PByteArray = ^TByteArray;  
    TByteArray = array [0..1023] of byte;  
  
function LoadTGATexture(Filename: String; var Texture:  
    Cardinal): Boolean;  
  
implementation  
  
{-----}  
{ Эта функция , является часто используемой,}  
{ которая выделяет место в памяти OpenGL и записывает в  
указанный блок}  
{ текстуру. }  
{ glGenTextures - выделяет место под текстуру в OpenGL }  
{ glBindTexture - делает текущей текстуру с нужным  
индексом }  
{ glTexEnvi - Установка взаимодействия «текстуры с  
миром» }  
{ glTexParameteri - Установка параметров текстуры }  
{ gluBuild2DMipmaps - Функция, которая записывает в  
память OpenGL'a }  
{ вначале делаются МирMap (уменьшенные копии той же
```

```
текстуры)},  
{ они позволяют повысить fps путём рисования уменьшенной  
копии }  
{ текстуры на дальних объектах }  
{-----}  
  
function CreateTexture(Width, Height, Format : Word;  
    pData : Pointer) : Integer;  
var  
    Texture : Cardinal;  
begin  
    glGenTextures(1, @Texture);  
    glBindTexture(GL_TEXTURE_2D, Texture);  
  
    {Прозрачная текстура с объектами на заднем фоне }  
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,  
        GL_MODULATE);  
  
    { Устанавливаем фильтры для текстур, в данном случае  
ставим линейный фильтр }  
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  
        GL_LINEAR);  
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  
        GL_LINEAR);  
  
    case format of  
        GL_RGBA, 32 : gluBuild2DMipmaps(GL_TEXTURE_2D, 4,  
            Width, Height, GL_RGBA, GL_UNSIGNED_BYTE, pData);  
        GL_RGB , 24 : gluBuild2DMipmaps(GL_TEXTURE_2D, 3,  
            Width, Height, GL_RGB, GL_UNSIGNED_BYTE, pData);  
    end;  
  
    result :=Texture;  
end;
```

SwapRgb – функция предназначена для конвертации цветов формата BGR в RGB. Targa, BMP и другие форматы часто используют BGRA, соответственно их нужно изменить вручную или проще всего поставить необходимое в **gluBuild2DMipmaps**.

Просто заменяем gl_rgb, gl_bgr. К сожалению, в стандартных библиотеках gl, glu не имеется gl_bgr, gl_bgra. Я указываю их константы ниже:

- gl_bgr = \$80E0;
- gl_bgra = \$80E1.

```
procedure SwapRGB(data : PbyteArray; Size,bpp : Integer);
var i : integer;
    C : Byte;
begin
  for I :=0 to Size - 1 do
    begin
      c := Data[i*bpp+2];
      Data[i*bpp+2] := Data[i*bpp];
      Data[i*bpp] := c;
    end;
  end;
end;
```

LoadTgaTexture – это функция загружающая 8, 24, 32-битные текстуры. Принцип загрузки несжатых Targa-текстур довольно простой. Мы просто загружаем заголовок файла, благо известна его структура. 8-битная текстура отличается тем, что она использует палитру для хранения RGB цветов и ссылается к ним по номерам от 0 до 255. 24 и 32-битные текстуры можно практически сразу «залить» в память, но так как мы пока не используем формат BGRA, то придётся поменять цвета местами, получив таким образом из BGR - RGB формат.

```
function LoadTGATexture(Filename: String; var Texture:
Cardinal): Boolean;
var
  { Это и есть наш заголовок Targa формата }
  TGAHeader : packed record
    FileType      : Byte;
    ColorMapType  : Byte;
    ImageType     : Byte;
    ColorMapSpec  : Array [0..4] of Byte;
    OrigX         : Word;
    OrigY         : Word;
    Width         : Word;
    Height        : Word;
    BPP           : Byte;
    ImageInfo     : Byte;
  end;
  TGAFFile      : File;
  image         : PbyteArray; {or PRGBTRIPLE}
  image256      : PbyteArray;
  i,
  ImageSize     : Integer;
  Pall          : Array [0..255] of Array [0..2] of Byte;
begin
  result :=FALSE;

  try
    AssignFile(TGAFFile, Filename);
    Reset(TGAFFile, 1);
    { Читаем заголовок из файла }
    BlockRead(TGAFFile, TGAHeader, SizeOf(TGAHeader));

    { Узнаём о его ширине, высоте и глубине цветов }
    with TGAHeader do
      IF BPP = 8 then
        Begin
          { Читаем палитру из файла 256 цветов * на 3 составных
          цветов R,G,B }
          BlockRead(TGAFFile, PALL[0][0], 768);
          ImageSize :=
            Width*Height;
          GetMem(image256, ImageSize);
          { Загружаем индексированную картинку , т.е. в
          картинке вместо цветов лежат индексы (номера), которые
          должны быть на нужных местах }
          BlockRead(TGAFFile, image256[0], ImageSize);
          GetMem(image, Width * Height * 3);
          { Палитра, к сожалению, тоже хранится в BGR-формате.
          Поэтому конвертируем цвета}
```

```
For i := 0 To ImageSize-1 do
begin
  Image[i*3] := Pall[ image256[i] ] [2];
  Image[i*3+1] := Pall[ image256[i] ] [1];
  Image[i*3+2] := Pall[ image256[i] ] [0];
end;
bpp := 24;
FreeMem(Image256);
end
else
begin
  ImageSize := Width*Height*(bpp div 8);
  GetMem(Image, ImageSize);
  BlockRead(TGAFFile, image^[0], ImageSize);
  { Меняем цвета }
  SwapRGB(Image, Width * Height, BPP div 8);
end;
Texture := CreateTexture(TGAHeader.Width,
TGAHeader.Height, TGAHeader.BPP, Image);
FreeMem(Image);
Result := True;
Except
  Result := false;
end;
end;
end.
```

«Простая» камера 2D

Итак, мы разобрались, как работать с текстурами. Давайте немного займёмся камерой 2D.

Ниже я привёл на мой взгляд самые необходимые вещи, ну а тех которых не хватает — вы сможете самостоятельно дописать, потратив на всё это пару десятков минут.

Немного о процедурах и для чего они нужны:

- Процедура Init, инициализирует камеру, устанавливает её размеры по высоте и ширине, а также начальную позицию;
- Процедура SetView используется для установки камеры в пространстве OpenGL;
- Процедура SetPosition отвечает за настройку позиции камеры;
- Процедура PointInCamera возвращает TRUE, если точка находится в поле видимости камеры.

Ниже я привёл простейший пример реализации нашей камеры:

```
Procedure T2DCamera.Init(W,H:Integer; P:TVector);
begin
  Width := w;
  Height := h;
  Position := P;
end;

Procedure T2DCamera.SetView;
begin
  with position do
    glTranslatef( -x, -y, 0 );
  end;
end;

Procedure T2DCamera.SetPosition( P: TVector );
begin
  Position := v_sub(P, vector(width*0.5,height*0.5));
end;

Function T2DCamera.PointInCamera( P: TVector ): boolean;
begin
  result := (abs(P.x - Position.x)< Width *0.5)
    and (abs(P.y - Position.y)< Height *0.5);
end;
```


РЕМОНТ Sauerbraten-3

Работаем с анимационными моделями



Автор



Петр Василевский

Работает системным администратором управления финансов. Увлекается электроникой и геймдевом. Обожает изучать что-нибудь новенькое.

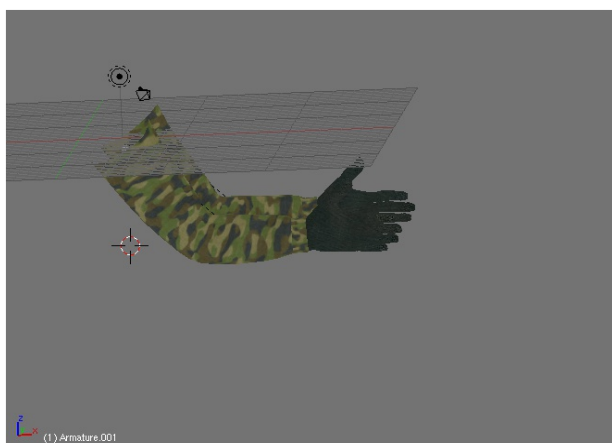
От редакции

В этот раз с помощью автора вы научите любимого героя держать в руках любимое оружие. Продолжение увлекательной эпопеи моддинга от Петра Василевского.

Статья воспроизведена с разрешения сайта: www.lor-ng.org

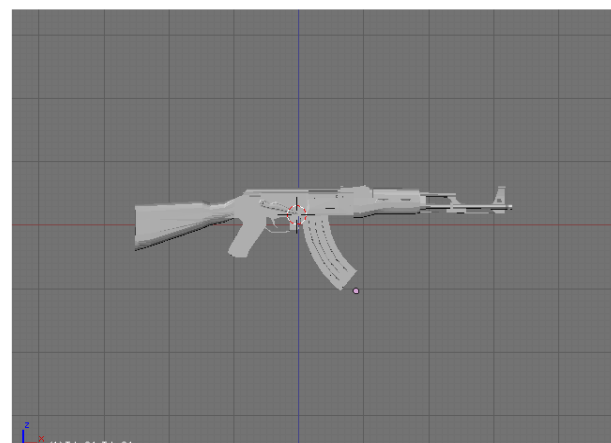
Сегодня мы продолжим обучение тонкостям работы с моделями в Sauerbraten. По умолчанию предполагается что у вас уже есть текстурированные модели рук и оружия содержащие анимации «gun idle» и «gun shoot» в формате .blend (Blender). Если это не так, то вы можете взять модели руки и автомата Калашникова в архиве к журналу.

Итак, приступим. Откроем Blender и загрузим модель рук. Если вы используете модель рук из приложений к журналу, то вам придется продублировать модель руки и арматуру **[Shift+D]**, а затем отразить ее по оси Y **[Ctrl+M]** и **[Y]** (рис. 1).



>> Рис. 1 Наша рабочая модель руки

Теперь загрузим модель оружия. В случае использования модели АК нужно будет добавить арматуру с костями Root, Barrel и Muzzle_Flash.



>> Рис. 2 Модель АК

Перейдите к виду спереди **[Num 1]**. Совместите руки и оружие. Удостоверьтесь, что арматуры рук и оружия разделены. Руки и оружие вместе должны занимать около 30 единичных отрезков Blender в длину и около 20 в высоту. А также отклонение против часовой стрелки по оси Y около 10 единичных отрезков. Теперь соедините теги оружия и рук так, как они будут выглядеть в Sauerbraten. Если вы меняли ракурс, то снова перейдите к виду спереди. Выберите кость арматуры рук более подходящую для роли корневой кости. Зафиксируйте курсор на ней **[Shift+S][Cursor | Selection]**.

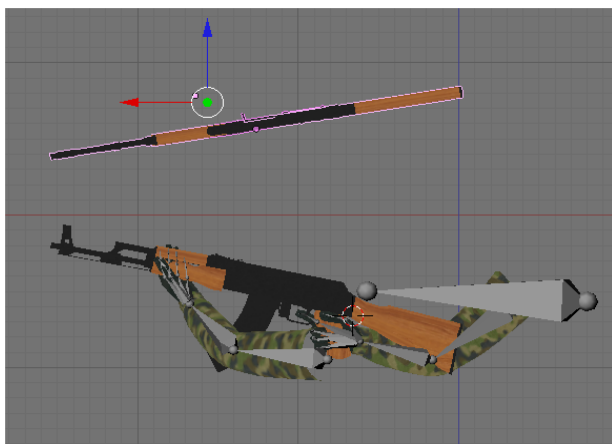
Сделайте объект Empty родителем указанной кости **[Ctrl+P] [Parent to bone]**. Назовите пока Empty подходящим именем, к примеру - 'tag_hands'.

Теперь продублируем модель. Удалите родительскую связь, но оставьте трансформацию **[Alt+P][Clear and keep transformation]**. Теперь корневую кость (Root) оружия. Сделайте Empty родителем указанной кости **[Ctrl+P][Parent to bone]**. Переименуйте Empty на 'tag_ak'. Не беспокойтесь по поводу осей, та как мы займёмся ими позже.

Приступим к анимации рук и оружия. Разделите окно Blender. В одной из частей откройте Action Editor, а в другой Timeline. Удостоверьтесь что анимации для рук оружия различны. Выберите арматуру и перейдите в режим позы (Pose Mode). Теперь с помощью клавиши **[I] [LocRot]** и Timeline нарисуйте анимацию.

Перейдите к виду сзади **[Shift+ Num 1]**. Теперь продублируем оружие, его арматуру и Empty, а также ориентируем последний к координатам «0,0,0».

Дубликат оружия выглядит ориентированным неправильно по отношению к оригиналу. Это потому, что выравнивая Empty к «0,0,0», вы должны зафиксировать курсор в центре **[Shift+C]** и переместить опорную точку на место курсора **[.]**. Выберите только дубликата арматуры и оружия и поверните их по оси X **[R][X][90][.]**. Руки и оружие теперь будут выглядеть как из вида сзади (рис. 3).



>> Рис.3 Правильная ориентация моделей

Сейчас мы должны скопировать анимацию с оригинала оружия на дубликат. Необходимо сохранить полезную анимацию и удалить перемещения дубликата относительно Empty.

Откройте два окна Action Editor. В одном из них выберите арматуру оригинала и соответствующую анимацию (например: AC:gun idle). Добавьте копию анимации с помощью элемента меню **ADD NEW**.

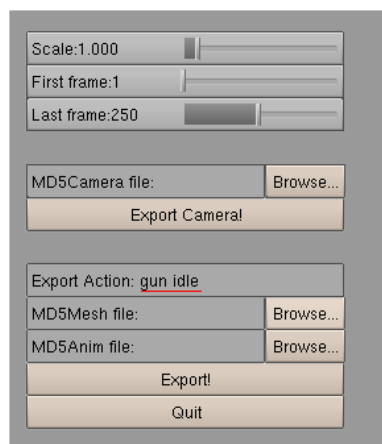
Теперь у нас появилась анимация AC:gun idle.001. Снова выберите анимацию AC:gun idle.

Выберите арматуру дубликата и затем анимацию AC:gun idle.001. Выделите и удалите из нее все ключи для кости Root. Так мы получили оружие, кото-

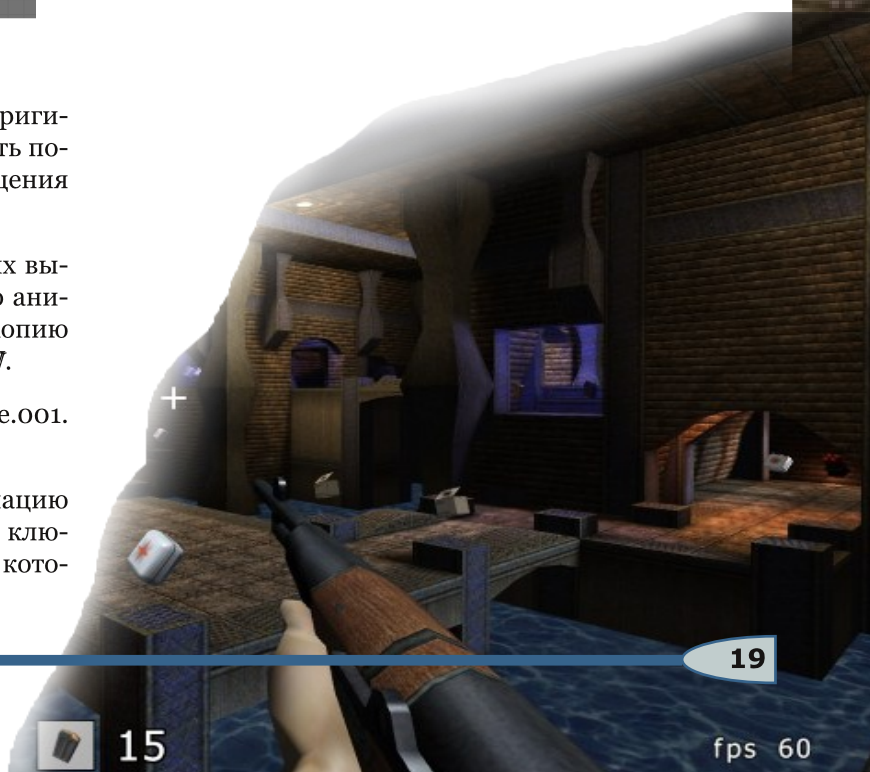
рое ведет себя как оригинал, но без отклонения от Empty.

Из-за различия осей в Blender и других графических приложениях (в том числе и Sauerbraten) следующим шагом убедитесь в правильности направления оружия и рук. Выберите родительскую кость арматуры рук и переместите курсор на нее. Удостоверьтесь, что оси модели X, Y и Z соответствуют осям X, Y и Z в Blender. Повторите тоже самое и для оружия.

Теперь все готово для экспорта. Убедитесь, что у вас открыт последний кадр анимации. Экспортируйте модель рук в файл (например: hands.md5mesh) и заранее заготовленные анимации «gun idle» и «gun shoot» в соответствующие файлы hands_idle.md5anim, и hands_shoot.md5anim. Теперь переименуйте Empty в 'tag_hands'. Выберите Empty дубликата оружия и переименуйте в 'tag_weapon'. Убедитесь, что в Action Editor выбрана нужная анимация, а также, что при экспорте в md5 указана нужная анимация. Теперь выделите модель оружия, и экспортируйте её в файл chaing.md5mesh анимации в chaing_idle.md5anim и chaing_shoot.md5anim (рис. 4).



>> Рис. 4 Настройка экспорта



Теперь осталось написать файл md5.cfg

```
md5dir "hudguns/chaing" //Указываем каталог модели рук
md5load "hands.md5mesh" hands //Загружаем модель рук
//В следующем блоке указываем текстуру в зависимости от
//цвета персонажа
if (>= (strstr (mdlname) "blue") 0) [
    md5skin "fixit_hands" "<dds>fixit_hands_blue.png"
    "<dds>fixit_hands_mask.png" .3 .15
] [
    if (>= (strstr (mdlname) "red") 0) [
        md5skin "fixit_hands" "<dds>fixit_hands_red.png"
        "<dds>fixit_hands_mask.png" .3 .15
    ] [
        md5skin "fixit_hands" "<dds>fixit_hands.png"
        "<dds>fixit_hands_mask.png" .3 .15
    ]
]
md5bumpmap "fixit_hands"
"<dds>fixit_hands_normals.png"//Указываем карту нормалей
для рук
md5spec "fixit_hands" 100 //Указываем уровень отражения
md5glare "fixit_hands" 0.5 1 //Указываем уровень блеска
md5tag Bicep.R tag_bicep
md5dir "hudguns/chaing" //Указываем каталог модели оружия
md5adjust Root 13.25 0 -4 //Указываем выравнивание
md5adjust Hand.R -5 -30 -5
md5anim "gun idle" "hands_idle.md5anim" 1 1 1 //Указыва-
ем анимацию.
md5anim "gun shoot" "hands_shoot.md5anim" 1 30 30 //Где
по порядку идут название анимации, название файла с анима-
цией, начало анимации с 1 кадра по 30 кадр, со скоростью
30 кадров в секунду
//Указываем параметры для оуужия по аналогии
md5load "chaing.md5mesh"
md5skin akm "<dds>m134.png" "<dds>m134_mask.png"
md5bumpmap akm "<dds>m134_normals.png"
md5spec akm 200
md5glare akm 0.5 1
md5tag Muzzle_Flash tag_muzzle
md5adjust Root 0 0 0 -0.75 0.5 0.1
md5anim "gun idle" "chaing_idle.md5anim" 1 1 1
md5anim "gun shoot" "chaing_shoot.md5anim" 1 1 1

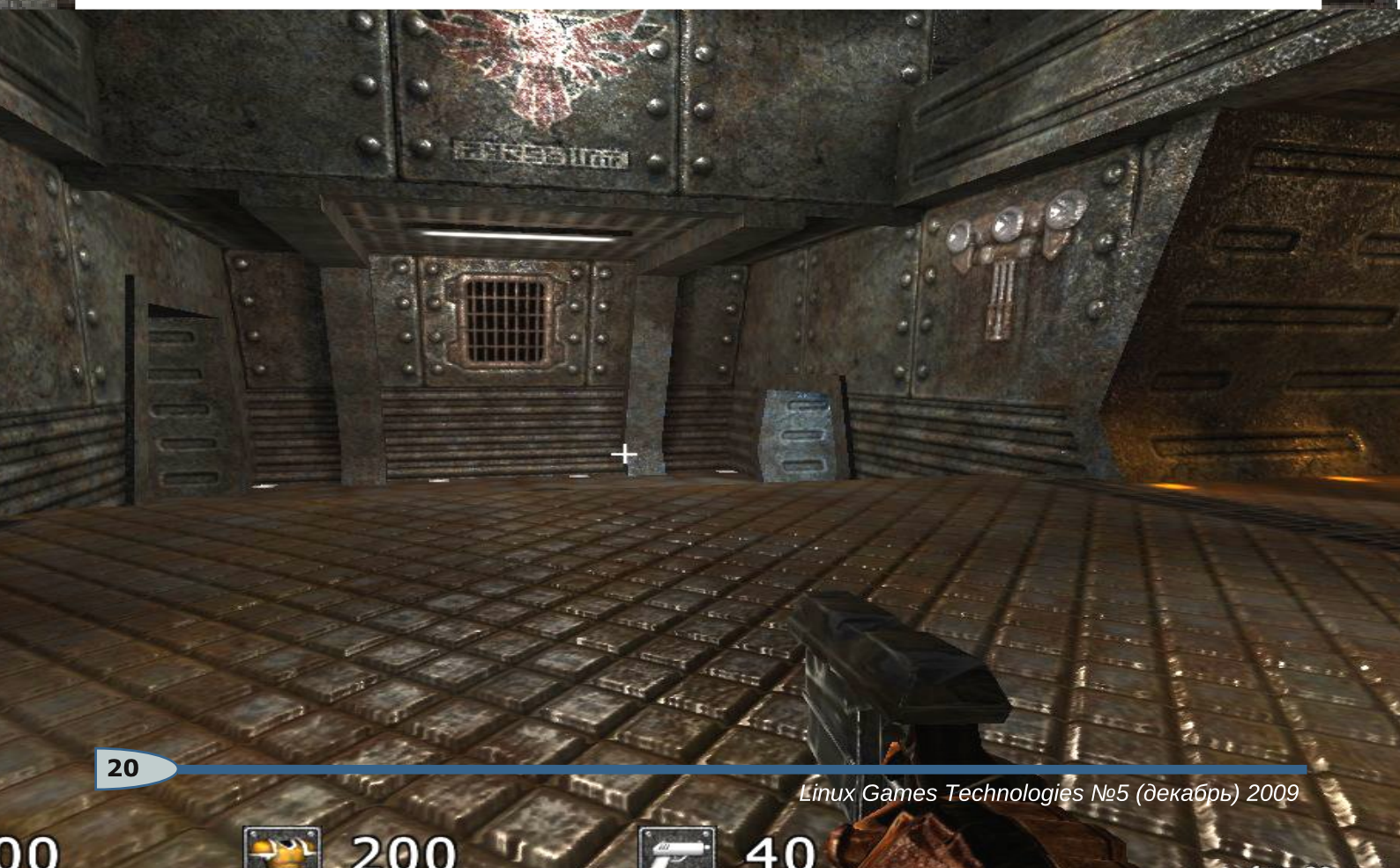
md5link 0 1 tag_bicep //Объединяем две модели по тегу
tag_bicep
mdlscale 300 //Указываем масштаб модели
mdltrans -0.5 -1.8 -2.25
```

Сохраните все в каталог Sauerbraten/packages/models/hudguns/chaing/. Теперь закоментируйте строку exec "packages/models/ PLAYERMODEL*/hudguns/md5.cfg" в файле packages/models/PLAYERMODEL/hudguns/chaing/md5.cfg (где PLAYERMODEL может быть mrfixit, ogro, snoutx10k в зависимости от персонажа, которому заменяются оружие и руки). По аналогии сделайте тоже самое и для вида от третьего лица. Если все прошло гладко, то в игре появится автомат и руки.

LGT



>> Рис.5 «С калашом в руках идём мы в бой»





LINUX GAMES TECHNOLOGIES

№5 (декабрь) 2009



С Новым Годом!