Руководство по Worldcraft 2.0

Автор: or@NGE (orange@igromania.ru)
Статья была опубликована в "Игромании" №7'2000

В данной статье описывается работа с редактором **Worldcraft 2.0**. Редактор поставляется вместе с лицензионной версией Half-Life и доступен для скачивания на сайте ***http://halflife.gamedesign.net*** (полный линк к файлу дать не могу, так как его местонахождение на ftp периодически меняется). Также вы можете найти последнюю версию Worldcraft на нашем компакте.

Кроме Worldcraft для H-L существует еще некоторое количество редакторов (QERadiant, Qoole, Quark, Therd и прочие.) Их рассматривать мы не будем. Хотите — скачивайте, пробуйте... Все ссылки можно найти на ***www.gamedesign.net***.

Настройка

Первым делом убедитесь, что Worldcraft и H-L установлены на одном диске, в противном случае скопируйте из директории H-L в директорию редактора файлы **halflife.wad, liquids.wad, xen.wad** и **decals.wad**. В Tools => Options укажите путь к файлам с текстурами (.wad), к программам-компиляторам (о них ниже), к директории H-L, к самой игре, к файлу **halflife.fgd** (живет в директории редактора), а также к директории **halflife\valve\maps** (если таковой не наблюдается — создать немедленно!).

Теперь немного о компиляторах. Это маленькие программки, которые создают уровень из того, что вы наворотили в редакторе. Их три: **qbsp2.exe** (отвечает за архитектуру, монстров и действующие объекты — entities), **qrad2.exe** (обрабатывает простые и цветные источники света) и **qvis2.exe** (используется чаще всего для финальной компиляции уровня, удаляет невидимые грани объектов, оптимизирует уровень и ускоряет его работу).

Замеченные глюки

Как известно, идеальных редакторов не бывает. Не лишен недостатков и наш с вами подопечный. Из замеченных "глюков" отметим следующие. 3D Preview при включенной hardware acceleration не очень хорошо работает под DirectX 7. При наклеивании текстур на плоскости редактор иногда вылетает, так что рекомендую почаще нажимать **Ctrl+S**.

Принципы и понятия

Интерфейс Worldcraft напоминает 3dMax, разве что расположение окон по умолчанию немного другое (см. картинку **w1.jpg**, на ней есть цифровые обозначения):

1 — Вид сверху (XY);
2 — Вид спереди (XZ);
3 — Вид сбоку (YZ);
4 — 3D Preview — примерно так будет выглядеть ваше творение в самой игре.

Итак, делаем первый шаг. Выбираем подходящую текстуру (в окне справа). Включаем **Block tool** (Shift+B) и рисуем, извините за выражение, параллелепипед. Удовлетворившись его размерами, нажимаем Enter. Вот и наш первый объект (браш). Пусть это будет пол. Выбираем другую текстуру, делаем таким же образом стены и потолок. Вот и готова наша первая комната!

Сделать комнату можно и более простым образом: создаем довольно большой браш, выделяем его и делаем **'hollow'** (все команды доступны с правой кнопки). Чтобы

вырезать с помощью одного браша дырки в другом, выделите его и сделайте **‘carve’**.

Entities

Действующие объекты. Делятся на **Point entities** — объекты, заранее созданные разработчиками (монстры, аптечки, источники света, оружие), и **Solid entities** — созданный вами браш с назначенными свойствами (дверь, лифт, стекло). Начнем с самого основного — с точки старта, без нее уровень просто не запустится. Врубаем **entity tool** (Shift+E), в менюшке справа выбираем **‘info_player_start’**, выбираем место, где хотим его поставить, и давим Enter.

Точно таким же образом создаем источники света (а то в темноте бегать, согласитесь, немного стремно). Выбираем в меню **‘light’** и размещаем его там, где, по вашему мнению, должно быть светло. Выделив созданный объект и нажав Alt+Enter, можно настроить его свойства (для каждой entity они свои, например, для источника света это яркость и цвет).

Для полного счастья сделаем еще и дверь. Выбираем подходящую текстуру, создаем браш, выделяем его, кликаем **‘to entity’** и в появившемся меню выбираем **‘func_door’**, назначаем свойства (скорость, направление, в какую сторону дверь будет открываться, звук, с которым она будет двигаться, и прочие ее характеристики).

Теперь разложим оружие и расставим монстров. Все делается точно так же, как и в случае с источниками света. Около места старта (или прямо на нем) рекомендуется разместить **‘item_suit’** — без него играть, мягко говоря, неудобно: вы не сможете пользоваться аптечками, не будете видеть уровень здоровья и даже не сможете переключать оружие.

Связи между объектами

Для этого служат пункты name и target в свойствах объектов. Объясню на простом примере.

Допустим, вам надо, чтобы при нажатии кнопки открывалась дверь. Создаем дверь, в пункте **name** у нее пишем, допустим, **door1**. Далее создаем кнопку (func_button), в пункте name у нее не пишем ничего, но в пункте target пишем имя двери (в данном случае **door1**).

Объекты типа ‘env’

Как правило, это мелкие события, осуществляемые посредством спрайтовых объектов и звуков. Используются исключительно для красоты.Например, при включении **‘env_spark’** появляются трещащие искры, а **‘env_explosion’** создает взрыв, а **‘env_sound’** управляет акустикой помещения. Чаще всего для активизации таких объектов используются триггеры...

Триггеры

Невидимые объекты, которые управляют другими, видимыми объектами. Триггеры делятся на **solid** и **point**. Чаще всего используются **‘trigger_once’** и **‘trigger_relay’**

Пример первый: вам хочется, чтобы при прохождении через определенное место неожиданно раздавался взрыв. Создаем **env_explosion**, называем его некоторым именем (как в случае с кнопкой и дверью). На том месте, при проходе через которое должно рвануть, создаем браш и превращаем его в **‘trigger_once’**, лезем к нему в самые **properties** и в поле **target** пишем имя, которым мы наградили **env_explosion**.

А теперь о применении **trigger_relay** (прошу заметить, это **point entity**, грузится оттуда же, откуда оружие и монстры). Например, нужно, чтобы через некоторое время после взрыва прозвучал еще один. Создаем второй взрыв, называем его другим именем, создаем **trigger_relay**, называем его тем же именем, что и первый взрыв, а в **target**

пишем имя второго взрыва. В поле **delay** пишем время в секундах (промежуток между взрывами). Все!

Декали

Это такие специальные текстуры, которые хранятся в файле **decals.wav**. Использовать их можно только накладывая поверх других текстур. Включаем **decal tool**, выбираем текстуру из **decals.wad** и в окне с **3d preview** наклеиваем туда, куда хотим. Набор декалей довольно ограничен. В основном это следы от взрывов, трещины, брызги крови и различные надписи.Раз уж речь зашла о текстурах, то позвольте также рассказать про фичу **texture application**. Врубается соответствующей кнопкой и действует следующим образом. Допустим, у вас есть куб. Вам надо, чтобы на одной плоскости была одна текстура, а на другой — другая... Пользуясь вышеназванной фичой, выбираем в появившемся окне нужную текстуру и кликаем правой кнопкой на плоскость, куда хотим ее прилепить. Так вот...

Компиляция

А теперь посмотрим, как выглядит то, что вы натворили!

Жмем F9 или кнопку с геймпадом на верхнем тулбаре, появится окно с опциями компиляции.

Для предварительного просмотра выбираем: **qbsp — normal, qrad — normal, qvis — no**. После нажатия OK появится окно с процессом компиляции... Чем уровень больше, тем дольше он компилируется (полная компиляция большого уровня может занять несколько часов). Если все нормально, то через некоторое время должна запуститься сама игра с вашим уровнем.

Уровень может не откомпилироваться (или откомпилироваться, но не полностью) по нескольким причинам:

— если в уровне есть дырки в окружающее пространство (так что старательно следите за герметичностью уровня);

— если какой-то из объектов (entity) полностью или частично находится внутри браша (особенно в этом плане привередливы источники света);

— если на уровне отсутствует **info_player_start**;

— если на уровне слишком много мелких или плоских брашей (при компиляции обычно проскакивает сообщение о количестве таких деталей и о их возможном максимальном количестве). Кстати, если у вас что-то вызывает сомнение, то бывает полезно нажать Alt+P — тогда редактор проверит уровень и укажет места с возможными ошибками (самая распространенная — invalid brush).

Поиск дырок: рекомендуется перед полной компиляцией сначала запустить уровень только с **qbsp** (так быстрее), а в самой игре, в консоли, написать волшебное слово **“pointfile”**. Если в уровне есть дырки, то они будут отмечены пунктирной линией (прямо в игре), тогда врубаем **noclip** и летаем по этой самой линии в поисках дырки.

Кстати, любая entity, а также жидкий браш (вода, кислота), находящиеся за пределами уровня, также считаются дыркой.

Несколько полезных советов

Следите, чтобы уровень был выдержан в одном стиле. Погуляйте по стандартным уровням с включенными god и notarget и внимательно рассмотрите архитектуру. Обращайте внимание, где и как используются текстуры.

Определитесь с тематикой уровня. Что это будет — завод по переработке радиоактивных отходов или военная база? Не перестарайтесь с использованием **prefabs** (готовых объектов)... Поверьте, очень убого смотрятся поделки горе-дизайнеров (в основном американских), сплошь заставленные диванами, унитазами и

газонокосилками...

Следите за реалистичностью. Стены из воды — вечная проблема начинающих левелмейкеров (они думают, что это круто). Оружие также должно быть разложено с умом. Не будет же egon просто так валяться на улице.

Не размещайте рядом монстров, враждующих между собой (например, десантники и пришельцы).

Небольшое замечание насчет вращающихся объектов: чтобы, скажем, вентилятор вращался вокруг оси, надо эту самую ось сначала сделать. Изготовляется она из текстуры **origin**. В противном случае объект будет летать по всему уровню. То же касается и створчатых дверей (func_door_rotating).

Worldcraft 3.3

Недавно увидел свет **Worldcraft 3.3** (вот так сразу, без всяких промежуточных версий... сразу с 2.1 на 3.3). Начнем с того, что это чудо инженерной мысли обзавелось долгожданной поддержкой OpenGL, но дело в том, что в данном случае ускоритель превращается в тормозитель... Отключить данную “полезную” функцию не представляется возможным. А теперь позвольте привести цитату из пресс-релиза: “OpenGL 3D Renderer. The real-time renderer is much faster, more robust, and allows for animating visuals in the 3D view”. Ну-ну...

Очень плохо сделана поддержка mouse wheel — колесо отвечает не за прокрутку 2D окон (как можно было бы подумать), а за уменьшение/увеличение.

Но и без приятных нововведений не обошлось. Например, такая фича — **Rotation texture lock** (когда вы поворачиваете объект, текстура поворачивается вместе с ним, а не остается, как это было раньше). Немного удобнее и функциональнее стала **Texture Application Tool**.

Entities теперь показываются в **preview** не в виде кубиков, а в виде спрайтов (хорошо, что не в виде моделей). Ну и плюс еще несколько совсем незначительных возможностей... Решайте сами, надо это вам или нет. Скачать новую версию редактора можно здесь:

http://halflife.gamedesign.net

****