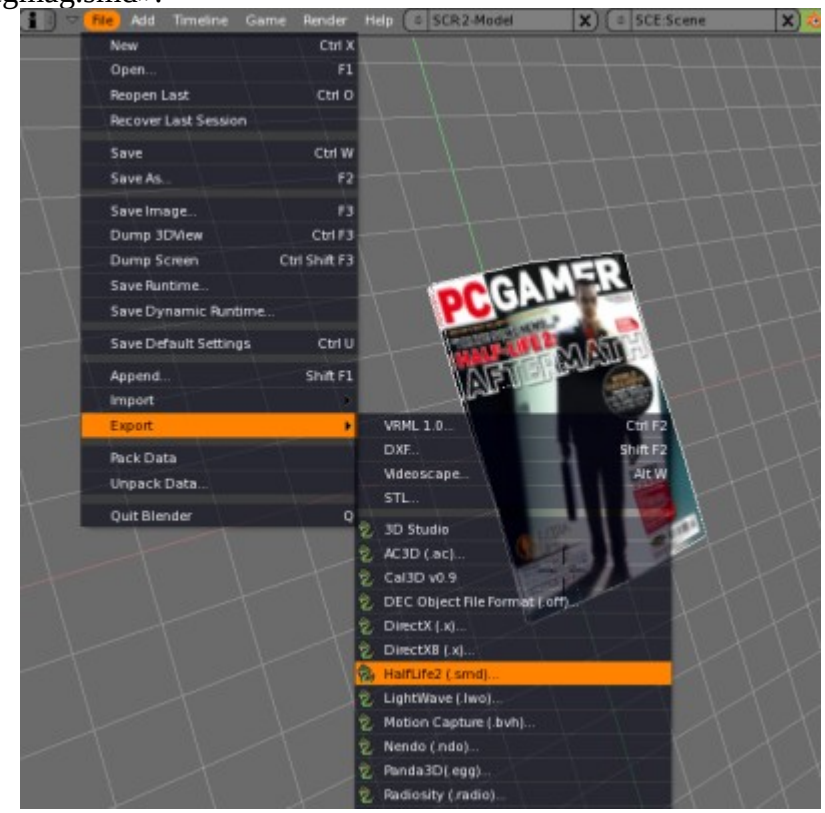
СОЗДАНИЕ МОДЕЛЕЙ В BLENDER'E

Примечание: предполагается знание основ работы в Blender. Хотя бы свободное создание простых моделей, применение UV-текстурирования и установки плагинов. Очень поможет практическое умение скомпилировать модель.

Создание базового нематериального пропа

1. Создайте модель и покройте её текстурой с помощью UV-маппинга. В результате получится файл в формате blend и все TGA-файлы, которые вы использовали в качестве текстур. Для примера использован журнал «PC Gamer UK», спрятанный на de_dust_pcg (secreted all around de_dust_pcg) (blender-файл модели - «pcgmag.blend», а файл текстуры - «pcgmap.tga»).
2. Установив плагин SMD exporter, экспортируйте вашу модель в формат Half-Life 2 SMD, в данном случае — «pcgmag.smd».



3. Если модель для Counter-Strike: Source, скопируйте SMD-файл в каталог SteamApps/username/sourcesdk_content/cstrike/modelsrc, а TGA-файл(ы) - в SteamApps/username/sourcesdk_content/cstrike/materialsrc/models.
4. SMD-файл, который вы видите - лишь ссылка на модель (is just the reference model). Нам также потребуется дополнительный SMD-файл для обозначения анимации «idle». Так что создаём в текстовом редакторе файл «pcgmag_idle.smd» (или как-то ещё, но с расширением SMD) и вводим следующее:

```
version 1
nodes
0 "joint0" -1
end
skeleton
time 0
0 0.000000 0.000000 0.000000 0 0.000000 0.000000
end
```

5. Компиляторам нужен QC-файл, содержащий информацию о модели. Это pcmag.qc:

```
$modelname pcgmag.mdl
$cdmaterials "models"
$scale 3.5
$surfaceprop "paper"
$staticprop
$body studio "pcgmag.smd"
$sequence idle "pcgmag_idle" fps 1
```

Обратите внимание на значение «Scale», который отвечает за правильность размера журнала в игре. Также взгляните на строку «$surfaceprop», устанавливающую такие свойства материала, как трение или издаваемые звуки. В отличии от обычного браша, все $surfaceprop'ы, заданные в материале пропа (VMT) игнорируются (any $surfaceprop set in the prop's material (.VMT) files is ignored); вместо этого они задаются в QC-файле.

6. Текстура должна быть скомпилирована в формат VTF, точно так же, как и любая нормальная текстура. В любом случае, она должна содержать объявление "$model". Для компиляции текстур в формат движка Source хорошо подойдёт программа VTFEdit. Она импортирует большинство форматов изображений и имеет графический интерфейс для создания VMT-файлов. Ниже приводится VMT-файл, использованный для pcgmag.tga:

```
"VertexLitGeneric"
{
	"$baseTexture" "models/pcgmag"
	"$model" 1
}
```

Какое бы имя файла текстуры не было использовано, модель в Blender'е будет использована в качестве названия файла - материала для этой модели. Например, если для текстуры использован файл pcgmag.tga, то модель будет использовать материал pcgmag.vmt. При использовании текстур с длинными именами файлов, Blender обрежет конец. Экспортируя модель, он будет вместо оригинального имени искать сокращённое, использованное Blender'ом. Дабы проверить, какое имя использовал Blender, взгляните в окно выбора текстур в редакторе изображений Blender'а (или окно UV-маппинга). Прежде, чем компилировать какие-либо текстуры, удостоверьтесь, что имена файлов такие же, как имена текстуры в окне. Вы также можете поменять имена файлов после компиляции, и всё будет работать, если вы также измените имена файлов в VMT-файле. Вы можете использовать много файлов-текстур для одной модели.

7. Наконец, перейдите в каталог исполняемых файлов в SourceSDK, содержащий studiomdl.exe, и скомпилируйте модель, используя команду вроде этой:

```
studiomdl.exe "..\..\sourcesdk_content\cstrike\modelsrc\pcgmag.qc"
```

Хорошей альтернативой для компиляции моделей является GUIStudioMDL. Вы загружаете ваш .qc и нажимаете кнопку компиляции, также имеется набор других опций, которыми могут воспользоваться более опытные пользователи.

8. Если всё прошло как надо, вы получите хороший результат (внимательно следите за любыми ошибками) и сможете найти модель в браузере моделей Hammer'а для использования в игре.

Добавление нескольких скинов

Сделать так, чтоб проп имел 2 и более скина несложно. Для модели журнала можно было бы сделать несколько вариантов обложки. Для начала создайте новую VTF-текстуру, следуя тому же UV-маппингу, что был использован в оригинальной текстуре. Также сделайте копию первого VMT-файла (например, сохраните под именем «pcmag_cover2.vmt»), убедившись, что обновили опцию «$baseTexture» для новой текстуры.

Теперь добавьте следующую команду в QC-файл, сразу после строки «$body»:

```
$texturegroup skinfamilies
{
   { "pcgmag" }
   { "pcgmag_cover2" }
}
```

Здесь оригинальной текстурой скина была «pcgmag», а «pcgmag_cover2» является материалом для второго скина. Чтобы создать больше скинов - добавьте ещё таких строк.

После перекомпиляции модели, при условии, что все файлы-текстуры находятся в нужных местах, теперь вы можете сменить скин пропа через соответствующую опцию в Hammer'e.

Добавление коллизий (collisions)

На данный момент модель можно использовать только в качестве prop_static, массы она не имеет. Игроки, объекты и пули будут проходить насквозь. Чтобы это исправить, надо добавить модель коллизий (collision model).

Для простого пропа вроде нашего, создать модель коллизий несложно. Мы можем использовать тот же меш, что у модели, но в Blender'e, выбираем весь меш и жмём «Set Smooth» (под панелью «Link and Materials» в режиме редактирования). Сделать это нужно по той причине, что компилятор ожидает того, что модель коллизий будет иметь единственную текстуру для каждой выступающей части в одной группе сглаживания (smoothing group). Теперь экспортируйте меш как «pcmag_phys.smd». Это будет нашей моделью коллизий.

Если reference-меш имеет не просто выпуклую форму, но и достаточно закрытую, компилятор создаст «выпуклую оболочку» вокруг специальной модели коллизий, заполняя все вогнутые куски (If the reference mesh isn't a simple convex shape, but is pretty close, the model compiler will build a «convex hull» around the specified collision model, filling in any concave bits). Также вы можете создать отдельную модель коллизий, создав куб точно тех же размера и формы, что и проп. Обратите внимание, что скрипту «SMD export» нужно, чтобы меш был с развёрткой (requires a mesh to be UV-mapped), так что придётся применить UV-маппинг к модели коллизий для её экспорта, даже если она никогда не будет отображена на экране. Более сложные пропы могут быть совсем бесформенными, ситуация будет рассмотрена ниже.

Измените QC-файл, чтобы добавить модель коллизий к пропу. Приведите файл pcmag.qc к следующему виду:

```
$modelname pcgmag.mdl
$cdmaterials "models"
$scale 3.5
$surfaceprop "paper"
$staticprop
$body studio "pcgmag.smd"
$sequence idle "pcgmag_idle" fps 1

$collisionmodel "pcmag_phys.smd"
```

Только что мы добавили строку «$collisionmodel», которая ссылается на колиззионный меш (collision mesh). Перекомпилируйте модель, и, будучи добавленной как prop_static в Хаммере, она будет иметь плотность. Игрок и другие объекты не смогут пройти сквозь неё, а выстрел будет остановлен и на модели появится след от пули.

Добавление физики

Теперь у нас есть модель коллизий, но проп по прежнему статичен: он не может двигаться, на него не влияют взрывы и гравитация. Чтобы это изменить, следует добавить некоторую физическую

информацию в проп.

Приведите файл «pcmag.qc» к следующему виду:

```
$modelname pcgmag.mdl
$cdmaterials "models"
$scale 3.5
$surfaceprop "paper"
$staticprop
$body studio "pcgmag.smd"
$sequence idle "pcgmag_idle" fps 1

$collisionmodel "pcmag_phys.smd"
{
   $mass 0.5
}

$keyvalues
{
   "prop_data"
   {
      "base"     "Cardboard.Small"
   }
}
```

Мы добавили две секции: после строки «$collisionmodel» мы добавили секцию в скобках, а также новую секцию «$keyvalues», содержащую некоторую информацию о пропе. Заметим, что «$staticprop» у нас всё ещё есть, несмотря на то, что мы больше не пытаемся создать prop_static. Несмотря на такое имя, эта команда используется не только при компиляции prop_static'ов, но и для любого другого пропа, не имеющего анимаций или нуждающегося в структуре «костей» (bome structure) (то есть что-то кроме prop_dynamic'ов или prop_ragdoll'ов).

Секция после «$collisionmodel», а точнее, команда «$mass» в скобках, определяет массу пропа в килограммах. Поставьте здесь до 0.5 кг - это обычная масса для забитого рекламой компьютерного журнала (advert-laden PC magazine). Раздел «$keyvalues» определяет дополнительные характеристики, которые могут быть использованы движком - здесь мы задаём prop_dat'у для модели. Базовый тип пропа «Cardboard.Small» («Картон.Мелкий»), наилучшим образом подойдёт для журнальной бумаги. Prop_data требуется для любого физического пропа и определяет такие вещи, как жизнь и повреждения пропа от разных эффектов. Полный перечень можно найти в файле scripts/propdata.txt в GCF-архиве «source engine.gcf» или «counter-strike source shared.gcf».

Теперь перекомпилируйте QC-файл с помощью studiomdl.exe. Теперь в Хаммере можно использовать модель как prop_physics. Она будет падать под влиянием гравитации, сталкиваться с объектами и игроком, двигаться от выстрелов и взрывов.

Сложные модели коллизий

Мы научились создавать модели коллизий для простого выпуклого объекта, вроде журнала или книги. Но для того, чтобы создать не выпуклый объект, такой как стол, придётся проявлять большую аккуратность для того, чтобы проп соударялся с другими вещами так, как ожидается.

Перед нами деревянный стол, созданный в Blender'e. Используем subsurfaces. У них есть вершина с закруглёнными краями и клиновидные цилиндрические отрезки. Можно сделать модель коллизий из того же меша, из которого сделан сам стол, но это займёт слишком много времени - колиззионные меши должны быть как можно более простыми, чтобы физическому движку пришлось меньше работать.

Чёрный каркас показывает наложение колиззионных мешей поверх модели путём добавления пяти отдельных объектов в Bledner'e. Верхняя часть стола - просто сплющенный куб, а каждая из ножек - клиновидный куб (или усечённая пирамида), повторяющая форму видимой ножки. На практике, возможно, лучше будет срезать углы верхней части стола, для лучшего соответствия reference-модели, но для примера сойдёт и так. Поместите коллизионный меш на отдельный от видимой модели слой, чтобы было легче редактировать их независимо друг от друга.

Важно, чтобы начала объектов reference-мешей и колизионных мешей были в одном месте (It's important that the object origins of the reference meshes and the collision meshes are in the same place). Самый простой способ этого добиться - выделить все меши, поместить 3D курсор в середину стола и нажать «Centre cursor». Это сдвинет все источники объекта (object's origins) к курсору; в противном случае модель коллизий и видимая модель могут оказаться не выровненными в игре, порождая некоторые странности.

Можно экспортировать SMD модель (table.smd), выбирая объекты и используя скрипт экспорта так же, как раньше. Убедитесь, что экспортируете в этот файл только те меши, которые желаете увидеть в игре.

Для экспорта модели коллизий нужна некоторая аккуратность. Во-первых, следует убедиться, что каждая выпуклая часть модели коллизий является в Blender'e отдельынм объектом. В объектном режиме, при выделении объекта (ПКМ), должна будет подсветиться только эта часть. Если после одного клика подсвечены две части, значит, они, фактически, находятся в одном объекте. Можно разделить их, перейдя в режим редактирования, выбрав все вершины одной части объекта и нажав клавишу «P».

Компилятор ожидает, что каждая часть сложной модели коллизий будет находиться в отдельной группе сглаживания (smoothing group). На самом-то деле Blender не знает такого понятия, как группа сглаживания, но его можно эмулировать, выбирая каждый объект модели коллизий по одному и нажимая «Set Smooth». Не пытайтесь выбрать сразу все объекты и нажать «Set Smooth» - результат будет не такой. Как только вы всё это сделаете, выберите все колиззионные объекты и экспортируйте их в «table_phys.smd». Помните, что скрипт экспорта требует, чтоб все меши были с развёрткой (exporter script requires all meshes to be UV mapped), так что перед экспортом приените простой UV-маппинг к колиззионным мешам (поскольку оно никогда не будет видимым, не важно, какую вы используете текстуру).

Как и до этого, для определения «idle»-анимации нам потребуется файл «table_idle.smd», который может быть идентичен файлу «pcgmag_idle.smd», использованному нами ранее.

Теперь нам нужен QC-файл для пропа стола:

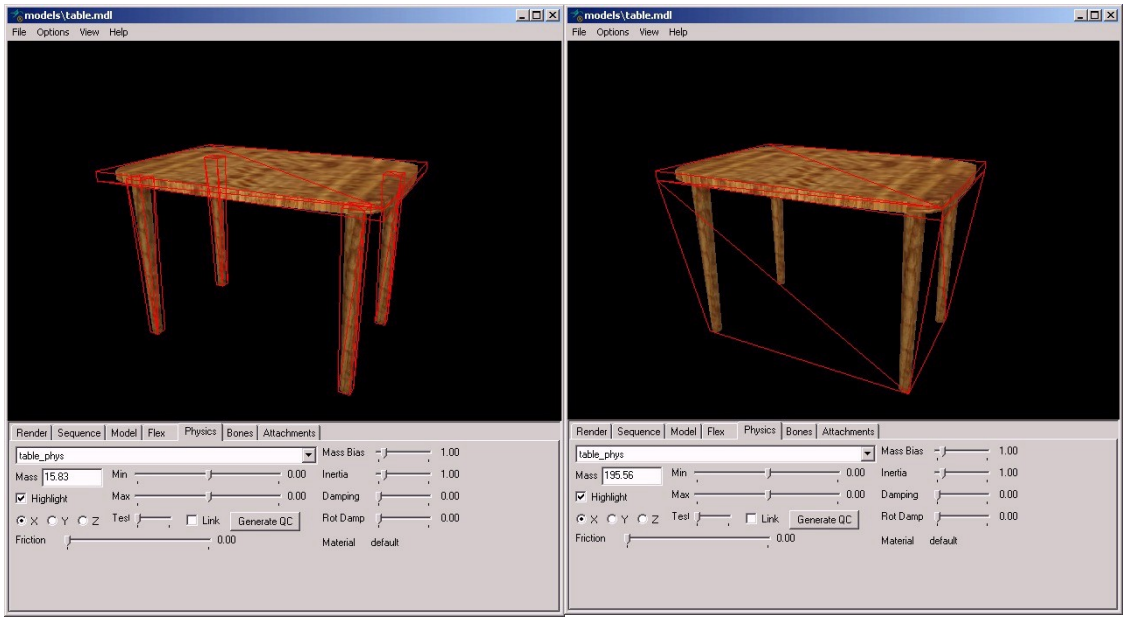
```
$modelname rof/table.mdl
$cdmaterials "models/rof"
$scale 12.0
$body studio "table.smd"
$sequence idle "table_idle" fps 1
$staticprop
$surfaceprop "Wood_Furniture"

$collisionmodel "table_phys.smd"
{
  $automass
  $concave
}

$keyvalues
{
  "prop_data"
  {
     "base"  "Wooden.Large"
  }
}
```

Обратите внимание на отличия от того, что было ранее. Мы немного увеличили размер и установили «$surfaceprop» и проп-дату на «Wooden.Large» («Деревянный.Большой»), дабы тип пропа соответствовал тому, что мы создаём. В разделе «$collisionmodel» мы, помимо строки «$mass», используем «$automass». Это заставит компилятор вычислить массу для объекта, взяв за основу его размер и тип материала. Вместо этого мы можем, как и в прошлый раз, задать массу объекта с помощью «$mass».

Также мы добавили в этот раздел строку «$concave». Это заставит компилятор создать не имеющий формы колиззионный меш (non-convex collision mesh) для каждой части файла «table_phys.smd», вместо того, чтобы смешать всё в одной выпуклой форме (lump them all together in one convex shape). Можно определить, сработало ли это при компиляции QC-файла. Первые строки в выводе studiomdl.exe должны быть чем-то вроде:

```
c:\steam\steamapps\username\half-life 2\hl2\modelsrc\, c:\steam\
steamapps\username\half-life 2\hl2\, path table
Working on "table.qc"
SMD MODEL table.smd
SMD MODEL table_idle.smd
SMD MODEL table_phys.smd
Model has 5 convex sub-parts
Collision model completed.
Computed Mass: 15.83 kg
...
```

Строка «model has 5 convex sub-parts» говорит нам о том, что модель коллизий работает так, как надо. Если вместо этого высвечивается «Model has 1 convex sub-parts», значит, мы либо не определили команду «$concave», либо зафейлили нормальное создание мешей колиззионных моделей (failed to make the collision model meshes correctly).



Мы можем проверить модель коллизий с помощью Model Viever'a (hlmv.exe). На вкладке «Physics» нажмите кнопку «Highlight» и колиззионный меш отобразиться красным каркасом. Если всё нормально, он должен выглядеть, как на картинке слева: каркас повторяет каждую часть по отдельности. Если что-то не так, вид будет, как на картинке справа: каркас, будучи единой формой, обёртывает весь объект. В игре модель стола справа будет странно себя вести - мы не можем выстрелить между ножек, а если его перевернуть кверх ногами, то помещённый на него объект зависнет в невидимой коробке под столом. Модель слева будет вести себя нормально. Если у вас возникла та же проблема, что со столом справа, когда физический меш является единой оболочкой, попробуйте увеличить размеры физической модели в Blender'e. Компилятору нужно, чтобы фейсы были больше, чем 1 дюйм, на каждой оси, так что если фейс короче или тоньше одного дюйма, компилятор выдаст предупреждение и создаст свой собственный физический меш. Не забудьте также увеличить размер нормальной модели, в противном случае физическая модель и отрендеренная модель не будут совпадать!

Добавление пользовательских моделей обломков

Создав модель стола, вы обнаружите, что она разбивается на несколько кусков дерева, будучи достаточно повреждённой. Это обломки модели, и они указываются в файле «propdata.txt» для пропов типа «Wooden.Large». Так или иначе, те обломки, что есть по умолчанию, не очень подходят нашему столу, поэтому можно создать свои модели вместо них.

Для начала нужно разбить стол на кусочки. Сначала сделайте копию (Shift+D) стола и уберите её на другой слой. Теперь порубите стол, используя «Knife tool» (Shift+K). Можно разрезать одну из ножек посередине, и поломка с одного угла приведёт к отламыванию в второй ножки (broken the tabletop across one corner so that comes away with the second leg). Повреждённые рёбра были огрублены, чтобы выглядеть достаточно повреждёнными. Две другие ножки просто получат повреждения (The

other two legs will simply drop-off whole). Не забудьте добавить грани (и UV-маппинг) в те области модели, которые станут видимыми после поломки - например, вершины ножек, видимость которых была лишней раньше, но не сейчас.

Доведите модели коллизий (чёрные каркасы) до соответствия сломанным кускам. Для некоторых ножек сделайте колиззионные меши более закруглёнными 0 это позволит ножкам красиво кататься, что им не было нужно, пока они были прибиты к столу. Заметим, что две из этих моделей обломков (стол с обрубком ножки и ножка с отломанным углом), имеют вогнутые модели коллизий, и поэтому должны быть созданы с использованием вышеназванных методов.

Как и прежде, следует удостовериться, что каждый из ориджинпоинтов (origin points) объекта (включая таковые у моделей коллизии) находятся в тех же местах (используя команду «Centre cursor»), а также они должны соответствовать оригинальным ориджинпоинтам первоначальной модели стола. Разместите обломки в тех же местах, где находится оригинальная, несломанная модель. Это заставит обломкам появиться в нужном месте, когда стол сломают (заметим, что на картинке модели для наглядности размещены чуть в стороне).

Теперь нужно экспортировать и скомпилировать модель для каждого обломка вместе с их колиззионными мешами. Ниже приводится содержимое QC-файла для части стола:

```
$modelname rof/table_gib_top.mdl
$cdmaterials "models/rof"
$scale 12.0
$body studio "table_gib_top.smd"
$sequence idle "table_idle" fps 1
$staticprop
$autocenter
$surfaceprop "Wood_Furniture"

$collisionmodel "table_gib_top_phys.smd" {
    $automass
    $concave
}

$keyvalues
{
  "prop_data"
  {
     "base"            "Wooden.Medium"
  }
}
```

Заметьте, что имя модели - «table_gib_top.mdl». Почти идентичный QC-файл используется для четырёх оставшихся кусков, с «table_gib_leg1.mdl» до «table_gib_leg4.mdl». Единственная новинка - команда «$autocenter», которая автоматически поместит модель в центре, таким образом, чтобы освещение и коллизии корректно работали.

Скомпилировав все пять моделей обломков, нам остаётся только добавить новую команду в оригинальный «table.qc»:

```
$collisiontext
{
        break { "model" "rof/table_gib_top" "health" "20" "fadetime" "0" }
        break { "model" "rof/table_gib_leg1" "health" "20" "fadetime" "0" }
        break { "model" "rof/table_gib_leg2" "health" "20" "fadetime" "0" }
        break { "model" "rof/table_gib_leg3" "health" "20" "fadetime" "0" }
        break { "model" "rof/table_gib_leg4" "health" "20" "fadetime" "0" }
}
```

Это заставит появиться обломки, когда первоначальная модель сломается. Каждый обломок может получить значение жизни, а также значение «fadetime», определяющее, как много пройдёт времени (в секундах) прежде, чем модель исчезнет из виду. Установка значения «0» означает, что модель не пропадёт никогда. Теперь скомпилируйте «table.qc», и модель будет готова.

Есть один странный эффект, заключающийся в том, что модели обломков не будут соударяться друг о друга (но они будут сталкиваться с другими объектами). Создание большего количества более мелких обломков сделает эту проблему менее заметной. Кроме того, каждая модель обломка может быть в дальнейшем при повреждении разбита на стандартные деревянные обломки. Для каждой модели это задаётся в разделе «prop_data».

Перевод: Mapper720 (27-28.10.2012)
Оригинал: https://developer.valvesoftware.com/wiki/Modeling_props_with_Blender