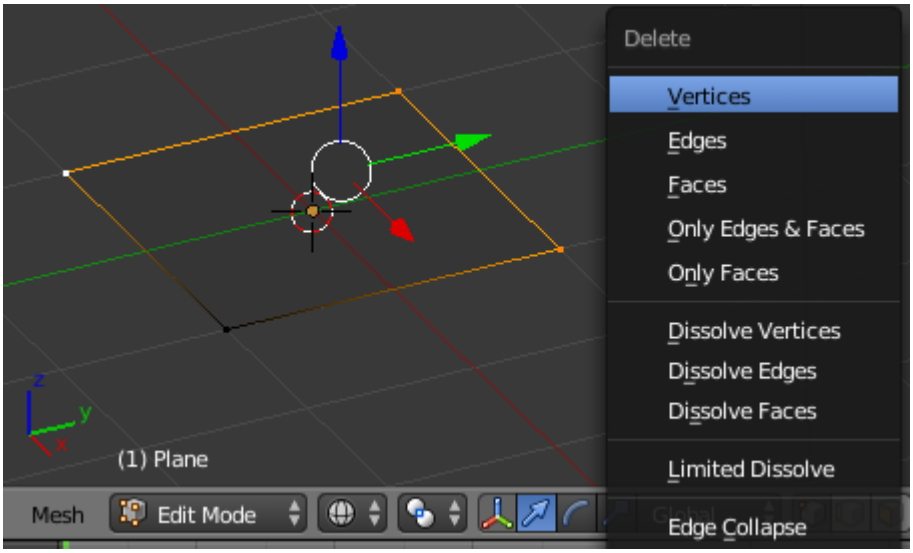
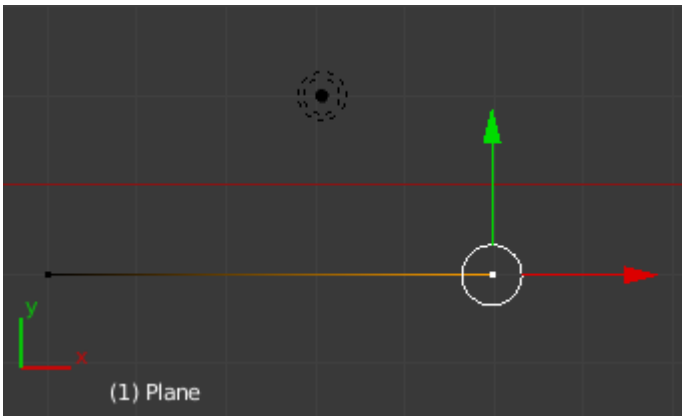
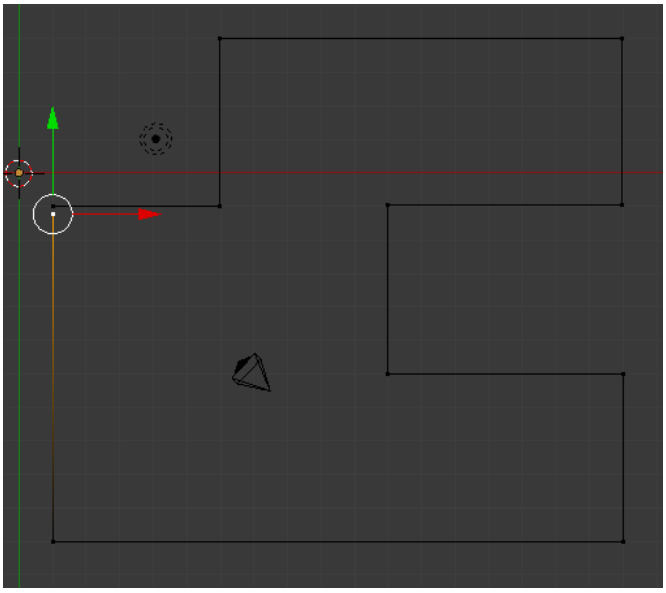
Прежде, чем мы будем рассматривать новые темы, давайте вспомним и закрепим вопрос текстурирования. Создадим произвольную комнату буквально по точкам. Для этого добавим план, перейдём в режим редактирования, выделим три вершины и удалим их (оставив одну):



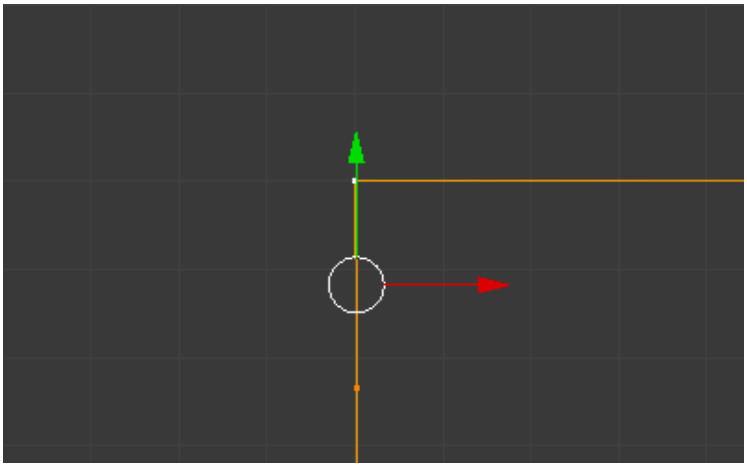
Перейдём на вид сверху, выделим оставшуюся вершину, экструдируем её в нужном нам направлении по оси X или Y. Для этого нажмём лат. «E» и за ней лат. «X» или «Y». Ведём мышью до нужного нам расстояния и закрепляем левой кнопкой мыши:



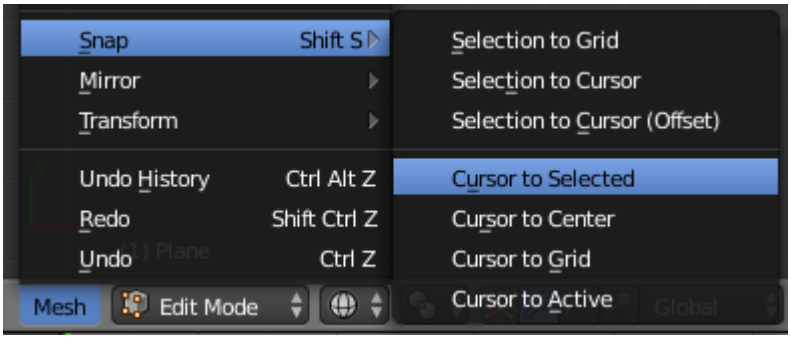
Снова жмём «E», затем «X» или «Y» и тянем в другом направлении. Так поступаем столько раз, сколько нам нужно:



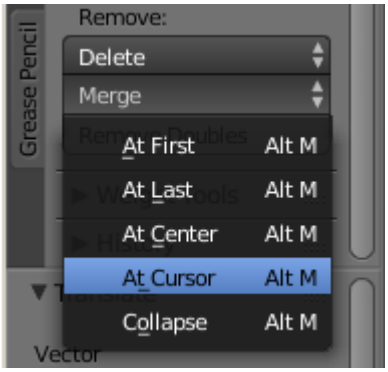
Если нам нужен замкнутый контур, то последнюю точку подводим как можно ближе к первой. Теперь выделяем обе и жмём «F». Между двумя точками появится дополнительная соединяющая линия:

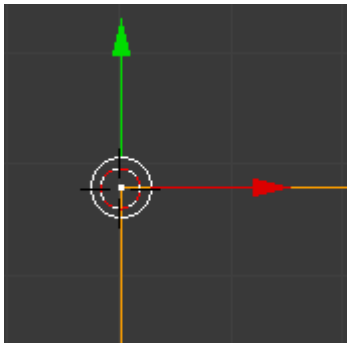


А можно поступить иначе. Выделив две точки, перенесем туда курсор:

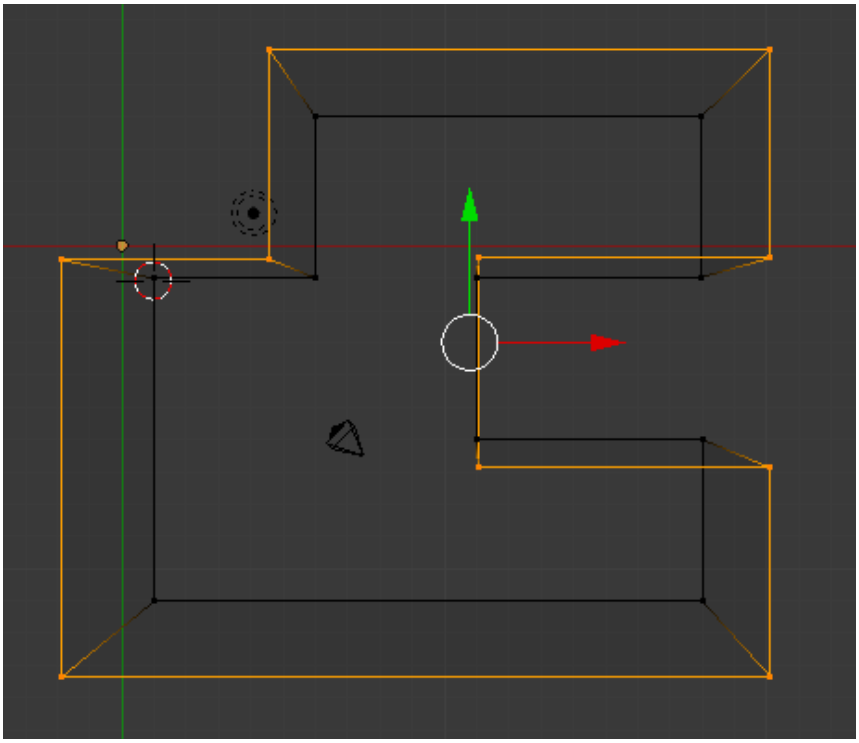


А теперь объединим две точки в одну:



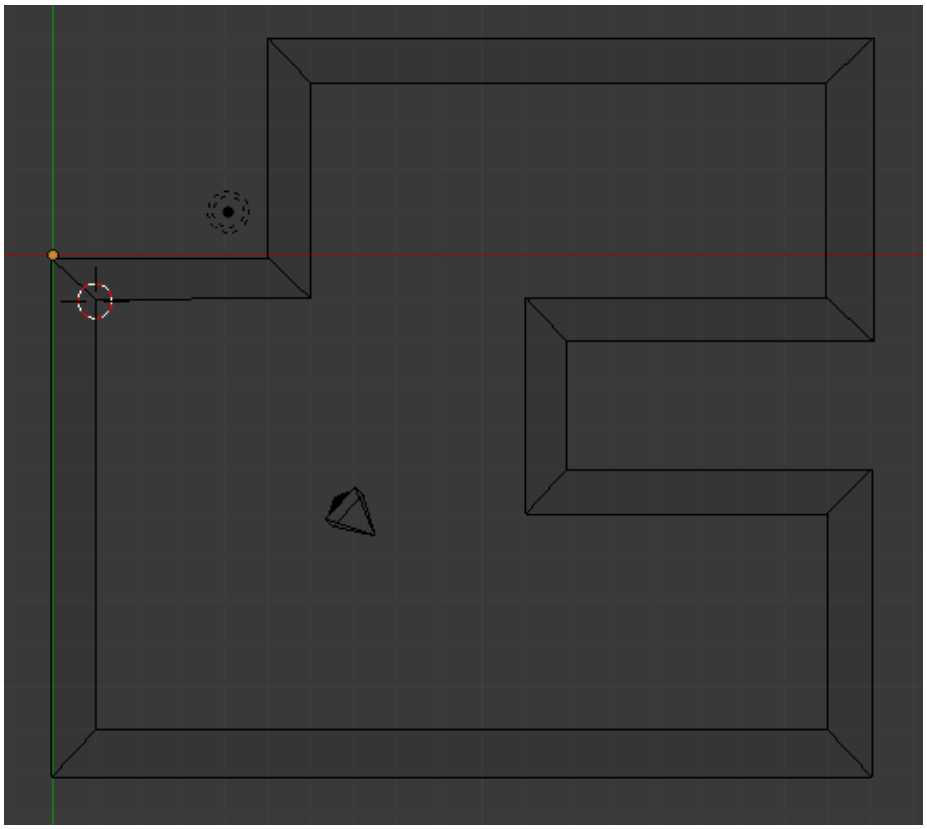


Поскольку я задумал покатые склоны у комнаты, мне нужны дополнительные полигоны по внешнему периметру. Выделяем все точки (лат. «A»), жмём лат. «E» и затем «S». Вытягиваем до нужных размеров (наружу):

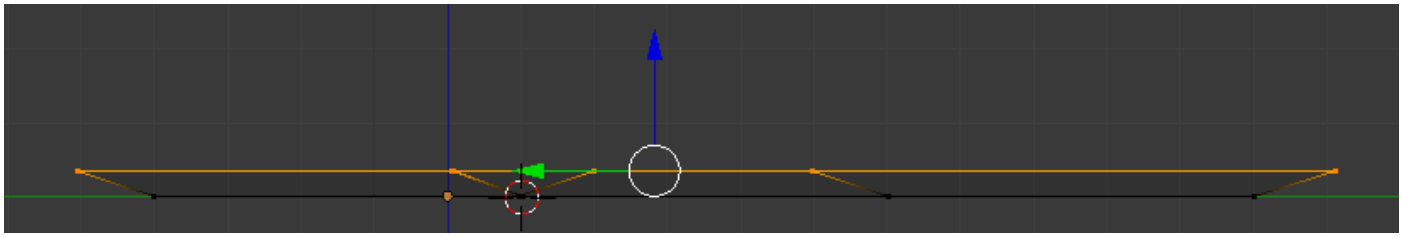


Как видно, некоторые грани выдавились не туда. Это легко исправить. Переходим в режим редактирования граней:

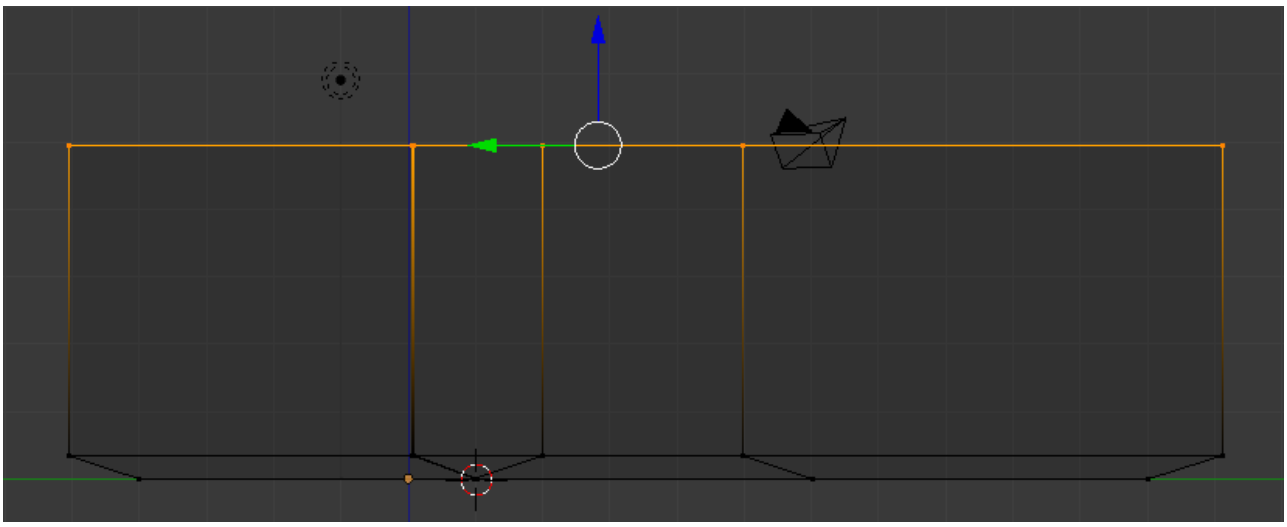
Выделяем нужную грань и тянем зелёной или красной стрелкой в любую сторону:



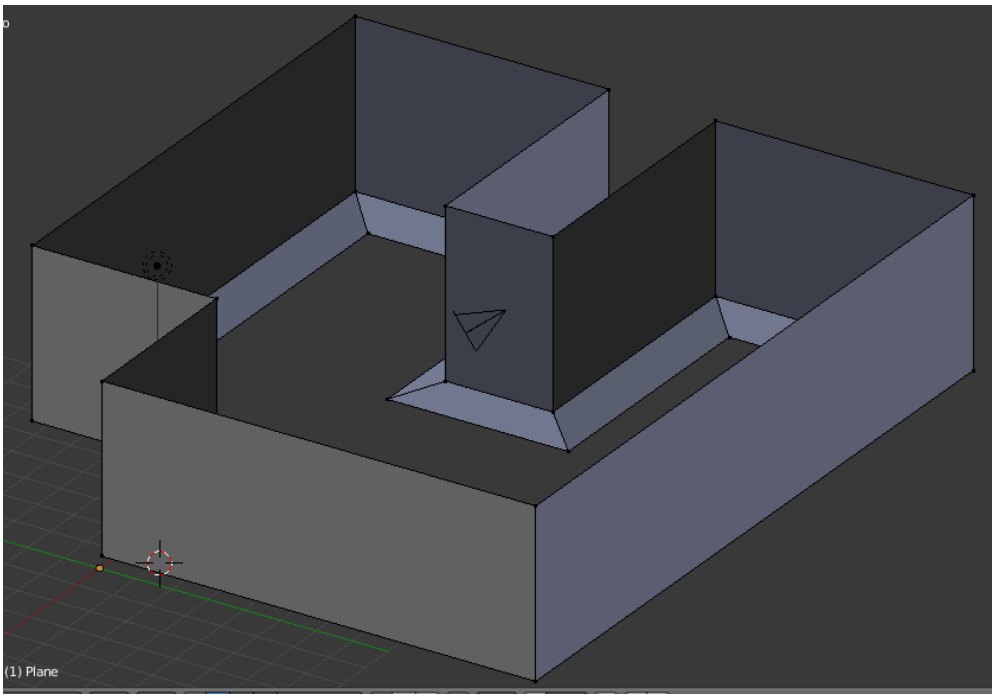
Исправив все грани, снова переходим в режим редактирования точек. Выделяем все точки внешнего периметра и включаем вид сбоку (слева или справа – всё равно). Тянем за синюю стрелку (т.е. по оси Z) столько, сколько нам нужно, чтобы получился уклон:



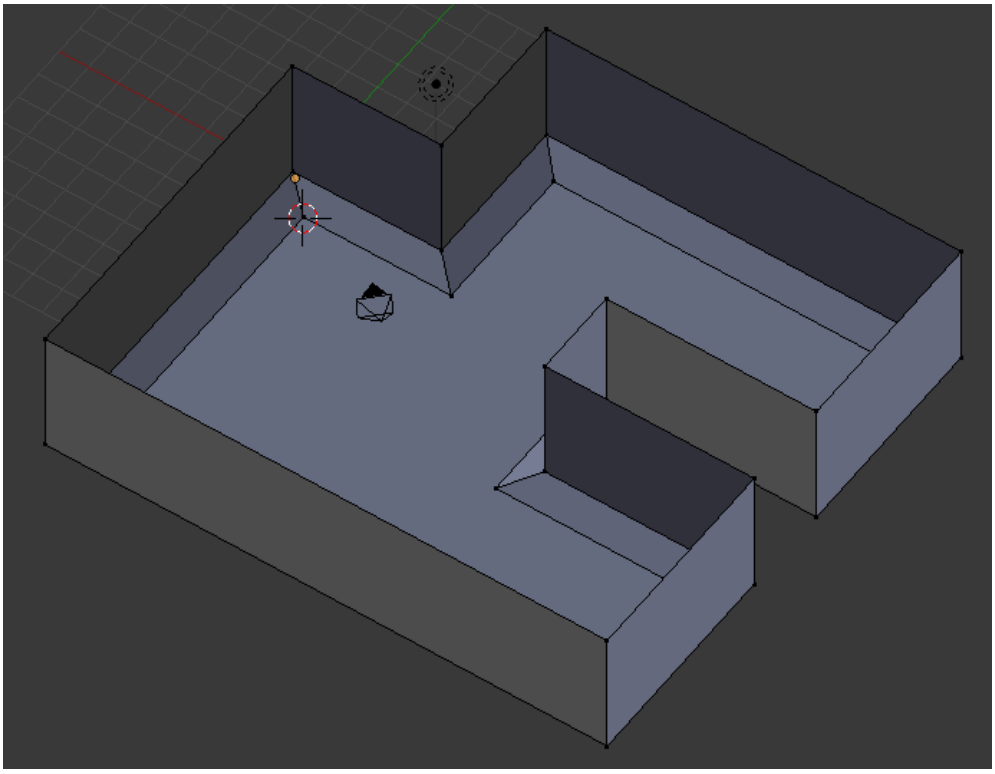
Теперь, не снимая выделения, жмём лат. «E» и затем «Z» и вытягиваем вверх наши стены:



Посмотрим, что у нас получилось, перейдя в режим отображения Solid:

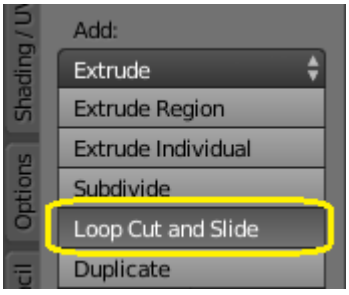


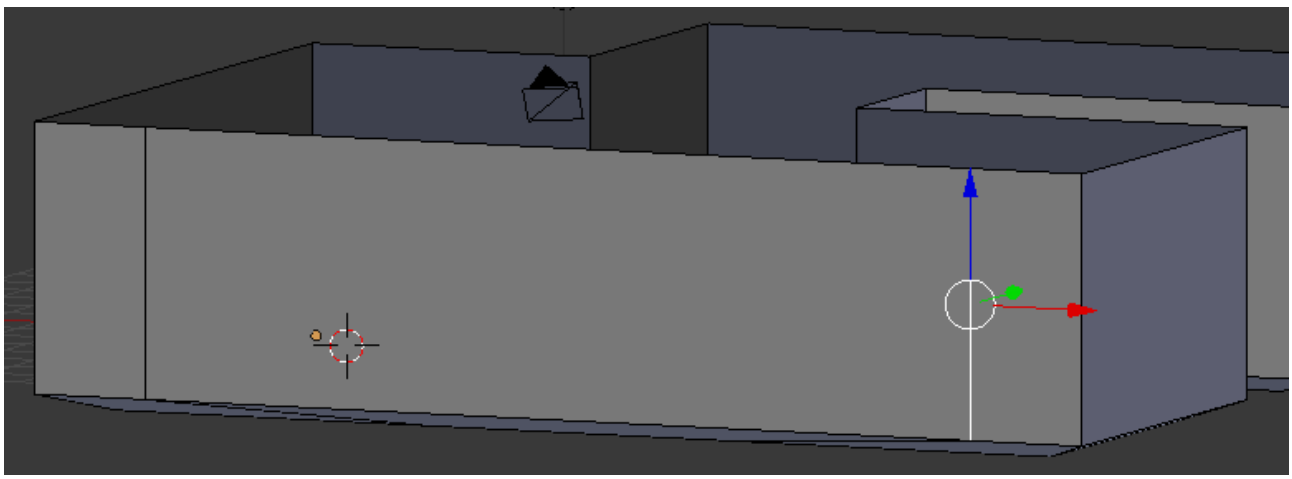
Не трудно догадаться, что полигоны есть везде, кроме пола. Вместо него у нас сквозное отверстие. Для заполнения этой пустоты необходимо выделить все нижние точки периметра пола и нажать « F»:



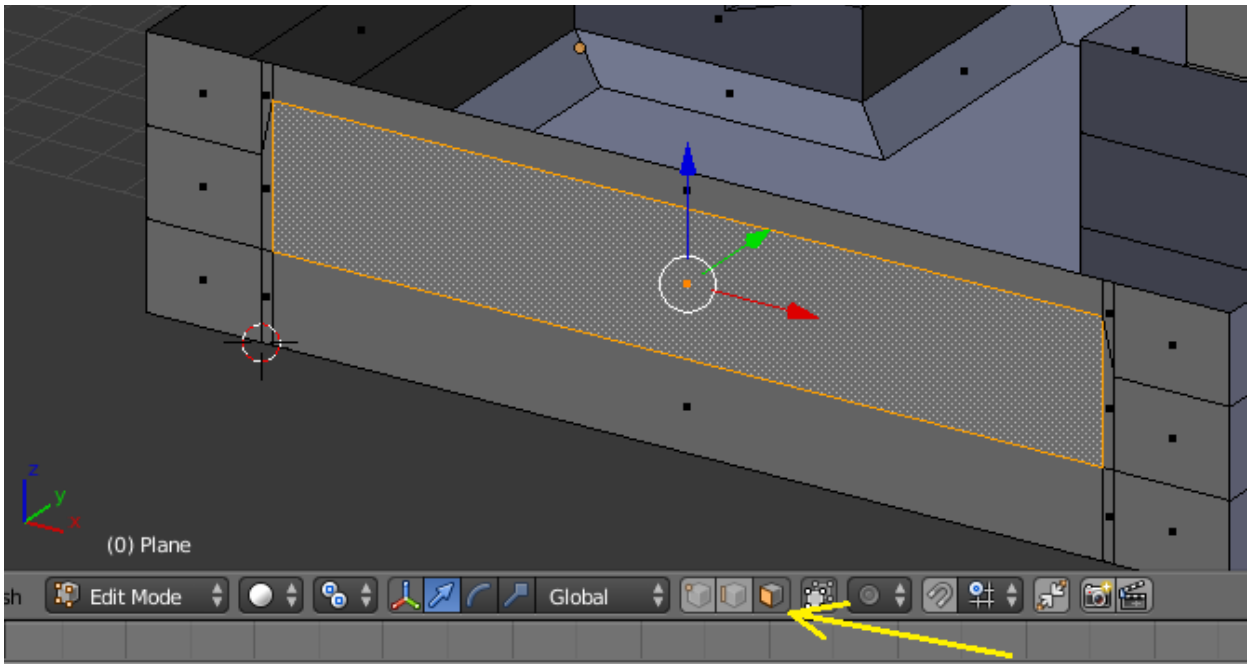
Точно так же можно сделать и потолок. Важно помнить лишь одно: для создания полигона должно быть выделено не менее 4 точек. Если нужно заполнить треугольник, например, то ничего не мешает нам создать дополнительную точку на любой грани ☺.

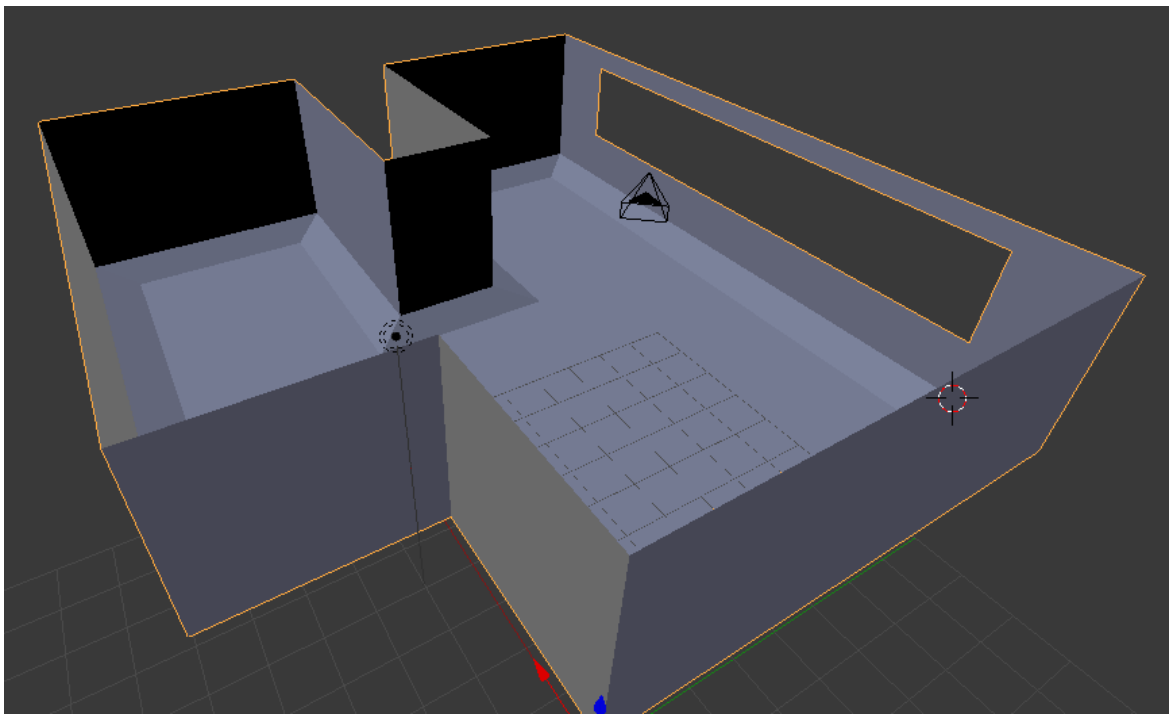
Создадим окно (точно так же можно сделать и дверной проём). Добавим 4 грани на длинную стену и раздвинем их (по очереди) как нам нужно:



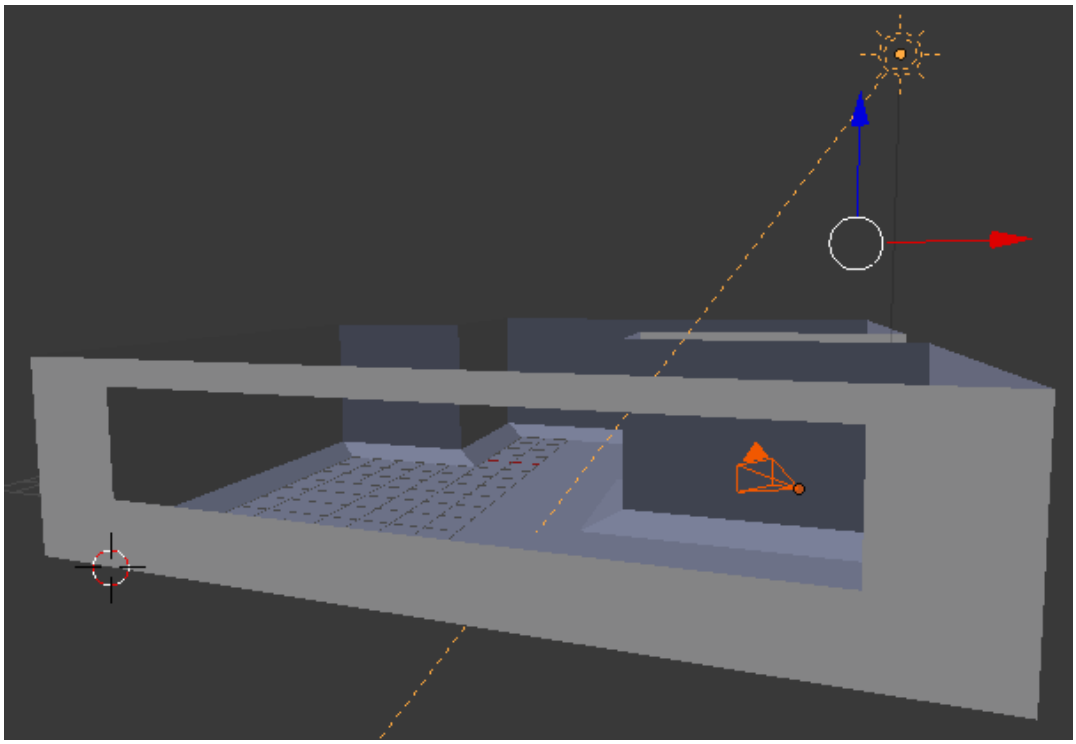


Выделяем внутренний полигон и удаляем («X» -> Faces):



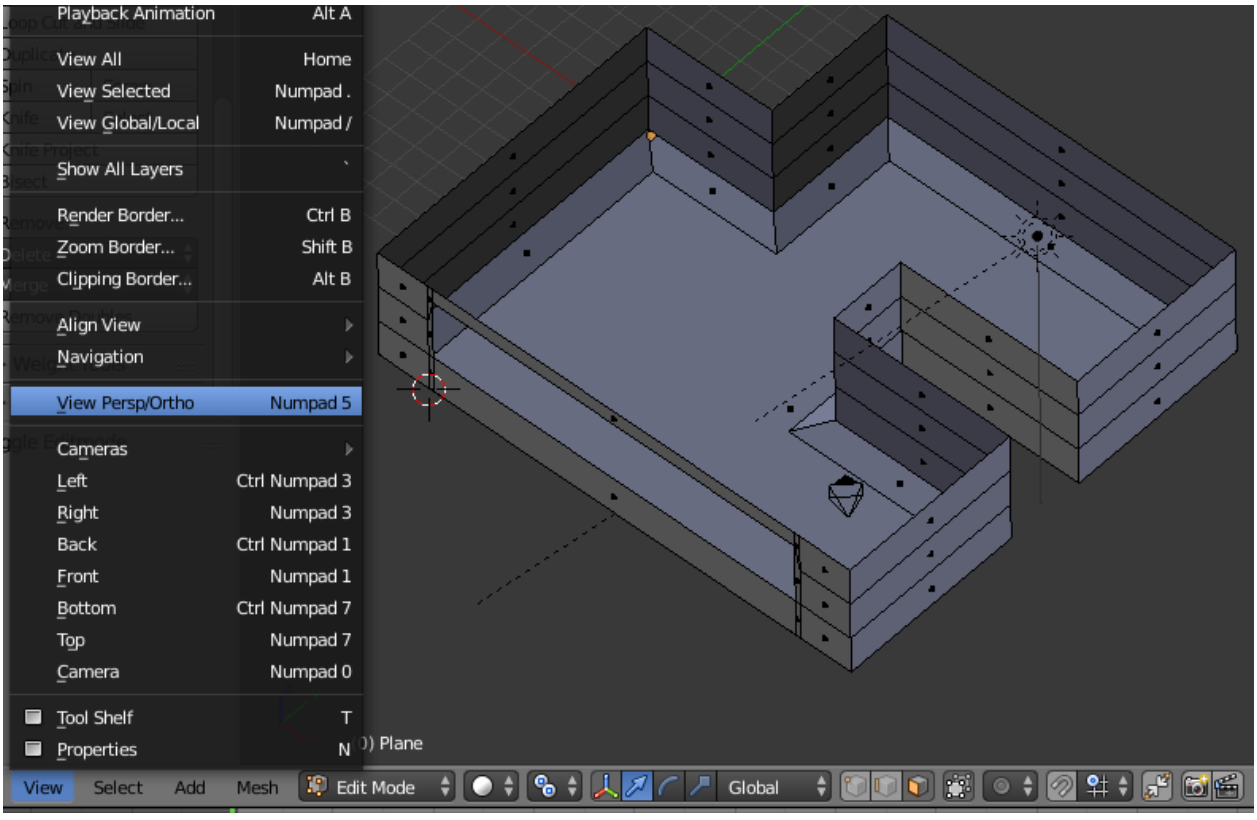


Готово. Поднимем лампу освещения и заменим её на Sun. Сместим камеру и развернём:

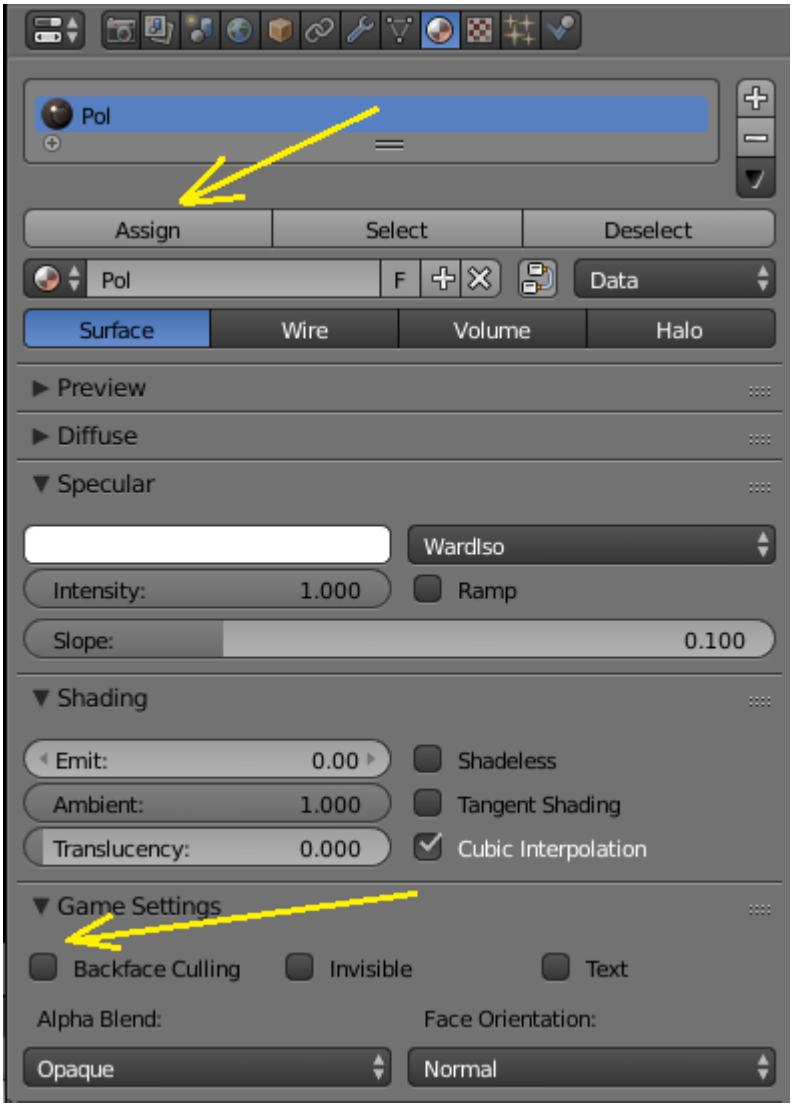


Хотя, если нам не нужно оконное отверстие, то можно было просто этот полигон покрыть прозрачной текстурой. Ну, да ладно, пусть будет так ☺.

Займёмся текстурами. У нас есть пол, склон и стены. Это три текстуры и соответственно три материала. Мы с вами помним (из первых уроков), что одному объекту вполне можно присвоить несколько материалов. Так и поступаем. Выделяем комнату и переходим в режим редактирования, предварительно отключив перспективу (с ней неудобно):



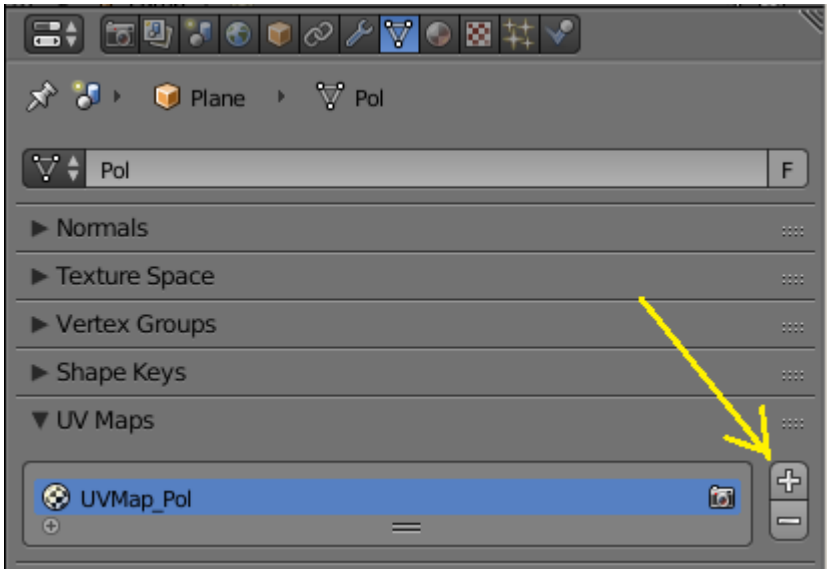
Выделяем полигон пола и назначаем ему материал:



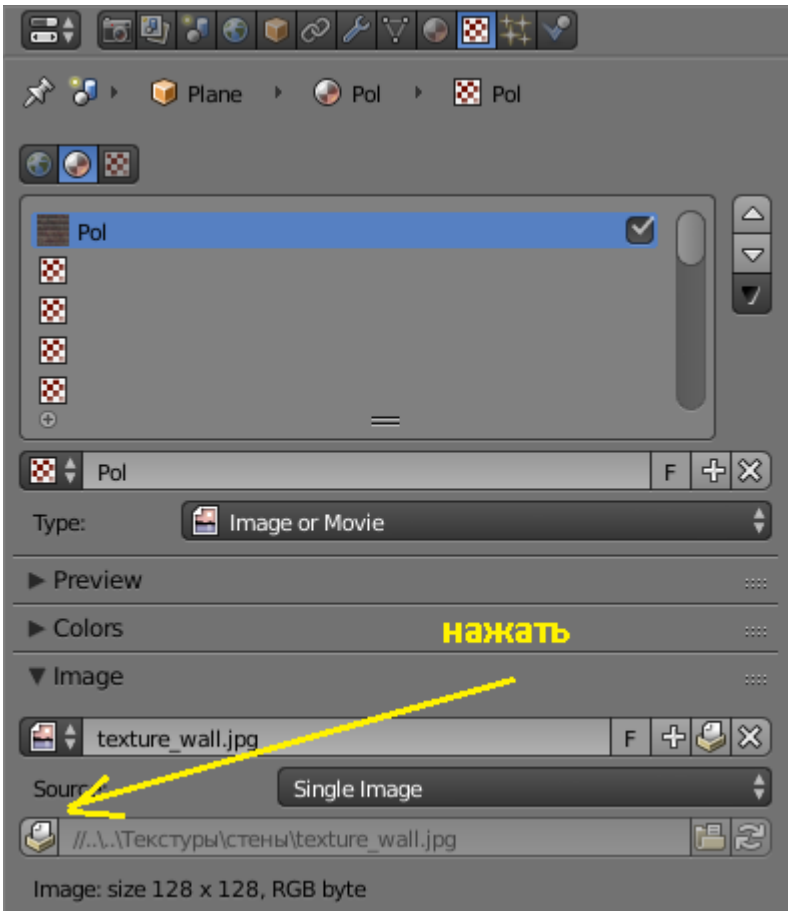
Обязательно снять галочку Backface Culling. Для непрозрачного материала мне понравились эти настройки. И жмём кнопку Assign, чтобы применить его только к полу.

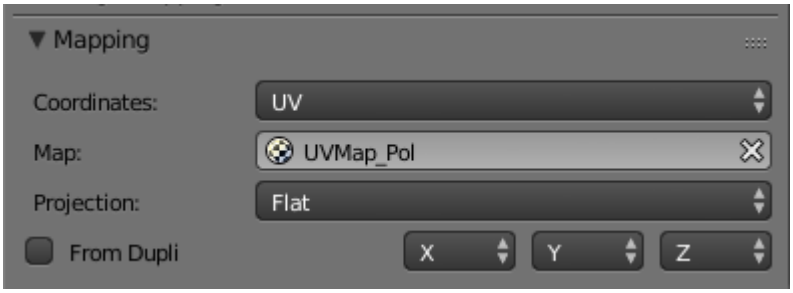
Важно! Всё это делается в режиме Blender Game и без GLSL !

В данных объекта создаём UVMap_Pol (плюсиком справа):

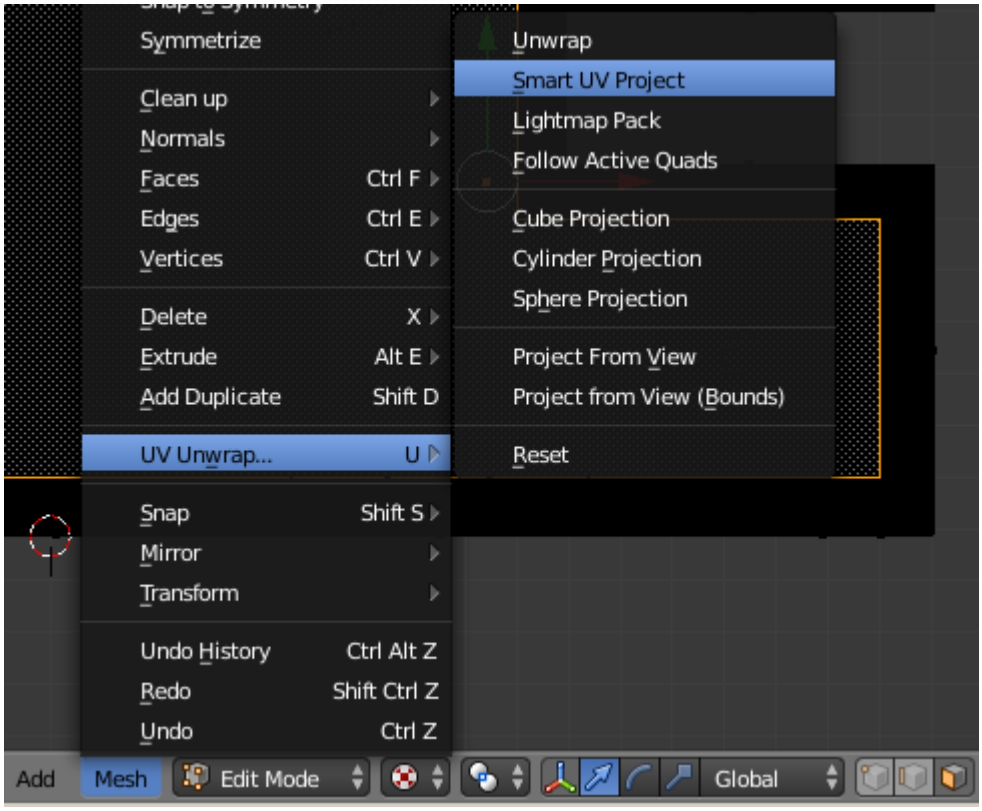


Создаём текстуру и загружаем картинку:

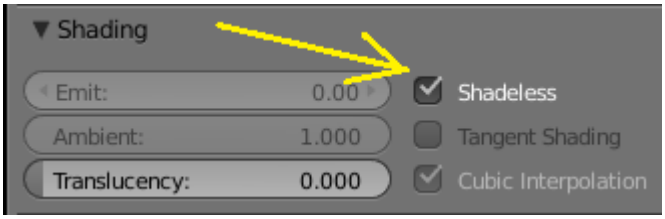




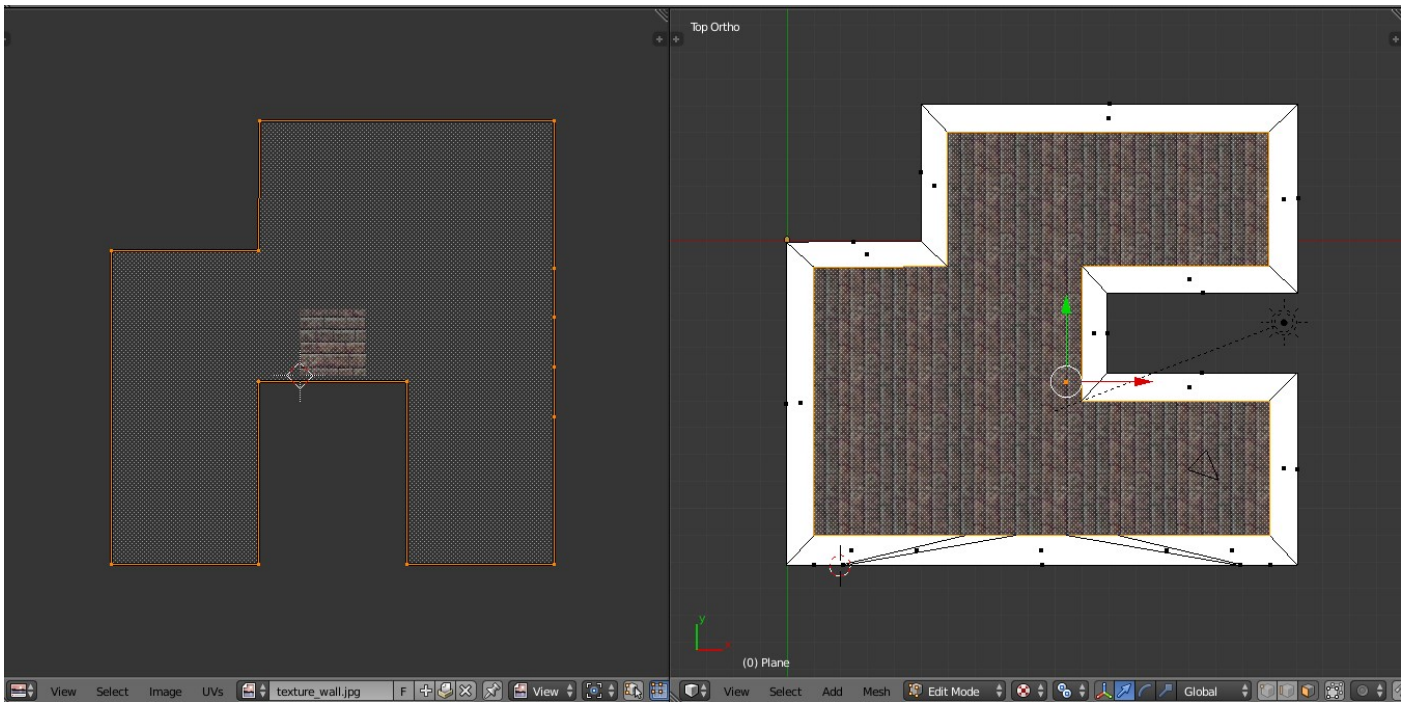
В режиме редактирования и при выделенном полигоне пола (а так же при виде сверху и включённом отображении текстур) переходим в UV редактор. Там создаём развёртку пола:



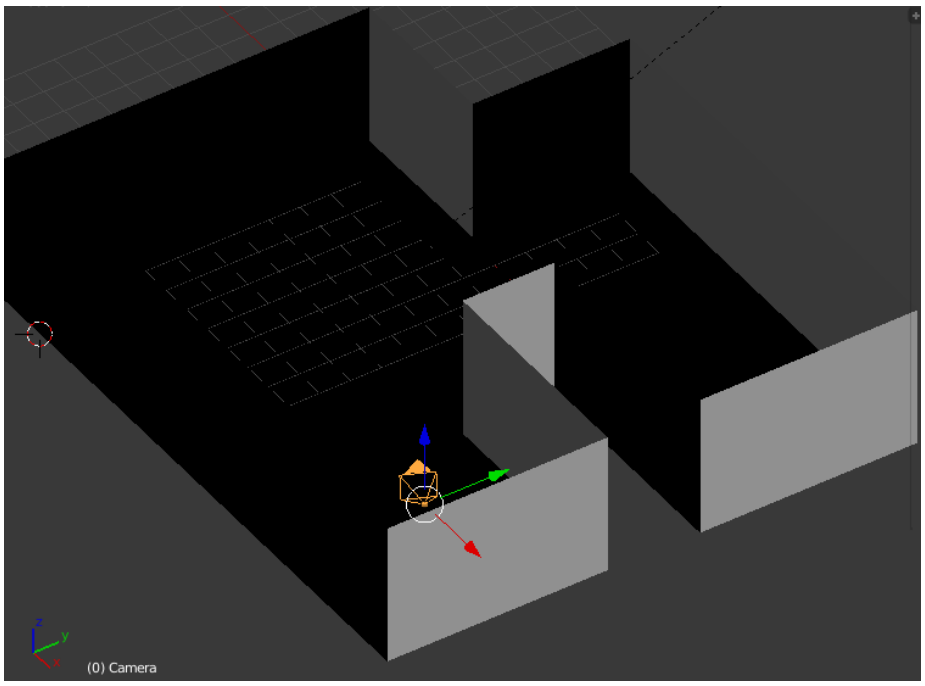
Если текстура (в правом окне) не видна, то вернитесь в материал и временно (!) включите:

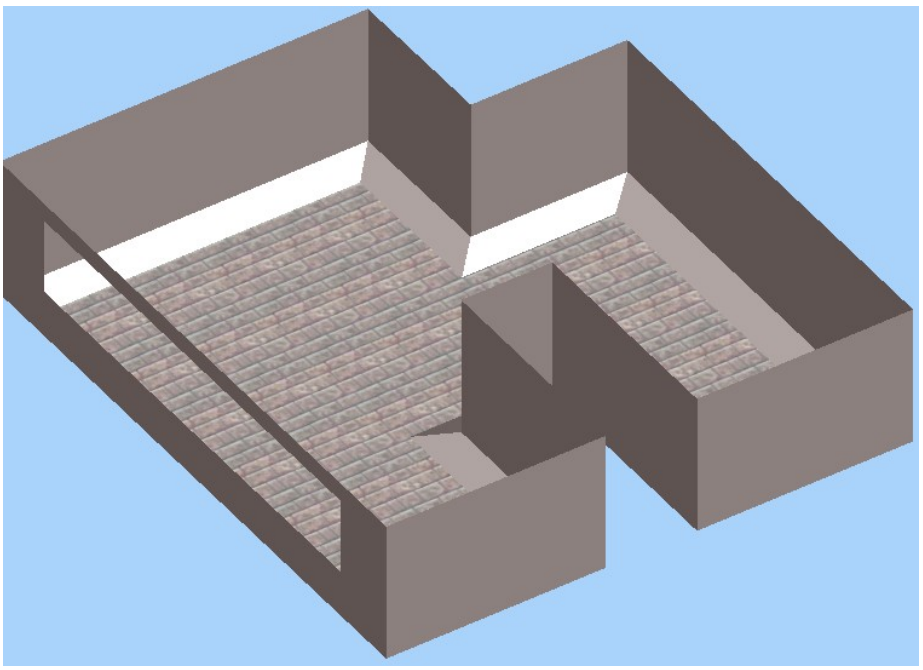


Не забудьте потом выключить! Иначе не будет затенения на стенах и гранях. Тогда пропадает смысл в источнике света ☺. Если размер рисунка текстуры справа велик (гигантские плитки!), то выделяем слева грани и масштабируем («S»):

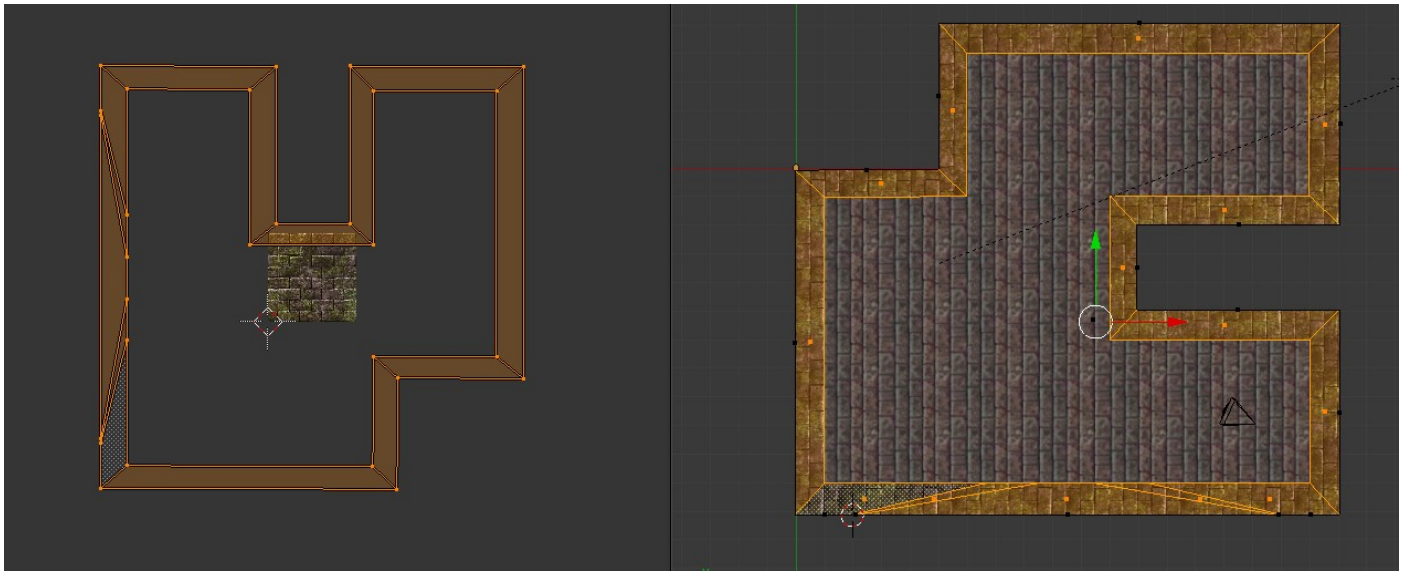


Если после всего этого у вас вот такой вид не страшно. Если вы нажмёте «P» (выйдя из режима редактирования), то увидите нормальную картину:

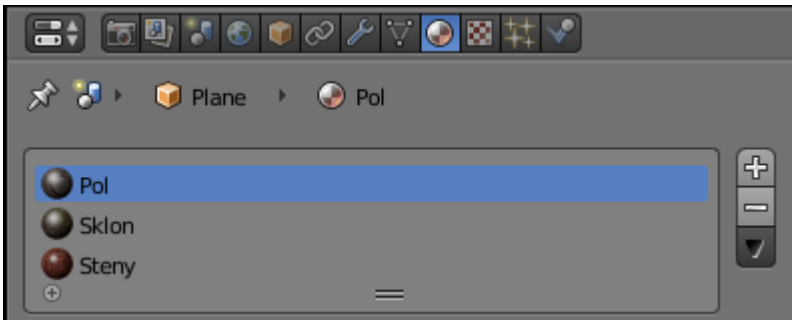




Теперь в режиме редактирования выделяем все полигоны склонов. Создаём новый материал. Применяем его (Assign) и т.д. Делаем всё так же, как и с полом, только выбираем другую картинку:

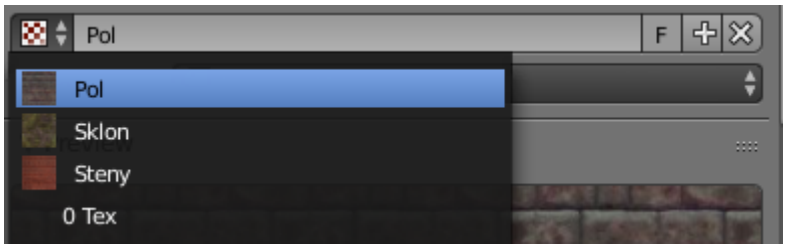


Точно так же текстурируем стены. В общей сложности у нас должно быть (пока) три материала:

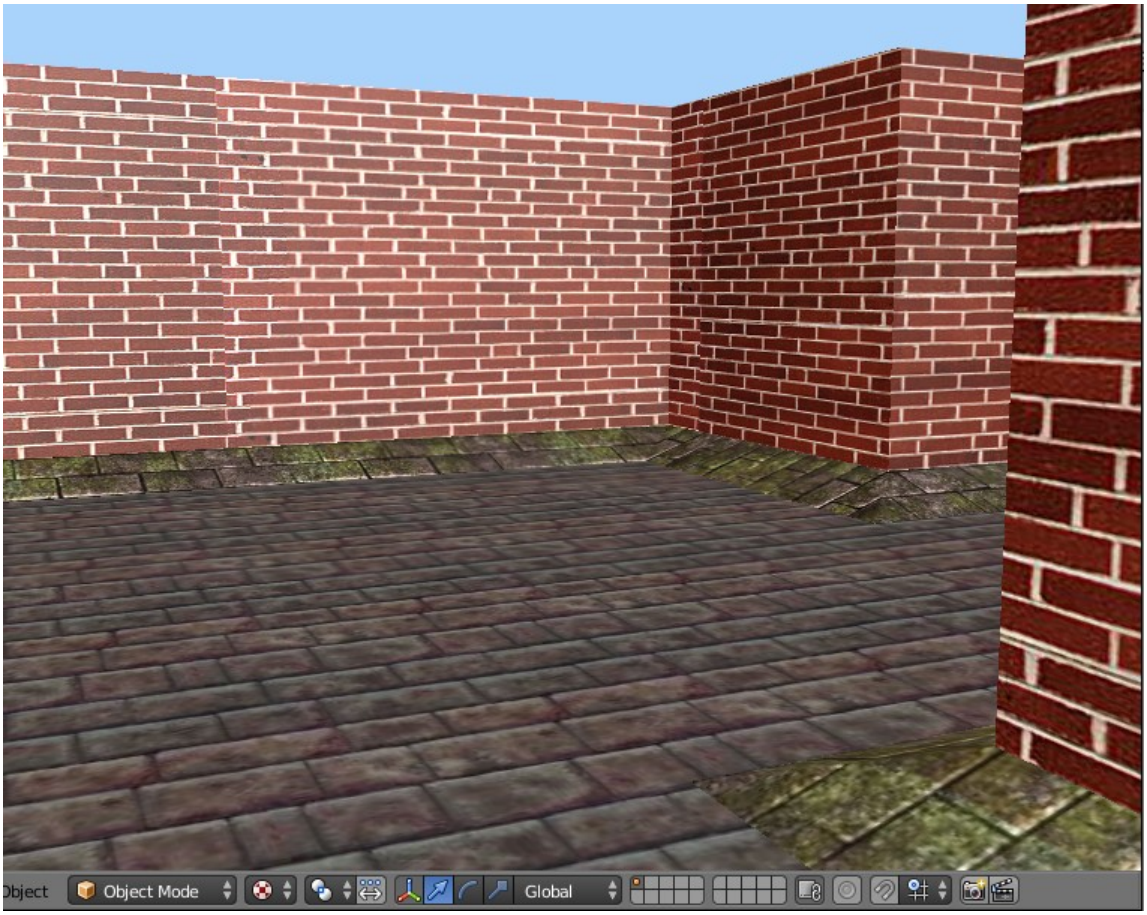


Три UVMap карты:

И три текстуры:

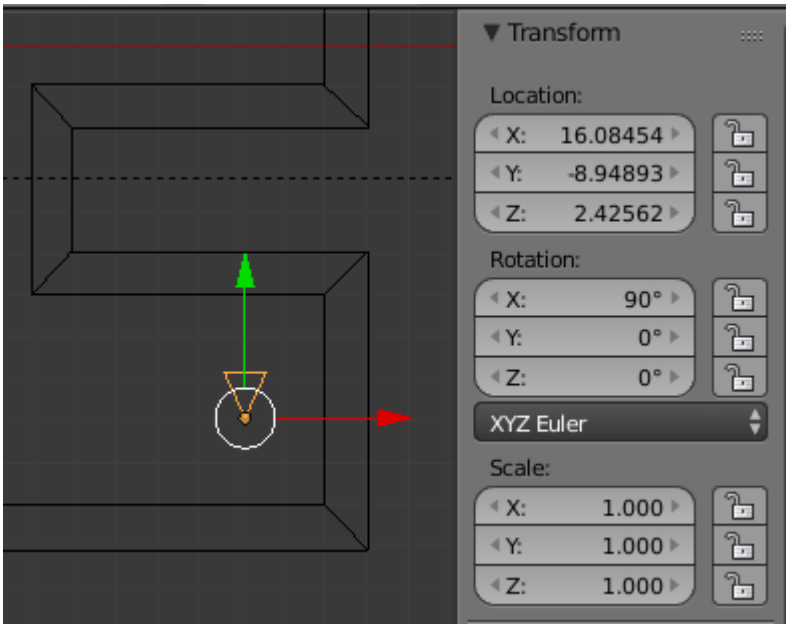


На виде с камеры всё будет выглядеть примерно так:

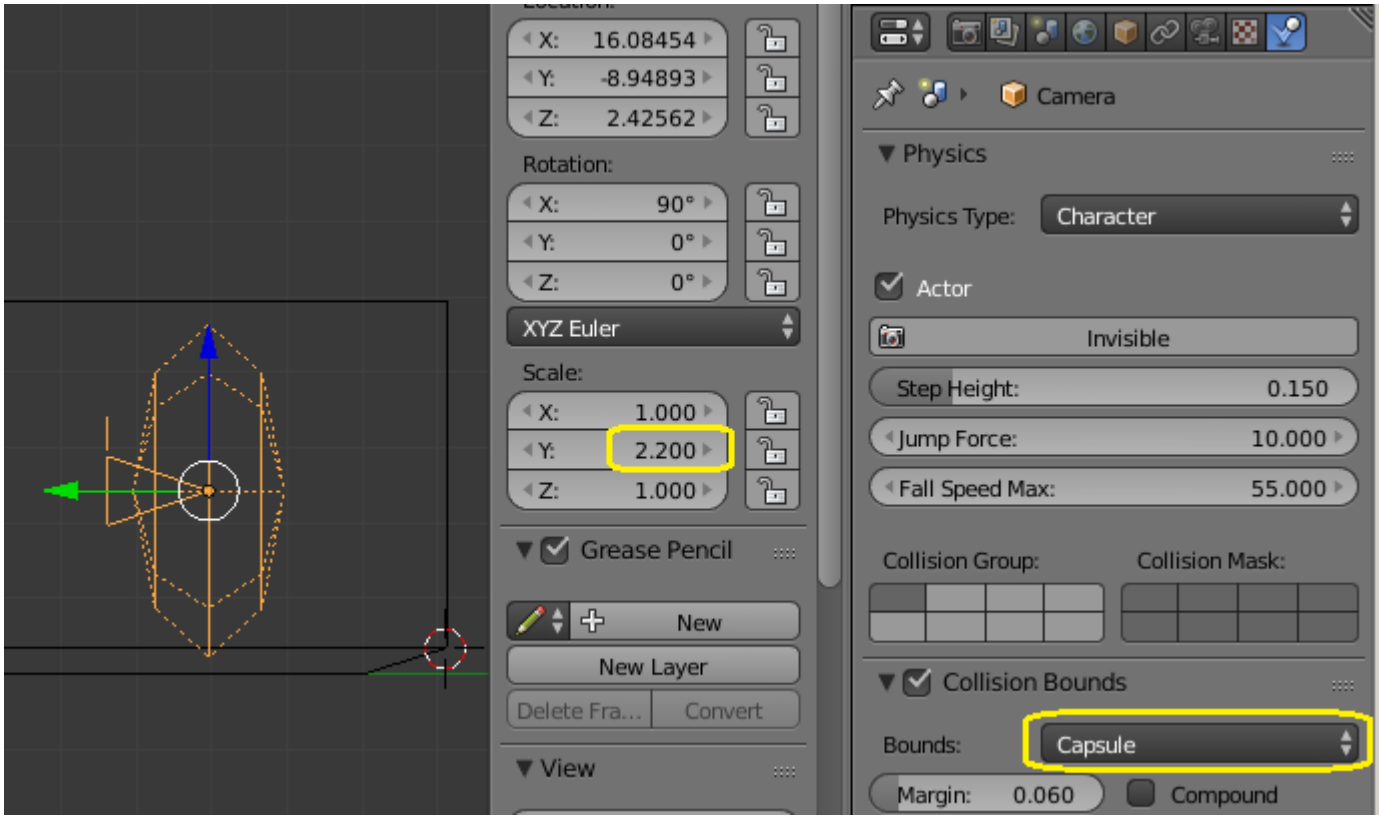


Важно! В следующих уроках я больше не буду подробно объяснять моделирование и текстурирование. За исключением чего-то нового в этих вопросах. Я буду подразумевать, что вы в режиме Blender Game и знаете, как создать модель и покрыть её текстурой, как поставить свет и т.д..

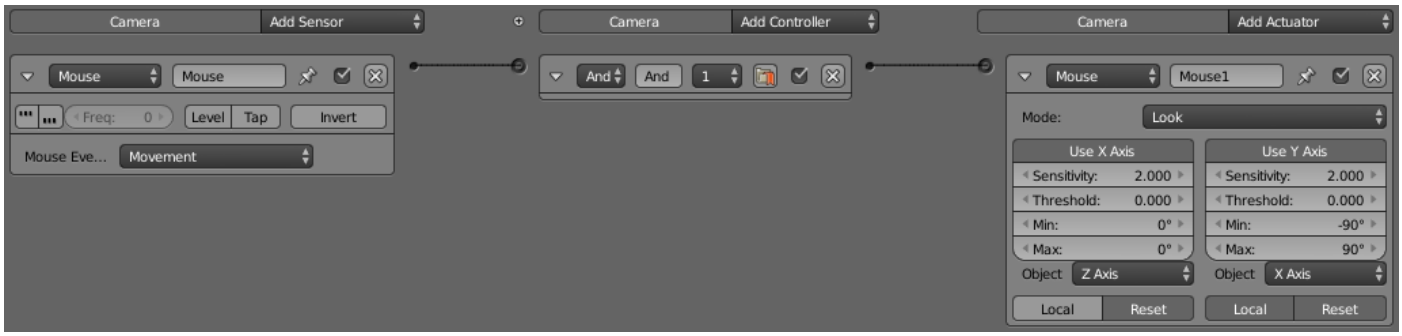
Наконец мы из камеры сделаем игрока. Оказывается, что можно камеру и не крепить к физическому объекту типа сферы. Она сама вполне может быть актёром. В Blender 2.72 (на котором я пишу все уроки) есть для этого всё необходимое. Обнулим у камеры все значения Rotation и повернём её по X на 90 градусов:



Пусть камера смотрит в сторону оси Y. Теперь зайдём в настройку её физических свойств:



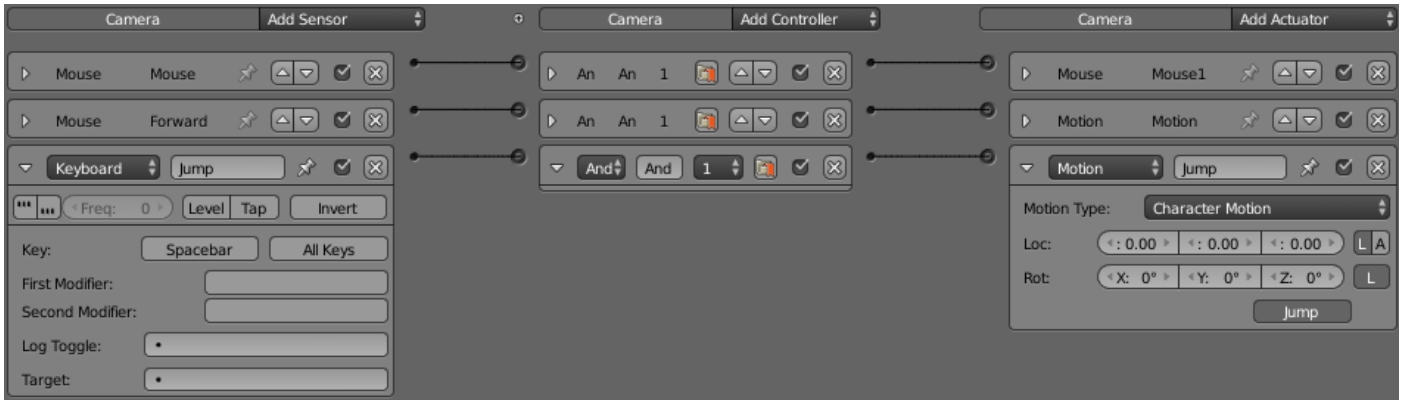
Как видите, мы выбрали тип столкновения Capsule, поскольку он более правильно взаимодействует с поверхностью. А ещё, мы растянули капсулу по вертикали. Почему по оси Y? Потому, что мы повернули камеру, и вертикальной оказалась ось Y. Далее заходим в логику и без всяких скриптов на Python создаём управление мышью:



Элементарно берём сенсор Mouse (Movement) и соединяем его с актуатором Mouse (Lock). Проверьте сами ☺. При движении мышью поворачивается камера (вращается в разные стороны). Добавим движение вперёд по нажатию правой кнопки мыши:

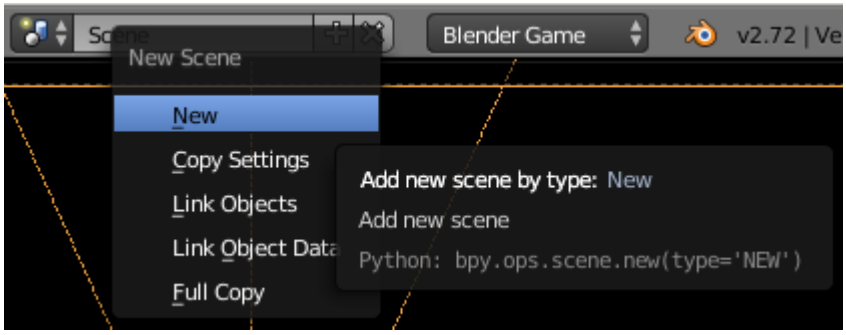
Сенсор Mouse (Right Button) соединяем с актуатором Motion (Character Motion), в котором делаем движение по оси Z потому, что у нас повёрнута камера. Осталось добавить прыжки:

aracte

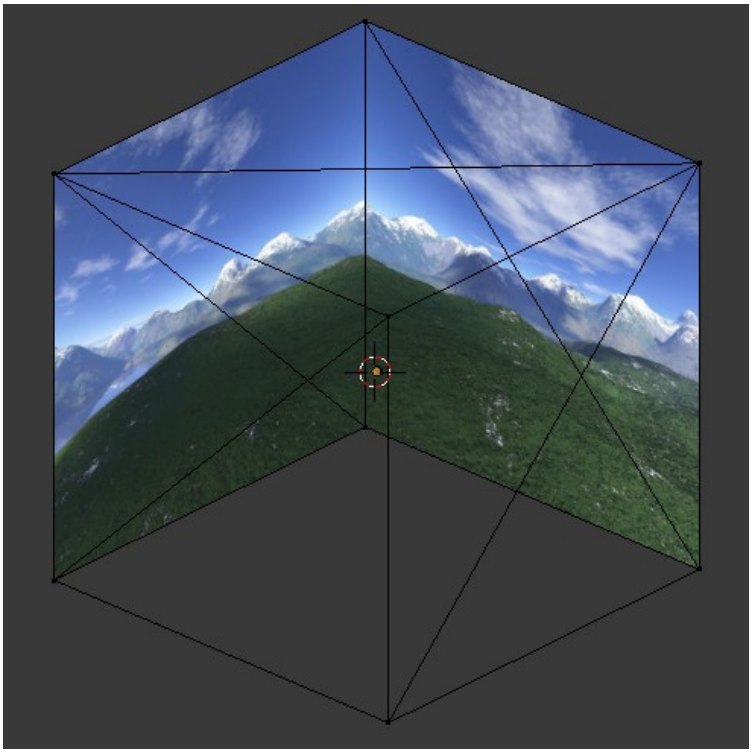


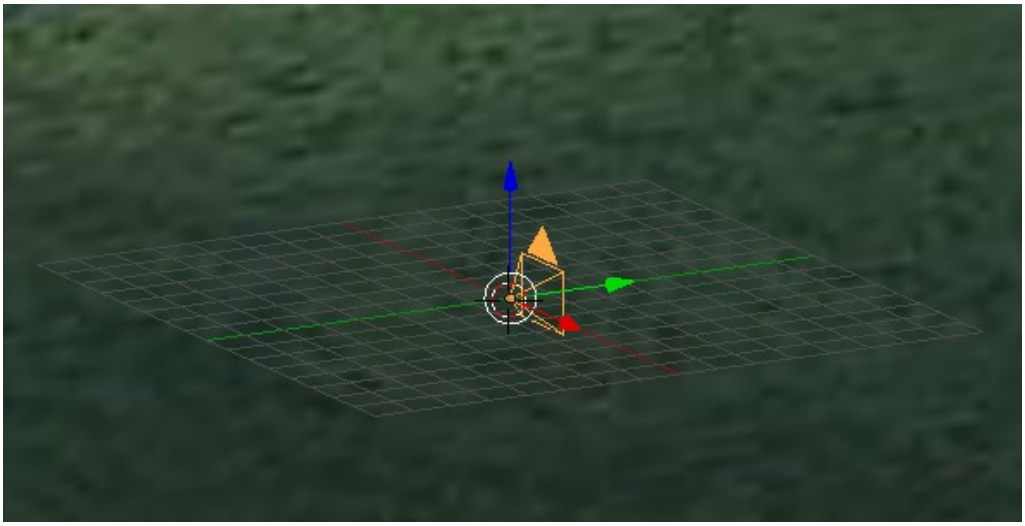
Здесь сенсор Keyboard подключаем к актуатору Motion (Character Motion), у которого просто нажата кнопка Jump. Если после проверки выяснится, что актёр прыгает слишком низко, то зайдите в физику камеры и измените параметр Jump Force на 20 .

Замечательно, теперь мы свободно перемещаемся по комнате. Хотелось бы сделать окружение в виде неба с облаками. Нет ничего проще! Скачиваем любой Skybox на странице «Полезное». Создаём ещё одну сцену (плюсик) и называем её «Sky»:

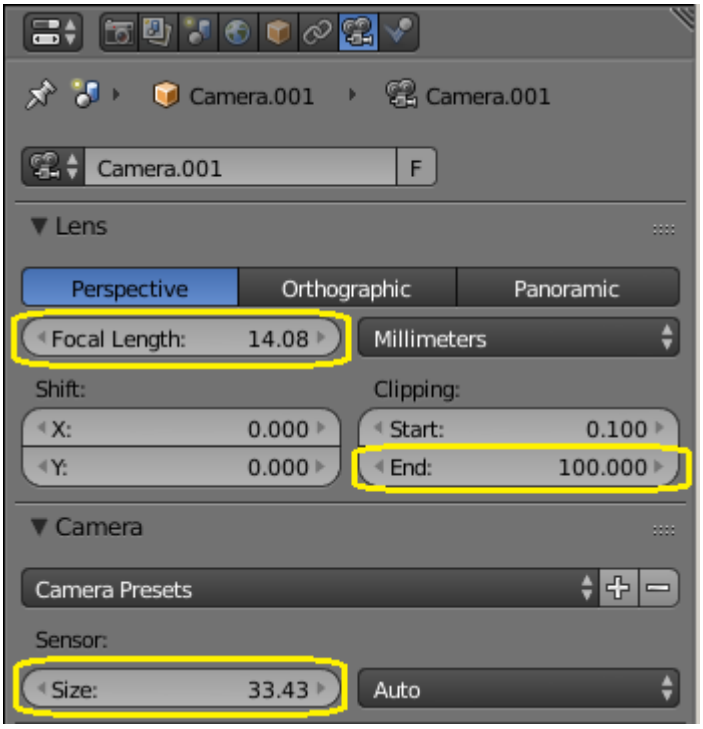


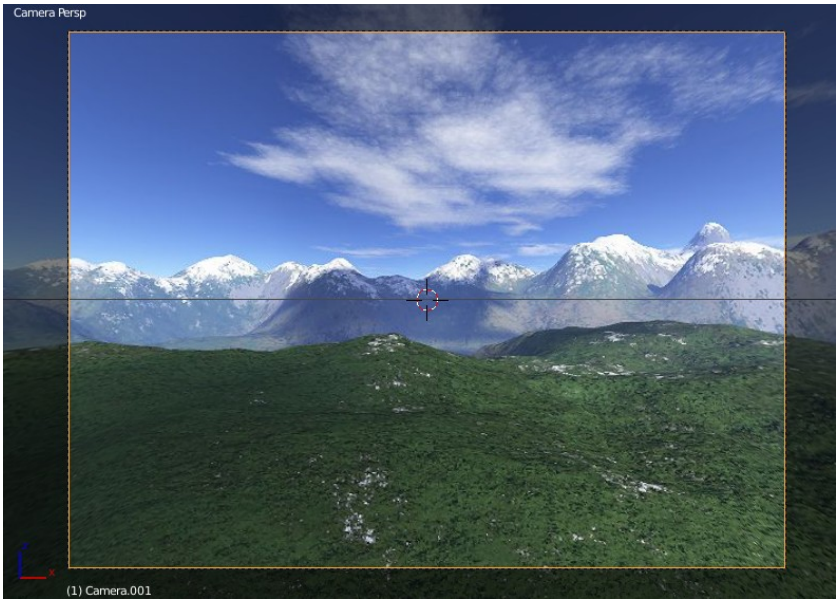
Поскольку скачанный Skybox в формате 3DS, импортируем его в новую сцену как 3DS. Немного увеличим. Установим камеру в центре и повернём её по X на 90 градусов:



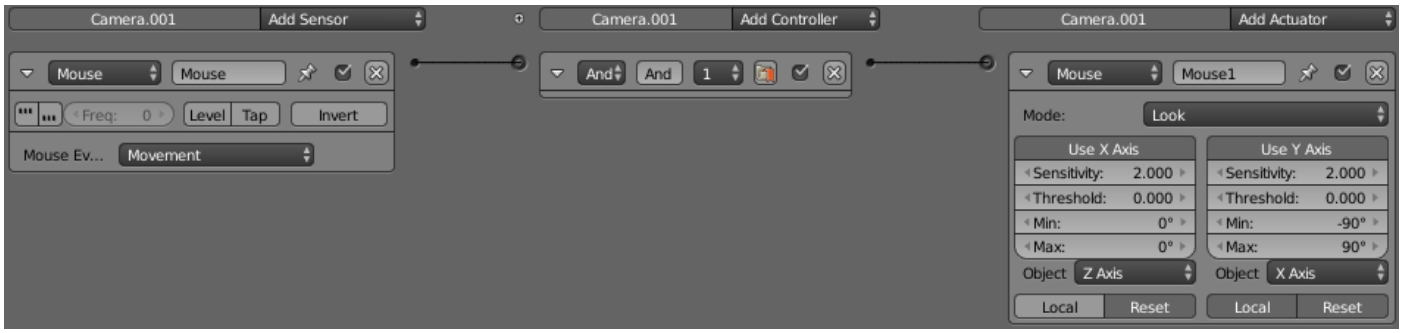


Чтобы улучшить отображение картинки неба на виде с камеры может понадобиться изменить настройки камеры:

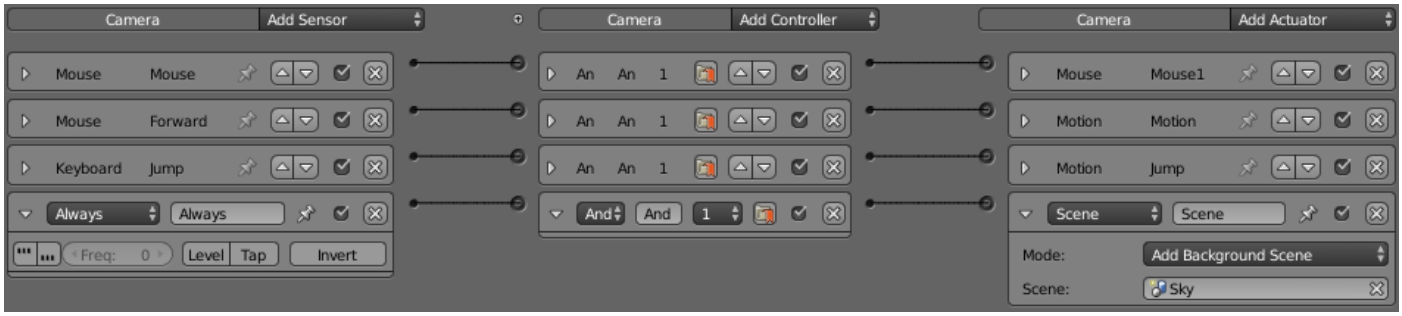




Переходим в логику и создаём управление мышью, аналогичное предыдущей сцене:

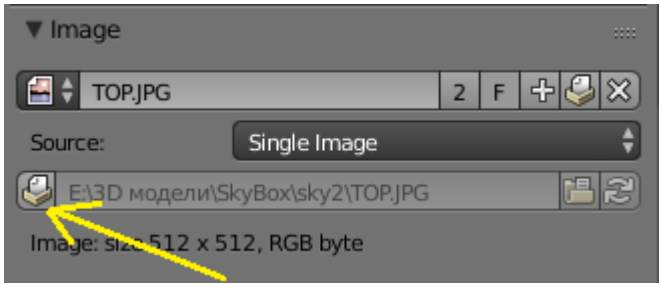


Возвращаемся в первую сцену и в логике создаём связь двух сцен:



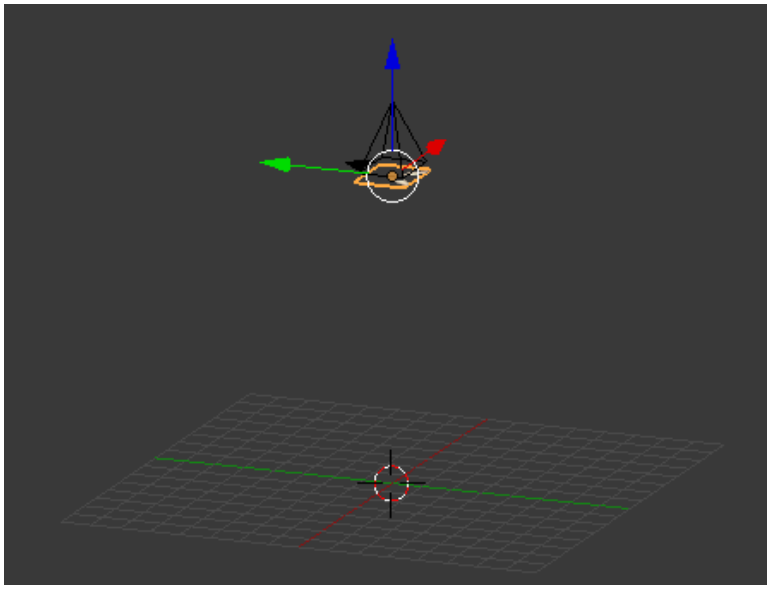
Т.е. добавляем сенсор Always и связываем его с актуатором Scene (Add Background Scene), где в поле Scene выбираем сцену с небом (Sky). Можно было не делать управление мышью в сцене с небом. Тогда при повороте камеры небо поворачивалось бы всегда одной и той же стороной.

И конечно, нужно не забыть запаковать текстуры. Иначе наше небо могут и не увидеть пользователи на других компьютерах ☺. Перейдём в сцену Sky. Откроем материалы и в каждом (основных должно быть 5) в Shading поставим галочку Shadeless. Затем откроем текстуры и в каждой нажмём кнопку запаковки:

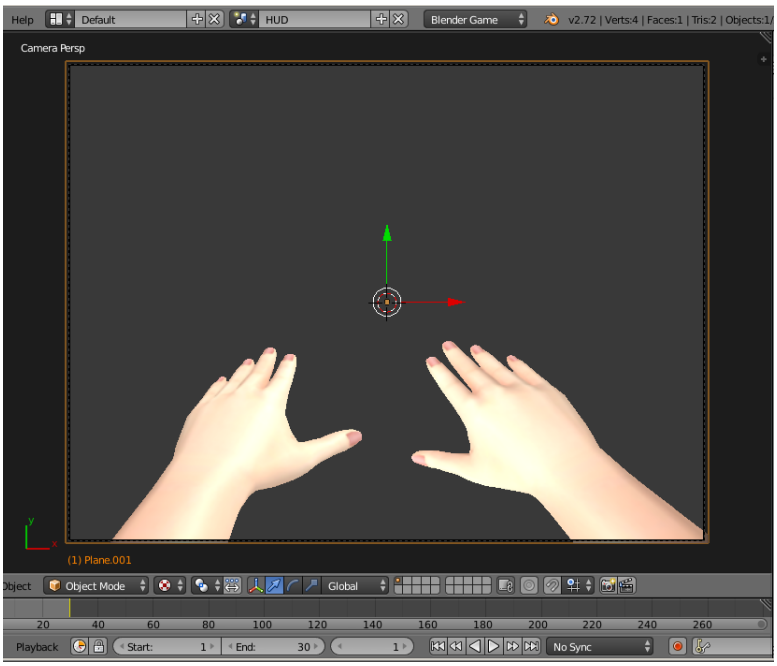


Настало время рассмотреть HUD. Что это такое? Прицел, который перемещается вместе с камерой; счёт на экране – всё, что перемещается вместе с камерой можно отнести к HUD. В прошлых уроках мы уже делали подобие HUD, прикрепляя нужные нам объекты к камере по принципу потомок-родитель. Однако в одном случае это может быть неэффективно, а в другом может некорректно работать вообще. Разработчики позаботились об этом, предложив нам в актуаторе Scene выбор режима Add Overlay Scene. Работает всё по такому же принципу, что и Add Background. Создаётся новая сцена, совершенно статичная, в которой есть камера и всё что нам необходимо. При этом ничего никуда прикреплять не нужно. При подключении этой сцены к основной всё будет отображаться на экране, перемещаясь вместе с камерой. Интересен тот факт, что больше не нужно беспокоиться о качестве шрифта на экране. Ведь в сцене с HUD мы спокойно можем его увеличить и отодвинуть на любое расстояние (для качества). В основной сцене текст не будет прятаться за предметы, с которыми мы сталкиваемся – он будет выглядеть, как положено. Но главное то, что две сцены могут обмениваться данными без глобального словаря – напрямую. Так, будто это всё происходит в одной сцене! Это позволяет подсчитывать предметы, патроны, жизни в режиме реального времени без тормозов.

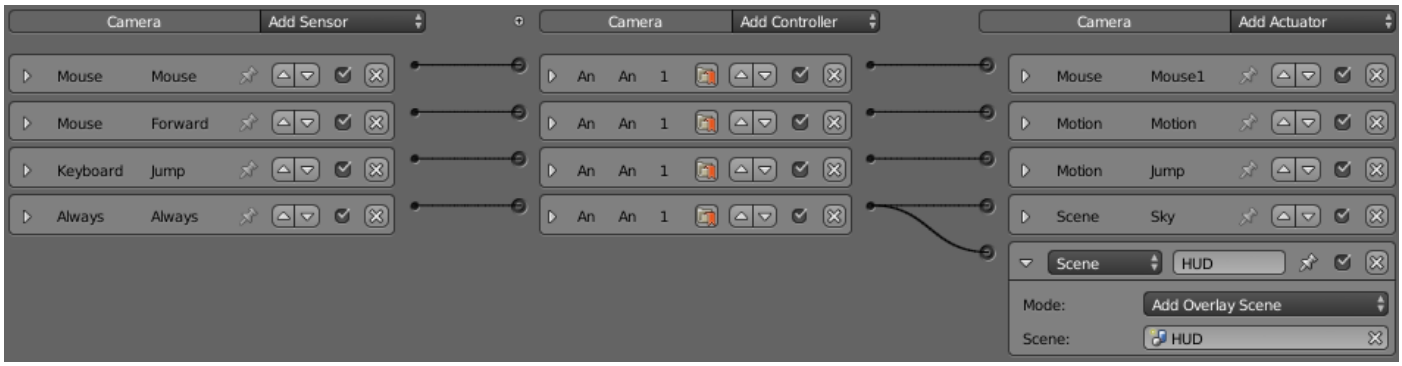
В предыдущем уроке был небольшой баг – руки периодический дёргались. Это как раз из-за подсоединения их к камере. А ведь та, в свою очередь, была соединена с физическим объектом. Данная ошибка в принципе невозможна, если разместить руки в сцене с HUD. Давайте сделаем руки и подсчёт столкновений с предметами как HUD. Создадим новую сцену и назовём её соответственно. Как обычно разместим в центре камеру. При этом поворачивать её уже никуда не надо (всё по нулям), но необходимо поднять на приличную высоту. Если не поднять, то сколько бы мы не отдаляли текстовый объект в результате размер текста будет всегда одинаковым:



Заодно сделаем анимацию рук, как в прошлом уроке (вспомните):

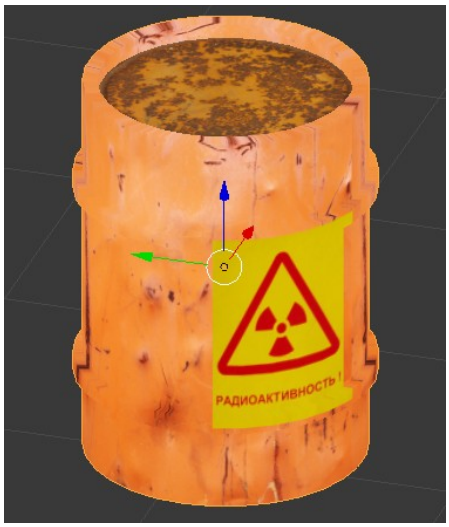


Перейдём в основную сцену и в логике добавим актуатор Scene (Add Overlay Scene), где в поле Scene выберем наш HUD:

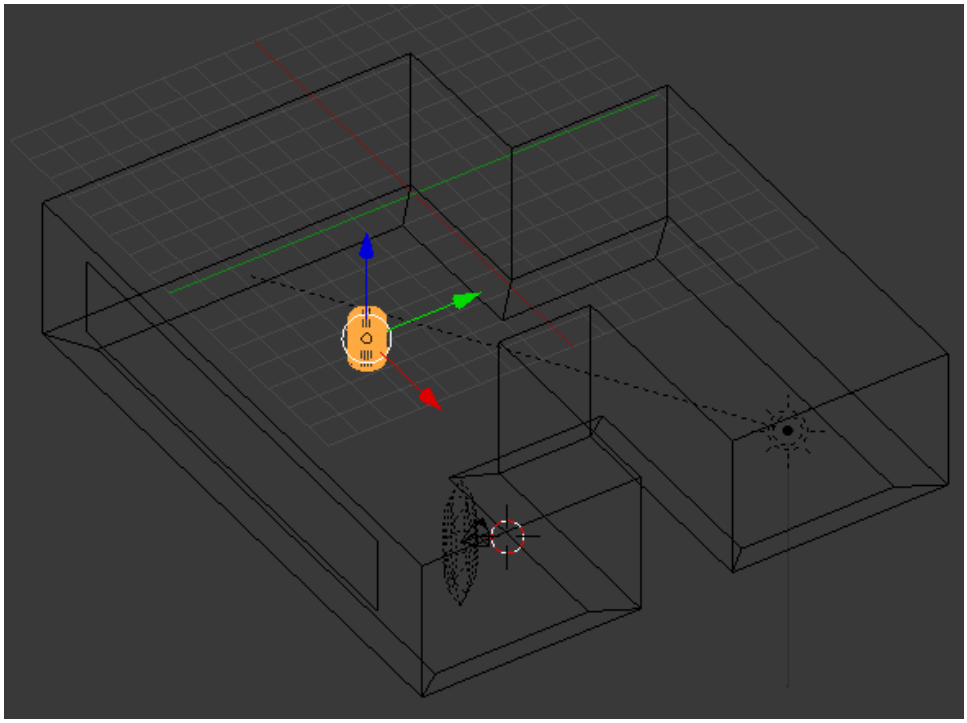


Как видим, всё замечательно:

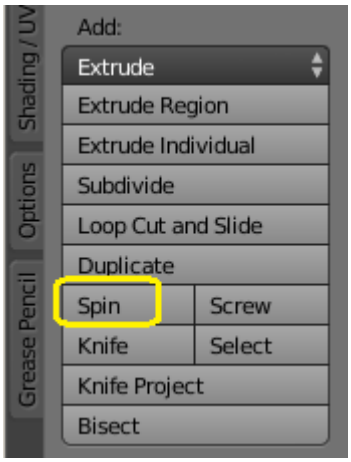
В основной сцене перейдём на новый слой и создадим бочку:



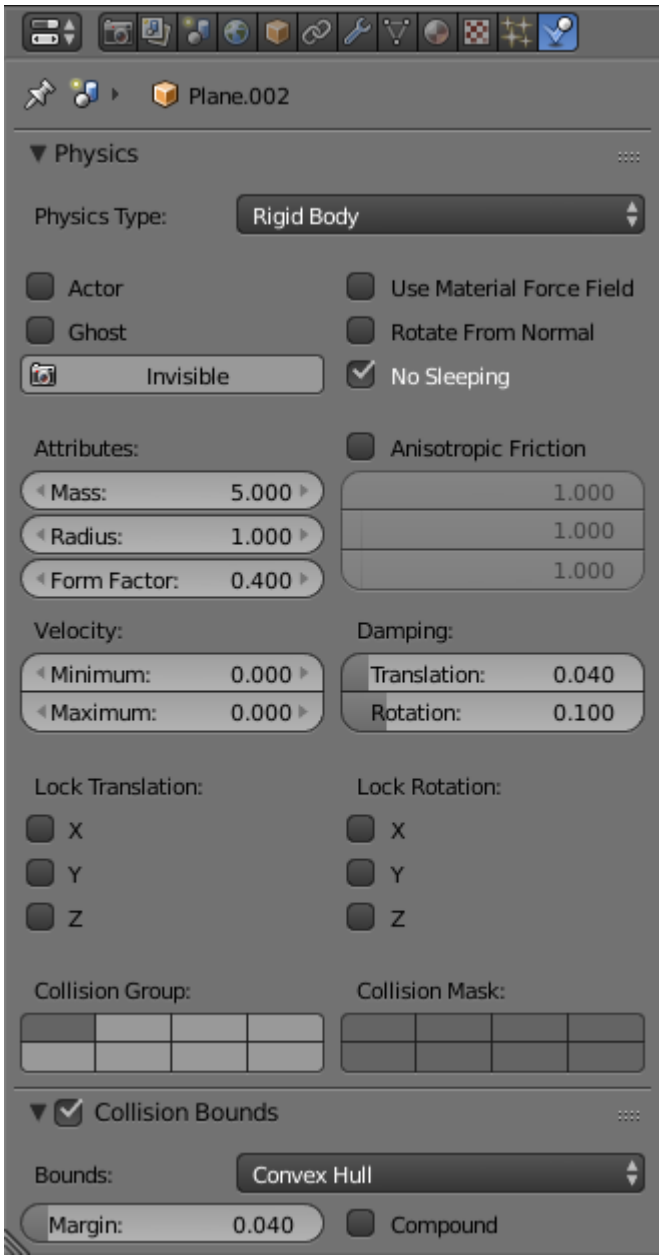
Сделаем её Rigid Body и разместим в комнате, включив два слоя одновременно:



Кстати, бочку я делал по тому же принципу что и комнату. А потом применил Spin. Об этом вы можете прочесть в книге Прахова (смотрим раздел «Полезное»):

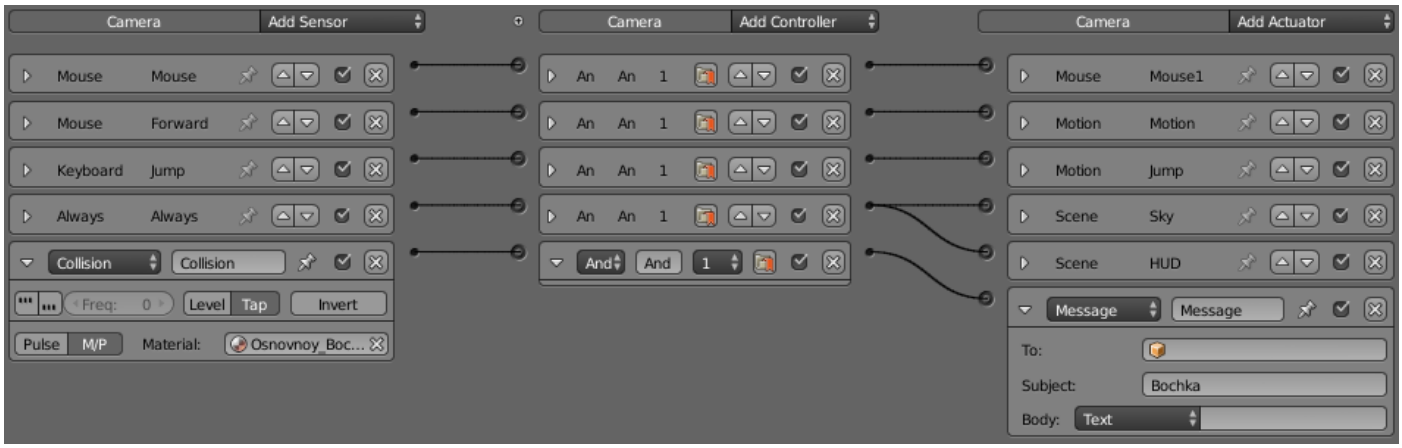


Ниже приведены настройки физики для бочки:



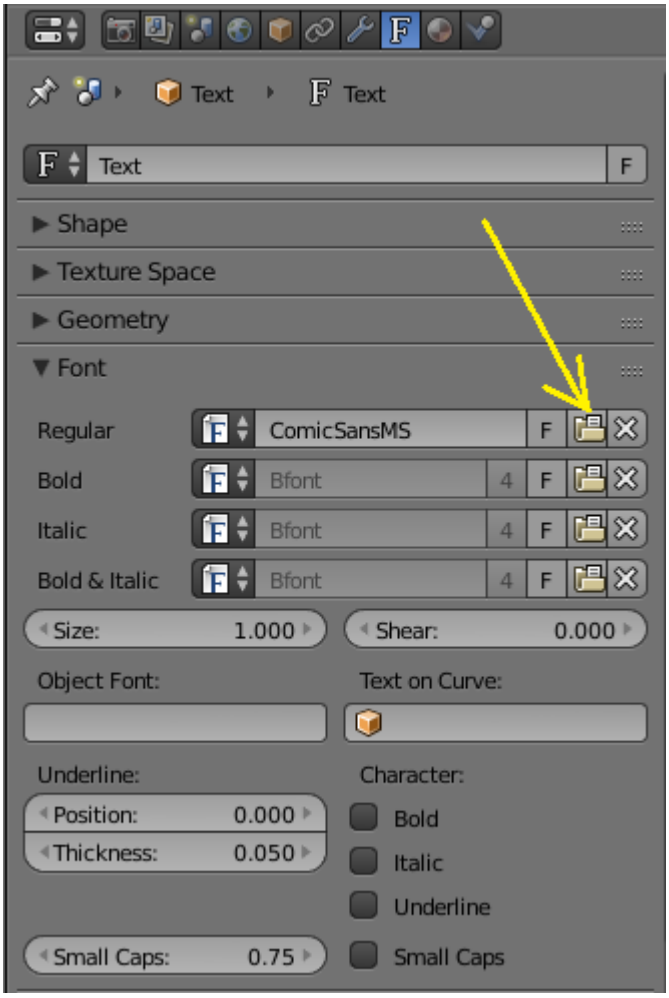
И так, включив два слоя:

Выделяем камеру в основной сцене и переходим в логику, где настраиваем столкновения с бочкой:

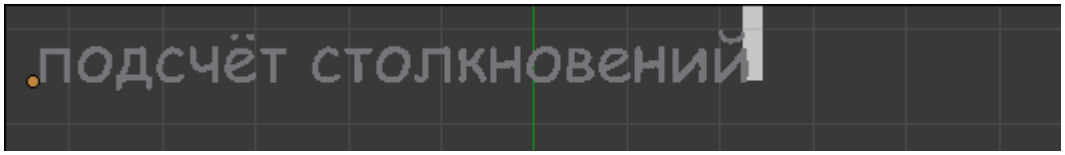


Что в сенсоре?

Включена кнопка M/P - отслеживать по материалу. В поле Material выбираем материал боковой поверхности бочки (физику крышки и дна вообще можно отключить в настройках их материала). Включена кнопка Tap – фактический в одно касание. Если не включить, то пока вы касаетесь бочки он будет считать десятки и сотни столкновений ! ☺ Нам это надо? ☺ В актуаторе Message в поле Subject пишем название метки, которую мы дадим принимающему сенсору Message в сцене HUD. Если этого не сделать, то сообщение будет отправлено на все возможные сенсоры Message. Переходим в сцену HUD. Добавляем текстовый объект, в котором меняем шрифт нажав на значок папки:

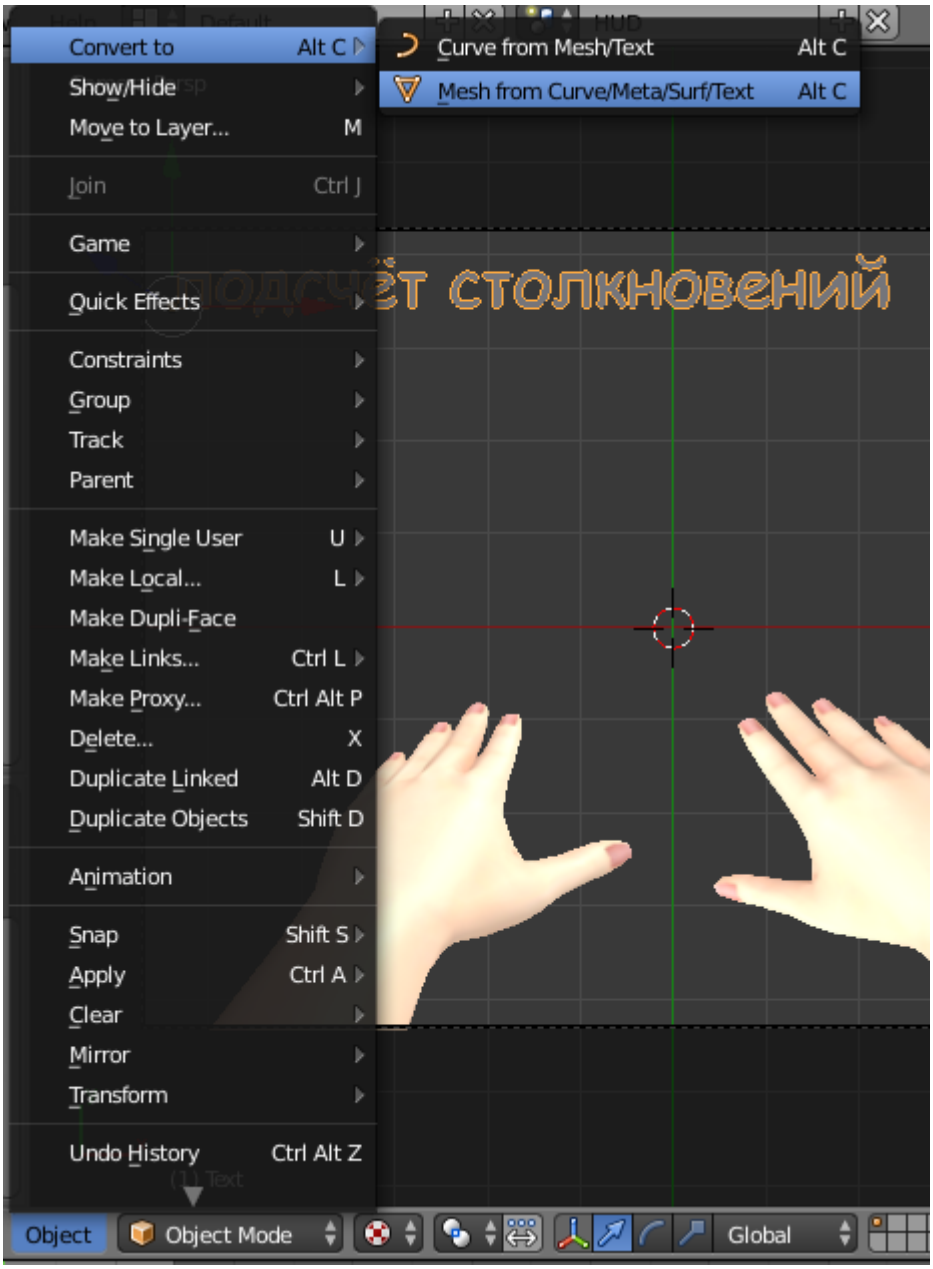


В Windows шрифты находятся по такому пути – C/Windows/Fonts. Выбираем любой, поддерживающий русский язык. При выделенном тексте переключаемся в режим редактирования и меняем надпись:

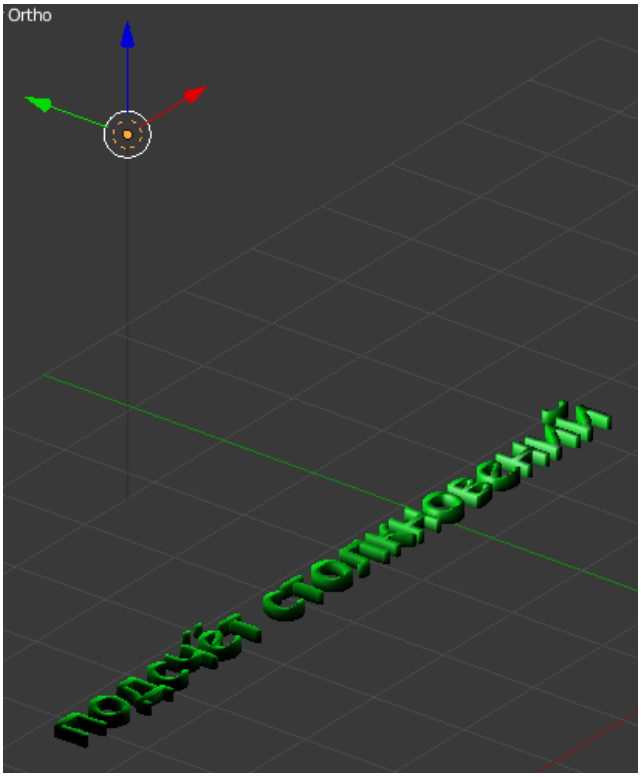


К слову, хочу признать свою неправоту. В самых первых уроках я писал, что Blender не дружит с русским… Не узнав всех возможностей я пытался рассуждать о программе ☺.

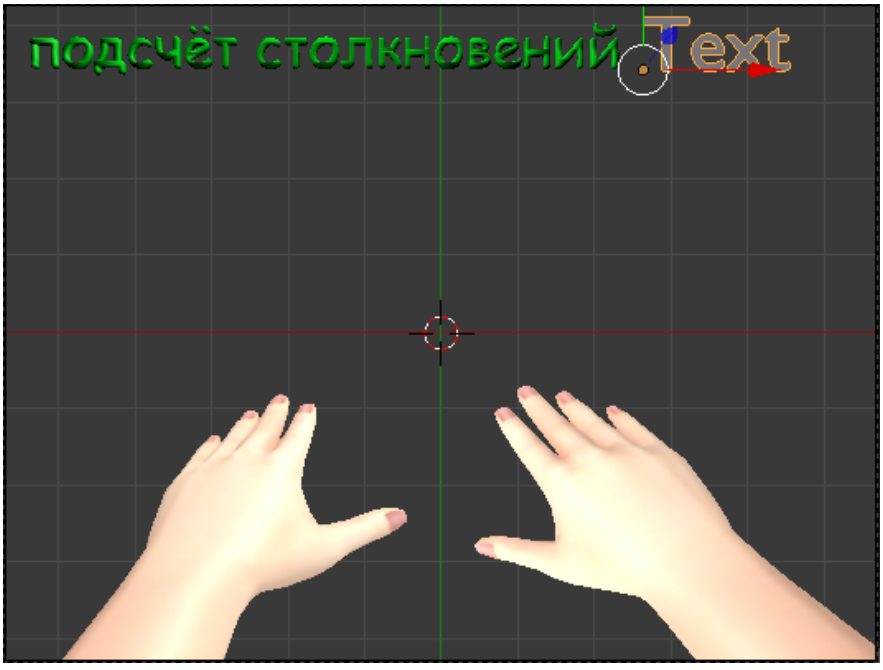
Ну да ладно. Далее. Поскольку мы не знаем на какой системе будет запущена игра, лучше сделать русский текст мешем, т.е. превратим его в объект. Конечно, после этого его нельзя будет изменить. Зато на всех машинах он будет выглядеть одинаково. Выходим из режима редактирования и делаем так:



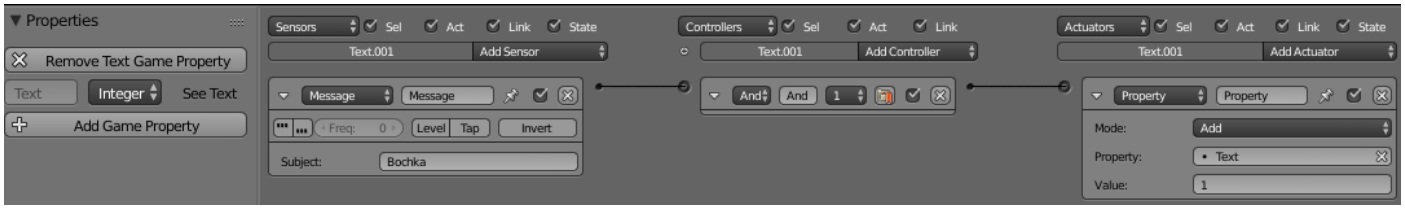
Теперь ему можно назначить материал, как любому другому объекту и в Diffuse выбрать нужный нам цвет. Или вообще покрыть текстурой! ☺ Я его немного выдавил, чтобы он стал объёмным и добавил источник света:



Сбрасываем выделение и делаем вид с камеры. Добавляем ещё один текстовый объект (рядом). А вот он уже будет показывать цифры, поэтому его изменять не нужно:

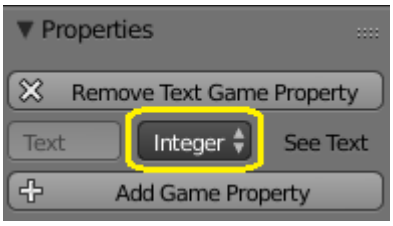


Не снимая выделения с текстового объекта переходим в логику (всё там же в сцене HUD):

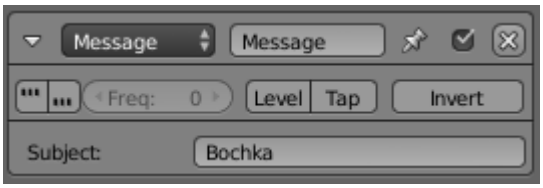


Что мы здесь сделали?

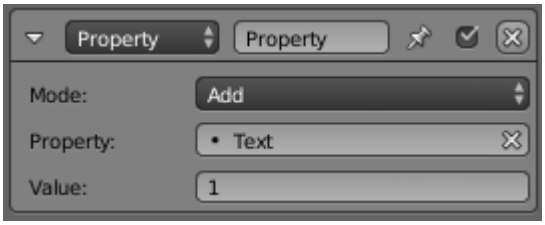
Добавляем свойство тексту:



Вставляем сенсор Message, где в поле Subject пишем метку Bochka (т.е. от кого принимать сообщения):



Вставляем актуатор Property (Add), где выбираем свойство Text. В поле Value пишем 1 (т.е. по сколько добавлять):



Соединяем сенсор и актуатор. Переходим в основную сцену. Делаем вид с камеры. Запускаем игру и убеждаемся, что мы всё сделали правильно ☺.

30 июня 2015 года.

Составил **Niburiec** для сайта **http://blender-game.ucoz.ru**