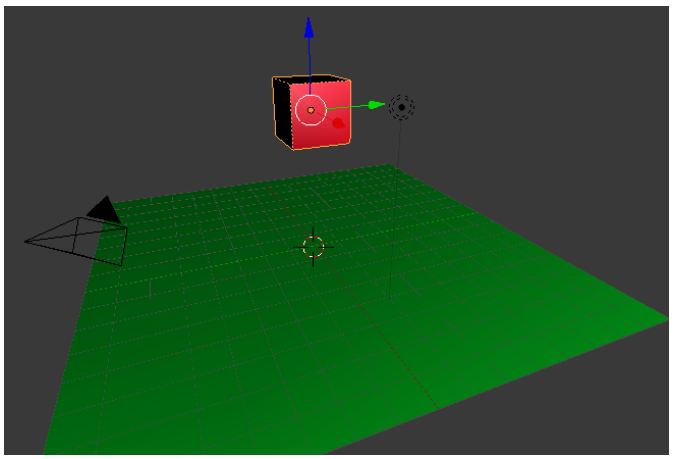
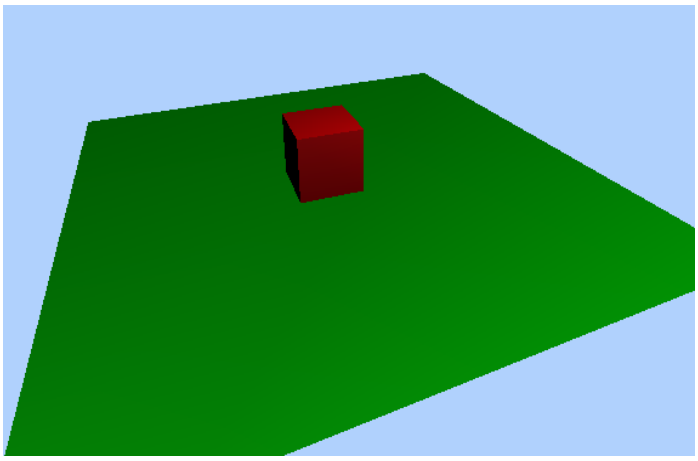
Постепенно подходим непосредственно к BGE. Начать изучение средств BGE я хочу с основ языка программирования Python. Язык, сам по себе, удивительный и открывает большие возможности. Но обсуждать его достоинства и недостатки мы будем в другом разделе. А здесь он нас интересует только в составе BGE. Чем он может нам помочь, и зачем он нужен?

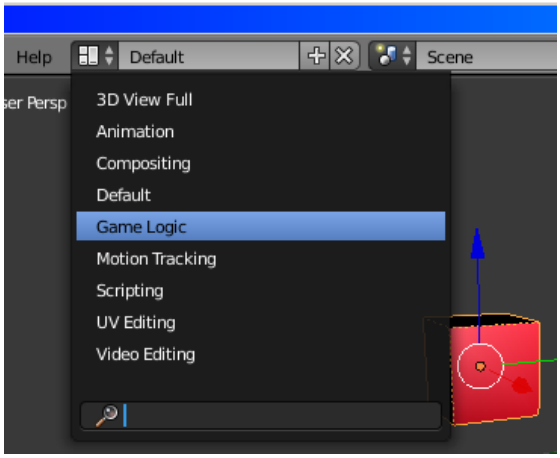
BGE имеет в своём распоряжении сенсоры (Sensor), контролёры (Controller) и актуаторы (Actuator). С их помощью осуществляется основное управление объектами и игровой логикой. Подробнее мы поговорим о них в уроке № 6. Но не всё можно реализовать только с их помощью. Есть некоторые функции, которые лучше запрограммировать, а иногда и только запрограммировать. Не стоит этого бояться. Если вы сумеете хорошо разобраться с применением Python в BGE, то возможно вам это понравится и написание скриптов станет основной частью ваших проектов.

Начнём. Откроем новый проект. Перейдём сразу же в Blender Game. Готовый куб в нём уже есть. Добавим плоскость «земли» и назначим им материалы (какие хотим). Поднимем куб немного над плоскостью и сделаем его физическим объектом Rigid Body (с массой и типом столкновений *см.предыдущие уроки*). Включим режим отображения текстур и нажмём лат. «P», чтобы проверить физику:

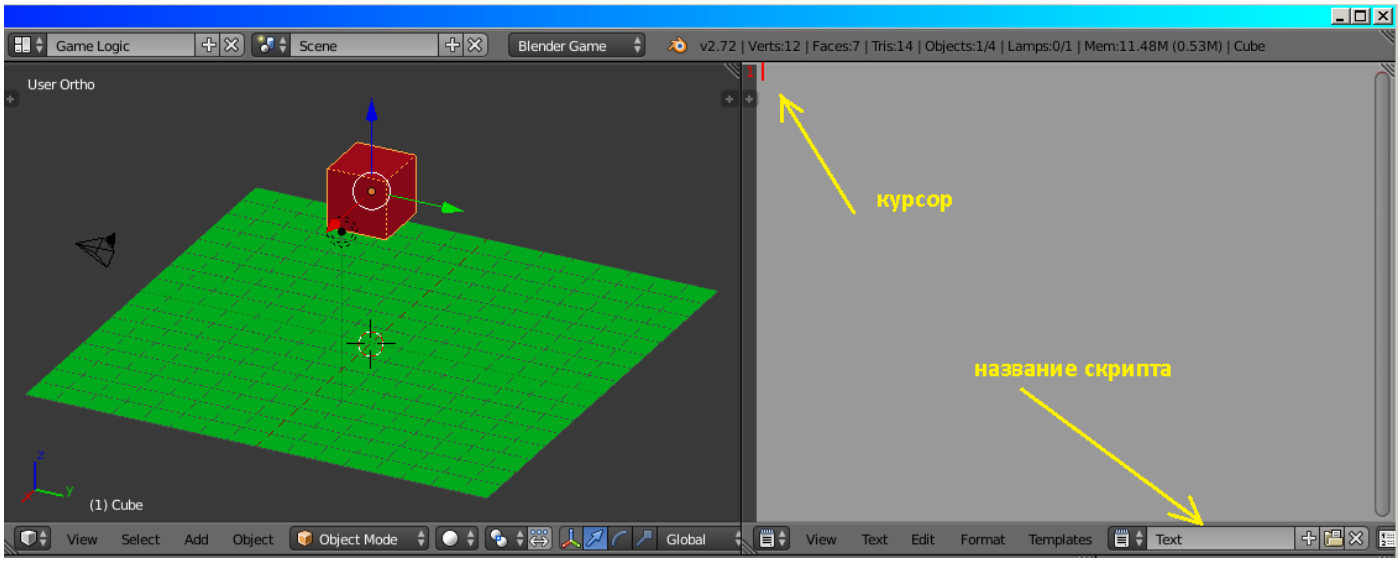




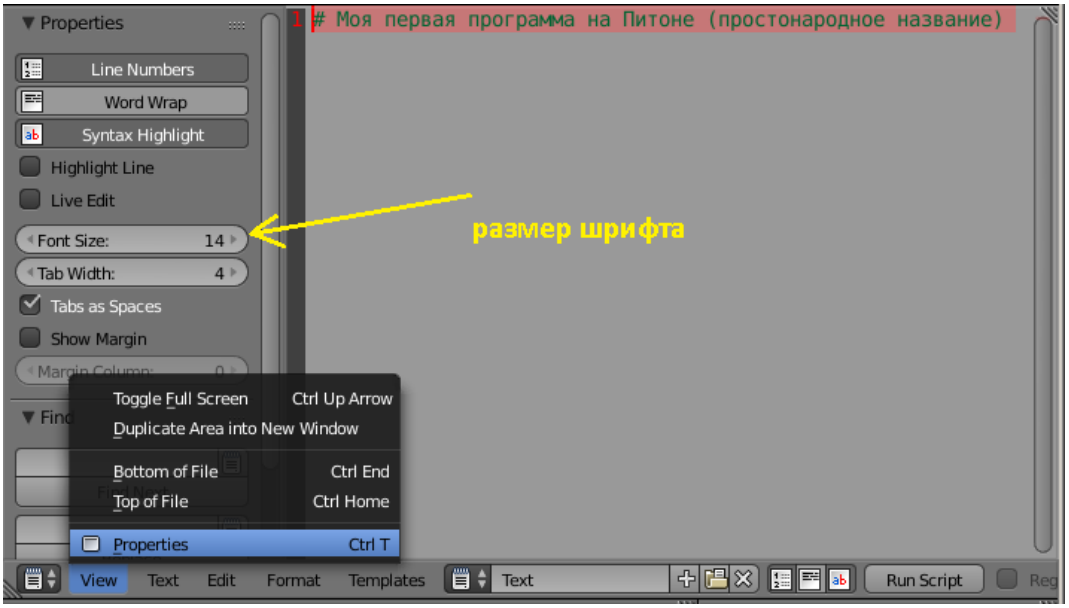
Переходим в режим программирования логики:



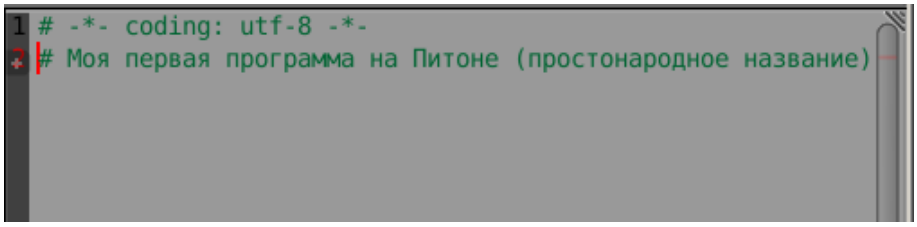
И тут у нас открывается в средней части экрана два окна: верхнее – обычное 3D окно, а нижнее – окно добавления контроллёров. В правой верхней части пустое окно. Это и есть окно добавления скриптов на языке Python. В нём же можно писать обычные комментарии к проекту. Это очень удобно. В окне 3D вида мы всё так же можем навести курсор и нажать лат. «P» для того, чтобы увидеть движок в действии. Ну, давайте напишем комментарий. Для этого в пустом окне для скриптов ищем снизу меню New и жмём её. При этом в окне появится мигающий курсор, а названию присвоится имя Text (хотя мы всегда его можем поменять):



Правда, справедливости ради, тоже самое можно было сделать через меню *Text -> Create Text Block* . В поле курсора пишем # Моя первая программа на Питоне (простонародное название) . Если шрифт по каким-то причинам вас не устраивает, то вы можете его изменить через меню *View -> Properties* . Откроется дополнительная вкладка, в которой я изменил размер с 12 на 14:



В меню Text есть две интересные строчки. Одна – Save As (сохраняет наш скрипт). А другая Open Text Block (загружает сохранённый скрипт). И так, вернёмся к нашей строке. Знак # обозначает, что после него идёт строка комментария. Вообще-то, чтобы в будущем можно было использовать строки на русском языке, нужно в самом начале вставить ещё одны строку # -*- coding: utf-8 -*-



Всегда ставьте её самой первой.

В принципе, сдесь мы можем писать пояснения ко всему проекту. При этом, при сохранении проекта, данный текст сохранится в нём автоматический.

Теперь давайте создадим новый скрипт, с именем Begin_Left (*Text -> Create Text Block*) . Вставляем туда нашу важную первую строку (см. выше) и второй строкой подключаем модуль BGE

```
import bge
```

К слову сказать, пробелы играют в Python важную роль. Например, в функции цикла или выбора тело цикла пишется с одинаковым отступом. Иначе программа не будет работать.

Далее, нам нужно получить объект, с которым мы будем работать. В нашем случае это куб (Cube). Сдесь есть два варианта. Первый – получить объект по его имени. Второй вариант – получить выделенный объект. Оба варианта применимы в разных ситуациях, но одинаково хорошо работают. Допустим, нам нужно чтобы при вызове скрипта наш куб всякий раз перемещался на единицу по оси X. Вот два возможных варианта кода:

```
scene = bge.logic.getCurrentScene()     # получить текущую сцену
objList = scene.objects                 # получить список объектов
car = objList["Cube"]                   # получить объект по имени Cube
car.worldPosition.x += 1                #  переместить объект на 1
```

Сдесь car всего лишь имя переменной, которое может быть любым. Или второй вариант, если объект выделен и мы пишем скрипт конкретно под него (как в нашем случае):

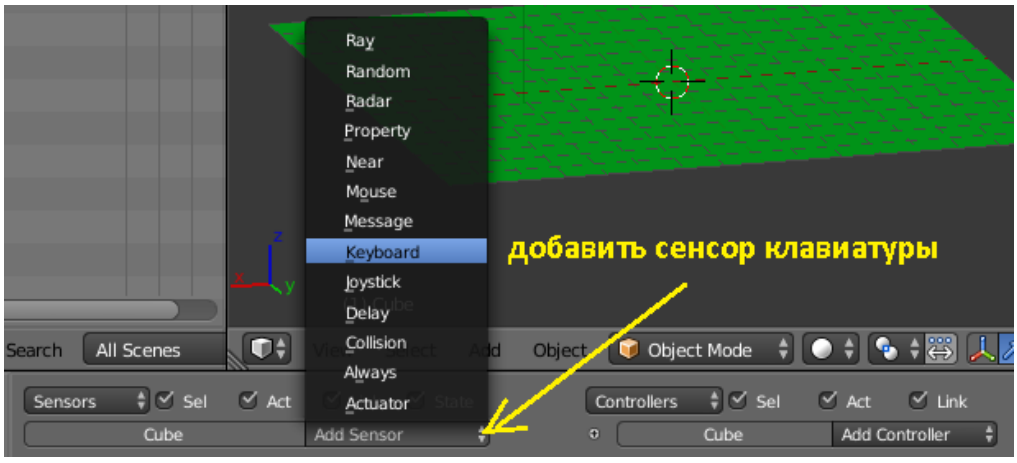
```
cont = bge.logic.getCurrentController() # получить текущий контролер
obj = cont.owner                        # получить выделенный объект
obj.worldPosition.x += 1                #  переместить объект на 1
```

Я выбираю первый вариант. Он более универсален:

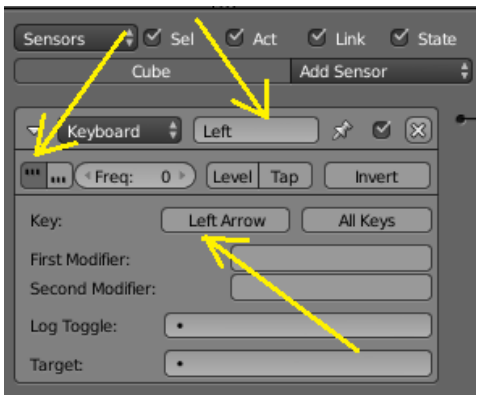
```
1 # -*- coding: utf-8 -*-
2 import bge
3 scene = bge.logic.getCurrentScene() # получить текущую сцену
4 objList = scene.objects       # получить список объектов
5 car = objList["Cube"]         # получить объект по имени Cube
6 car.worldPosition.x += 1      #  переместить объект на 1
7
```

Нужно заметить, что в действительности Python работает очень быстро и значение +1 нужно поменять на +0.1 , иначе наш куб не просто сместится, а улетит в мгновение.

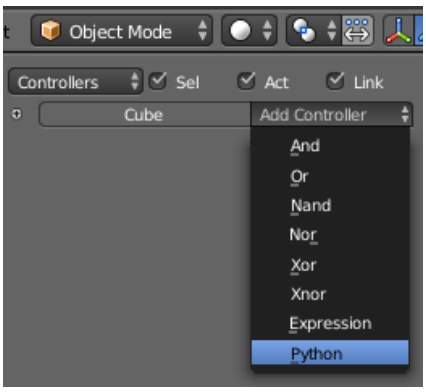
Теперь подключаем сенсор. Чтобы наш скрипт сработал, ему нужно дать понять, при каком действии он должен включаться. Для этого существуют сенсоры. Мы возьмём сенсор нажатия клавиши (стрелка влево):



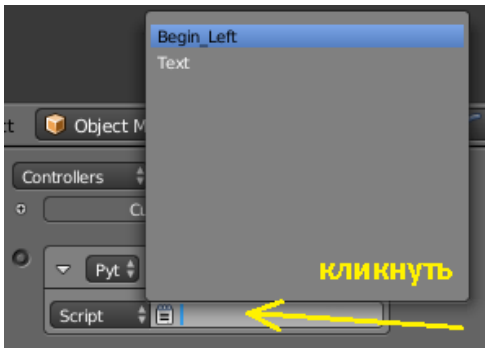
Где в поле Keyboard мы делаем двойной клик и вписываем название Left. А в поле Key кликаем один раз, и пока высвечивается фраза Press a key жмём клавишу со стрелкой влево.



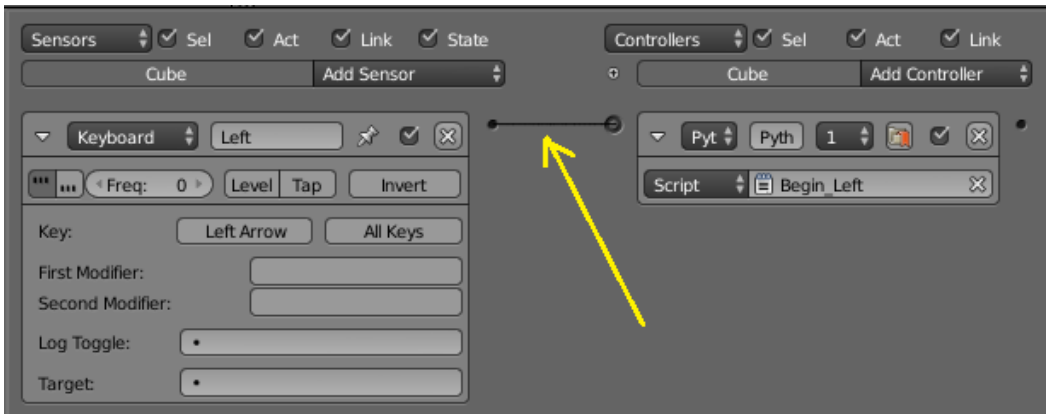
Добавляем контролёр:



Где в списке выбираем наш скрипт управления Begin_Left :



И соединяем всё ЛКМ:

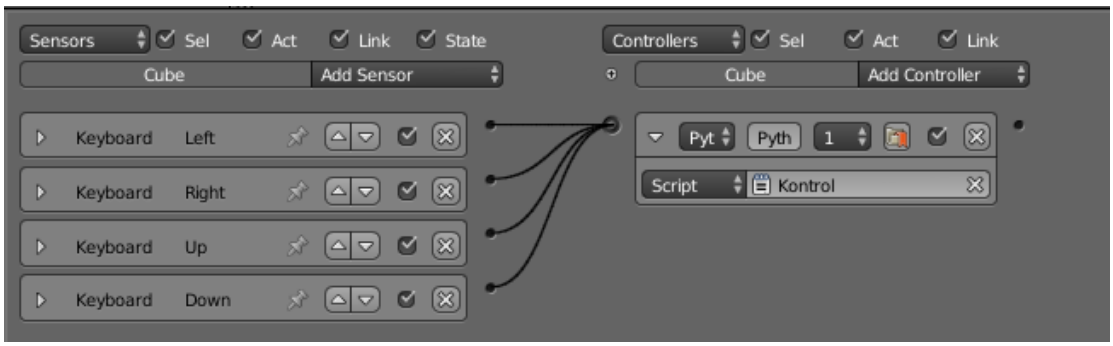


Теперь если запустить движок и нажать клавишу «стрелка влево», то кубик будет перемещаться в левую сторону, пока мы эту клавишу не отпустим. Напомню, остановить движок можно клавишей Esc. Для наглядности переведите окно 3D вида в вид сзади (*View -> Back*).

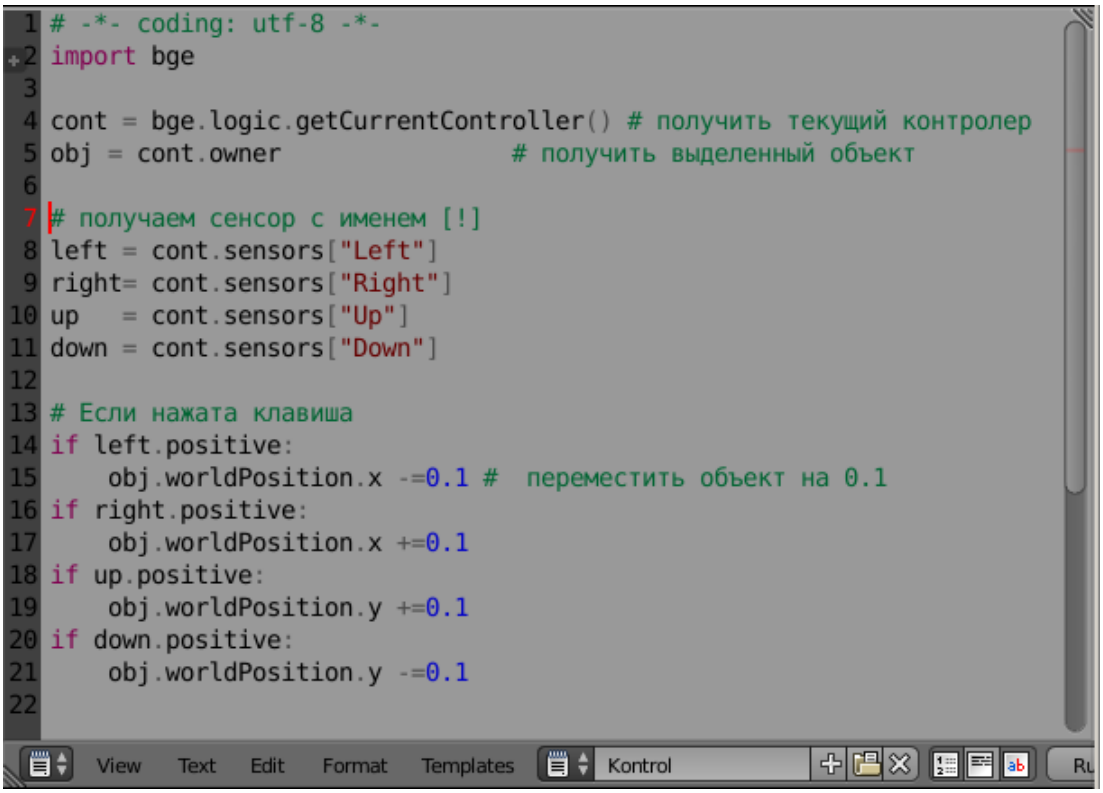
И конечно, если в скрипте поменять + на минус 1, то куб будет двигаться в противоположном направлении. Так что домашним заданием будет добавить ещё один скрипт под названием Begin_Right, добавить ещё один сенсор под клавишу «стрелка вправо» и заставить двигаться куб вправо. А ещё, подумайте, как можно его заставить двигаться вперёд и назад ? Правильно, в скрипте изменить ось X на Y и добавить дополнительные сенсоры под соответствующие клавиши.

Со своей стороны скажу, что можно в одном скрипте прописать управление всеми четырьмя клавишами через функцию If (если). Я использовал получение текущего контролёра выделенного объекта. Сначала добавим ещё три сенсора управления, чтобы получилось «влево», «вправо», «вперёд», «назад». Сразу их подключим к

скрипту. Если хоть один сенсор не подключить, то весь скрипт может не работать, поскольку все сенсоры в нём уже определены:



Не менее важно следить за правильным написанием имён сенсоров. Они должны совпадать с тем, что вы пишите в скрипте. Вот весь скрипт:



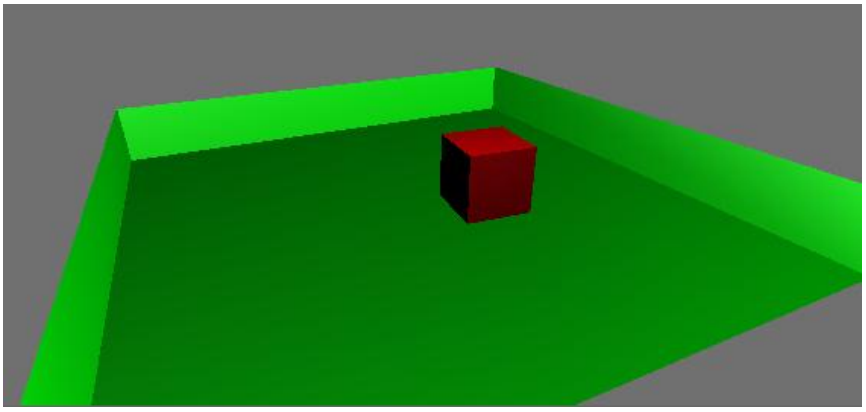

```
# -*- coding: utf-8 -*-
import bge

cont = bge.logic.getCurrentController() # получить текущий контролер
obj = cont.owner            # получить выделенный объект

# получаем сенсор с именем [!]
left = cont.sensors["Left"]
right= cont.sensors["Right"]
up   = cont.sensors["Up"]
down = cont.sensors["Down"]
```

```
# Если нажата клавиша
if left.positive:
    obj.worldPosition.x -=0.1 #  переместить объект на 0.1
if right.positive:
    obj.worldPosition.x +=0.1
if up.positive:
    obj.worldPosition.y +=0.1
if down.positive:
    obj.worldPosition.y -=0.1
```

Как видим, нет ничего сложного. Достаточно школьных навыков программирования. Изучать Python лучше всего на практике. Поэтому наше дальнейшее знакомство с ним продолжится в следующих уроках. В Blender 2.7 и выше используется Python 3.** . Будет прекрасно, если вы почитаете дополнительную литературу по этому интересному языку. Заходите в раздел «Полезное».



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