The Blender Game Engine

Blender has a powerful **built-in Game Engine** that is rapidly growing in popularity. It is one of the very few 3D software that have Game development functionality incorporated and the best thing about the GE is that it provides the opportunity to create games without writing any code. That is possible via a straight forward **Logic Brick system**, where the user incorporates logic interactions to the 3D environment using 'drag and drop'.

Although code is not necessary there is the possibility to use a **scripting language called Python** to create more elaborated effects for games.

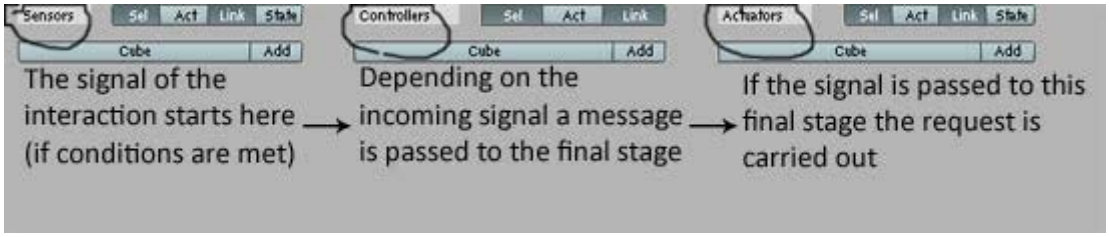
The first thing you need to know about the game engine (GE) is how to start it. Move the cursor over the 3D view and press **P** (Play). Now you are in the Game mode!! Hit **Esc** to get out and return to Blender's original interface.

It will be helpful in the future if you know how to change the 3D view size to see the **Game in full screen**. Hover over the 3D view and hit **ctrl+uparrow** (max) or **ctrl+downarrow** (min).

To use the GE we will need to be in the **Logic** panel. Go down to the Buttons panel and hit the first button (Logic – F4) that looks like a Packman head. In here we will control the GE.

Blender uses two ways of Game development. Either the visual click and drag system that creates interaction without the need of code, and this is what we will use, and a scripting language called Python that can be integrated in the GE.

Each Game we will create will use Logic bricks, as a visual way to set up interactions. These can be connected visually (with lines that run from one brick to the other). There are three types of logic Bricks: **Sensors, Controllers and Actuators** that are used together.



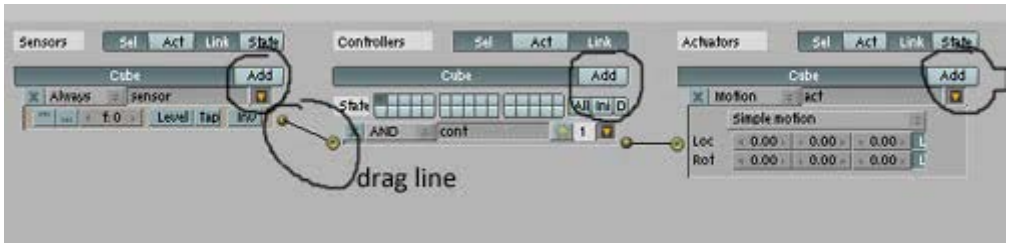
1 Sensors: detect some input (this can be anything from key press, to mouse click, timers, collision and so on)

2 Controllers: are used to link Sensors and Actuators and allow more complex control on how those two interact.

3 Actuators: carry out interactions within the Game (e.g moving an object, playing a sound, triggering an animation and so on)

We will now create our first interaction example.

Go to the Logic Panel and click on the **Add** button once for the sensor, controller and actuator (this is called a line, in the GE). To connect them together, click and drag from the socket to the start of the next brick. To delete a connection, move the cursor over the line and press either **X or Delete**.



Move the cube (no-physics)

We will first use direct manipulation to move the cube in the scene. Later we will use the build-in physics engine to allow more complex interactions such as collision and gravity.

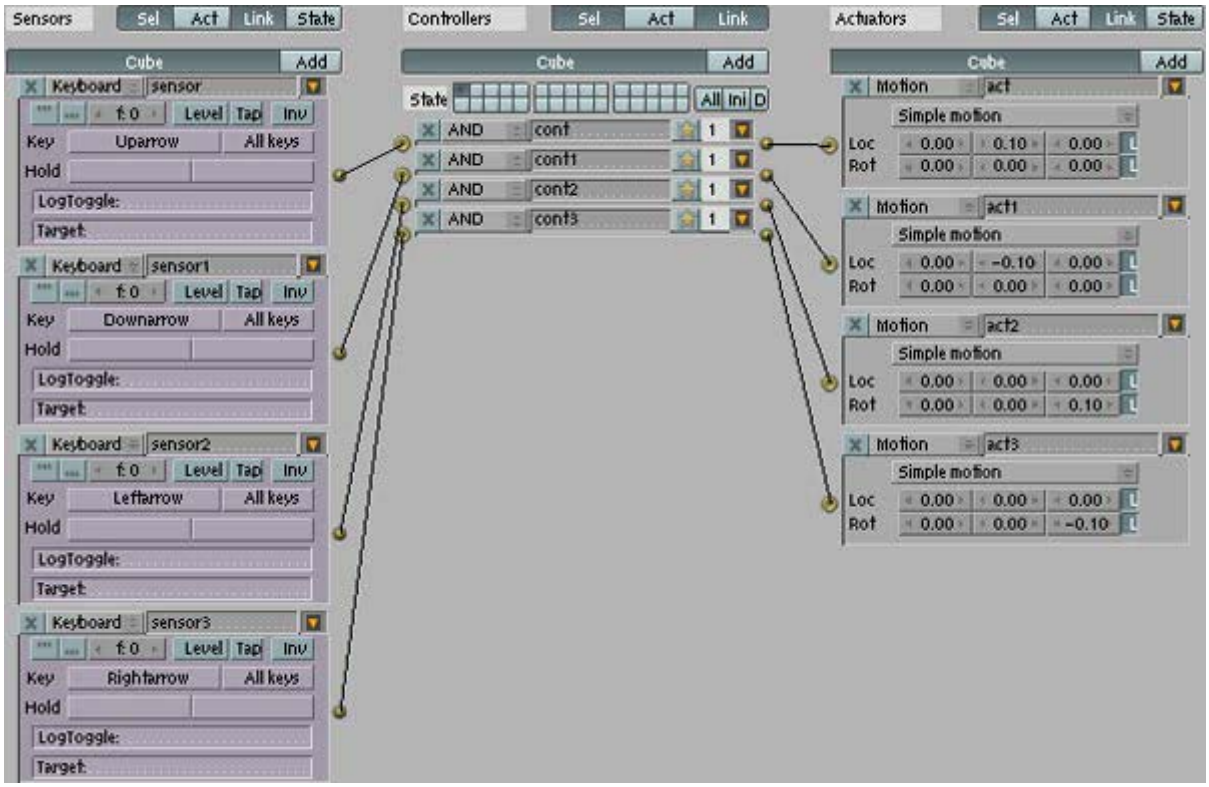
As you see the Loc row in the Actuator we added before has three values (all set to zero). Those define the three axes along which the cube will move. So give a value of 0.10 to the second one (Y axis) and press P. The cube is moving!! Note that the cube moves indefinitely since the controller is set to ‘Always’.

We will now do something more exciting than this.

Go and click on the double arrows button on the Always sensor and change it to > Keyboard. Now go on the Key area > click > hit uparrow.

Press P again. Now the cube doesn’t move unless we hit the uparrow key.

Define three more actions (as seen on the image below) to add more interactivity to the cube.



Now hit P again and see how the cube moves, using the arrows.

GE housekeeping

It is very important to name Sensors, Controllers and Actuators, so as to keep track of them later on. Also try to minimize the ones you are not using to make room for the new ones. To **name** a Sensor, Controller or Actuator click on the text filed beside its name and type and to min it hit the small yellow arrow button beside the name you just typed in.

To erase a Sensor, Controller or Actuator hit the **X** button.

Move the cube (with-physics)

Hit the ctrl+X buttons again to start a new Blender and select the cube.

To make the object follow the laws of physics go to the Logic buttons and hit the **Actor** button (this will make other objects in the scene detect it and interact with it). From the expanded list beside the Actor button go and select **Dynamic**.

Most common chooses we will use from this list:

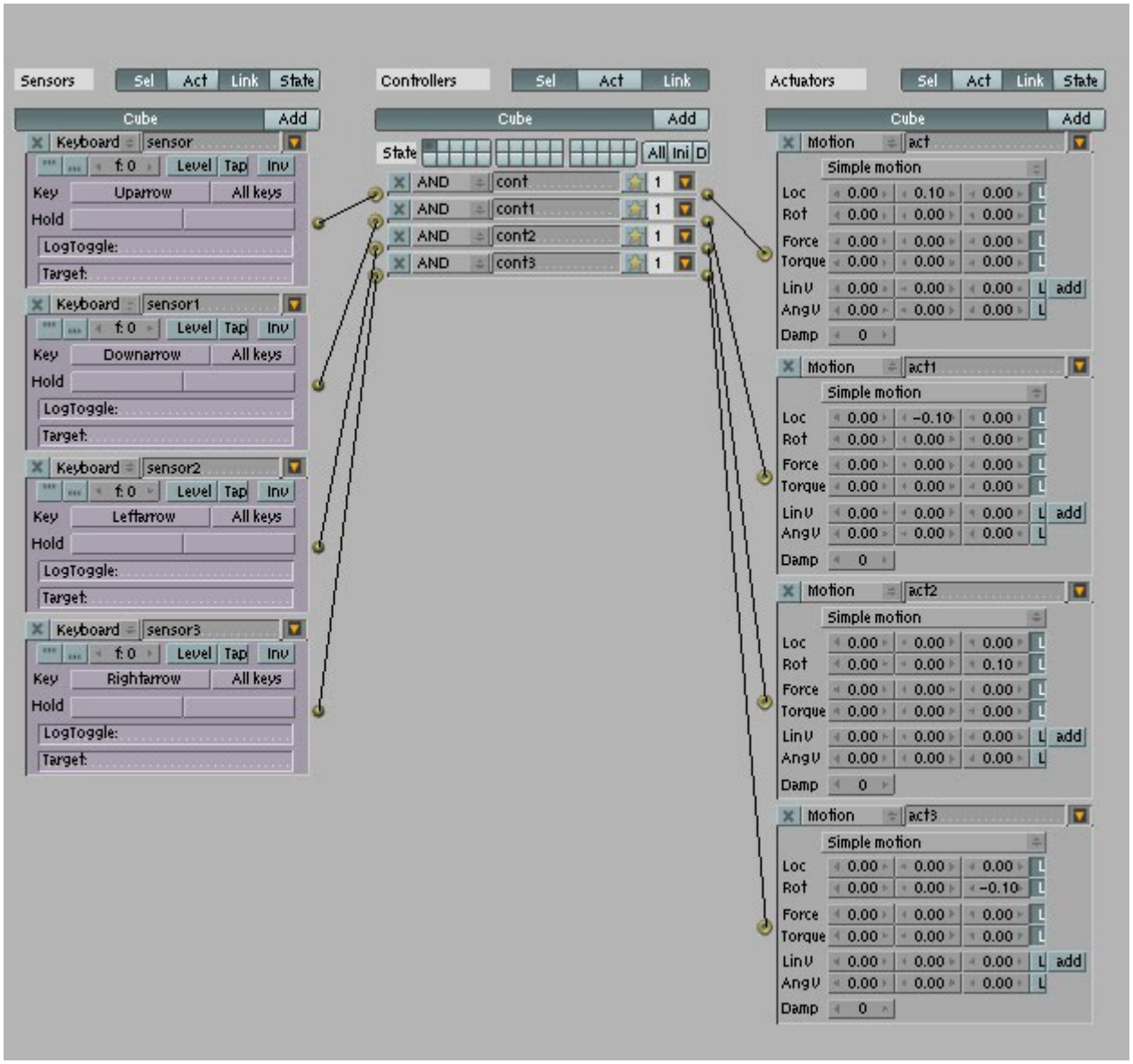
1 Dynamic: this option tells the GE that the object is a physical one.

2 Static: this object is unaffected by gravity

If you now press P you will see that the object moves although there are no Sensors applied to it. That is because it is now a physical object affected by gravity.

Try to put a plane under it and when you hit P you will see it landing on the plane.

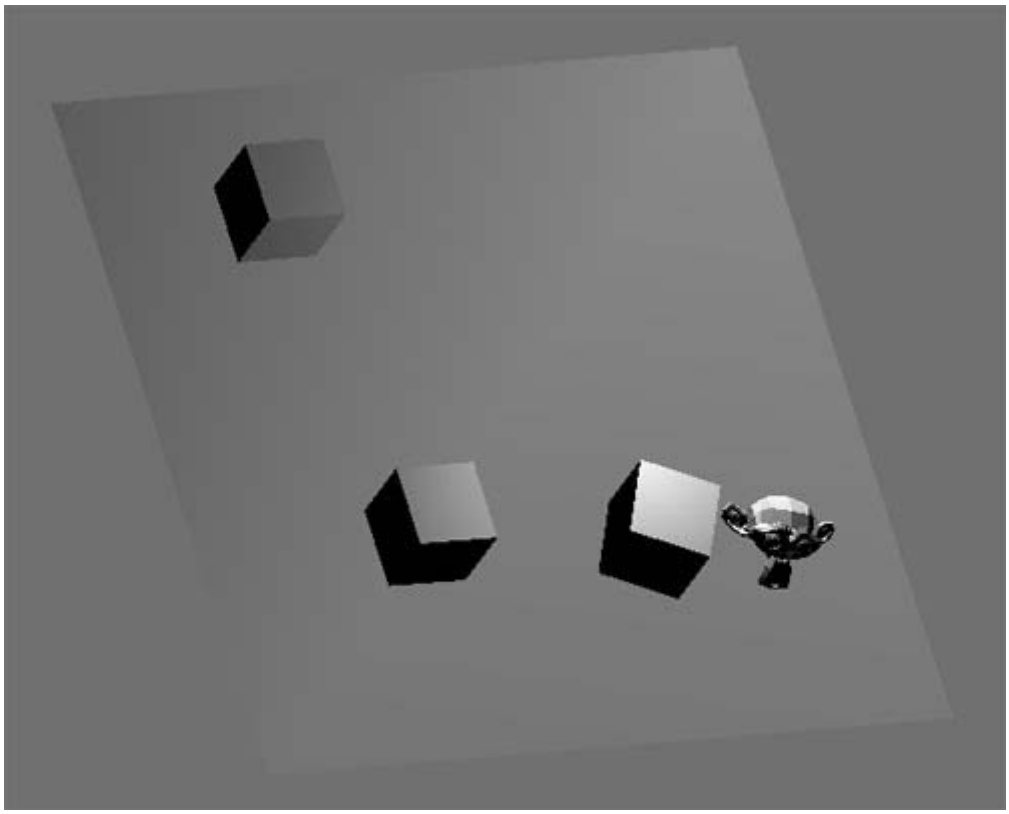
As we did before we will now create keyboard interactions to move the cube on the plane. Create four lines and set them as in the image below.



Finally we will add some obstacles in the scene to see how the cube interacts with them. Add two cubes and a monkey to the scene. If you play now you will see that the cube hits the objects but they don't move. So we will make them physical. Make one of the cubes > **Actor > Dynamic** and the other one **Actor > Rigid Body**.

Notice the difference between their motions when the main cube hits them. The Rigid Body also rotates in an odd way, almost like a sphere. This is because the GE assumes that the newly added object has a spherical mesh. To fix that go to the **Bounds** button hit it and select **Box** as the bounds type.

Do the same for the monkey but choose **Convex Hull** instead of Box (best for complex meshes).



Our first Game - 'Pumpkin Run'

Now that you know the basics on how to create motion in the GE environment and have seen the difference between physics and no physics interaction we will create a simple game called Pumpkin-Run*. This example will demonstrate most of the core techniques for making a 3D game and will present various functionalities of the Logic Bricks.

Go to the **top view** of the scene by pressing PAD-7.

*note: some pre made materials (e.g models will be used) to save time

<u>Create the Ground/UV Unwrap:</u>

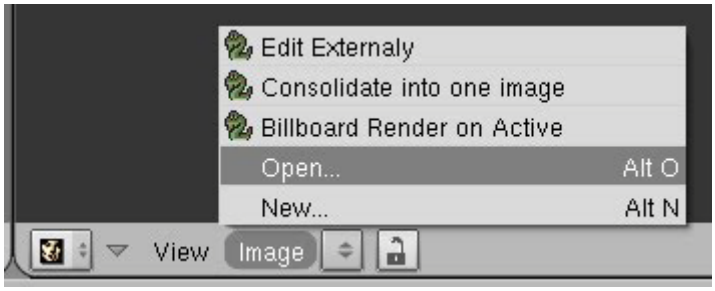
In the next steps we will create a ground for our game. Select the cube again by pressing RMB with your mouse over it. Now press XKEY to delete it.

Press SPACE to call the Toolbox. Choose **Add->Mesh->Plane**. Scale the plane since this will be the ground of our scene.

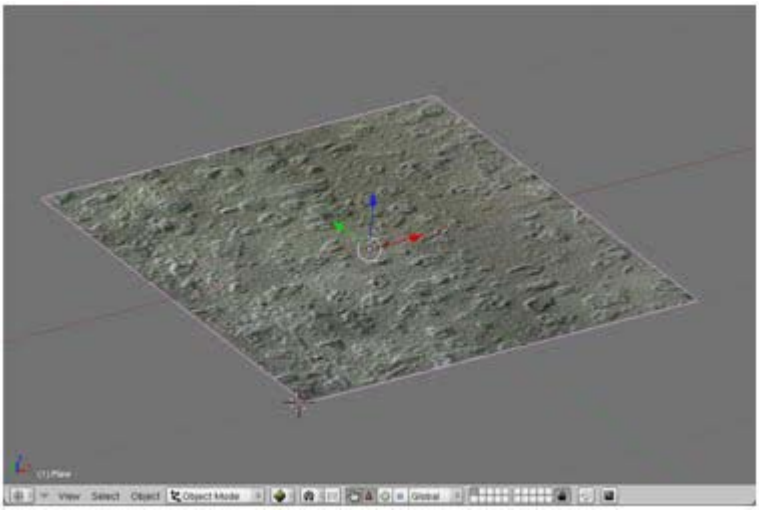
Now split the screen and **load the UV/Image Editor** to the second view.

Select the plane again hit TAB to go to Edit mode, now hit **U** and choose "**Reset**" from the "UV Calculation"-menu.

Move your mouse to the right window and select **Open** from the Image-menu.



Navigate to Pumpkin/textures/ and locate the concgray_q.jpg image. Right click on it and then choose "**Open Image**".



The texture now shows up in the 3D View to the left (to see it go to **Draw Type >Textured Mode**). Now leave Edit Mode by pressing TAB.

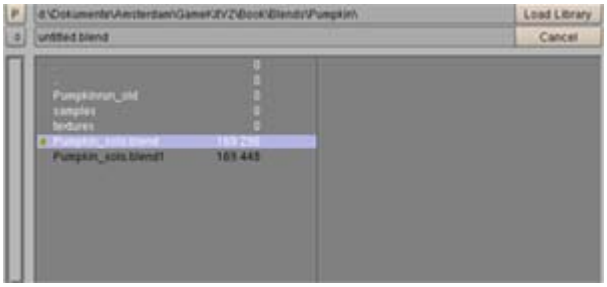
It is now time to save your scene. To ease the process we will **include the texture in the saved scene**. To do so, choose the menu "**File->External Data->Pack into .blend file**". A little parcel-icon will appear in the menu bar to indicate that this scene is packed. Now use the "**File->Save As**" browse to your hard disk and save.

Note: although packing textures into the .blend file is quite convenient it will soon make your file too large (especially the more you add image textures) so avoid using this in your animations!!

Appending an object from another scene:

Just because we don't have the time available to model everything for the needs of this game we are going to append models to make things quicker. The main character of this Game will be a Pumpkin.

Go to **File > Append or Link** and browse to find the folder called Tutorials/Pumpkin/Pumpkin_solo.blend.



Now click on "**Objects**" with LMB. You will see the objects contained in that scene. Select all objects by pressing **AKEY**. Confirm by pressing ENTER or clicking with LMB on the "**Load Library**" button.

You can now see the pumpkin as an orange spot, sitting in the middle of the plane in the left 3D View.

Now **go to camera view** (PAD-0) and start the Blender game engine! While pointing with the mouse over the camera view, press **PKEY** and you can see the pumpkin on our textured ground. The pumpkin character has an animated candle inside and you will see it flicker. To stop the game engine and return to Blender press **ESC**.

Make the Pumpkin Dynamic and Move it:

Move your mouse over the right 3D view and press PAD-3 to get a **view from the side**. Select the character with the RMB (click somewhere on the pumpkin), and it will outlined pink to indicate that it is selected.

Now go to the **Logic Buttons** icon in the Buttons bar.

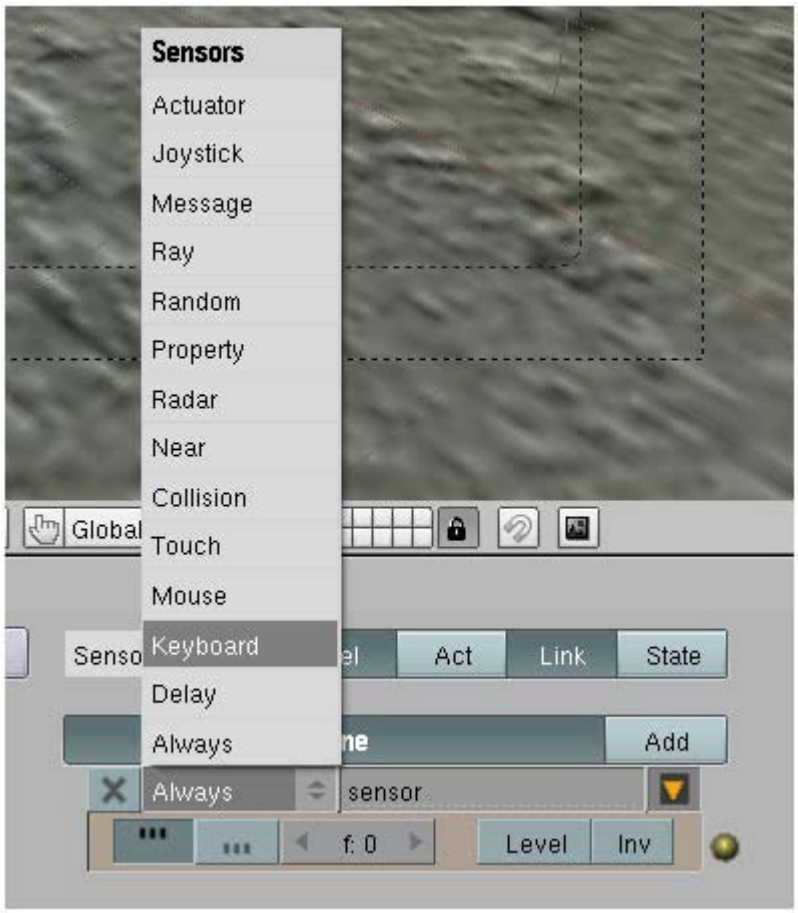
Change the pumpkin to "**Dynamic**". This setting will enable the character to follow the laws of physics, like falling, bouncing and reaction to collisions.

Now **go back to camera view** and select the object. Use **G to move it up** so as it disappears from the camera view. Then move the mouse to the left 3D View (the camera view) and press **PKEY** to start the game engine. The Pumpkin falls and rests on the ground. Press **ESC** to exit the game engine.

The Logic Buttons (F4) are logically divided into four columns. We have already used the left-most column to set up the object parameters and make the pumpkin fall. The three right columns are used for building the interactivity into our game.

Now we will move the pumpkin at our request using lines of **Sensors, Controllers and Actuators**. **Add** a line.

For the first task, press and hold the Menu Button now labelled with "Always" and choose "**Keyboard**" from the pop up menu.



Now LMB click into the "**Key**" field and press **uparrow**.

Now have a closer look at the Motion Actuator. We will now define how the player should move. The third line of numbers labelled "Force" defines how much force will be applied when the Motion Controller is active. The three numbers stand for the forces in X, Y, and Z-Axis direction.

Since the character is Dynamic we will use **Force instead of Loc** (Location).

Change the second number (**Y**) **in the "Force" row to 10**.

Wire the line, go to **Camera view** and press **PKEY** to start the game engine and when you press the UPARROW key briefly, the player moves passing the camera.

To make the movement more interesting, **we can now add a jump**.

To do so enter a 20.0 in the third (z-Axis) field of the Motion Actuator. But now, If you try it again in the game engine, you can see that there is a problem: **If you hold the key pressed, the pumpkin will take off into space**! This is because you also apply forces when in the air.

To solve this we have to ensure that the forces only get applied when the pumpkin touches the ground. That's where the Touch Sensor kicks in. **Add a new Sensor** by clicking on "Add" in the Sensors row. Change the type to "**Touch**" like you did for the Keyboard Sensor.

Wire the Touch Sensor to the first AND Controller. Now the Keyboard and the Touch Sensor are connected to that controller. The type "AND" of **the controller will only trigger the Motion Actuator when the key is pressed AND the player touches the ground**.

This is an easy way to add logic to your interactive scenes. As well as the AND Controller there are also OR, Expression and Python (Blender's scripting language) Controllers which all offers more flexibility to make your game-logic.

Because the forces are now applied only a fraction of a second we might adjust them to higher values, 50.0 for force Y and 100.0 for force along the Z-axis should get you started, feel free to experiment here to get a idea how the forces work.

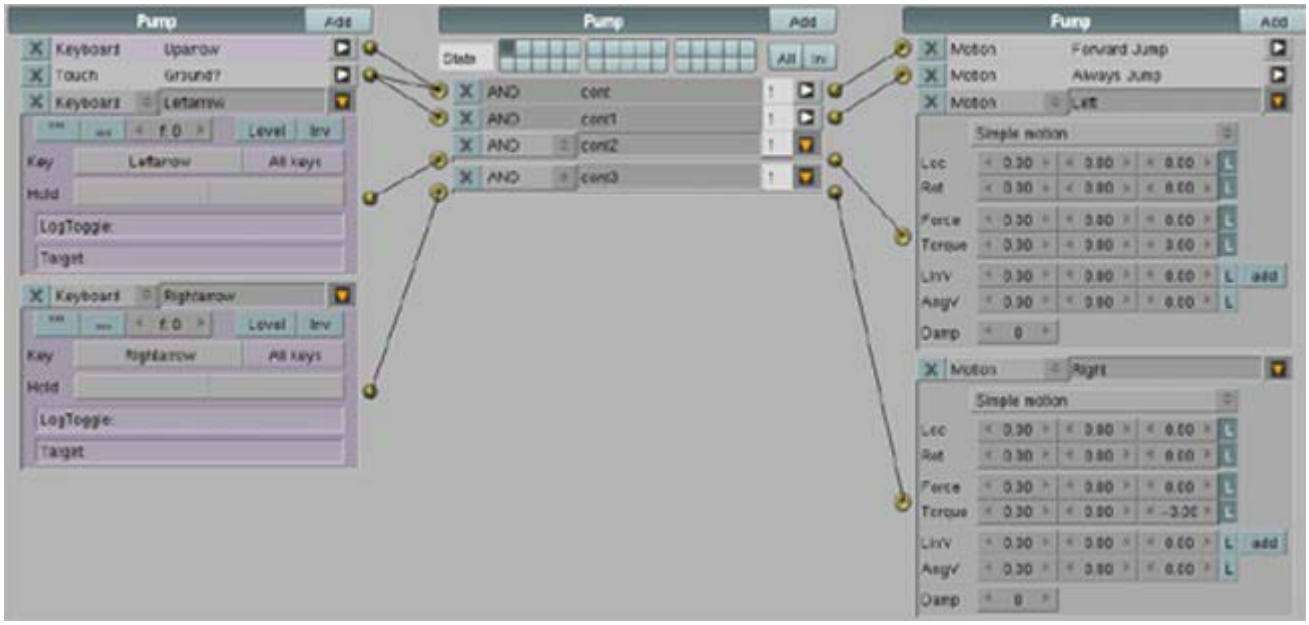
To make the movement more dynamic, we will now add Logic Bricks to **make the pumpkin jump constantly**. **Add a new Controller and** a new **Actuator** by clicking "Add" in the appropriate row. Name the new Actuator "Always Jump". **Wire the Touch Sensor with the new AND Controller input** and the output of the Controller to the new Motion Actuator "Always Jump".

On the second Motion Actuator give the value of 4o to the Z axis.

Yes, not only one Controller can be connected to two Sensors, **a sensor can also "feed" two or more controllers**. Start the game again with PKEY, the pumpkin jumps, UPARROW moves it forward.

Now we add more Logic Bricks to steer the player with the cursor keys.

Add a new Sensor, Controller and an Actuator by clicking on the "Add" Buttons. Change the Sensor type to "**Keyboard**" with the Menu Button and assign the **LEFTARROW** Key to the Sensor. Don't forget to **name the Logic Bricks** by clicking on the name field in the bricks. Wire the Sensor ("LeftArrow") with the Controller ("pass2") and the Controller output with the Actuator ("Left")



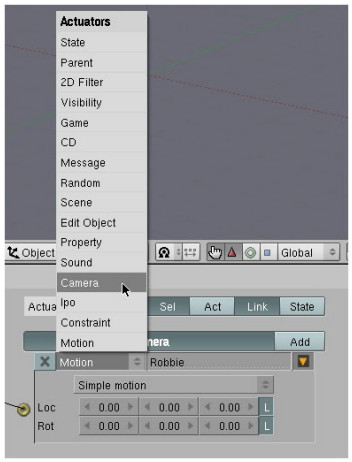
Enter "**3.**0" in the third field (**Z-axis**) in the "**Torque**" row. **Torque is the force that turns the object**. In this case it will turn the actor around its longitudinal axis. Try it in the game engine, the pumpkin will turn left when you press LEFTARROW. **Repeat the steps but change it to turn right**. To do so use RIGHTARROW and enter a torque of "**-3.0**".

It is most likely that we need to adjust or fine tune the controls later in the game, this is a iterative process which need some experience.

Control the Camera:

In this section, we will see how you can set up a camera which follows the actor, trying to mimic a real cameraman.

Locate the camera and select it with RMB.



Ensure that the Logic Buttons (F4) are still open. Now **add a Sensor, Controller and an Actuator** as you learned above. **Wire** the Logic Bricks and Change the Actuator into a **Camera Actuator**. The Camera Actuator will follow the Object in a flexible way which gives smooth motions.

Click the "**OB:**" field in the Camera Actuator and enter the name of the pumpkin object, here "**Pump**". The camera will follow this object. Click and hold the "**Height**:" field with the LMB and move the mouse to the right to increase the value to about **2.0**. This is the height the camera will stay at.

The **Min: and Max:** fields determine the minimal and maximum distance the camera will get to the object. I chose "Min: 4.0" and "Max: 7.0". Start the game engine to test the Camera Actuator. Experiment a bit with the values.

Object Animation:

Here we will cover the basics of combining Blenders animation system with the game engine. The animation curves (Ipos) in Blender are fully integrated and give you full control of animations.

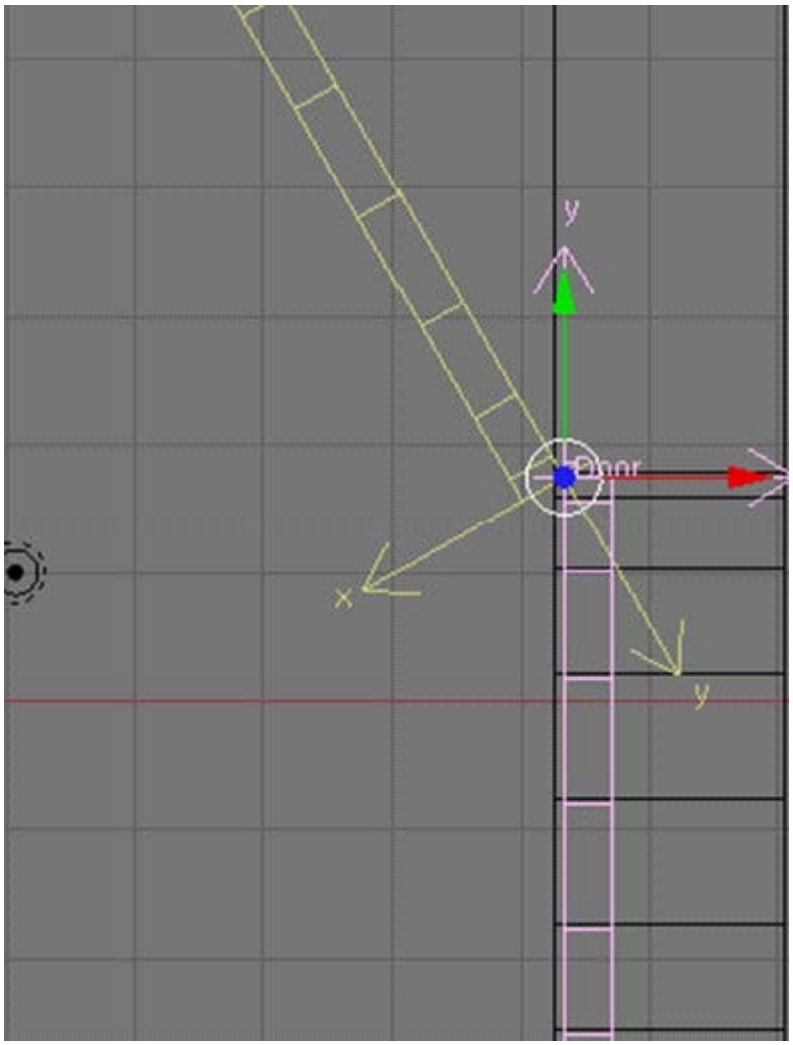
Use the File-menu "**Append or Link**". Browse to Pumpkin/Door.blend, click on **Object**, select all objects with **AKEY**, confirm with ENTER. This will append a wall with a wooden door to the scene. The pumpkin will bump against the walls and the door. The collision detection is handled by the Blender game engine automatically.

Switch the right 3D View to a **Top View** (PAD7) and zoom (PAD+ or PAD-) as needed to see the appended door completely. The door has the name and the axis enabled, so it should be easy visible, for easier access switch the view to a wire frame view (ZKEY). **Select the door with RMB** (it will turn pink).

We will now make a simple key frame animation:

- Ensure that the Frame Slider (the current animation frame) is at **frame 1**.

- **i** KEY, select "**Rot**" from the menu
- Now advance the animation time by going to **frame 61**. With the game engine playing 60 frames per second our **animation will play now 2 seconds**.
- Press **RKEY** (be sure to have your mouse over the top view) and rotate the door **150°** clockwise.
- Now insert a second key by pressing **i** KEY and again choosing "**Rot**"
- Move to **frame 1** and press **ALT-A**, you will see the animation of the door being played back. After 61 frames the animation will continue to frame 250 and then repeat.
- Press **ESC** to stop the playing animation.



With the door still selected switch the Buttons Window to the **Logic Buttons**, by pressing F4. **Add a Sensor, Controller and Actuator**, wire them, name them, change the Sensor to a Keyboard Sensor and change the **Actuator to "Ipo" type**.

Change the type of the Ipo Actuator to "**Ping Pong**" mode using the Menu Button. SHIFT-LMB "**End**" and change the value into "**61**". This way the Ipo Actuator plays the door animation from frame 1 to

61 which opens the door. A new invocation of the Ipo Actuator will then close the door (because of playing it "Ping Pong").

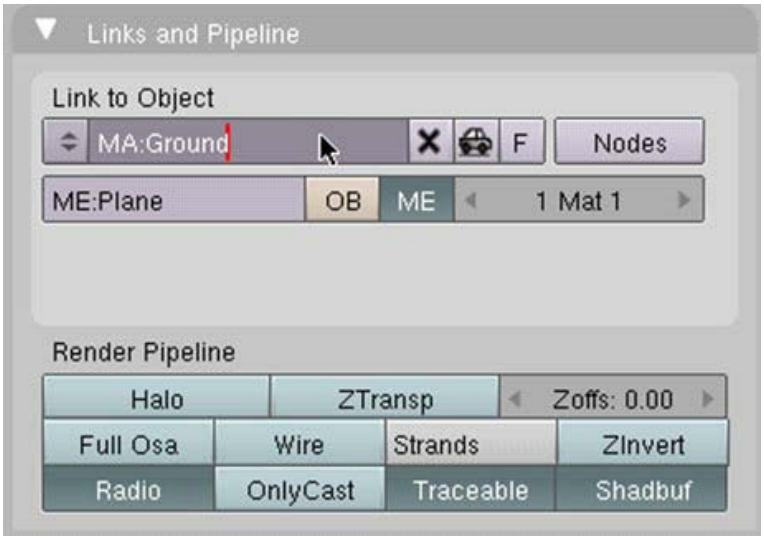
Now go and change the Sensor to the **Keyboard** type and hit **SPACE** at the KEY field. So now the door animation will play every time we press SPACE.

Play the scene (**PKEY**) in the textured view, and the door will now open and close when you press SPACE and can push the actor around if he gets hit by the door.

Refining the Game:

You may have noticed there is currently a problem in the file: The actor can "climb" (nearly takes off like a rocket) the wall because we can jump on every touch of any object even on a wall.

Select the pumpkin and look at the Touch Sensor, the "MA:" field is empty. "**MA:**" stands for a material name. If you fill in a material name here, **the Touch Sensor will only react to objects with this material**. In our scene this would be the ground we created at the beginning.



So now **select the ground plane** and hit **Sift+S > Cursor to selection**. Then Add a new plane over the ground and hit **R > X > 180**. With the second plane selected switch to the **Material Buttons** F5. Locate the "**Links and Pipeline**"-panel and change the name of the material by clicking it with the left mouse button and entering "**Ground**".

Now, select the pumpkin again, switch back to the **Logic Buttons** F4 and enter "Ground" into the "**MA:**" field **of the Touch Sensor**.

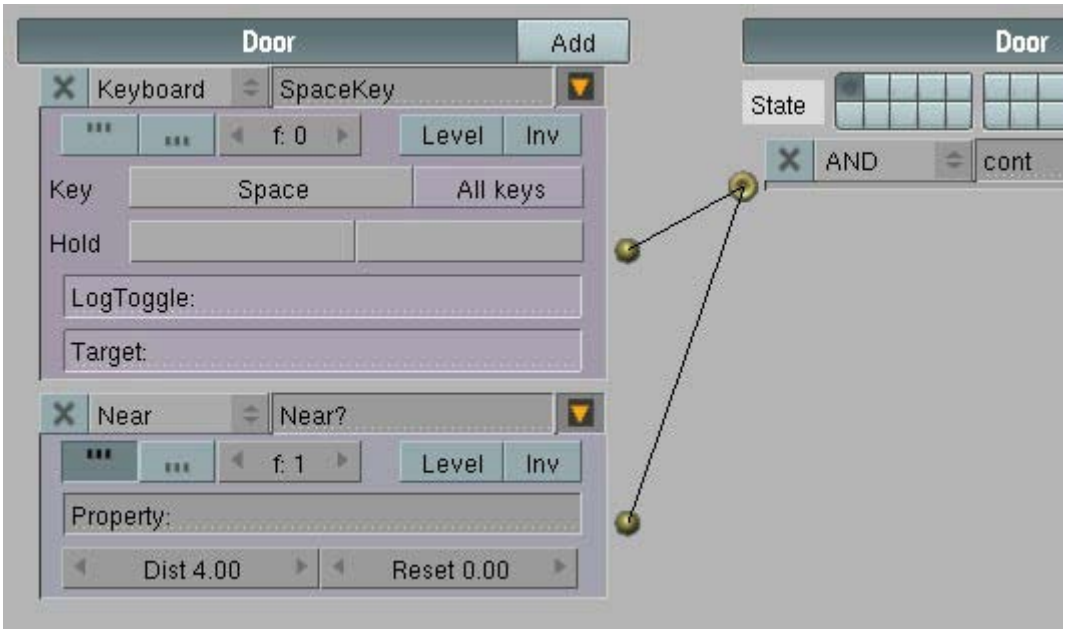
Try again to hop around, now you now cannot climb the wall anymore.

The reason why we add the new plane over the existing ground is to use it as a Platform to add Logic without ruining the Textures of our original Ground Plane.

At this point you may want to place the camera more behind the pumpkin, so that the view during the game is better and also move the light in a way that the pumpkin gets lit better. It is best to do such things in the top view, using **GKEY** to move and **RKEY** to rotate and controlling the effect in the camera view.

One last thing: **it would be nice if the door would only open when the actor is close to it**. The Near Sensor will help us here.

Add a new Sensor to the door and change the Sensor type to "Near". Wire it to the existing AND Controller. The "**Dist:**" field gives the distance at which the Near Sensor starts to react to the actor. By the way, it will react to every actor if we leave the "Property:" field empty. The "Reset:" field is the distance between the Near Sensor and the object where the Near Sensor "forgets" the actor. Try to **make the "Dist:" setting to 4.0**, now you need to come close to the door first but then you can back up before you press SPACE to open the door without the danger of getting hit. **Now the door only open when you press SPACE and the pumpkin is near the door, good for levels with more than one door.**

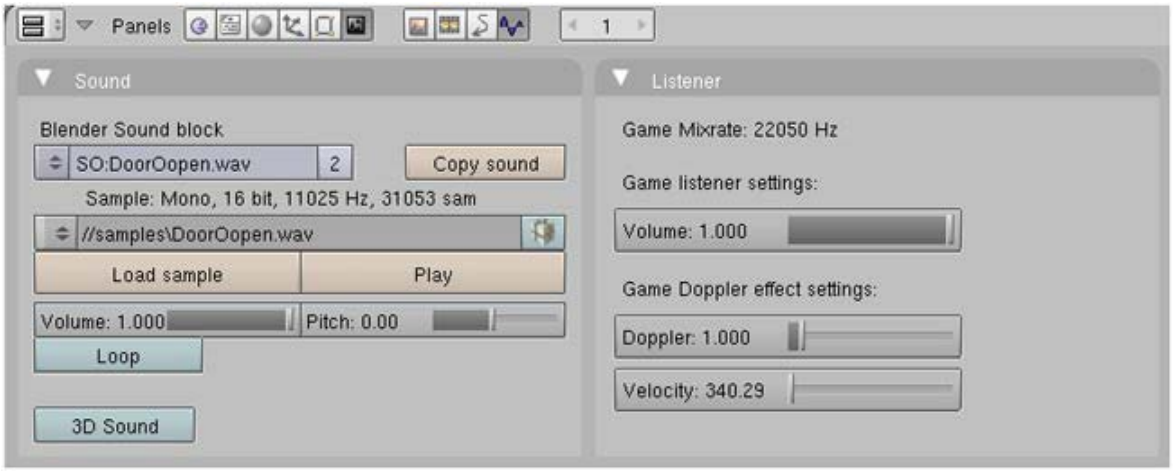


Adding Sound to the Game:

There is no game without sound, so we will see here how to add a sound to an event in Blender.

Locate the **Sound Buttons** icon in the icons of the **Scene Buttons** F10. Click the icon with the left mouse button to switch to the Sound Buttons. Because there is no sound in the scene the "Sound"-panel will be near empty. Use the Menu Button under **"Blender Sound block**" (click and hold) to

choose "**OPEN NEW**". A file window opens, browse and load DoorOpen.wav from the directory Pumpkin/samples/.



Blender is capable of creating 3D sound (Sound located spatial in the scene) and provides many ways to influence the sound. But for the moment we can go with the defaults and don't need to touch any of these buttons. Of course you can play the sound by clicking on the big "Play" button.

Select the door object (RMB) and switch to the **Logic Buttons** with F4. **Add a new Actuator** and change the type to "**Sound**". Wire it to the Controller. Click and hold the solitary Menu Button in the Sound Actuator and **choose the sound file "DoorOpen.wav"** from the pop-up menu. As a last step before you can try it, change the mode "Play Stop" to "**Loop End**" this means the whole sound is played without stopping too early.

Quit the Game:

Another effect we can add to the Game is to quit it in case the Pumpkin falls off the Ground. To do that **add a new Plane** and use GKEY to move it down under the Ground. **SKEY** to Scale it so as to be bigger than the Ground and **RKEY > X > 180** to turn it upside down. This way it will be invisible to the Game Mode.

Then Select the plane > Logic Buttons > Add a line and wire it. Change the Sensor to a **Collision** Sensor and the Actuator to '**Game**' > '**Quit this Game**'. Now hit PKEY to Play and note than when the Pumpkin falls over the Ground the Game will Quit.