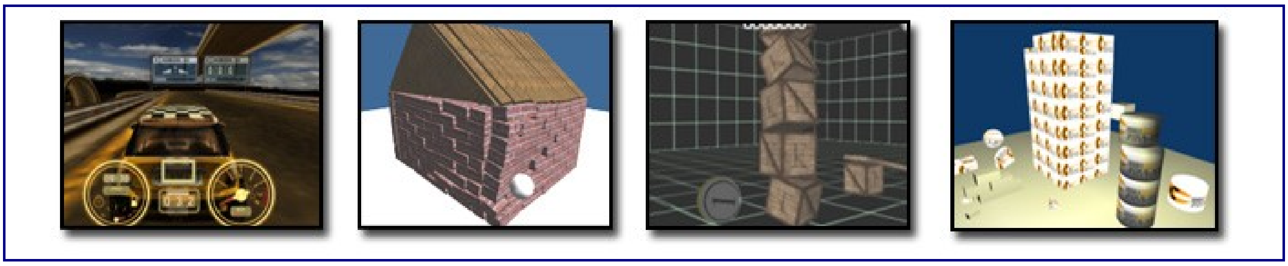
Blender Summer of Documentation: Learning the Blender Game Engine



The format of the tutorial

This tutorial will show the reader how to create a simple 3D game from scratch in Blender. After completing the tutorial, the skills learned will allow readers to extend the game levels.

It assumes no prior knowledge of Blender, so experienced users will be able to skip over certain sections.

Only aspects of Blender that are relevant to making simple games will be touched on. This will cover only a fraction of Blender's overall functionality, and will hopefully encourage readers to use Blender for other purposes, such as creating 3D animations.

The tutorial should take around 2 hours to create a working 3D game from scratch.

Intended Audience

This tutorial is aimed at anyone with an interest in making 3D games, but who may have no prior experience of either Blender or 3D.

It will teach the basics of Blender's Game Engine ("GE") to both new Blender users, and also to existing Blender users who may not yet have enjoyed this aspect of their favourite application.

Blender is a very powerful 3D modeling and animation package, with an excellent and constantly improving built-in Game Engine. It is known as having quite a steep learning curve, although when you get used to the workflow, it is very fast to work with.

What skills will you need to know in advance

This tutorial is aimed at all levels of user, so it doesn't assume any prior knowledge of Blender.

If you are a new user with no experience of Blender, the tutorial will step you through the important aspects of Blender and the GE, so that you will be able to create your own game.

If you are experienced with the Blender UI with regard to the important GE buttons and areas, you might want to quickly scan over some of the initial sections.

Starting a game project?

Before posting your Game project announcement in a forum please read this.

Blender is a great application for all kinds of 3D artwork, be that stills, animations or even... games!

Most of these projects are solo, but once in a while someone or a team of people decides/decide to recruit people to the development team or just want to let us know he/they is/are making a game. If you are working on a game and would like to recruit or just show your project in development, please take the following into account:

1. We frequently get overly excited newbies who post about their project to create a MMORPG (or any other type of game, really, but MMOGs are the most common). In 99% of the cases, the project never goes anywhere and the actual announcement itself would make any professional representative cringe. That's the reason why we are quick to be skeptical about any posted project which does not follow the following guidelines.

2. As mentioned before, make sure you represent yourself in a professional way. Provide us with details on you/your team/your company including current team members and their responsibilities, your website (if you have one; most games only have a website once their game is about to be published) and contact information.

3. Present your game in a good way. That is to say, show us the game concept (including concept art, a plot concept and a game script concept) and the work in progress (models, screen shots, videos showing the game in action) which you already have. Even if you are recruiting, you must be able to prove that your game is not the dream of an overly-excited newbie, so show us more than a dream. Give us a plan.

4. Give us a reason to be interested in your game and/or participating in the creation of it. Tell us what your game is about, and what its major features will be. Tell us who your audience is and what your plans are for when the game nears the end of its production. Using which medium will the game be transported to the client? Will the game be free to play or will you charge a single/monthly fee to play? Tell us about your goals in general. Make us anticipate the finished version of the game.

5. If you are recruiting, tell us what's in it for us. Many people ask for others to join their project, but why should we? We will not make your game, at least not for free.

6. Please use proper grammar, spelling, syntax and punctuation. There is nothing worse than presenting yourself by misspelling a word in every other sentence.

I hope this post will avoid any future flames caused by an inexperienced person beginning a game project he can't handle (yet) and will allow the professional game developers to gain acceptance and support.

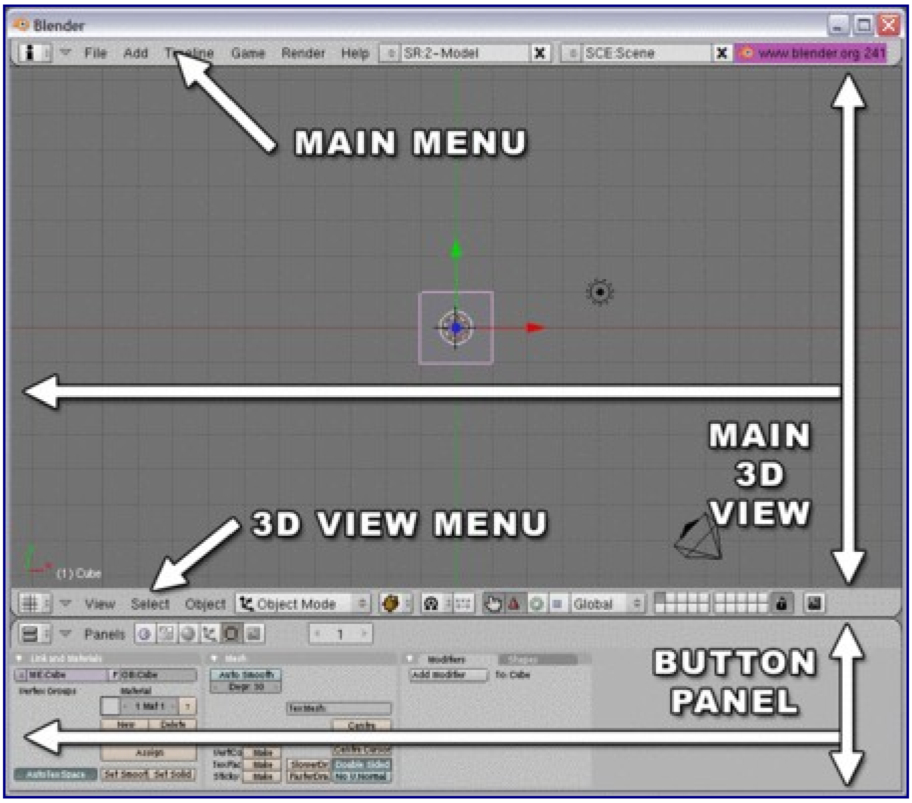
-The above was provided by Peter Charlesworth via the blenderartists forum.

Downloading the correct version of Blender

This tutorial requires the use of Blender 2.42a or newer. If you have an older version of Blender, please upgrade to this version. It can be downloaded here...

Click here to download the latest version of Blender

The basics of the Blender UI, and how to quickly use it



Initial Blender screen setup

It is assumed at this stage that you have installed and downloaded Blender.

When you run Blender for the first time, you will be presented with the screen to the right.

- The main menu panel is located at the top of the screen. The GE has its own *Game* menu option here, which we will have a look at later on.
- The main 3D view is shown as a area with a grid overlaid. It shows the current scene from the top.
- The menu for the 3D view is located below it.
- At the bottom of the screen, you will see the *Buttons* panel. Again, the GE has its own panel F4 (also represented by a purple PacMan style icon) which again we will come to later on.

Taking a closer look at the default scene

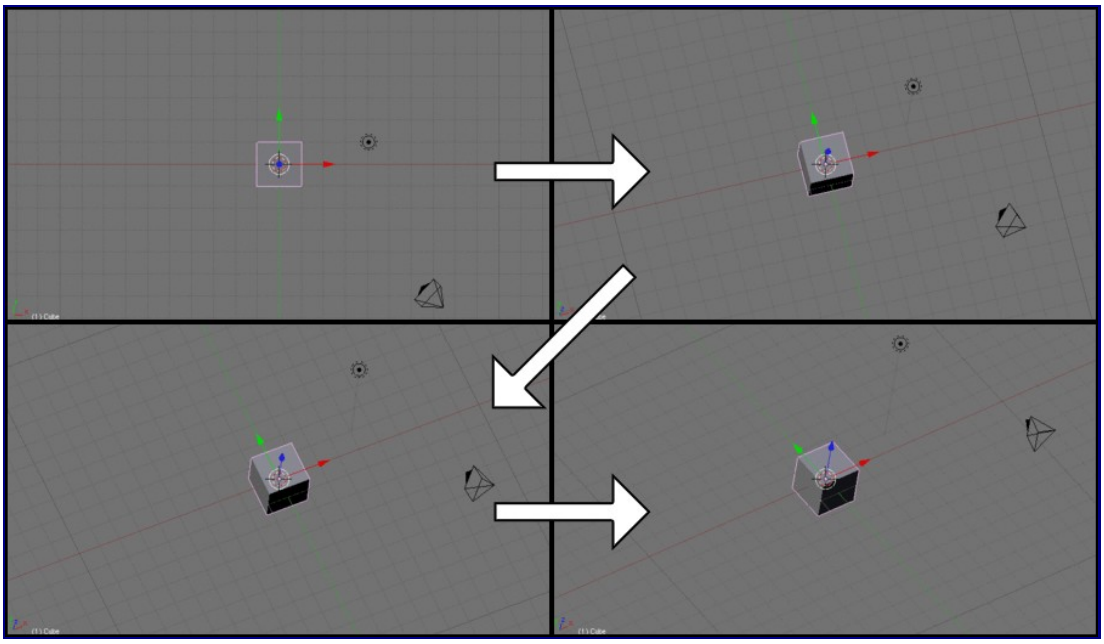
The default scene contains...

- A cube - the square in the middle. This is selected, indicated by both having a pink outline and by having a 3D Transform manipulator (the arrows that you can drag to manipulate the object).
- A light - the small round object, with dashed line.
- And a camera - you can just about see this, clipped to the bottom of the screen.

Interacting with the 3D scene

Hold down the middle mouse button MMB and drag to rotate the scene. If you are on a laptop, or have a mouse with just two mouse buttons, you can hold down Alt LMB to emulate the middle mouse button.

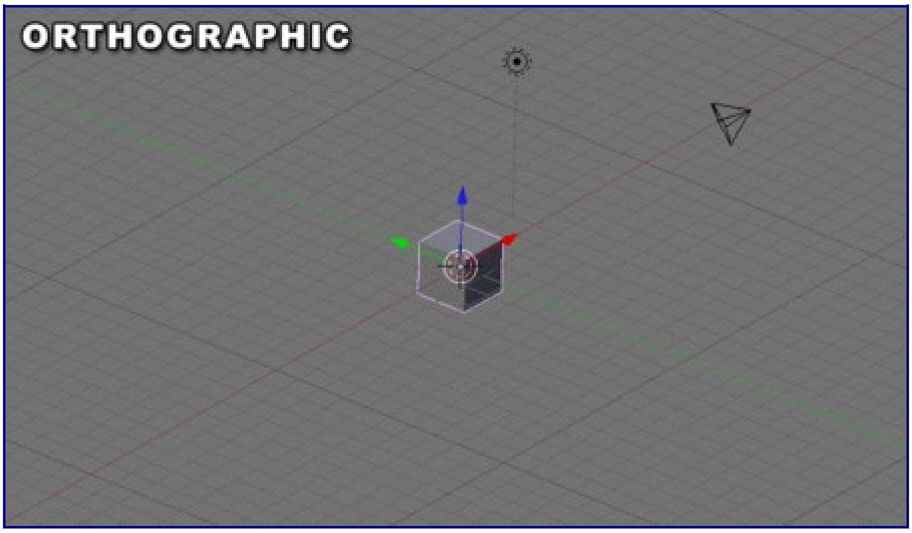
Rotate the view around until you are looking at the cube from a top, left viewport, as shown in the image sequence below.



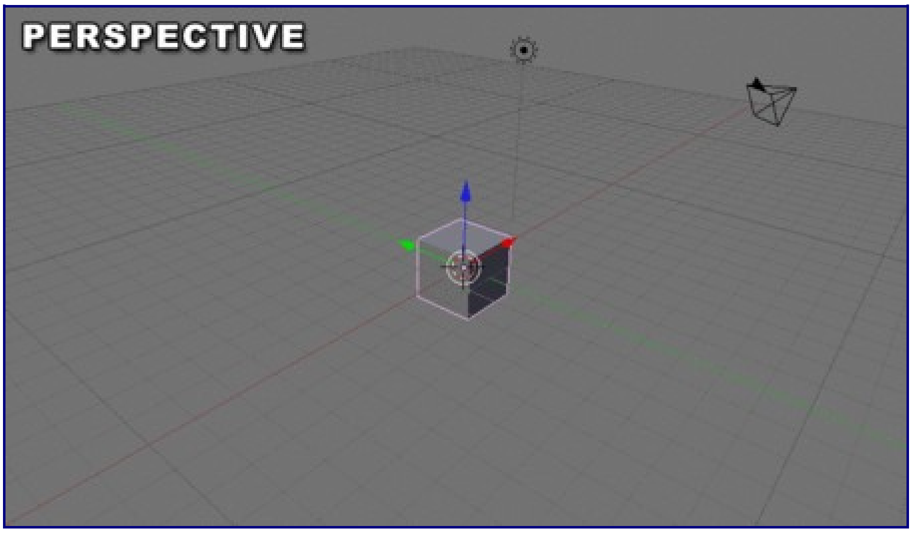
You can also zoom in and out on the 3D scene by scrolling the middle mouse wheel MMB or by holding down Ctrl MMB and moving the mouse up/down.

To pan around the scene, hold down Shift MMB and move the mouse around.

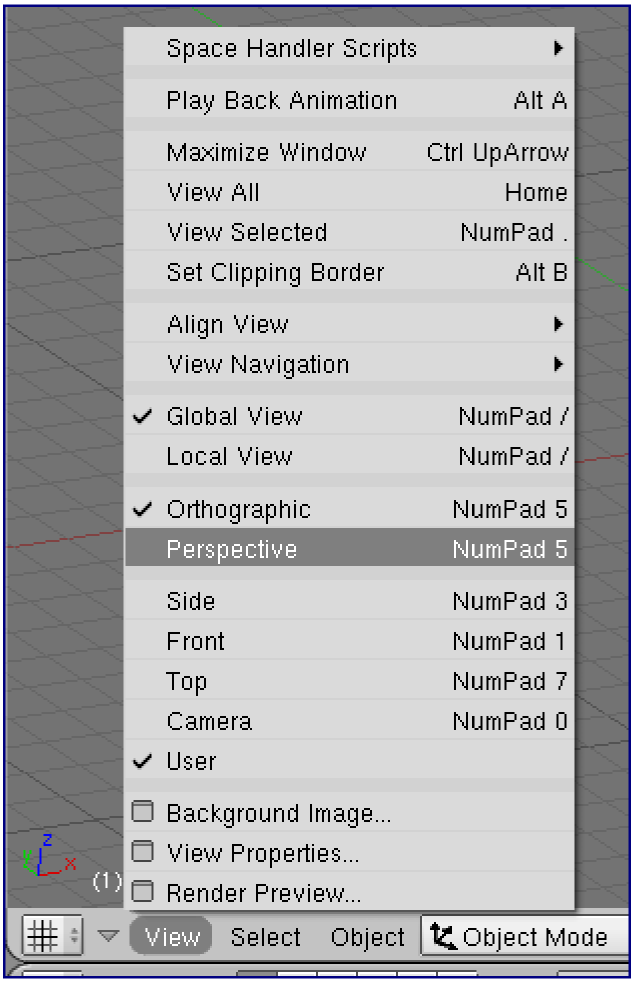
Changing the 3D scene to Perspective mode



This current view is known as an orthographic view. It can be easier to view a scene in perspective mode, which takes into account depth.



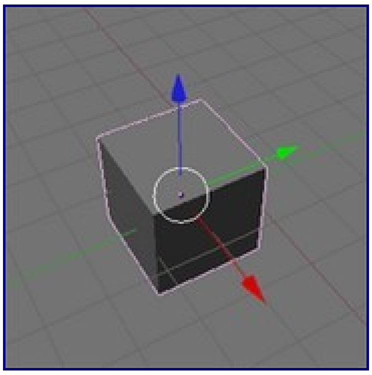
To change into perspective, select *View* -> *Perspective* option from the 3D View menu.



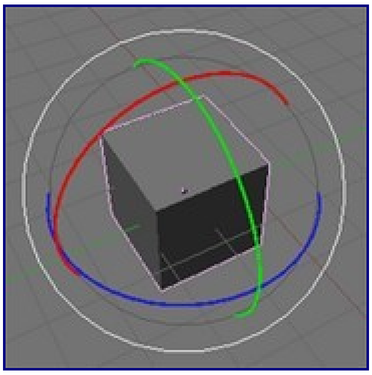
Manipulating objects within the scene

You will be able to move objects around by clicking and dragging the handles on the 3D Transform manipulator. Click and drag the handles on the box to move it around in the scene.

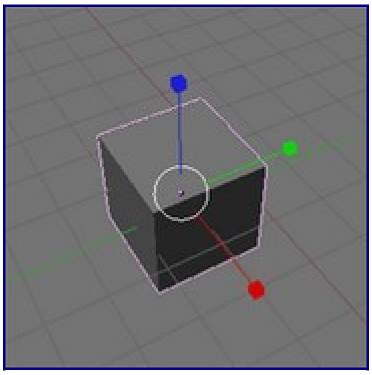
The move, rotate and scale 3D manipulators are shown below, along with a combo 3D manipulator that includes all three.



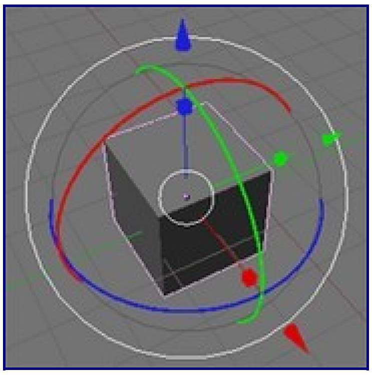
Translate



Rotate

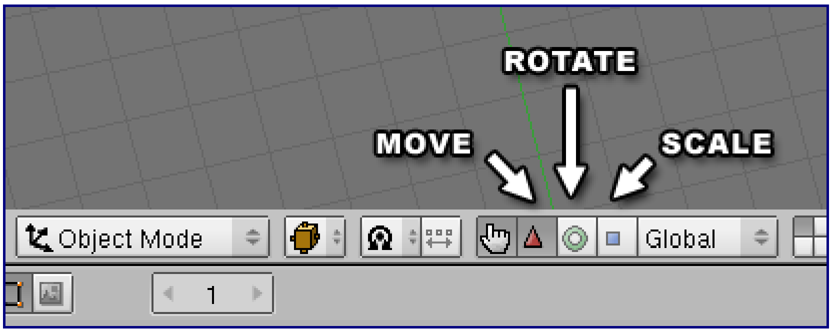


Scale

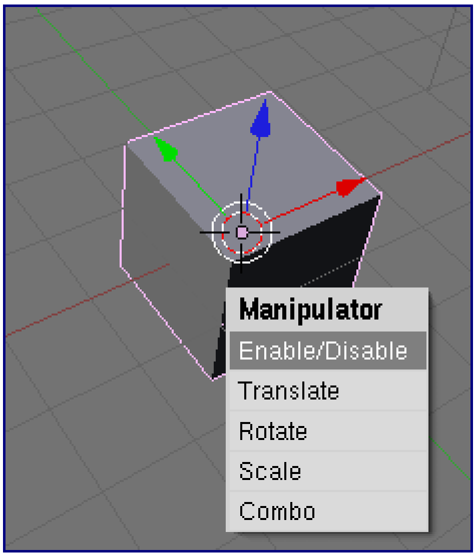


Combo

To change the 3D manipulator type, click on the icons shown below in the 3D View menu.



Select manipulator icons.



Manupulator selection menu.

Another quick way to change the 3D manipulator type is to press Ctrl Space and select *Translate*, *Rotate* and *Scale* from the menu.

For ease of use, this tutorial will focus on using the 3D manipulators to manipulate the objects in the 3D viewport. Other tutorials you will read will explain how to use the Grab, Rotate and Scale hot keys to manipulate objects without using the 3D manipulator.

For more information on the 3D manipulator you can refer to Manual/Manipulators.

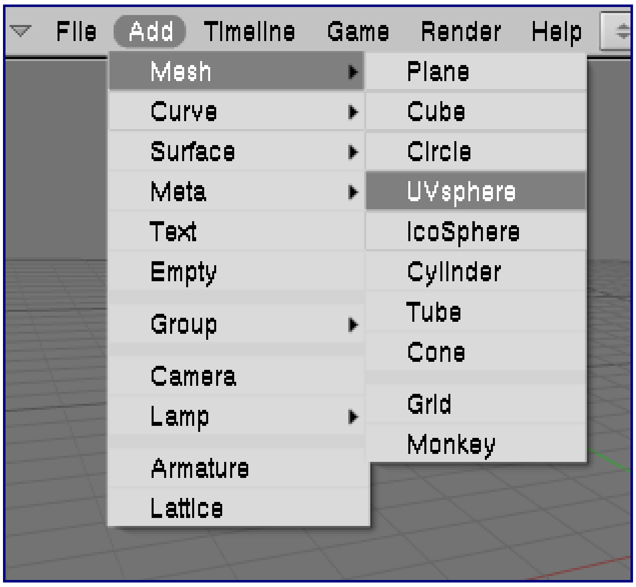
Selecting other objects within the scene

In Blender, clicking the right mouse button RMB over an object will select it.

Left-clicking LMB will position the Blender 3D cursor in the environment. New objects will be added at the location of this 3D cursor.

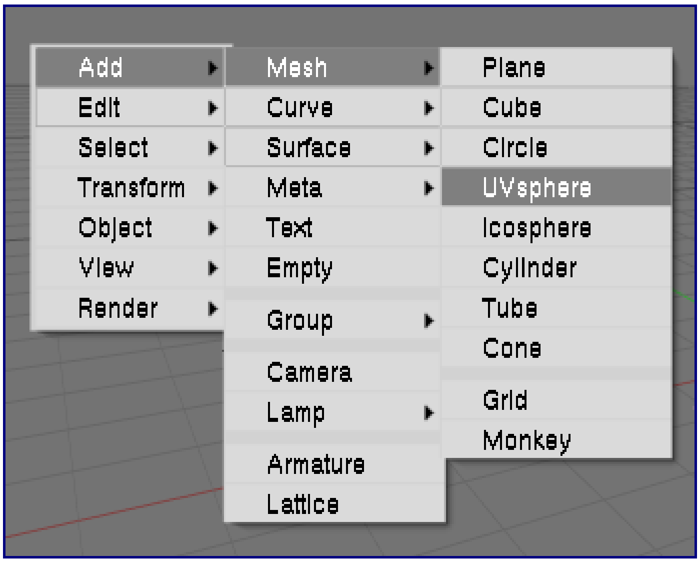
Right-click RMB on the lamp to select it, and then move it around using the 3D manipulator. Right-click RMB on the box again to select it, and move it around again using the 3D manipulator.

Adding and removing objects in the scene



Removing the objects is easy... select an object, and press the Del key.

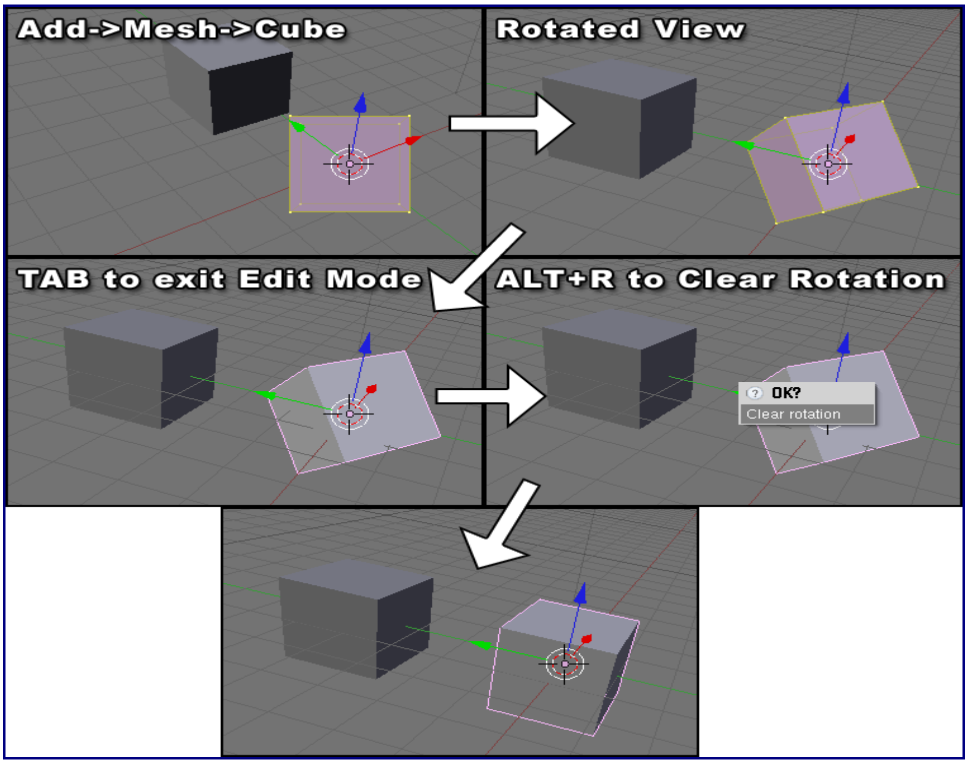
To add an object to the scene, use the *Add* menu item at the top of the screen, and then select the Mesh sub-option.



You can also press the Space key within the 3D view to display a menu with a number of useful functions, including the *Add* menu.

This will then allow you to add a number of simple (or primitive) models to the scene, including planes, cubes, spheres and cylinders. You can also add Suzanne, a model of a monkey head (primitive model / monkey model... it's a Blender developer joke :))

Fixing newly added 3D objects



The image sequence shows a cube newly placed into the scene, and the steps required to have it placed in the scene as expected.

When you add a new object, Blender will do two things.

- The new object will automatically be placed in Edit mode. Extensive editing of object meshes is outside the scope of this article, so for now you will need to press the Tab key to toggle off Edit mode and return to Object mode, every time you add a new object into the scene.
- The new object will initially be rotated as orientated to the current view.

This can be very confusing for new users, so always follow this sequence when you add a new object to the scene. You can press Alt R to clear the rotation on the object.

To summarise, every time you add a new object to the scene as part of this tutorial, immediately press TAB followed by Alt R.

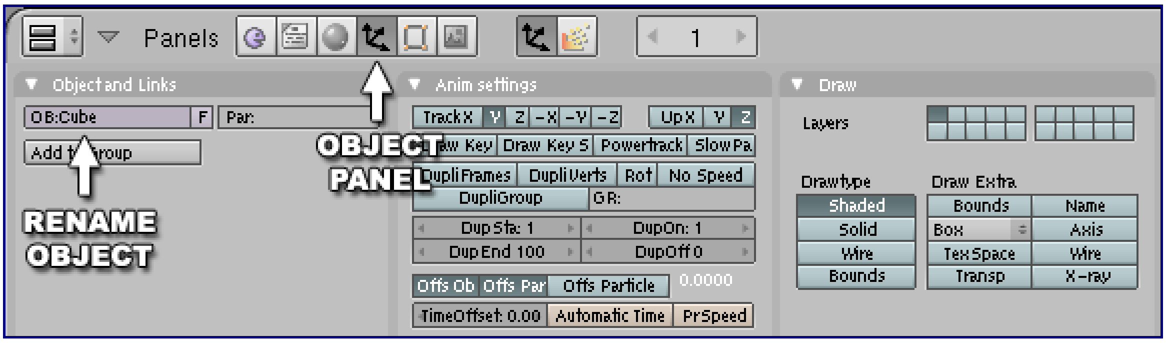
NOTE – In the current CVS, the Auto-Editing and Auto-Rotating of models are optional, so this step will soon be unnecessary.

Renaming objects

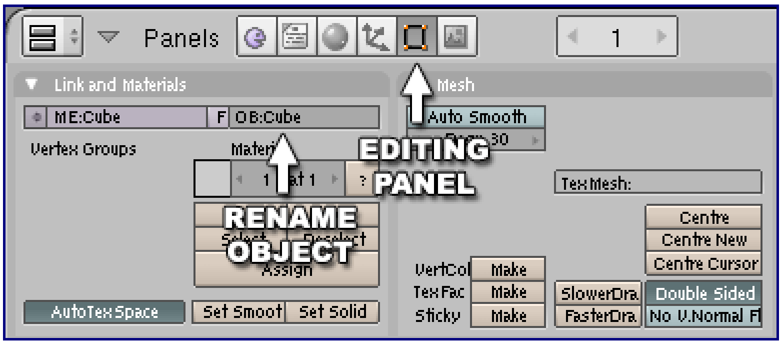
When you add a new object, Blender will assign it a default name (e.g. `Cube`, or `Cube.001` if `Cube` already exists).

It is very good practice to rename your objects using more relevant terms, such as player, crate, pickup, etc... This will make your scene a lot more readable when it gets more complex, as well as making it more readable for other people viewing it.

To rename an object, you can select the *Object* panel, and change the name within the *OB:* area.



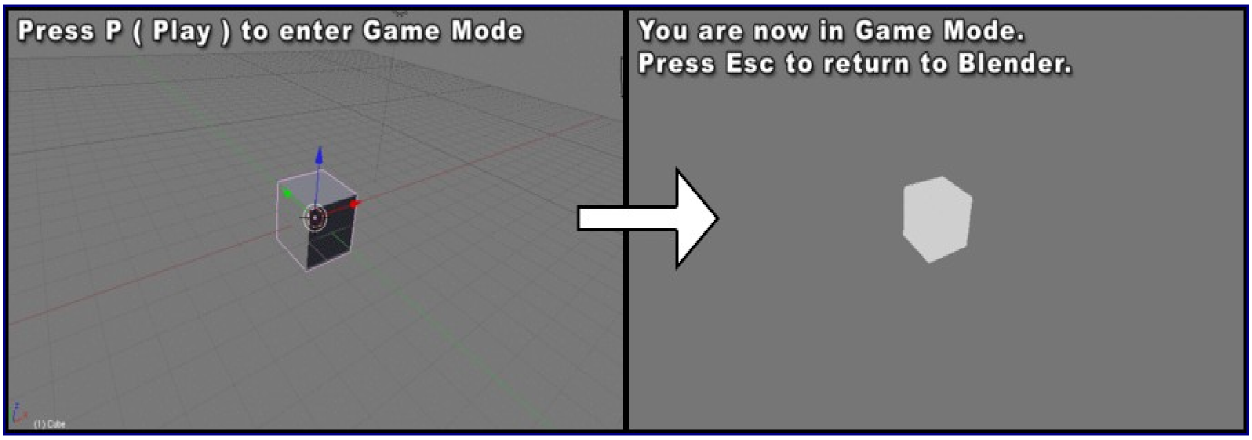
You can also rename the object within the *Editing* panel.



Practice, Practice, Practice

You should spend a bit of time getting used to navigating within the 3D viewport, and also selecting and manipulating objects within the current environment. When you feel that you are comfortable with the basics of rotating around models with a 3D environment, and moving them to new locations, then continue on to the next section of the tutorial below.

Learning the basics of using the Blender Game Engine



Press P (Play) to enter Game Mode

You are now in Game Mode.
Press Esc to return to Blender.

Entering and exiting Game Engine mode using the P key

Now that the basics of Blender have been covered, we will now focus on the GE related features of Blender.

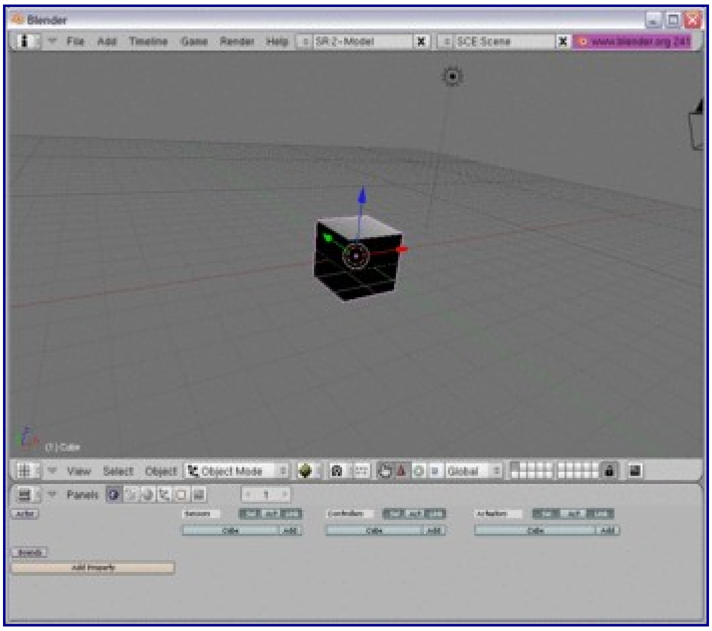
We will start with the most important key in Blender, the one that starts the GE.

Move your mouse cursor over the 3D scene, and press P to Play the Game.

Congratulations, you have just played your first game within Blender!!! How easy was that?

As we haven't told the GE to do anything yet, nothing will occur within the scene. Press Esc to return to Blender.

Setting up the default scene



Before we start, reset the scene in Blender to get back to the defaults.

This can be done be either of the three methods below...

- Selecting the *File* -> *New* menu, and clicking on the *Erase All* option.
- Press Ctrl X, and click on the *Erase All* option
- Quit Blender (Ctrl Q - just Q in 2.42 or older) and restart it.

It's time to see how well you can use Blender, using the information you learnt from the previous section.

Set up the 3D view in Blender to look like in the image.

To do this, you will need to carry out the following tasks...

- Rotate the scene using the middle mouse button MMB
- Change the view into perspective mode using *View* -> *Perspective*
- Add a cube and a lamp (assuming you deleted them previously and that they are not part of the default scene when Blender loads a new file).

A very useful keypress sequence when working with the GE

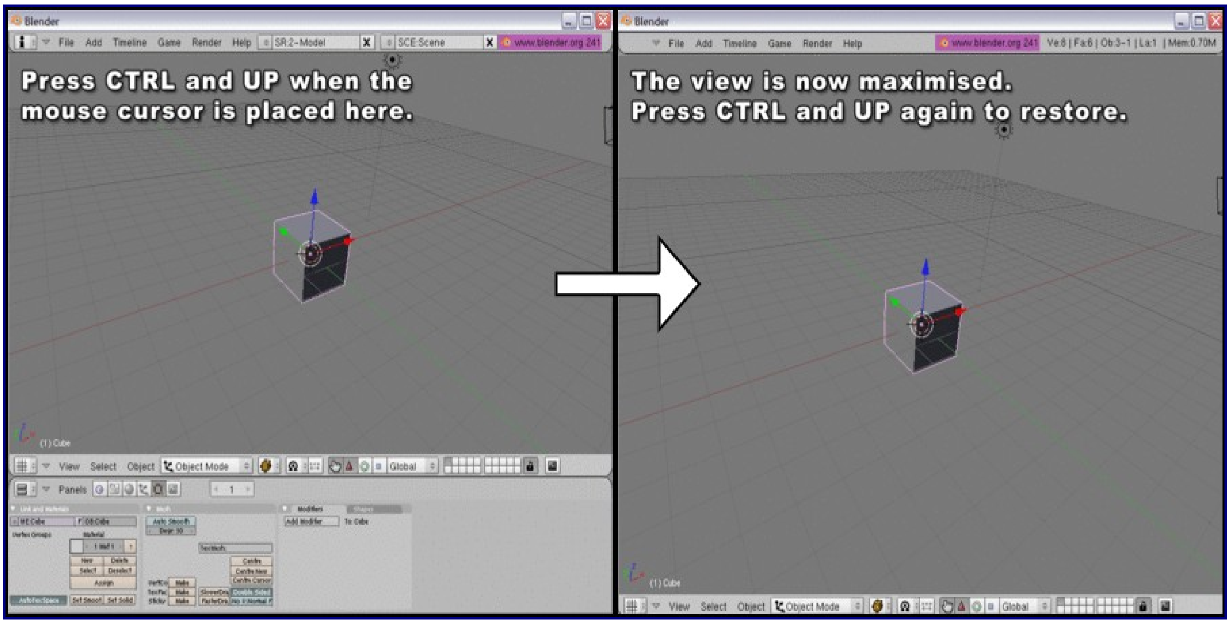
One useful keypress to remember when working with the Blender GE is the one that maximizes the current 3D window.

Move your mouse over the 3D window, and press Ctrl Cursor Up (or Ctrl Cursor Down). This will make the current window scale to the full size of the Blender area. If you press Ctrl Cursor Up again, the window will be restored to its previous size and location.

Go ahead and practice this now:

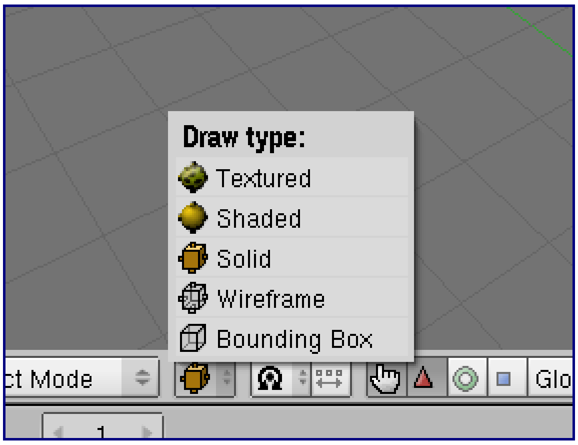
- Maximize the 3D window using Ctrl Cursor Up
- Press P to play the current scene within the GE (nothing will happen)
- Press ESC to return to modelling mode
- Restore the 3D window to it's original size again using Ctrl Cursor Up.

The image sequence below shows the effects of using the Maximize/Restore keypresses within the default Blender screen setup.


Press CTRL and UP when the mouse cursor is placed here.
The view is now maximised. Press CTRL and UP again to restore.

Maximising the Blender panels using Ctrl Cursor Up

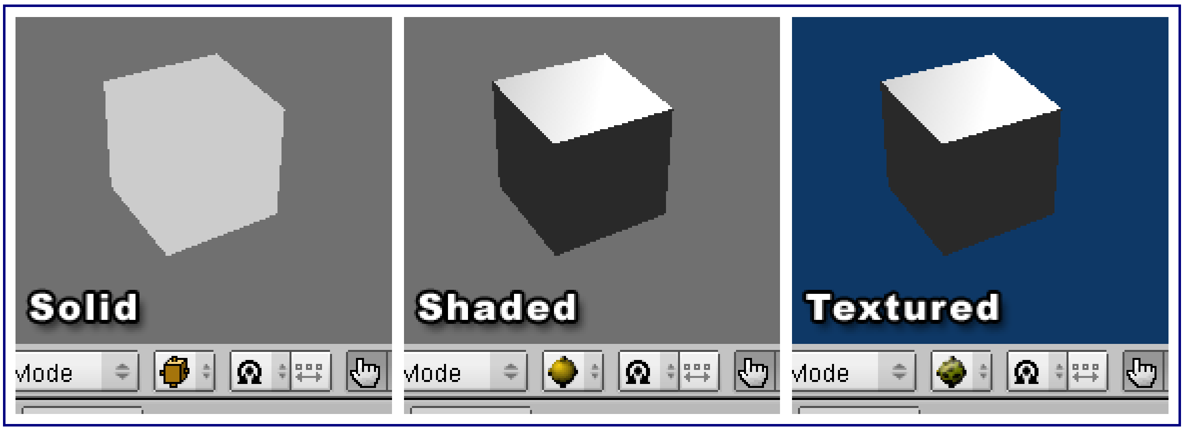
Choosing the correct Shading Mode (or Draw Type) for the GE



Enter GE mode again by moving over the 3D panel and press P. You will notice that the scene within the GE appears flat. Press Esc to return to Blender.

Blender has a number of draw types for the viewport that are useful for different tasks. The panel to change the current shading mode is shown below...

The image below show how the basic scene looks in the GE, in the various Shading Modes



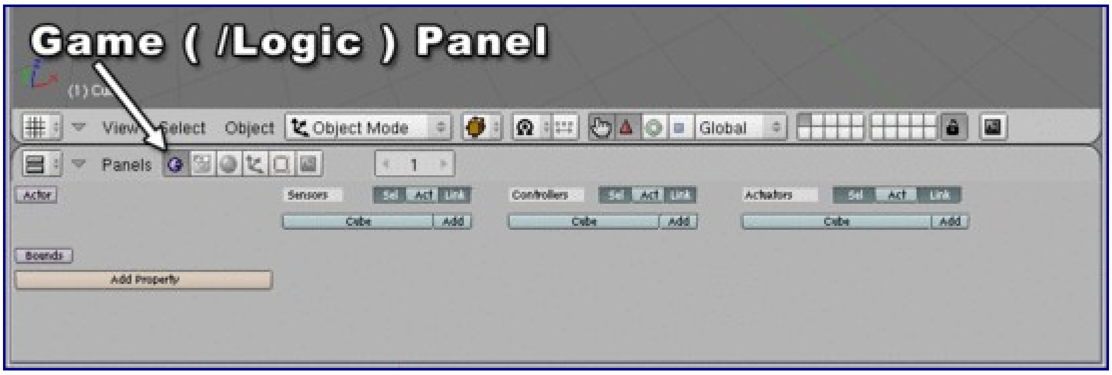
- Solid mode doesn't take into account the lights in the scene
- Shading mode takes into account the lights in the scene
- Textured mode takes into account the lights in the scene, and also shows any textures live in the viewport. This will be as close to the actual in-game view as you can get, and should always be selected when you start a new GE project.

The best draw type for the GE is Textured. Select this Textured draw type from the list, and press P again. You will notice that the lighting affects the environment within the GE now, making it look more realistic. Always remember to set this option to Textured if you play your scene in the GE and it appears flat.

The Main Game Logic Panel

Below the 3D window, you will see the panel that contains many different buttons for controlling different aspects of Blender.

You can view the panel related to the GE by clicking on the purple Pacman-like icon , just like below or by pressing F4.



The Logic panel.

This panel is where you will control what will happen within your game.

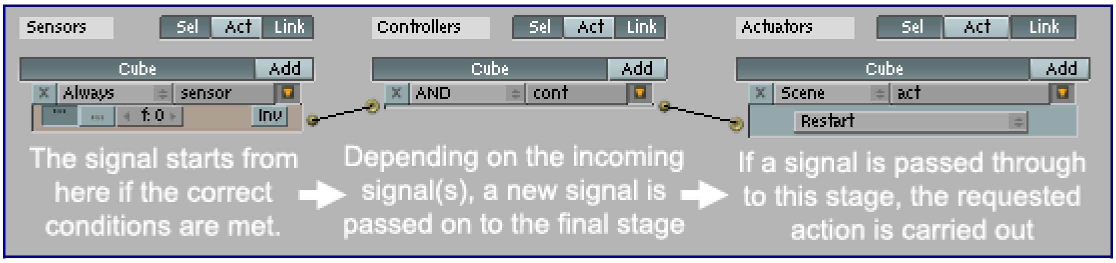
Blender uses a visual click-and-drag system to create basic game interactions. This allows the GE to be used by 3D artists who may not have access to a coder. Blender also has a programming language, Python, which can be used to create more complex game interactions.

For the purposes of this tutorial, we will focus on the visual system for creating games. When you have grasped the basics of using the Blender GE, you can then follow more advanced tutorials showing how to implement Python scripting to create more complex games.

Visually controlling the GE - Sensors, Actuators, Controller Logic Blocks

The GE system uses Logic Blocks as a visual way to set up interactions within the game. These logic blocks can be connected together visually to allow for complex game actions to take place.

There are three different types of Logic Blocks - *Sensors*, *Controllers* and *Actuators* - each with a number of different sub-types.



The signal starts from here if the correct conditions are met.

Depending on the incoming signal(s), a new signal is passed on to the final stage

If a signal is passed through to this stage, the requested action is carried out

Sensors

A *Sensor* will detect some form of input. This input could be anything from a keypress, a joystick button or a timer that triggers every single screen update (or frame) of the game.

Controllers

Controllers are used to link *Sensors* to *Actuators*.

They allow for some more complex control over how sensor and actuators interact with each other.

Actuators

An *Actuator* will actually carry out an action within the game. This can include moving an object within a scene, playing an animation, or playing a sound effect.

Setting up a basic Sensor, Controller, Actuator Logic Block sequence.

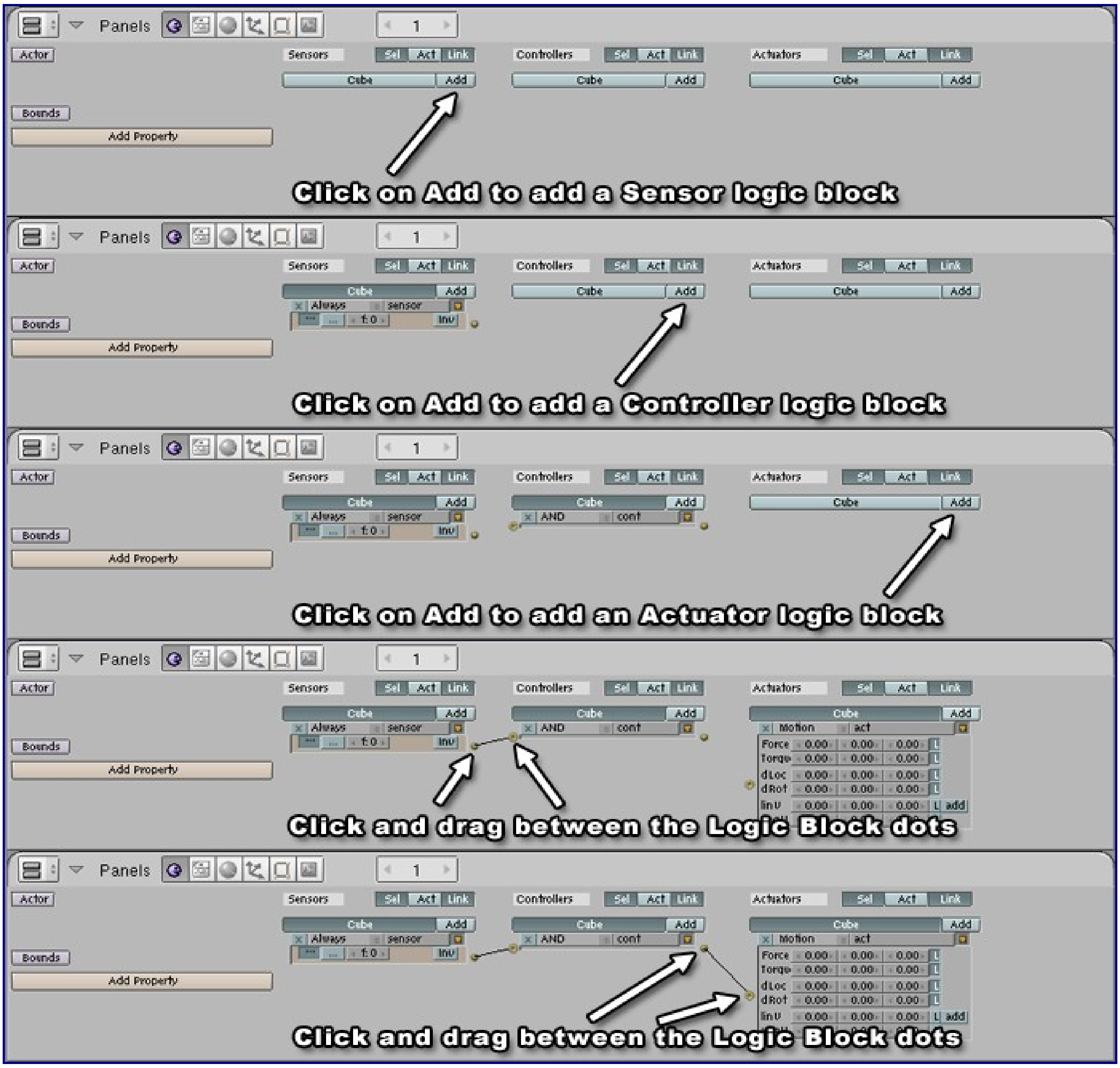
We will now set up a very basic system within the game panel by adding and connecting a sensor, controller and actuator together.

GE Panel: Make sure the Game Logic panel is visible (click on the purple pacman in the Buttons window), and re-select the cube within the 3D scene

Below each of the 3 main sections, you will see the selected object's name, and an *Add* button. Click this *Add* button once for each of the 3 sections: *Sensor*, *Controller* and *Actuator*.

We will now connect this system together. Click and drag from the socket (small circle) at the end of the first sensor to the socket at the start of the controller. Then click and drag from the socket at the end of the controller to the socket at the start of the actuator.

The image sequence below shows the steps involved in setting up a simple *Sensor*, *Controller*, *Actuator* chain and connecting them together.



Click on Add to add a Sensor logic block

Click on Add to add a Controller logic block

Click on Add to add an Actuator logic block

Click and drag between the Logic Block dots

Click and drag between the Logic Block dots

Press P to now play the game. You will notice that, although we have added some control to the GE, nothing seems to happen again. This will be explained in the next section. Press Esc to return to Blender.

Breaking down the events in the GE system

We will now look into what is happening with our newly created GE system.

The sensor is an *Always* timer. This will send out a signal every single frame, so the linked controller will be activated continuously thoughout the duration of the game.

The controller is an *AND*. When it has just one active sensor input (like in the current case), it will automatically call the connected *Actuator*.

The *Actuator* controls the Motion aspects of the selected object.

Because of the *Always* sensor connected to the controller, this actuator will be called every single frame.

If we press P now, it is being called every single frame, but because all of the values are set to zero in the *Motion* actuator, the object will not move within the GE. Press Esc again to return to Blender.

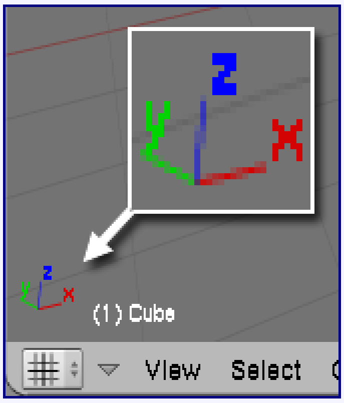
Making the default cube move, using no physics

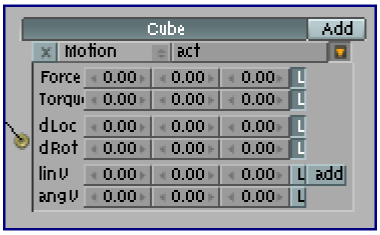
We will initially use direct manipulation to move the cube within the scene. Later on, we will set up a similar scene, but will use physics to move the cube around the environment. By using the built-in physics engine (called Bullet), more complex scene interactions such as collisions and gravity will be handled automatically.

Have a look at the Motion actuator, especially the 3 numerical boxes beside the *dLoc* label. Each of the 3 boxes in this *dLoc* area can be used to specify a change to the location of the object along the X, Y or Z axis specifically.

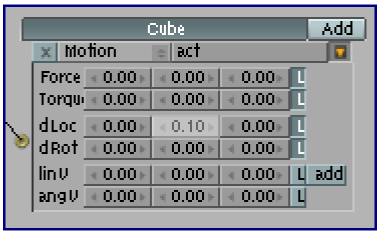
Keeping Track Of Directions

To keep track of what direction X, Y and Z are in, keep an eye on the visual axis in the bottom left of the 3D view.




Cube Add
Motion act
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Change the middle *dLoc* numeric value (Y axis, or forward) to be 0.1.

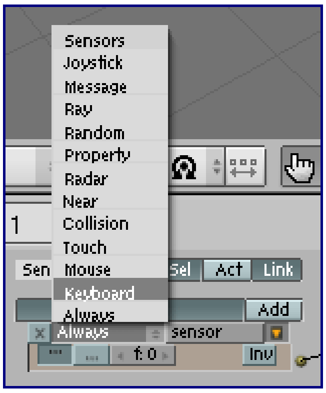

Cube Add
Motion act
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Now press P. You will notice that the cube continually moves along the Y axis. Press Esc to return to Blender.

Press P again and you will notice that the exact same sequence of events in the GE occurs again. Press Esc to return to Blender again.

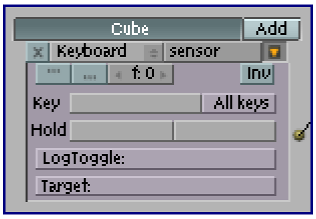
To recap on what is happening in the GE - the timer (Sensor) sends out a signal every frame to the Cube's Controller. This then sends the signal through to the Cube's Actuator, which will then call the Motion object, which moves the cube object 0.1 units in the Y axis. As it is called every single frame, it looks like the cube is continually moving. If you let this game run, the cube will eventually glide off to infinity. This is about as fun as watching grass grow, so read on!

Controlling the cube via the arrow keys



In the example above, the timer is responsible for moving the cube forward. We will change this now, so that it is controlled using a keyboard press.

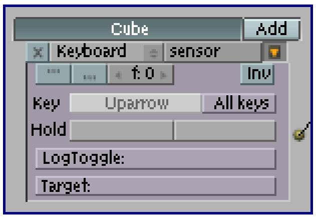
On the *Always* sensor, click on the <> beside the *Always* label. This will bring up a list of the available sensor types.



Select *Keyboard* from the list. The *Sensor* will now change to show the *Keyboard* options.

This new sensor panel will allow us to choose a key that needs to be pressed before a signal is sent to the controller, which in turn will pass it on to the actuator.

Click on the button area beside the *Key* label, and when it changes to say *press a key*, press the ↑ key.


Key Uparrow

Press P to play the game again. The cube will not automatically move now. Press the ↑ key, and it will start to move forwards. When you stop pressing the ↑ key, the cube will stop moving. Press ↑ again to move the cube again. Press Esc to return to Blender.

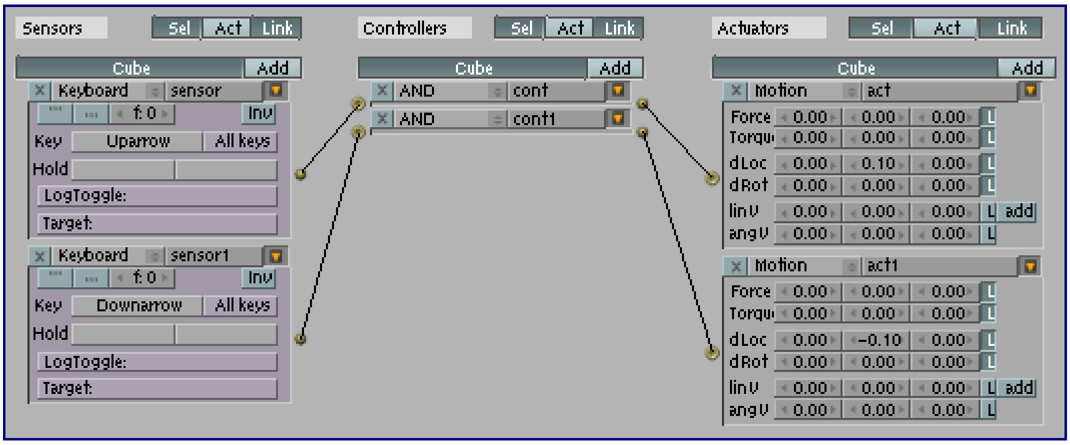
Adding in additional keyboard controls

We will now add in the ability to move the cube backwards, as well as rotating it, to allow you to move it around the 3D environment.

To add the ability to move backwards, we will add in a new (but nearly identical) set of GE logic bricks. Click the Add button again on the Sensor, Controller and Actuator areas, to create 3 new logic bricks on the GE panel. Connect these up as before.

NOTE - You might want to use the Ctrl ↑ key to maximise and restore the game logic panel when working in it, as it will become cluttered very quickly.

With the newly added sensor, change it to keyboard type, and set it to use the ↓ arrow key. In the newly added *Motion* actuator, change the Y value (2. column) of the *dLoc* area to be -0.1.



Sensors — Sel Act Link
Cube — Add
Keyboard — sensor — f: 0 — Inv
Key: Uparrow — All keys
Hold
LogToggle:
Target:
Keyboard — sensor1 — f: 0 — Inv
Key: Downarrow — All keys
Hold
LogToggle:
Target:

Controllers — Sel Act Link
Cube — Add
AND — cont
AND — cont1

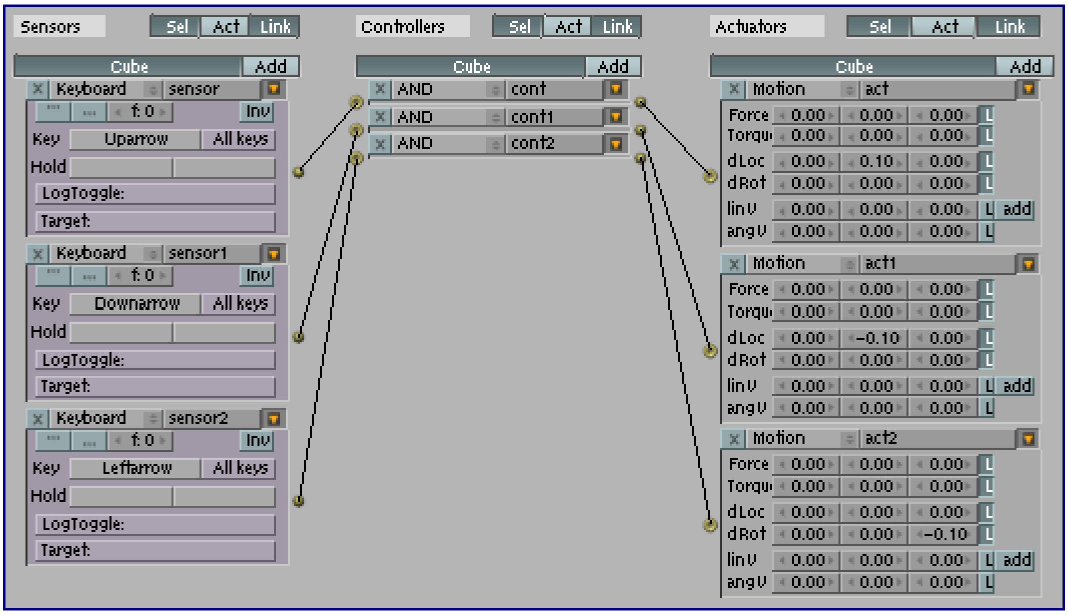
Actuators — Sel Act Link
Cube — Add
Motion — act
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L
Motion — act1
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 -0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Press P to play the game. If the logic blocks are set up correctly, you should now be able to move the cube forward and backward just by using the ↑ and ↓ arrow keys.

To complete this "no physics" part of the tutorial, we will add in the ability to rotate the object as well, so that you will be able to "drive" your model around the 3D environment.

Add in 3 new logic blocks as before, connect them up, and change the sensor type to keyboard. Set the keyboard entry to use the ← arrow key.

Now, in the Motion Actuator, we will set it up to rotate the cube. To do this, we will rotate the object around its Z axis. In the *dRot* area, change the 3rd number (*Z*) to be -0.1.



Sensors — Sel Act Link
Cube — Add
Keyboard — sensor — f: 0 — Inv
Key: Uparrow — All keys
Hold
LogToggle:
Target:
Keyboard — sensor1 — f: 0 — Inv
Key: Downarrow — All keys
Hold
LogToggle:
Target:
Keyboard — sensor2 — f: 0 — Inv
Key: Leftarrow — All keys
Hold
LogToggle:
Target:

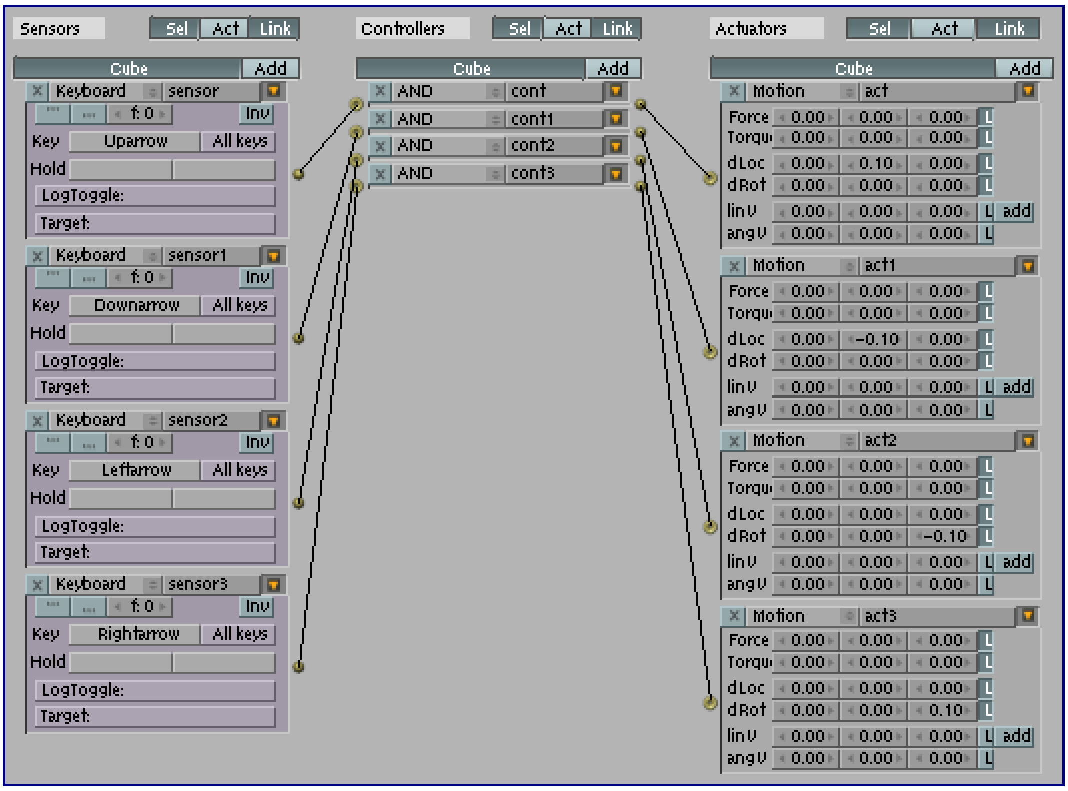
Controllers — Sel Act Link
Cube — Add
AND — cont
AND — cont1
AND — cont2

Actuators — Sel Act Link
Cube — Add
Motion — act
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L
Motion — act1
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 -0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L
Motion — act2
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 -0.10 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Press P to play the game. When you click the ← key, the cube will rotate. When you press the ↑ arrow key, the cube will move forward in that direction. Press Esc to return to Blender, and we will add in the ability to rotate the other way.

As before, add in 3 new logic blocks as before, connect them up, and change the sensor type to keyboard.

Set the keyboard entry to use the → arrow key, and the Z value (3. column) in the *dRot* section in the *Motion* actuator to be 0.1.



```
Keyboard  sensor
f: 0  Inv
Key  Uparrow  All keys
Hold
LogToggle:
Target:

Keyboard  sensor1
f: 0  Inv
Key  Downarrow  All keys
Hold
LogToggle:
Target:

Keyboard  sensor2
f: 0  Inv
Key  Leftarrow  All keys
Hold
LogToggle:
Target:

Keyboard  sensor3
f: 0  Inv
Key  Rightarrow  All keys
Hold
LogToggle:
Target:

AND  cont
AND  cont1
AND  cont2
AND  cont3

Motion  act
Force  0.00  0.00  0.00  L
Torqu  0.00  0.00  0.00  L
dLoc   0.00  0.10  0.00  L
dRot   0.00  0.00  0.00  L
linV   0.00  0.00  0.00  L  add
angV   0.00  0.00  0.00  L

Motion  act1
Force  0.00  0.00   0.00  L
Torqu  0.00  0.00   0.00  L
dLoc   0.00  -0.10  0.00  L
dRot   0.00  0.00   0.00  L
linV   0.00  0.00   0.00  L  add
angV   0.00  0.00   0.00  L

Motion  act2
Force  0.00  0.00  0.00   L
Torqu  0.00  0.00  0.00   L
dLoc   0.00  0.00  0.00   L
dRot   0.00  0.00  -0.10  L
linV   0.00  0.00  0.00   L  add
angV   0.00  0.00  0.00   L

Motion  act3
Force  0.00  0.00  0.00  L
Torqu  0.00  0.00  0.00  L
dLoc   0.00  0.00  0.00  L
dRot   0.00  0.00  0.10  L
linV   0.00  0.00  0.00  L  add
angV   0.00  0.00  0.00  L
```

Press P again to run the game. You can now "drive" the cube around the 3D environment using the arrow keys. Note that you can add this game logic to any model in Blender (no matter what shape or size it is), and it will move around just like the cube. Press Esc to return to Blender.

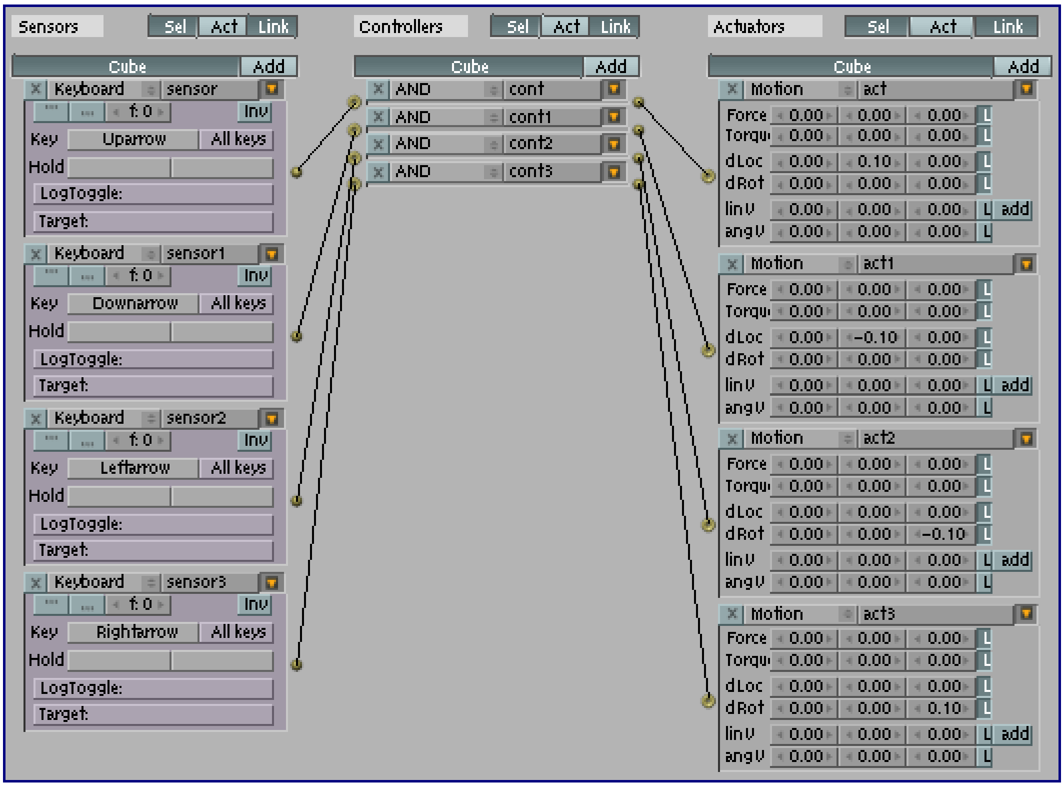
Some GE house keeping

We will now name some of the sensors and minimise them, to keep the game panel as uncluttered and readable as possible. This won't affect what the actual logic does, but it will help keep the game logic area more manageable.

Press Ctrl ↑ in the game panel to make it full-screen.

In the first sensor panel, beside where it says Keyboard, you will see an area where the text *Sensor* appears. This can be changed to something more descriptive. Change this text to "up key". Also, press the arrow button beside this to minimise the *Sensor* panel.

Repeat this for the other keys. Also, do the same in the Actuators, giving them a descriptive title (for example move forward, turn left, etc...).



Sensors Sel Act Link

Cube Add

Keyboard sensor f: 0 Inv
Key Uparrow All keys
Hold
LogToggle:
Target:

Keyboard sensor1 f: 0 Inv
Key Downarrow All keys
Hold
LogToggle:
Target:

Keyboard sensor2 f: 0 Inv
Key Leftarrow All keys
Hold
LogToggle:
Target:

Keyboard sensor3 f: 0 Inv
Key Rightarrow All keys
Hold
LogToggle:
Target:

Controllers Sel Act Link

Cube Add

AND cont
AND cont1
AND cont2
AND cont3

Actuators Sel Act Link

Cube Add

Motion act
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Motion act1
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 -0.10 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

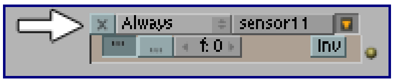
Motion act2
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 -0.10 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Motion act3
Force 0.00 0.00 0.00 L
Torque 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 0.10 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Press Ctrl ↑ again to restore the game panel to its original size.

As you can see, having the various blocks named make it a lot easier to figure out what is going on.

Removing logic blocks and connections



Always sensor11 f: 0 Inv

Remove a logic block.

To remove logic blocks, press the X button at the top left corner of the logic blocks.

To remove the connections between logic blocks, move the mouse over the line connecting the blocks and press the Del key.

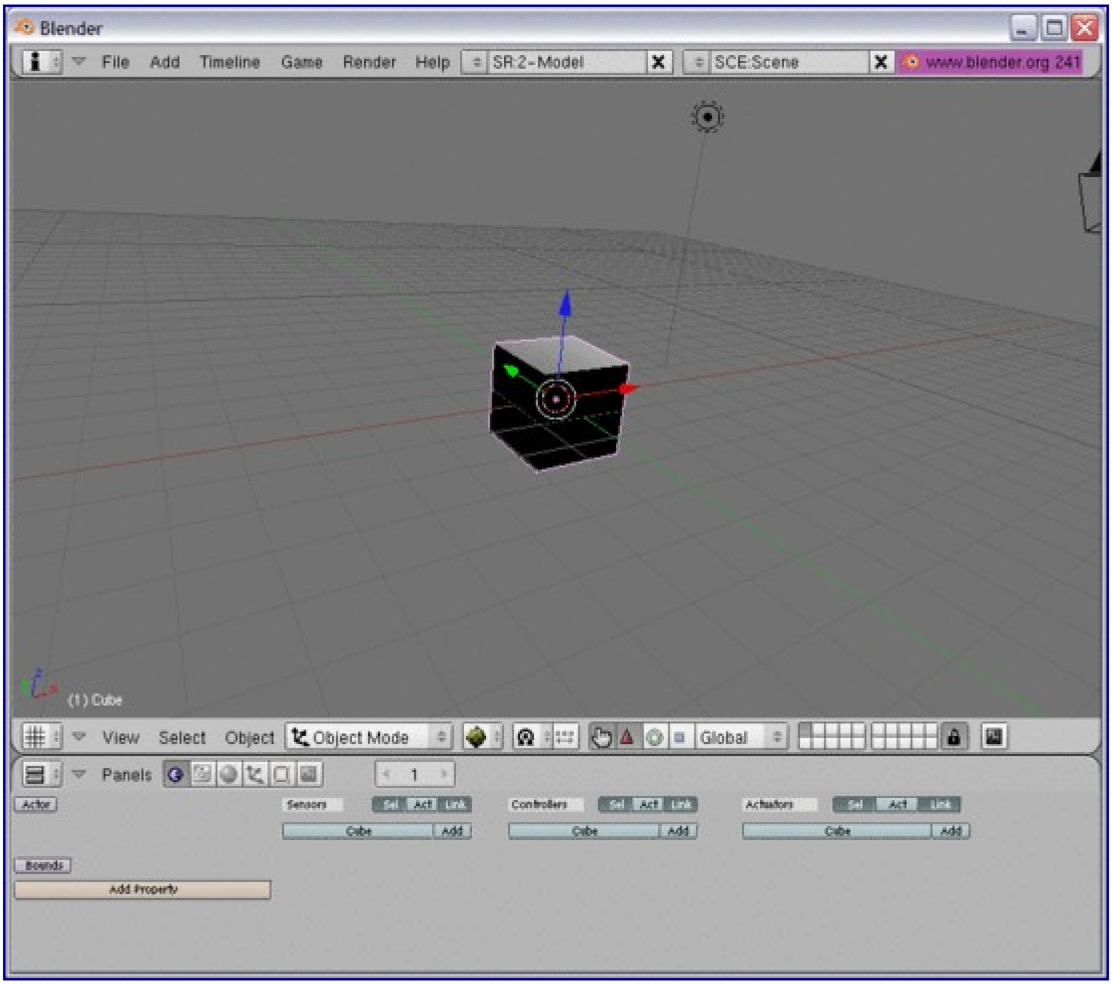
Making the default cube move, using physics

One of the most powerful features of the Blender GE is the built-in physics engine (Bullet).

By using forces to move an object around the scene, the physics system will automatically handle complex interactions such as resolving collisions with other objects in the scene. For many game types, using Physics will solve more complex issues, but it does require slightly more work to set it up.

We will now create a basic game from scratch using physics. The scene will consist of a sphere, which we will move around the scene using physical forces.

Before we start, reset the scene in Blender to get back to the defaults, and set up the 3D view in Blender to look like the image below.



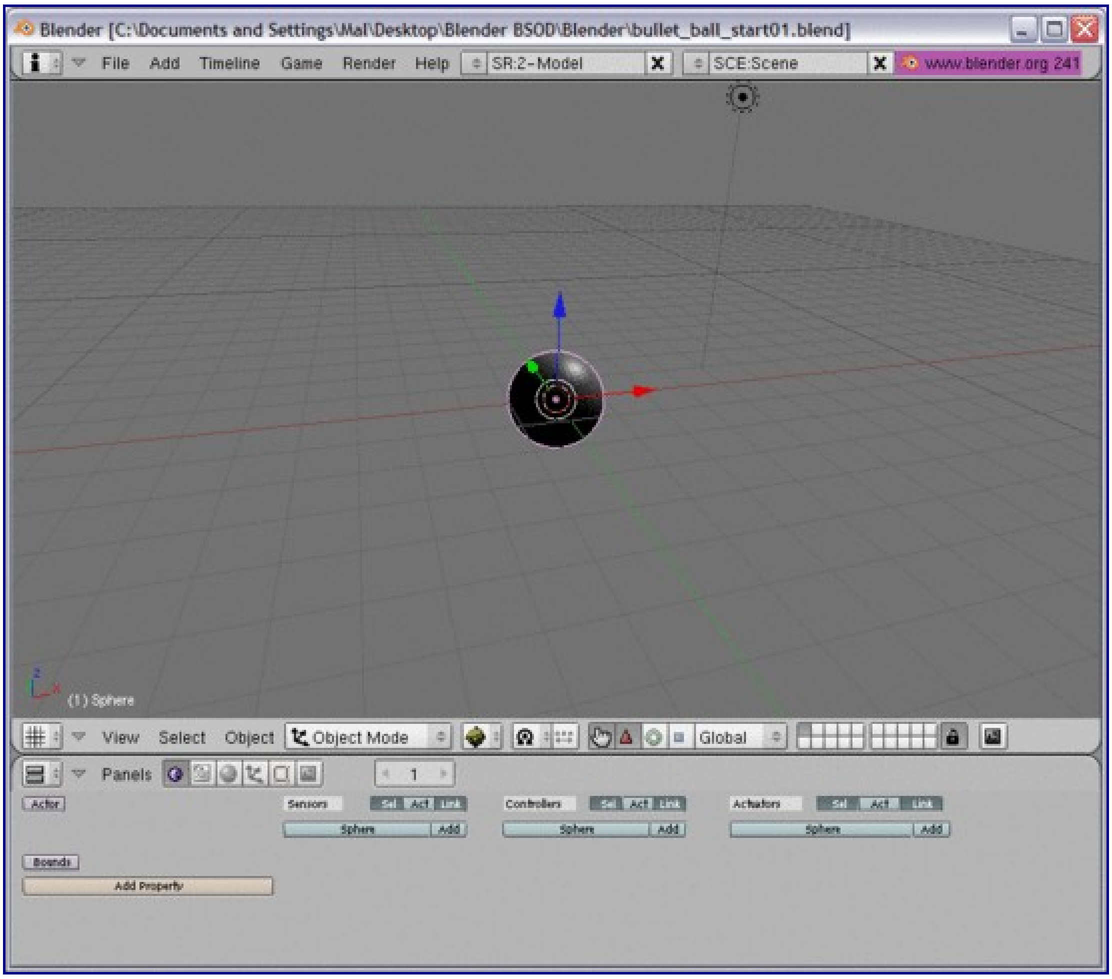
Set up Blender to look like this for the following tutorial

- Select the *File* -> *New* menu, and click on the *Erase All* option.
- Rotate the scene using the middle mouse button MMB
- Change the view into perspective mode using *View* -> *Perspective*
- Change the 3D view to use the Textured shading mode (/draw type)
- Open the Game (/Logic) panel

Setting up the game scene

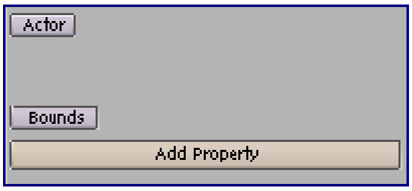
To start creating the game scene, delete the cube by pressing Del.

- Add in a sphere using *Add* -> *Mesh* -> *UVSphere*.
- When asked, select the default 32 segments and 32 rings as the setup of the sphere.
- Exit Edit mode straight away on the sphere by pressing Tab, to return to Object mode.
- Also, press Alt R to clear the rotation on the object (The object must still be selected).

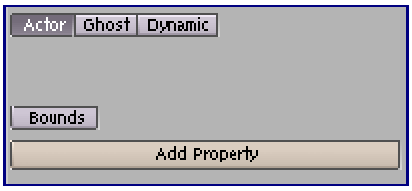


The Scene.

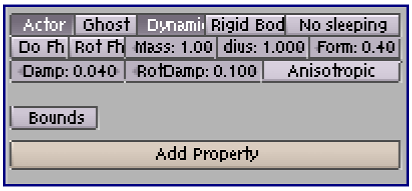
Making a model physical within the GE



In the game panel, you will see an unselected Actor button.



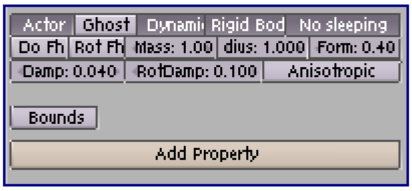
Click on this button, and a number of other features will become available for the selected model.



Mass: 1.00 | dius: 1.000 | Form: 0.40

Damp: 0.040 | RotDamp: 0.100

Select the *Dynamic* option - This will tell the GE that the model is a physical object. More options will then also appear.



Actor Ghost Dynami Rigid Bod No sleeping

Do Fh Rot Fh Mass: 1.00 dius: 1.000 Form: 0.40

Damp: 0.040 RotDamp: 0.100 Anisotropic

Bounds

Add Property

Select the following options...

- *Rigid Body* The physical object will automatically rotate correctly using the GE physics engine. If this isn't selected, the object will be able to move, but not rotate.
- *No sleeping* - The physical object will never be de-activated (also known as sleeping).

Now, press P to enter the GE. You will notice that, even though we haven't added in any game engine logic blocks, the ball starts moving. This is because gravity is affecting the ball, so it falls down. This illustrates one of the features of interacting within a physical world. Press Esc to go back to Blender. Press P again and you'll notice that the same thing happens. Press Esc to return to Blender.

We need to add something for the ball to fall onto, for example a ground object.

- Add a plane model to the scene using *Add* -> *Mesh* -> *Plane*.
- Press Tab to exit Edit Object mode, and return to Object mode.
- Press Alt R to reset the rotation on the model.

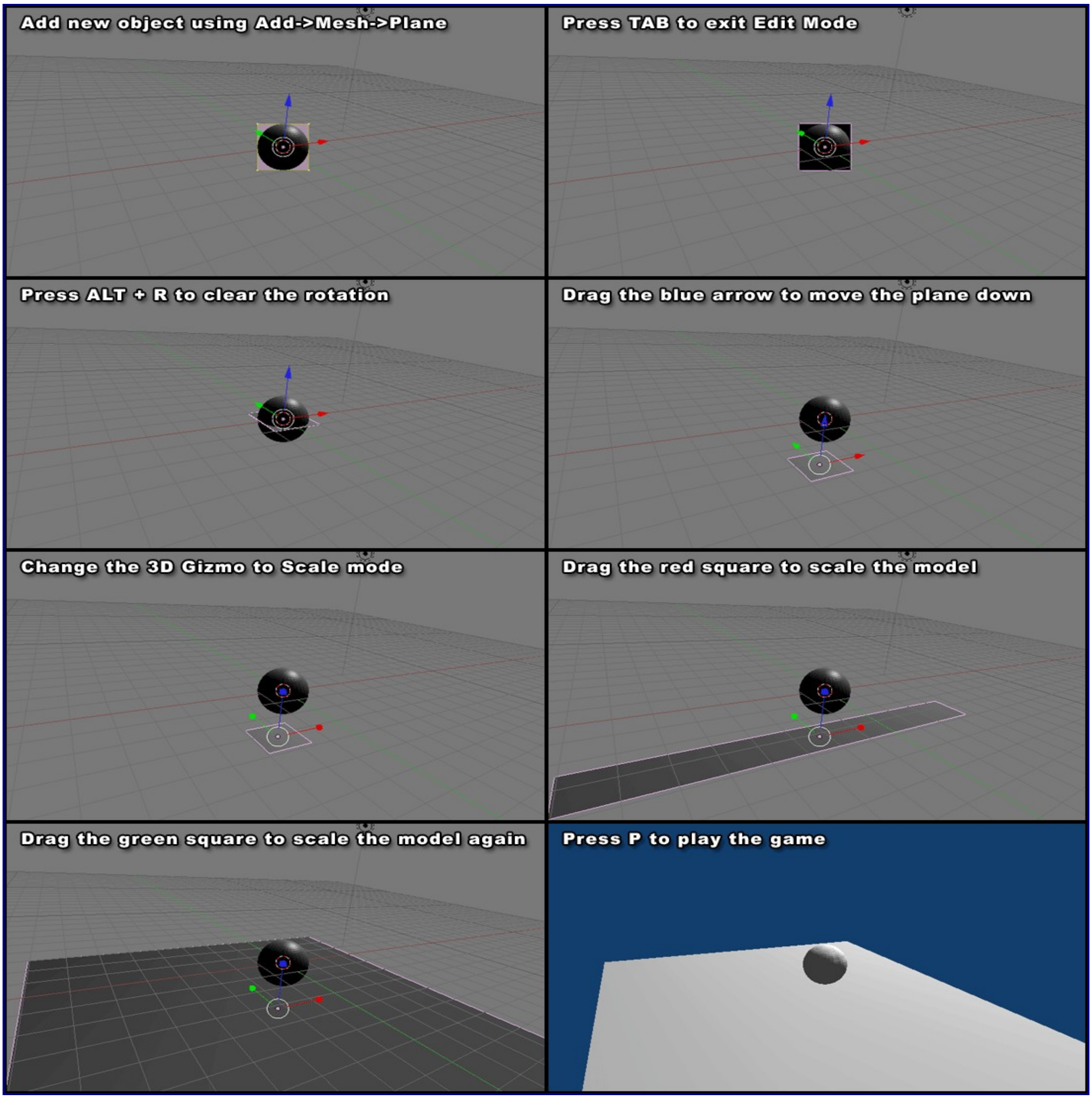
Use the 3D Transform Manipulator to move the plane underneath the sphere, and press P. You will notice that the physical sphere now falls due to gravity, but will land on the plane below it and come to a rest.

We will now scale the plane up (S), so that we have plenty of room to move the ball around within.

Change the 3D transform manipulator from move mode into scale mode.

Grab the scaling handles and size the plane so that it is around 10 times larger in the X and Y directions.

The image sequence below illustrates the sequence of steps indicated above.


Add new object using Add->Mesh->Plane
Press TAB to exit Edit Mode
Press ALT + R to clear the rotation
Drag the blue arrow to move the plane down
Change the 3D Gizmo to Scale mode
Drag the red square to scale the model
Drag the green square to scale the model again
Press P to play the game

Adding and placing a ground object

Moving the physical object within the GE

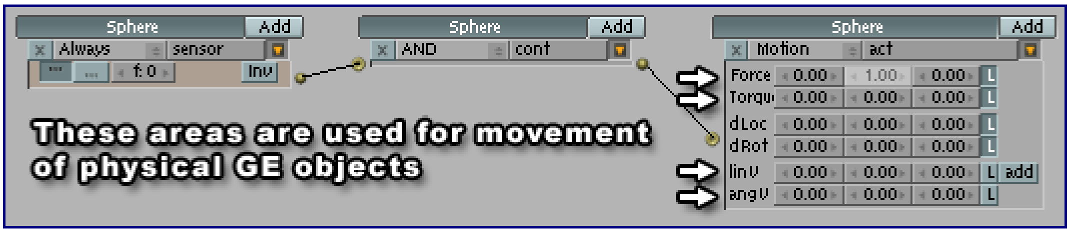
We will now apply physical forces to the sphere, to make it move around the 3D environment.

IMPORTANT - Make sure the Sphere model is selected. If you have just created the ground plane model, it will currently be selected, so you will have to right-click on the sphere to re-select it.

Add in a new sensor, controller and actuator object in the game panel, and connect them together by clicking and dragging between the dots.

In the motion actuator, we will need to set the values in the Force section in order to move the physical object around the scene.

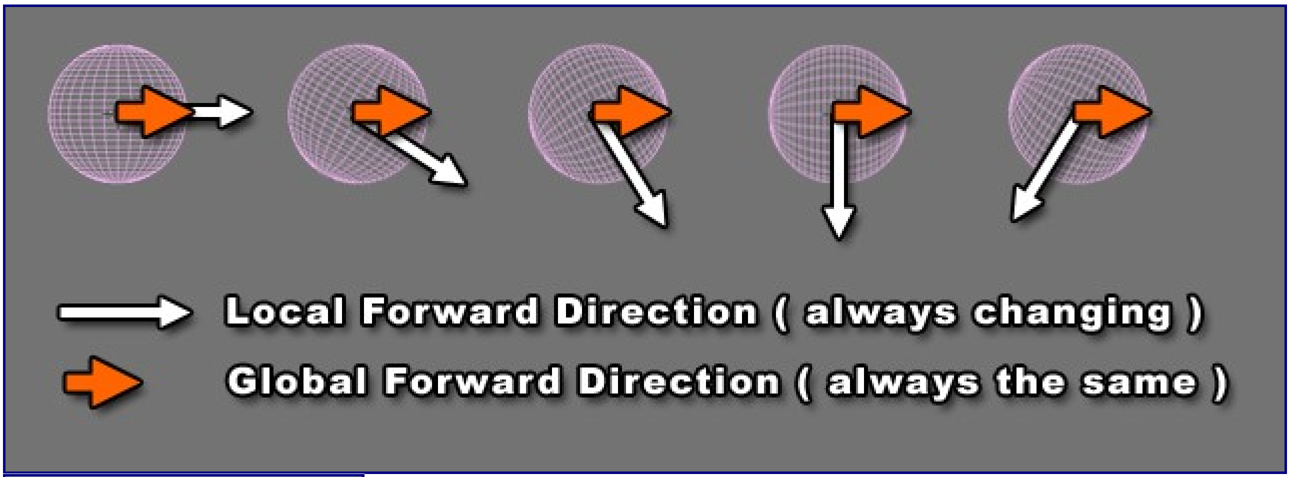
In the *force* section, set the 2nd value (Y) to 1.

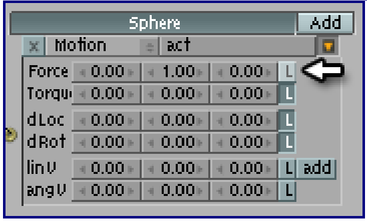


These areas are used for movement of physical GE objects

Now press P. You will then see that the ball falling onto the ground / plane object, and starting to roll.

After the ball rolls a certain distance, you will notice that it starts rolling back on itself. This is due to the fact that we are applying the force locally, along the balls Y axis. as the ball rotates, it's Y axis also rotates, as seen below. Press Esc to return to Blender.


Local Forward Direction (always changing)
Global Forward Direction (always the same)



In order to fix this, we will change the force movement from local to global. To do this, click on the L at the right of the Force section to deselect it. Press P again, and you will see that the ball continually moves in the correct direction.

You may not have noticed how the physics engine is working in the background.

As we apply a sideways force to the ball, it will start to roll. This rolling is caused by friction between the surface of the ball and the ground.

Also, as the ball reaches the end of the plane, it will realistically tip off the end of the object, and continues falling.

These are a few of the advantages of using physics within the GE.

Press Esc to return to Blender.

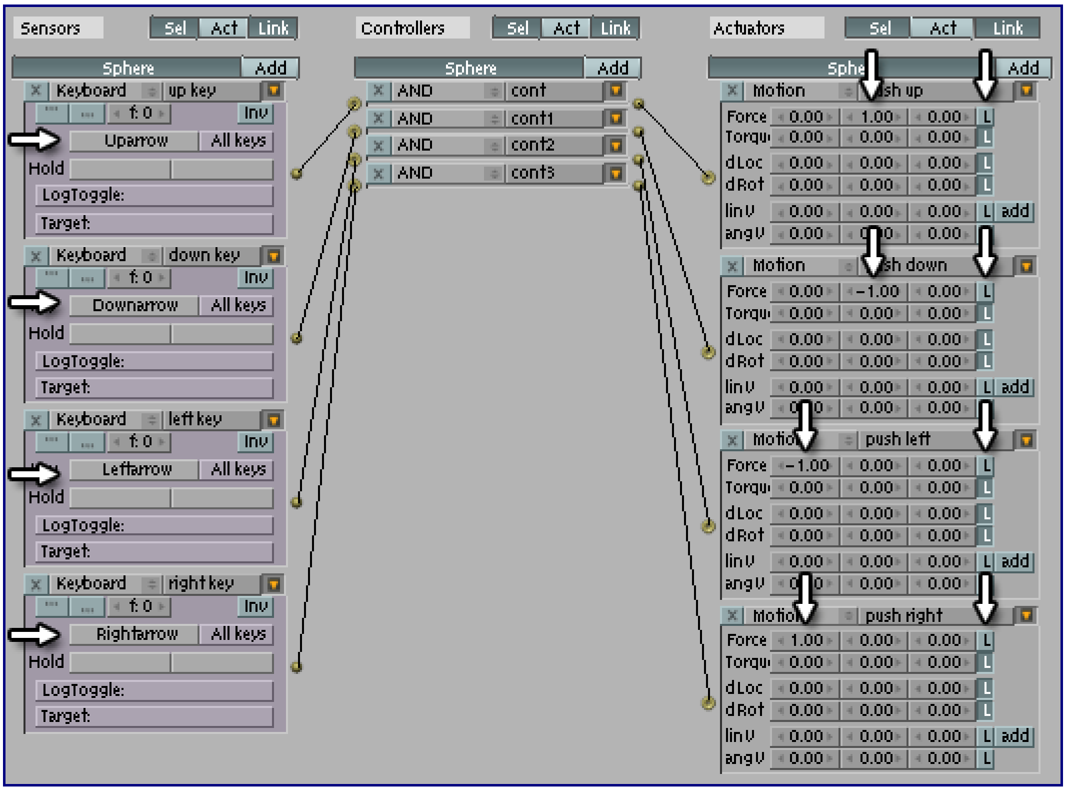
Controlling the sphere using the arrow keys

We will now take control of the sphere using the arrow keys.

Change the Always sensor to Keyboard, and set it to use the ↑ arrow key.

Now, add and connect additional sensors, controllers and actuators, where...

- The ↓ keyboard sensor controls the motion actuator with a -1 in the Y (2nd) location of the *Force* section (with *L* for Local de-selected.)
- The ← keyboard sensor controls the motion actuator with a -1 in the X (1st) location of the *Force* section (with *L* for Local de-selected.)
- The → keyboard sensor controls the motion actuator with a 1 in the X (1st) location of the *Force* section (with *L* for Local de-selected.)



Sensors (Sphere):
Keyboard – up key – Uparrow – All keys
Keyboard – down key – Downarrow – All keys
Keyboard – left key – Leftarrow – All keys
Keyboard – right key – Rightarrow – All keys

Controllers (Sphere):
AND – cont
AND – cont1
AND – cont2
AND – cont3

Actuators (Sphere):
Motion – push up – Force 0.00 1.00 0.00; Torque 0.00 0.00 0.00; dLoc 0.00 0.00 0.00; dRot 0.00 0.00 0.00; linV 0.00 0.00 0.00; angV 0.00 0.00 0.00
Motion – push down – Force 0.00 -1.00 0.00; Torque 0.00 0.00 0.00; dLoc 0.00 0.00 0.00; dRot 0.00 0.00 0.00; linV 0.00 0.00 0.00; angV 0.00 0.00 0.00
Motion – push left – Force -1.00 0.00 0.00; Torque 0.00 0.00 0.00; dLoc 0.00 0.00 0.00; dRot 0.00 0.00 0.00; linV 0.00 0.00 0.00; angV 0.00 0.00 0.00
Motion – push right – Force 1.00 0.00 0.00; Torque 0.00 0.00 0.00; dLoc 0.00 0.00 0.00; dRot 0.00 0.00 0.00; linV 0.00 0.00 0.00; angV 0.00 0.00 0.00

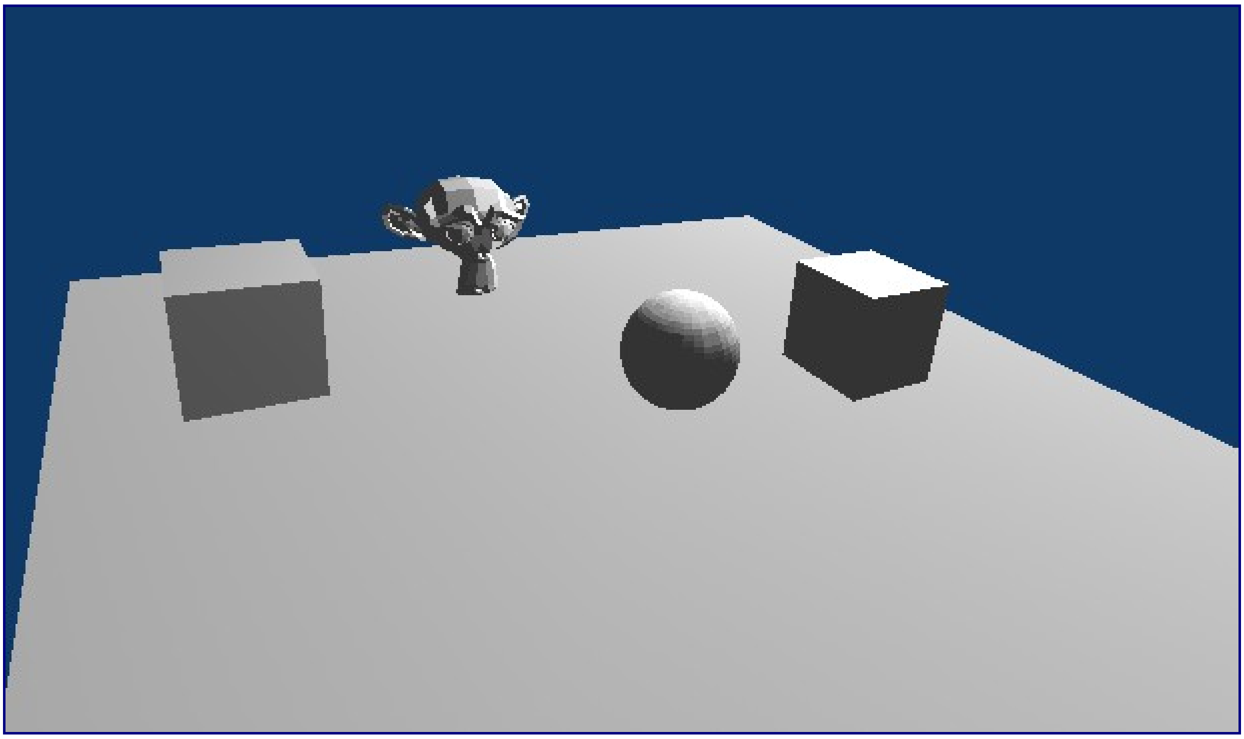
When you press P again, you will be able to control the ball and move it around the ground plane. Press Esc to return to Blender.

To make the game panel screen more readable, give the various sensors and actuators suitable labels, such as "up key" for the first sensor, and "push up" to the first actuator.

Also, try changing the values in all of the motion actuators from 1 to 2, and from -1 to -2. Press P now, and you will see the difference in the speed of the ball when using the arrow keys.

Adding some obstacles into the level

Add a cube to the environment, using *Add* -> *Mesh* -> *Cube*. Press Tab to exit Edit mode and return to Object model. Press Alt R to clear the rotation on the object, and drag the arrows of the 3D transform manipulator to place the box somewhere on the surface of the plane object. Repeat this step to add a few more objects to the surface of the plane, including cylinder and monkey objects.



Now, press P. You will see that the ball will automatically collide off the objects that you just added. Again, this is one of the advantages of using physics within the GE. Press Esc to return to Blender.

You might want to try adding in planes and scaling them to make ramps and jumps.

If you have experience with editing models in Blender, you can spend some time now creating a more complex level layout. If you don't have experience modelling in Blender, hopefully this tutorial will have given you an interest in Blender and learning more, including how to edit models. You can see some links to tutorials on editing at the end of this tutorial.

Making some of the objects physical

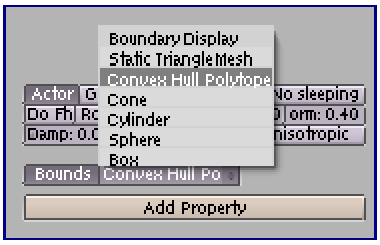
Select the cube that you have added into the new scene. In the Game Panel, select the following options, the same ones that were set for the main sphere (except that *No Sleeping* is not selected this time, to allow the objects to settle and rest / sleep - when an object is sleeping, it takes less time to compute within the physics system)

- Actor
- Dynamic
- Rigid Body

Now, press P to play the game, and move the main sphere into the cube. You will see that the cube now gets knocked out of the way.

However, the box moves in a very odd way - it actually moves as if it was a sphere.

Currently the physics system will assume that a newly added rigid body will have a spherical collision shape. Press Esc to return to Blender.



You will notice that there is a *Bounds* button below the *Actor* area. Click this button, and an additional dropdown will appear, with the default setting of *Box*.

In the case of the cube, a *Box* collision type will work fine. However if you have a more complex shape, you will want to select the *Convex Hull Polytope* option.

Select some of the other objects you have added to the scene, and carry out the same steps as above, selecting *Convex Hull Polytope* as the bounds type of the object.

Press P again to play the current level, and roll into the various physical objects to move them out of the way. Press Esc to return to Blender.

Completion of the basic GE tutorial

Congratulations on completing the basic Blender Game Engine tutorial!

You should now have a general overview of the basics of using the GE.

You will have practical experience of...

- Connecting sensors, controllers and actuators in the game panel
- Using the motion actuator to move objects directly
- Using the motion actuator to move objects using physical forces
- Taking keyboard control of game objects
- Creating a simple 3D game scene
- Making new objects physical within the GE

With the skills you have learned so far, you will be able to extend this simple environment as you learn more about modelling within Blender.

At this stage, you might want to recreate the final scene again, starting from scratch, to see how far you can get without reading the instructions. If you can recreate it all from memory, you are on your way to becoming a true Blender GE power user!

The next few additional areas below will cover some more complex issues, such as making the ball jump, and adding materials to the scene.

Creating more complex game levels and interactions

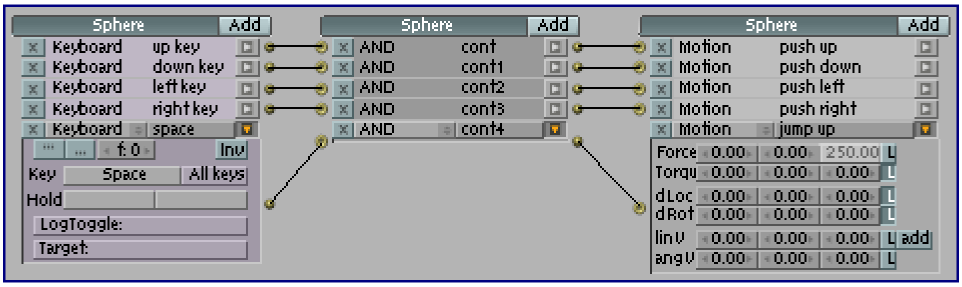
Making the ball jump

We will make the ball jump when the Space key is pressed.

Add a new sensor, controller and actuator to the scene and connect them up. Change the *Always* sensor to a keyboard sensor, and set it up to use the Space key.

In the *Force* section of the *Motion* actuator, set the 3rd value (Z / up) to be 250. Also, click on the *L* to turn it off, so that the force is applied along the global direction, rather than the local Z direction.

This will give the ball quite a bit of a knock, making it look like it is jumping up in the air.



Sphere Add
Keyboard up key
Keyboard down key
Keyboard left key
Keyboard right key
Keyboard space
f: 0 Inv
Key Space All keys
Hold
LogToggle:
Target:

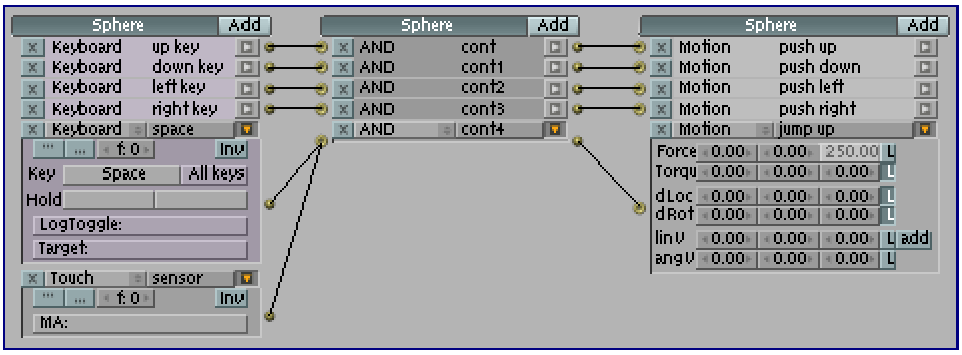
Sphere Add
AND cont
AND cont1
AND cont2
AND cont3
AND cont4

Sphere Add
Motion push up
Motion push down
Motion push left
Motion push right
Motion jump up
Force 0.00 0.00 250.00 L
Torqu 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

Press P, and in the game, tap the Space bar. You will notice that the ball will jump directly up in the air. However, if you continually hold down the Space button, the ball will continually go up in the air.

We need to add in an additional rule - the ball can only jump when it is touching the ground. To do this, we will use the *Touch* sensor.

Add just an additional sensor to the scene, and change it to a *Touch* type. Now, connect this sensor to the controller that the Space keyboard sensor is connected to.



Sphere Add
Keyboard up key
Keyboard down key
Keyboard left key
Keyboard right key
Keyboard space
f: 0 Inv
Key Space All keys
Hold
LogToggle:
Target:
Touch sensor
f: 0 Inv
MA:

Sphere Add
AND cont
AND cont1
AND cont2
AND cont3
AND cont4

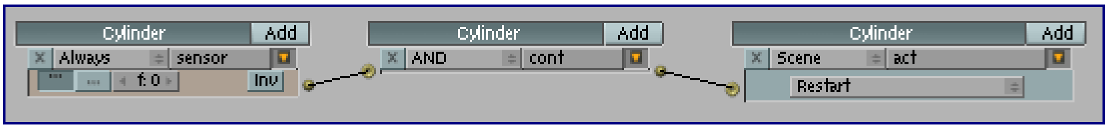
Sphere Add
Motion push up
Motion push down
Motion push left
Motion push right
Motion jump up
Force 0.00 0.00 250.00 L
Torqu 0.00 0.00 0.00 L
dLoc 0.00 0.00 0.00 L
dRot 0.00 0.00 0.00 L
linV 0.00 0.00 0.00 L add
angV 0.00 0.00 0.00 L

The controller will only send a signal to the connected motion actuator (for the jump action) when BOTH Space is pressed, AND the ball is colliding with another object, such as the ground.

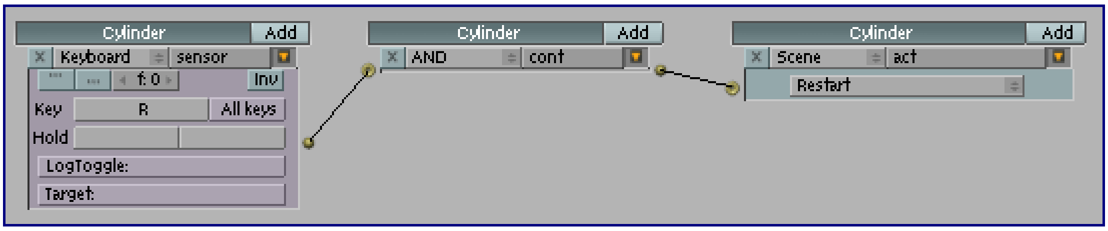
Restarting the game when a goal is reached

When the ball touches a certain object, we will set it to restart the level.

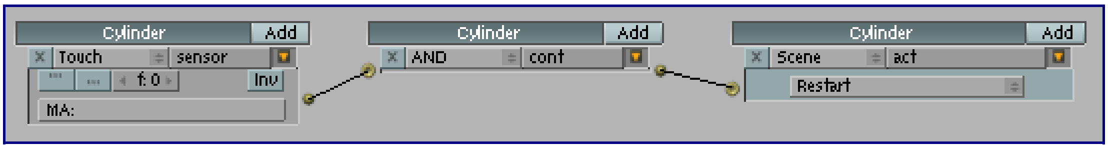
- Add a new object into the scene, and place it somewhere near the main sphere (but not touching it).
- Make sure the new object is selected (and not the main sphere).
- Open the *Game Logic* panel, add a sensor, controller and actuator and connect them together.
- Change the type of Actuator to *Scene*. The default setting of the *Scene* actuator is *Restart*.



- Now, press P to play the game. It will look like nothing is happening. However, as the *Always* sensor is continually triggering the *Restart Scene* sensor.
- Press Esc to exit the game.
- Change the sensor type to *Keyboard*, and set it up to use the R key.



- Press P again to play the game, and you will be able to move the sphere around.
- Pressing R will restart the game, and allow you to continue.
- Press Esc to exit the game.
- Finally, change the sensor type again to *Touch*.



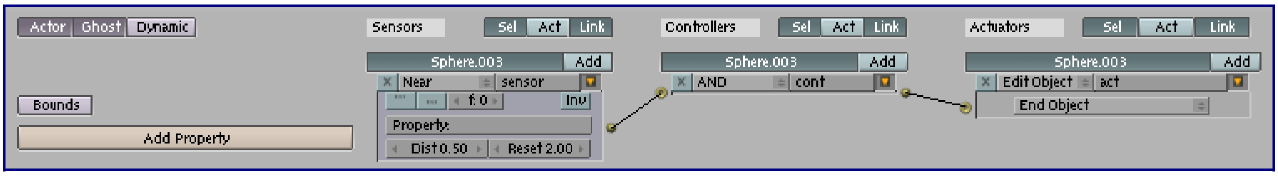
- Press P again to play the game. The scene will restart now when you run into it with the sphere.
- Press Esc to exit the game.

This illustrates some basic game engine scene management. You could have the game go to a different scene (not covered in this tutorial) that might contain a game win or game lose sequence, or the scene might be an additional level of the game.

Collecting pickups within the level

We will now add in an object that the player can collect when they move close to it.

- Add a sphere to the scene, and place it at a reasonable distance from the main sphere.
- Add and connect a *Sensor*, *Controller* and *Actuator*.
- Change the *Sensor* type to *Near*
- Change the *Actuator* type to *Edit Object*.
- For *Edit Object*, change it from *Add Object* to *End Object*.
- Also, activate both the *Actor* and the *Ghost* buttons. The *Ghost* object means that other objects (including the main player) will NOT be able to collide with the object.



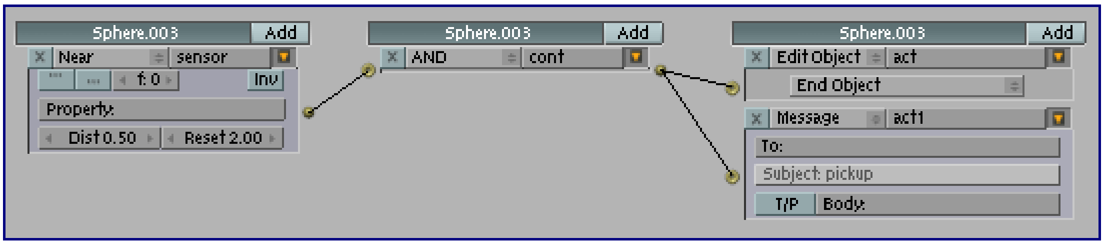
- Press P now to play the game. When you move near the object, it will now disappear (by ending itself).
- Press Esc to return to Blender.

Counting the collected objects

When a pickup is collected, we will update a property value to reflect the total number of collected items so far.

We will use the *Message* actuator to send a signal to another object in the scene, which will have a *Message* sensor. This sensor will trigger an actuator that will increase the value of a property.

- Add another actuator to the pickup object, and change its type to *Message*.
- Connect this *Message* actuator to the existing Controller.
- Set the subject name to be something like "pickup". It's important to remember this name, as it will be required for the *Message* Sensor later on.



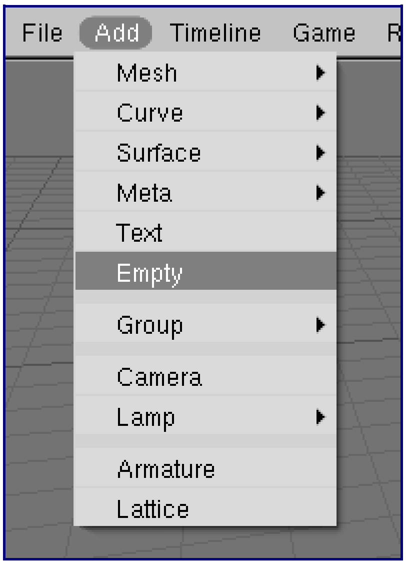
Sphere.003 Add
Near sensor
f: 0 Inv
Property:
Dist 0.50 Reset 2.00

Sphere.003 Add
AND cont

Sphere.003 Add
Edit Object act
End Object
Message act1
To:
Subject: pickup
T/P Body:

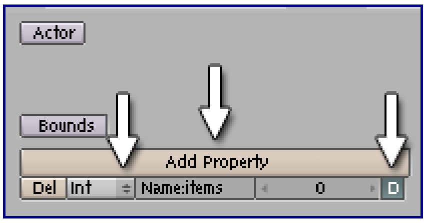
When the near sensor is activated, then both of these actuators will be triggered, and the message will be sent to all of the objects in the scene.

We will use another object to store the pickup count information. For this task, an *Empty* object is useful - this is an object that will exist in the scene, but as it doesn't have any geometry it will not be visible in the game engine.



Adding an *Empty* object.

- Add an *Empty* object to the scene (*Adding an Empty object.*).

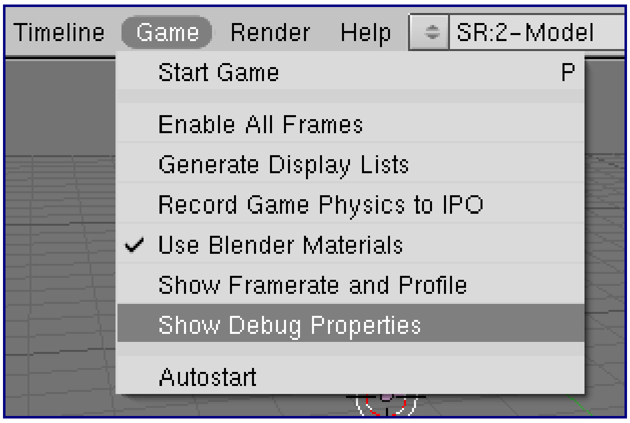


Int Name:items 0 D

Add Property

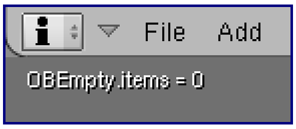
- When this object is selected, click on the *Add Property* button in the *Game Logic* panel (See *Add Property* image).

- Change the name of the newly added object to items, and change the type from *Float* to *Int* (Integer, or whole number, e.g. 0, 1, 2, 3). This is where we will store the number of items collected so far.



Activate *Show Debug Properties*

- Finally, press the D (for "Debug") button to the right of the property. This will allow you to see the value of the property in-game. In order to see the debug properties in-game, select the *Show Debug Properties* menu option.

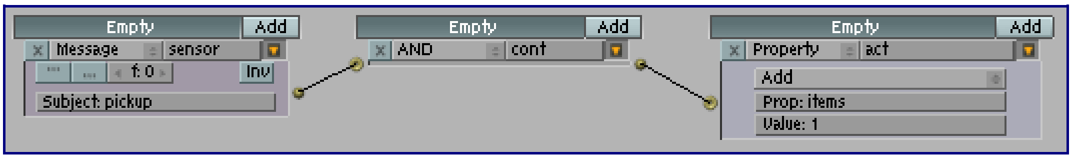


OBEmpty.items = 0

- Press P to play the game now, and you will notice some text in the top left corner of the 3D screen, showing the value of the items property (currently set to 0).

- Press Esc to return to Blender.

- On the *Empty* object, add in a *Sensor*, *Controller* and *Actuator* Logic Block sequence, and connect them together.
- Change the *Sensor* type to *Message*, and set the subject name to pickup (the same name that you placed in the *Message* sensor on the "pickup" object).
- Change the *Actuator* type to *Property*, and change *Assign* to *Add*.
- Change the *Prop* name to "items" (the name of the property to add to), and set the value to 1.



Subject: pickup

Add

Prop: items

Value: 1

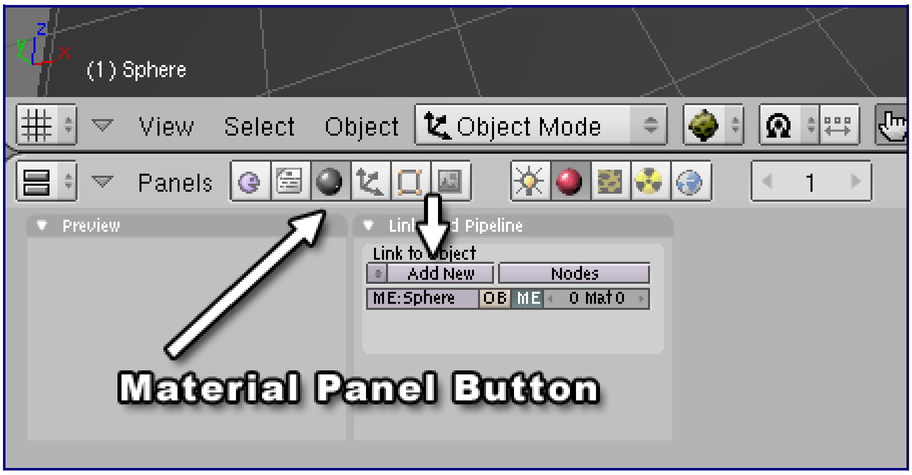
Now, press P to play the game, and collect a pickup - you will notice that, when you collect it, the value of the property increases. Press Esc to return to Blender.

You can use the result of this property to affect your game, such as restarting or going to a new scene when a certain number of pickups are collected.

Adding color to the levels using Materials

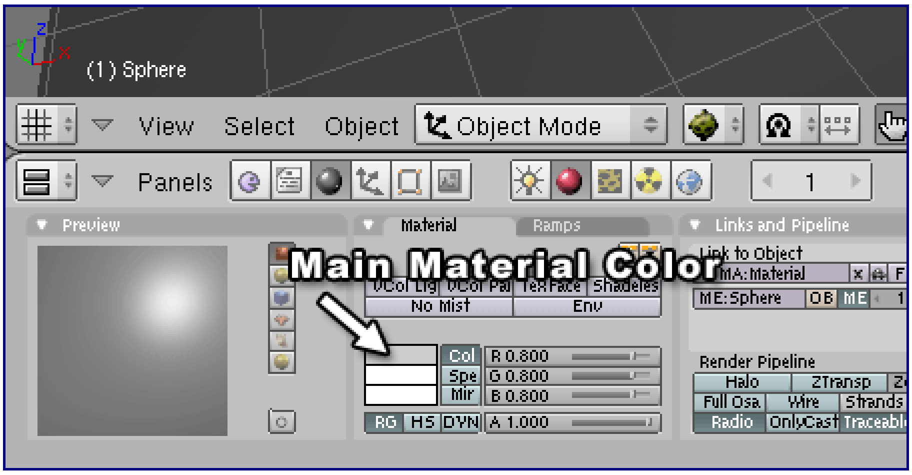
Up until now, the added objects have used the default gray color. In order to change the basic look of the scene, we will now change the colors of the models in the scene by creating new materials for the them.

- Open the *Materials* panel by clicking on the gray sphere on the panel (●) or by pressing F5, as shown below.
- Select the main sphere model in the 3D view with a right click RMB .



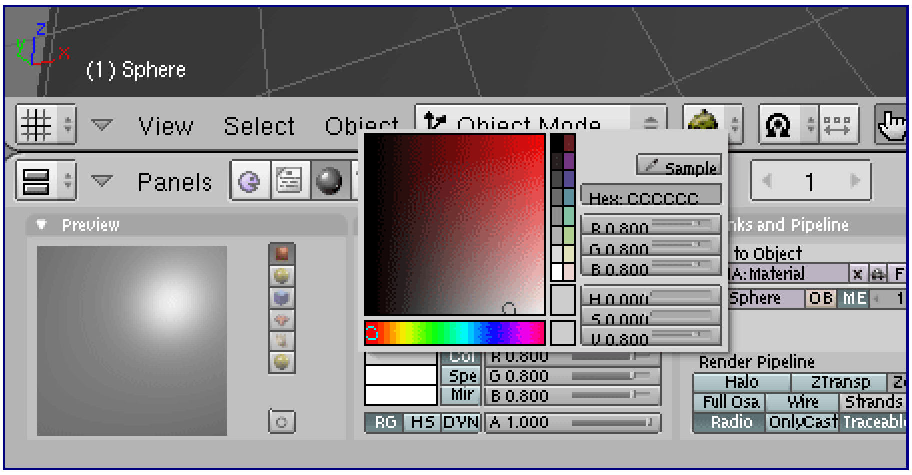
Material Panel Button

- Click on the *Add New* button. This will add a new material to the sphere model. A more complex set of panels will appear. For now, we will just change the color of the material.
- Click on the area beside *Col*, which indicates the main material color.



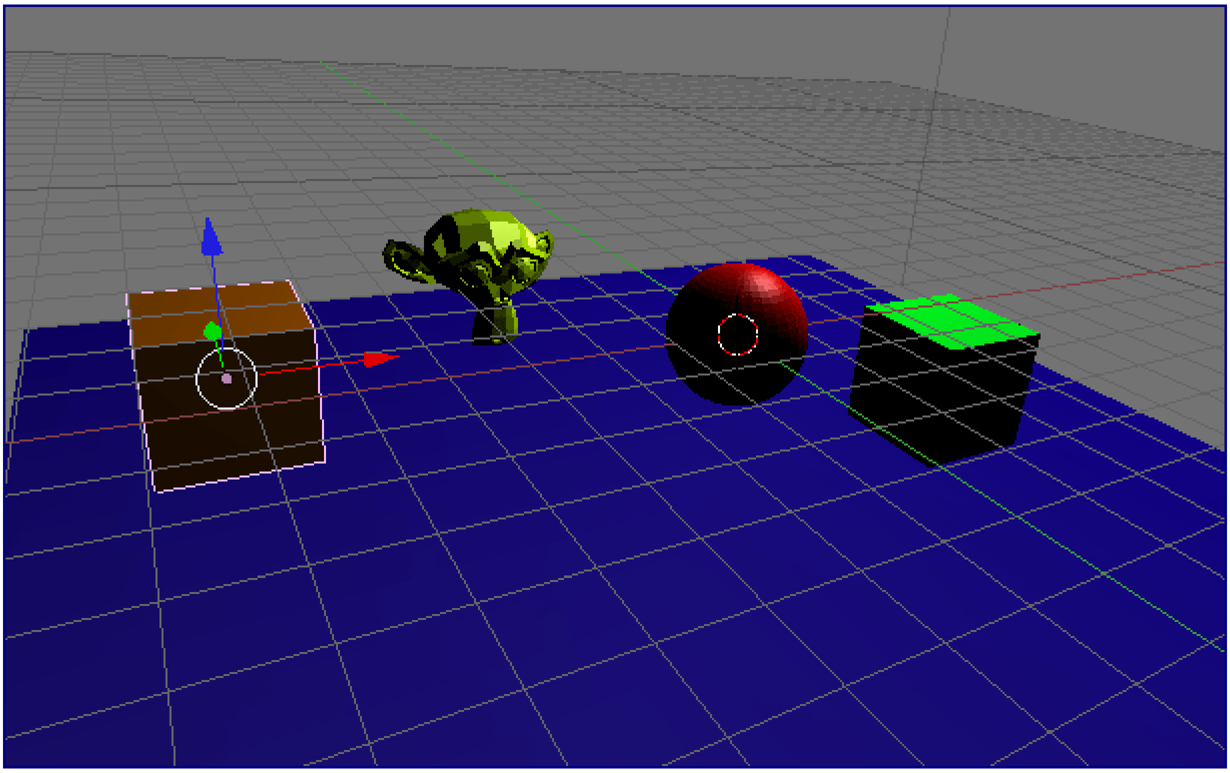
Main Material Color

- A color picker will then appear. Use it to choose a red color, and then move the mouse cursor away from the picker. The sphere will appear red in the 3D viewport.



Hex: CCCCCC
R 0.800
G 0.800
B 0.800
H 0.000
S 0.000
V 0.800
R 0.800
G 0.800
B 0.800
A 1.000

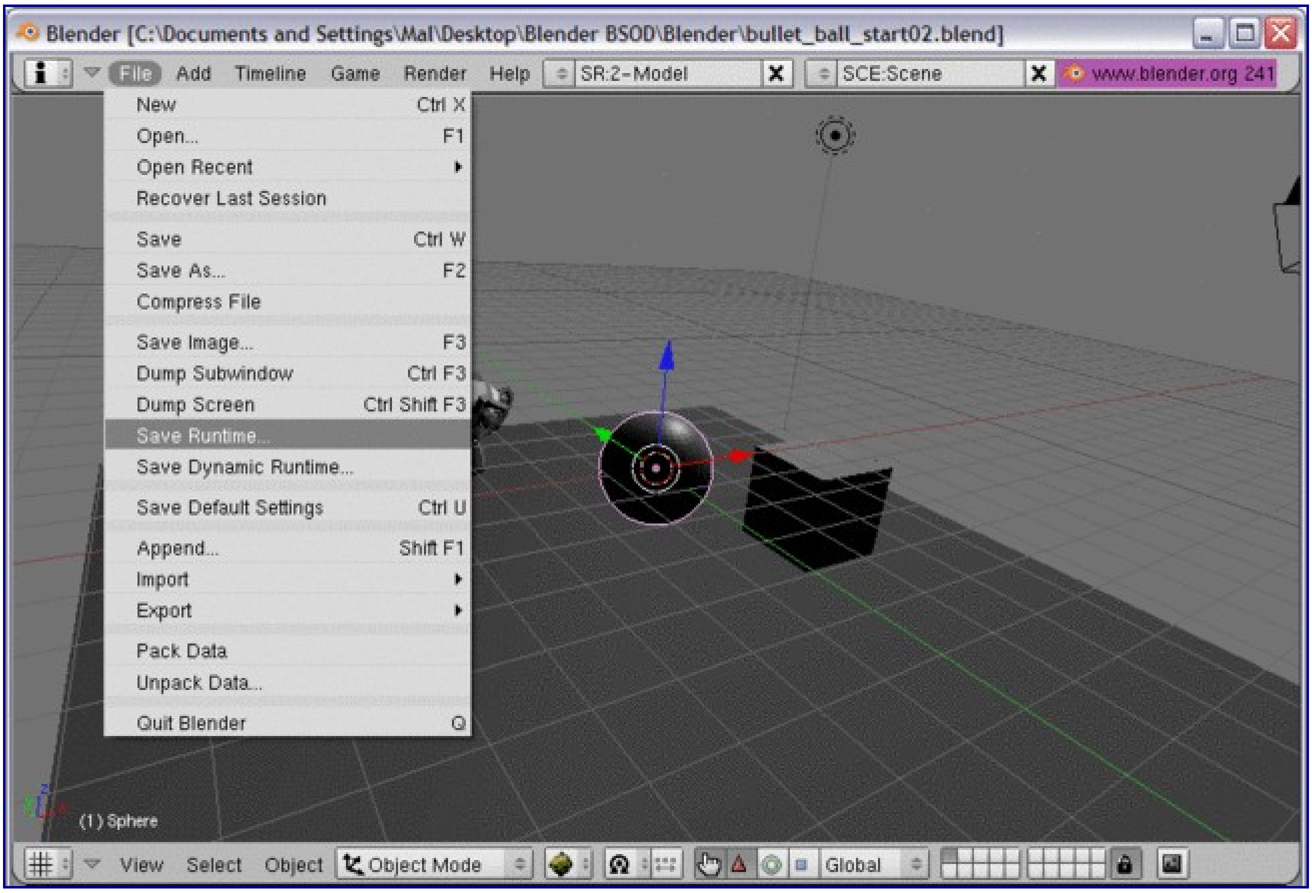
- Repeat this process for some of the other models in the scene until they are all different colors.



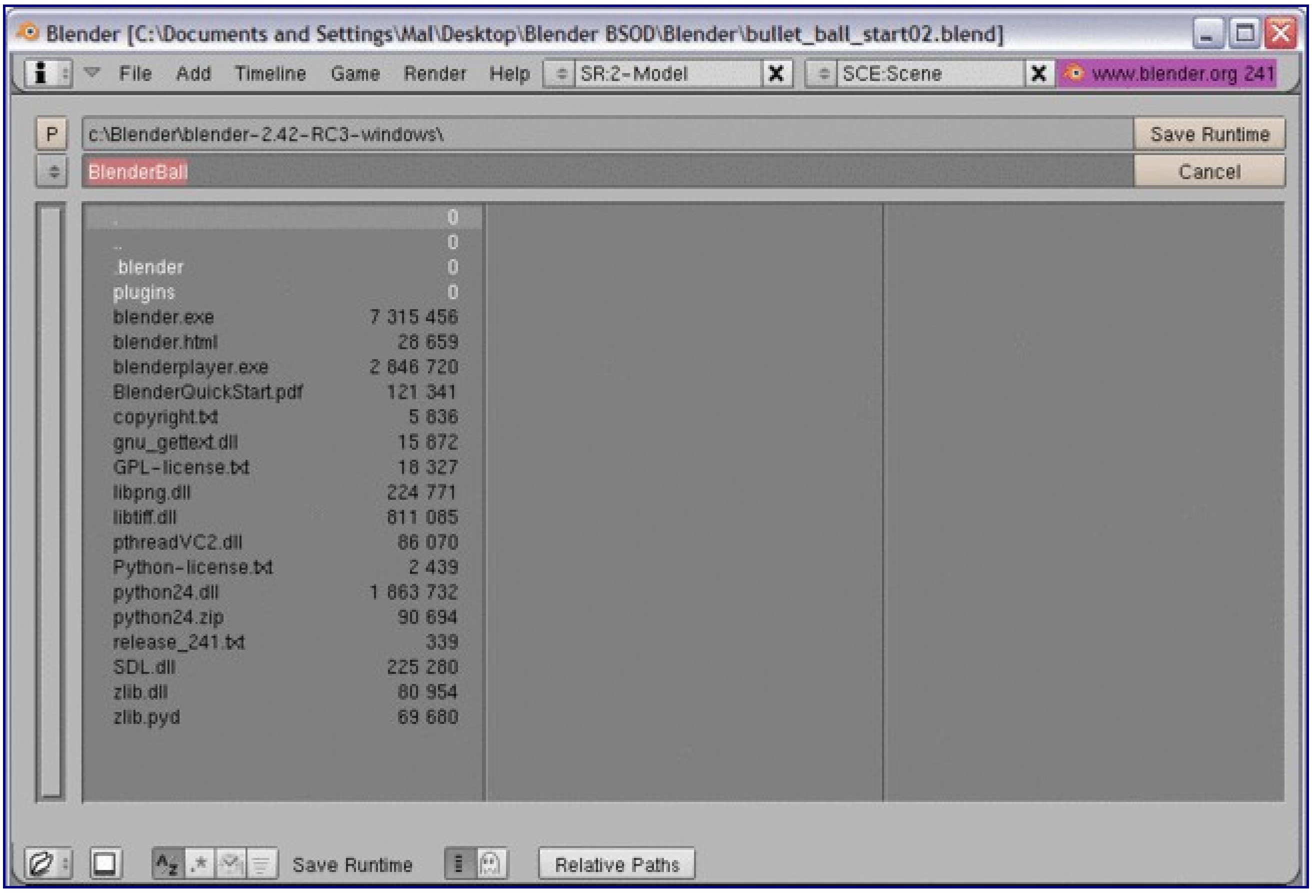
Making a stand-alone version of the game

Blender allows you to create a stand-alone version of your game to distribute to colleagues, without them having to have Blender installed. Your game will automatically run when the program is run.

In Blender, select *File* -> *Save Runtime*.



In the *Save* screen, enter a name for the game executable (for example `ball_game`). This will create a `ball_game` executable in that folder, which you can distribute to your friends.



c:\Blender\blender-2.42-RC3-windows\

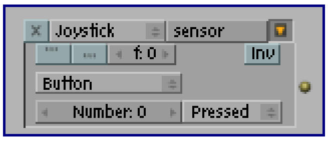
BlenderBall

.	0
..	0
.blender	0
plugins	0
blender.exe	7 315 456
blender.html	28 659
blenderplayer.exe	2 846 720
BlenderQuickStart.pdf	121 341
copyright.txt	5 836
gnu_gettext.dll	15 872
GPL-license.txt	18 327
libpng.dll	224 771
libtiff.dll	811 085
pthreadVC2.dll	86 070
Python-license.txt	2 439
python24.dll	1 863 732
python24.zip	90 694
release_241.txt	339
SDL.dll	225 280
zlib.dll	80 954
zlib.pyd	69 680

NOTE: You may need to include a few other files along with that executable. Copy the file to a new folder and run it. If it gives an error that a file is missing, copy that file (probably a `.dll` file) into the same folder. Continue this process until the game runs. You will then be able to distribute those files to your colleagues.

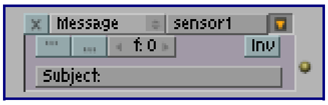
Overview of all of the Sensor, Controller and Actuator Logic Blocks

Sensors



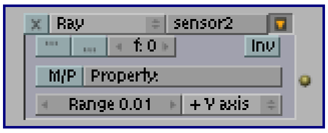
Joystick sensor
f: 0 Inv
Button
Number: 0 Pressed

- *Joystick* - Triggers when either a joystick button is pressed, or when a joystick is moved along a certain direction (left/right, up/down etc).



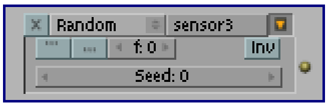
Message sensor1
f: 0 Inv
Subject:

- *Message* - Triggers when a message is received. You can send messages to other objects using a *Message* Actuator.



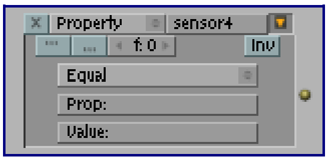
Ray sensor2
f: 0 Inv
M/P Property:
Range 0.01 +Y axis

- *Ray* - This will trigger when an object is detected along a certain axis. You can additionally check for the detected object having a certain material or property value.



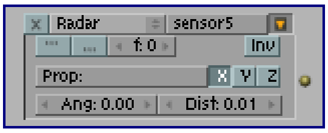
Random sensor3
f: 0 Inv
Seed: 0

- *Random* - Triggers randomly - change seed for different sequence numbers (or use python for real random generator).



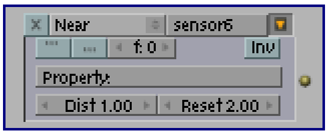
Property sensor4
f: 0 Inv
Equal
Prop:
Value:

- *Property* - Triggers when a property changes, is between certain min and max values, or is equal or not equal to a certain value.

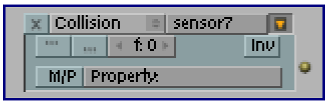


Radar sensor5
f: 0 Inv
Prop: X Y Z
Ang: 0.00 Dist: 0.01

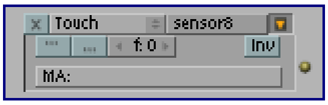
- *Radar* - Triggers when an object is detected within a certain range (distance, and angle). You can specify a property that the detected object must have.



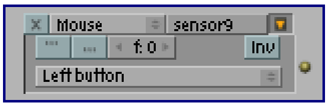
- *Near* - Triggers when an object is detected within a certain distance. You can specify a property that the detected object must have.



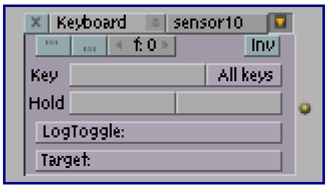
- *Collision* - Triggers when the object is in collision with another object. You can specify a material or a property that the collided object must have.



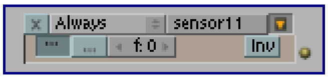
- *Touch* - Triggers when an object is touching another object. You can specify a property that the touched object must have.



- *Mouse* - Triggers when certain mouse events occur, such as mouse button clicks, mouse movement etc.



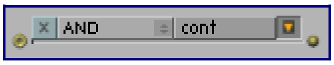
- *Keyboard* - Triggers when a certain key is pressed.



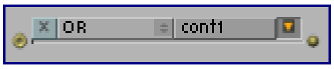
- *Always* - Triggers every single frame.

Controllers

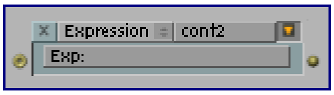
Controllers are triggered by their attached sensors.



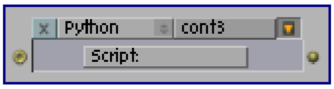
- *AND* - Runs the connected actuator if ALL of the connecting sensors are triggered.



- *OR* - Runs the connected actuator if ANY of the connecting sensors are triggered.



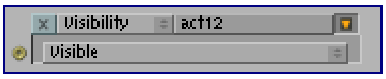
- *Expression* - Evaluates an expression.



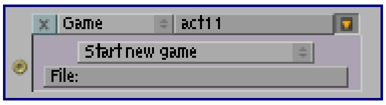
- *Python* - Runs a python script.

Actuators

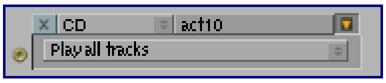
If the relevant sensors are triggered, the controller will call the connected actuator(s).



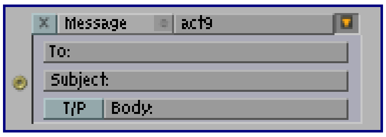
- *Visibility* - Show and hide the current object.



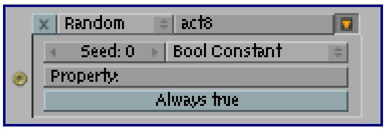
- *Game* - Restart and Quit the current level. Can also load a new scene.



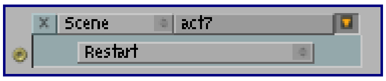
- *CD* - Allows for control over CD music tracks.



- *Message* - Send a message to all objects, or to a certain object. This message will trigger the Message Sensor.



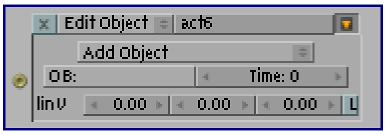
- *Random* - Sets a random value into a property of the object



- *Scene* - Allows for control over scenes - loading, playing, suspending etc.

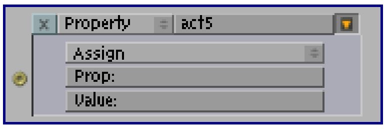
  This is very useful for showing different scenes, such as a start-up scene or menu. When the user wants to play the actual game, a keyboard sensor (e.g. press Space to play) could be connected to a scene sensor, which would then load up the game scene.

  This actuator also allows you to specify what camera to look from, within a 3D scene.

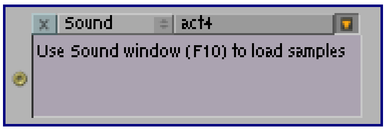


- *Edit Object* - Allows for control over adding, editing and deleting objects within the scene at run-time. This could be used to fire bullets from a weapon.

  It also has an object tracking feature.

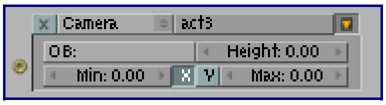


- *Property* - Sets the property value of the object (or of another object).

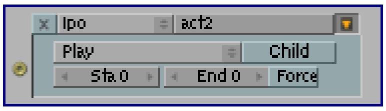


Use Sound window (F10) to load samples

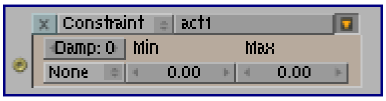
- *Sound* - Allows you to control sounds from within Blender.
  Only sounds that have been loaded into Blender will be accessible.



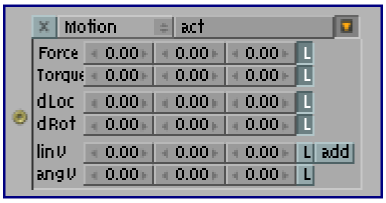
- *Camera* - Allows the camera to track an object. The camera can be placed behind the object within a certain distance (*Min* and *Max*) and *Height*.



- *IPO* - Allows control over playing object animations.



- *Constraint* - Constrains the object's position.



- *Motion* - Allows control over the motion of the object. This includes direct positioning and rotating of the object (*dLoc* and *dRot*), as well as applying forces to a physical object to move it (*Force* and *Torque*).

Continuing your Blender Game Engine

I hope you enjoyed learning the basic concepts of the GE. Hopefully you will continue to work with the GE, using the basic skills you have learnt here, and improving them with plenty of practice, and by learning from and sharing with members of the large GE community.

Additional links and tutorials

Blender Artists Community Forum - GE Section

This is one of the best resources for GE users.

If you want to ask any questions about how to do something with the GE, post some examples of your current game, or just keep up to date on all things GE, this is THE main place to visit.

http://blenderartists.org/forum/forumdisplay.php?f=34