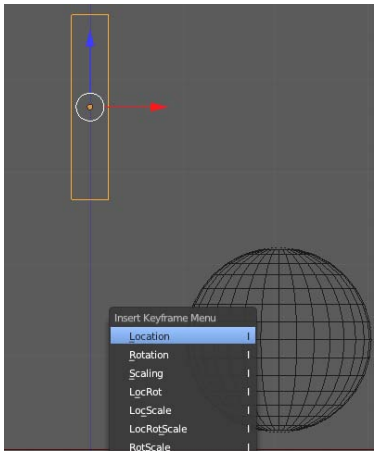
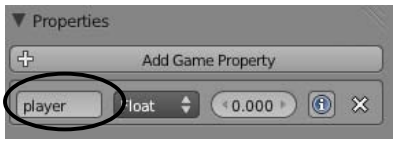
Using Animation in a Game:

Now that we have basic motion down, let's try an animation in the game. We will make the Cube act like a rising door when the actor gets close to it. We first need to add some animation keys to the cube. With the *Cube* selected and at *Frame 1*, hit "**I**" to insert a *Location* key. Move up to *Frame 60*, raise the cube high enough for the actor to pass under it and hit "**I**" again to insert another *Location* key. If it helps, change back to the *Animation* or *Default* screen layout during this step, then return to the *Game* screen.





Back in the *Game* window layout, select the *Sphere* and give it a *Property*. Name it something like "*player*"

Select the *Cube* once more and add a *Sensor-Controller-Actuator* to it. This time, you will add a *Near* sensor, *And* controller, and an *F-Curve* actuator. Set is up as shown:

Distance-Reset:
Adjust for actor distance when trigger is activated. The reset distance (usually higher than distance) resets the trigger.

Start-End Frames:
Set these numbers to match the range of frames you wish to play during the action.

When the actor with the property name "player" gets within the sensor's trigger distance, the actuator occurs. There are several different playing options in the *F-Curve* actuator- *Play* plays the frames and stops; *Ping-Ping* plays frame forwards and backwards; *Flipper* plays forward, stops, then plays backwards during the trigger reset; and *Loop* occurs the entire time when activated.

These are just the basics of the Game Engine. With practice, experimentation, and a little research, you will be able to build some amazing games. Games are played through the *camera's view* so you will want to set the camera's location or child-parent it to the *Actor*. When you're ready to test the game outside of Blender, you need to enable exporting through the *User Preferences* in the *File* menu. Go to *Add-Ons* and select "*Game Engine:Save As Run time*". Now go to File-Export and save as a .exe file.

RoboDude Says: *When making a game, try to keep face counts on meshes as low as possible. The game must actively count and deal with the faces in a game. Detailed meshes will slow things down considerably. The best way to simulate detail is through detailed textures, which will be discussed in the next chapter.*