

# Blender Game Engine Tutorial: Ammo

The author releases permissions for this tutorial to DarkScarab to distribute this tutorial as desired. Go to [www.DarkScarab.com](http://www.DarkScarab.com) to acquire authorizations. Also, join the community!

Thus far, none of these tutorials has required Python, this one is no different. No Python experience required.

The following tutorial will be indexed by color and font styles. Bolded font will mostly indicate **terms**. Red will show alternative **keyboard shortcuts** or **controls**. Green is for **numbers**. Blue will point out **locations** and, in most cases, will tell you where to go.

If you already know what is being presented, feel free to skip ahead!

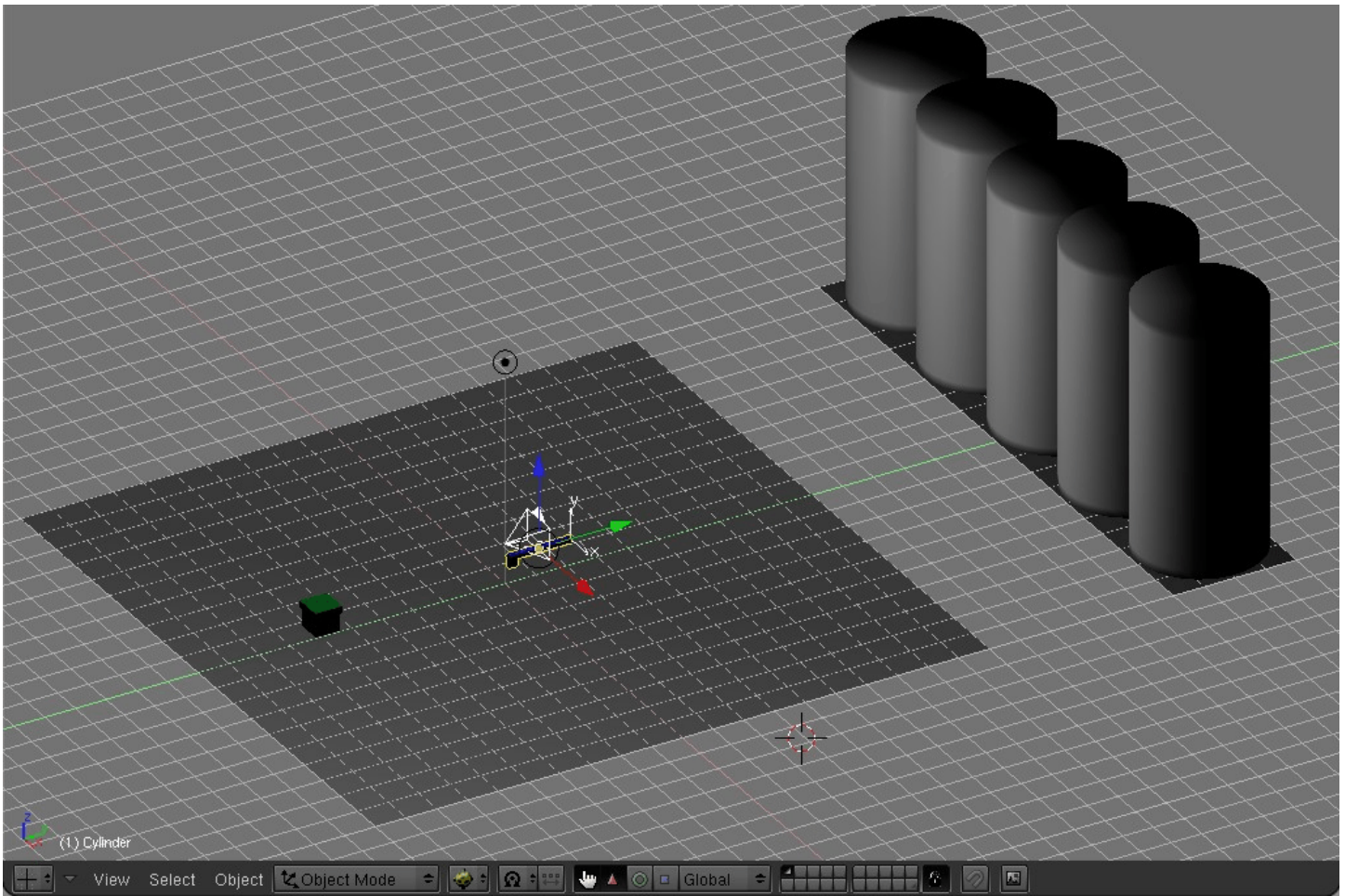
This tutorial will teach you how to set and implement a limited amount of ammunition for a first person styled shooter. It will also inform you on how to utilize ammo pickups so you can add to your overall ammunition count.

This tutorial will cover:

- A review of the '**near**' sensor.
- **Properties:**
  - Adding properties.
  - The difference between "**int**" and "**float**" numbers.
  - Setting **values** to properties.
  - Properties in **Sensors** and in **Actuators**.
- Animating materials (i.e. **alpha**).
- **Editing** meshes.

When first opening the .blend for this tutorial and running the game, you should notice that some things have already been done for you. The camera controls have already been applied to the camera and the gun already shoots when you push the spacebar. If you do not know how I accomplished this, it is recommended that you do not continue in this tutorial. There are tutorials at [DarkScarab.com](http://DarkScarab.com) that cover those concepts. BUT if you do go on, you should have no problem completing the tutorial.

Let us begin!



Our starting scene, complete with stuff to knock down, a loaded gun, and an ammo refill box.

To start, you probably want to go into **camera view (NUM0)** and run the game (**p**). Standard **arrow keys** will allow you to move, while **spacebar** will permit you to shoot. Shoot at the cylinders, they fall down, no problem. You have an unlimited amount of ammo. To add more realism to our “game,” we will want to set a number of bullets that the player can access.

Right click on the **Empty** “mesh” and go into **Logic (F4)**. Click **add property** in the left hand side of the panel.



Change the **Name**: from “prop” to “ammo” (without the quotations) and click on **Float** and change it to **Int**. Then change the number value (click on the column second to the right, next to the “D”) to **10**. You will end up with this:



The general difference between an Integer (**Int**) and a **Float** is that an integer is a whole number, while a float can be a decimal. In our case, we want the gun to count a single bullet as just one. We have no reason to fire off half a bullet or anything like that, we just want whole numbers. Float numbers are useful for things like mph, distance, or any other measurements that are precise.

The number 10 we set for the value of “ammo” is exactly what it sounds like. Our gun will have 10 bullets. You can change it to whatever value you want.

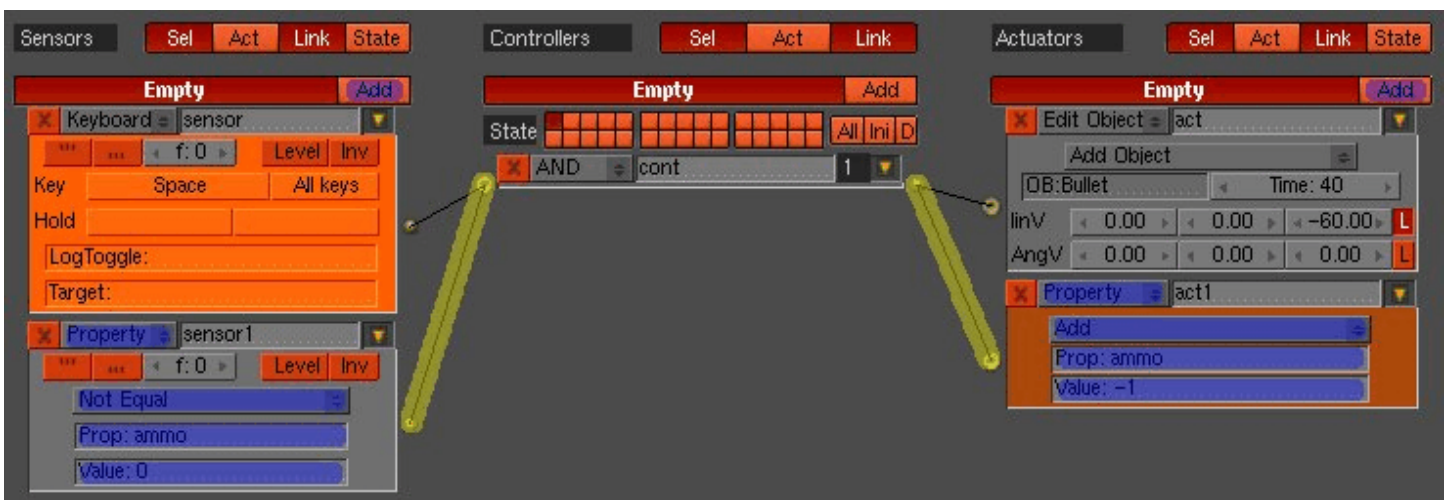
Our next task is to tell the computer that we only want the gun to shoot off a bullet when we actually have bullets. Every time we fire a bullet, the computer should subtract one from our “ammo.” To do this, (still in **Empty > Logic**) **add** a **sensor** and an **actuator**.

**Connect** the new sensor and the new actuator to the controller in the middle. You should now have two bonds to the **controller** from the left and two bonds to the **controller** from the right.

Set ‘**Property**’ to your new **sensor**. In **sensor/property**, change **Equal to Not Equal**. Set **Prop** to “ammo” and fix **value** to 0.

Set ‘**Property**’ to your new **actuator**. In **actuator/property**, change **assign to add**. Set **Prop** to “ammo” and fix **value** to -1.

It should now look like this:



You have just instructed the computer, if spacebar is pressed **AND** “ammo” is not 0 (or that the gun still has rounds left), fire off a bullet and subtract that used bullet (“add” a negative) from your total “ammo.”

Run the game (**p**) and you should find that you can only fire 10 bullets. After that, when you press the spacebar, nothing happens; no bullets come out.

That’s all good, and you could just stop there, but what if you wanted to be able to add bullets to your arsenal?

To do this task, we will review using the ‘**near**’ sensor in a process VERY similar to the one described in [DARK SCARAB – BLENDER GAME ENGINE: PICK UP A WEAPON.](#)

To start, if you are no longer in **Empty/Logic**, get yourself there. **Add two sensors, a controller, and an actuator.** **Connect** the two **sensors** to the one **controller** and in the same way, connect the new **actuator** to the same **controller**.

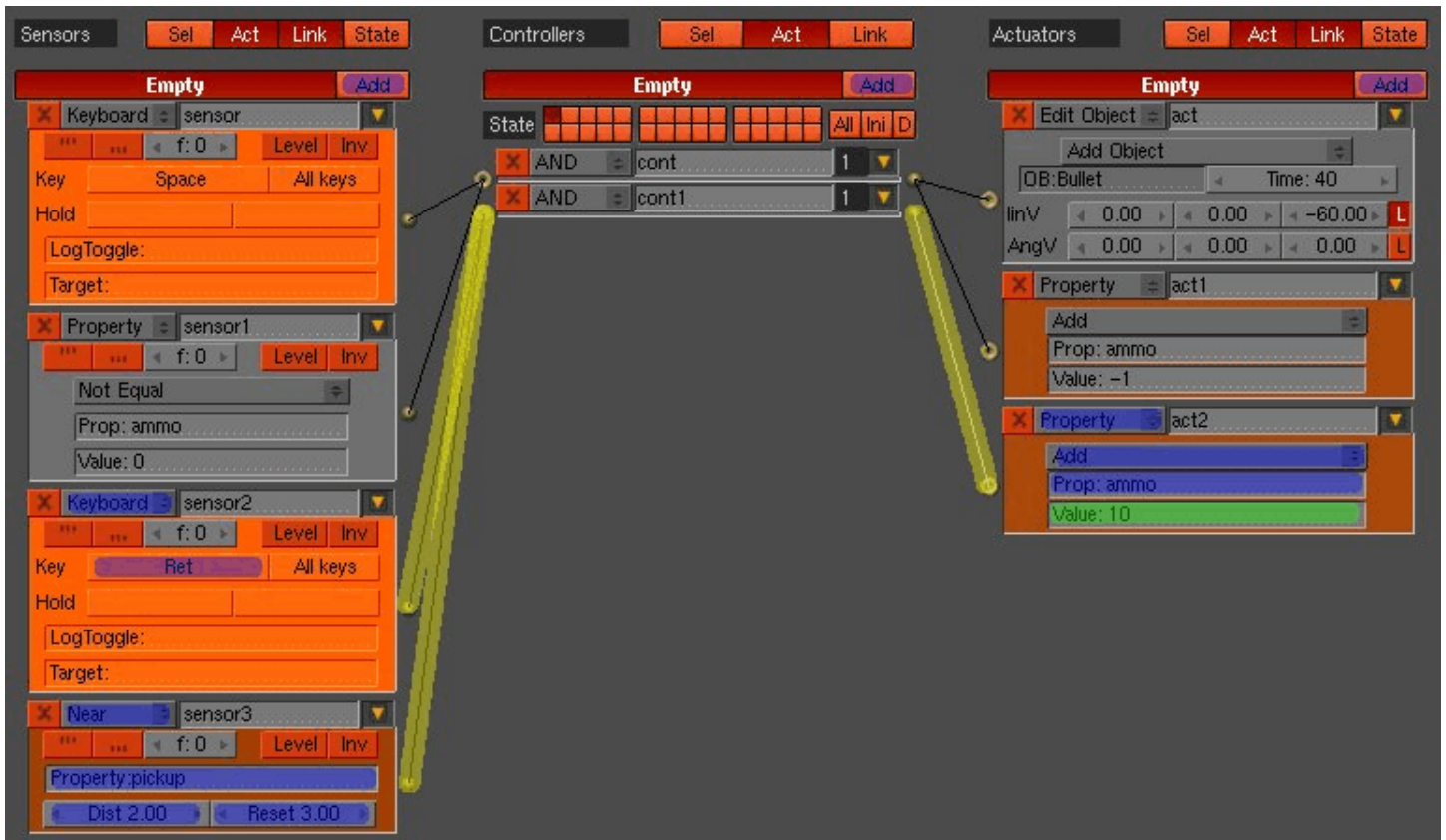
In **sensors**, change one of them to **Keyboard** and the other to **Near**.

In **Keyboard**, click on the **box** right of the word 'key.' Press **enter (return)** to plant it as the **key code**.

In **Near**, set **property** to "pickup," **dist** to **2**, and **reset** to **3**.

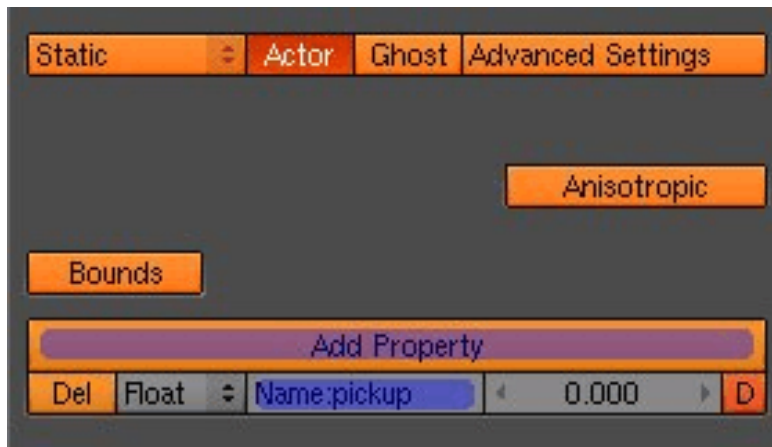
Under **actuators**, set your newest addition to **Property**. Change **Equal to Add**, name **Prop**: "ammo," and **value**: **10**.

It should end up looking like this:



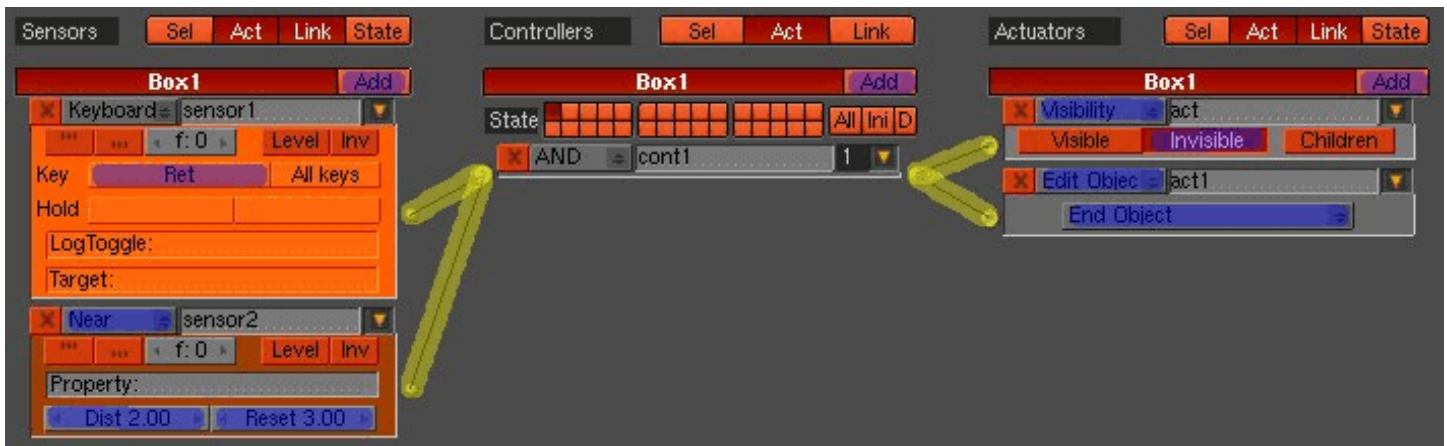
Upon completion of the previous, we have done the following: When **Return** is pressed AND you are within two units of an object with the property "pickup," 10 bullets are added to your arsenal. Pretty good, huh? Only, we don't have an object with the property of "pickup" yet.

Get out of **camera view** and go find the premade ammo box. It is green and when right clicked, you should find that it is named "**Box1**." After selecting it, go to its **Logic (F4)**. **Add** a new property to it. Call it "pickup."



Run the game (p). Fire off 10 bullets and you are out of bullets. Go near the box, press **enter**, and when you push **spacebar**, you will find that you have 10 more bullets you can fire off. There is already a glitch in our little “game.” If you go near the box and press **enter**, each time you press **enter**, 10 bullets are added to your arsenal. That means that if the user presses **enter** 5 times while near the box, they will get 50 more bullets. That is a problem. Usually, when an ammo box is used, it disappears. We will do that for our game.

Select “Box1” and go into its **Logic** (F4). Set it up like in the following picture. By now, I am assuming that you can do this without step-by-step instructions.



We have just set it up so that when we are within 2-3 units of the box and **enter** is pressed, this box will turn **invisible** and it will **end** itself. That way, when the box **ends** itself, users will no longer be able to press enter and get 10 additional bullets to the 10 given. Also NOTE: the near sensor was set up with parameters the same as the ones given to the Empty. This way, everything is consistent; the gun gets ten bullets when 2-3 units from the box - the box disappears only when the gun is 2-3 units from the box. This is very important! It doesn't allow for any backdoors or glitches to occur.

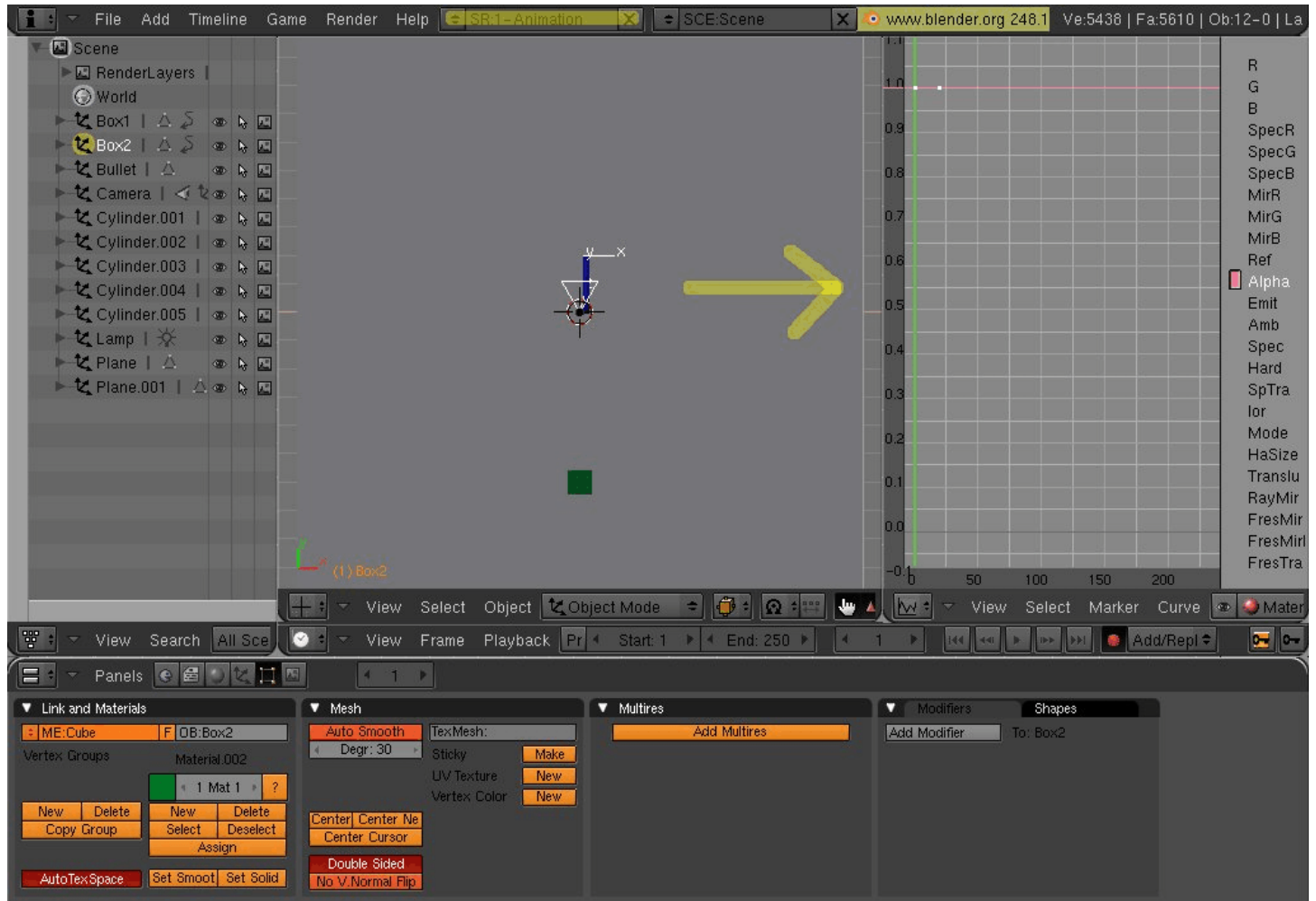
Run the game now (p) and you will find that there are only 20 bullets in the whole scene! Congrats! The tutorial COULD just end here, but we will now take the liberty to make the game look better for the player.

When you press **enter** near **box1**, you will note that **box1** just disappears in a split second. It makes the game look pretty cheap, so we are going to fix that. We will now make it so that the box fades away instead of just disappearing.

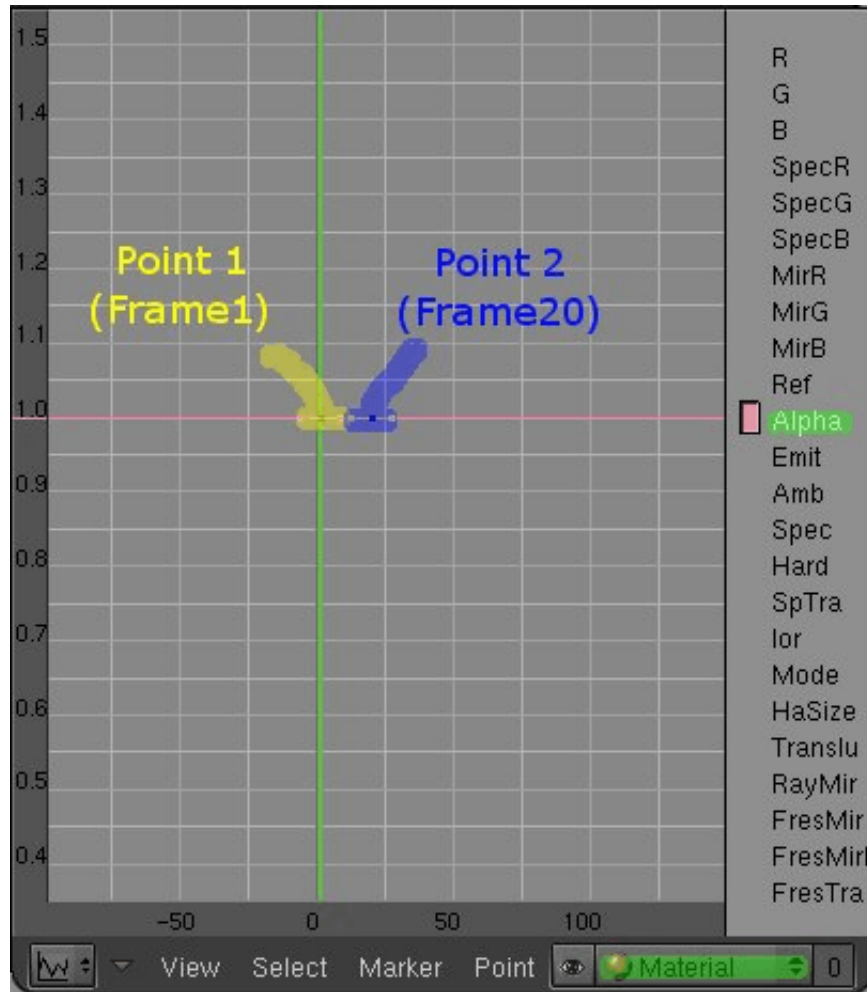
To accomplish this, go to the very **last layer** and find box1's duplicate, **box2**. Select **box2** and go into its **Shading (F5)**.

Enable **z-transparency** by clicking on it.

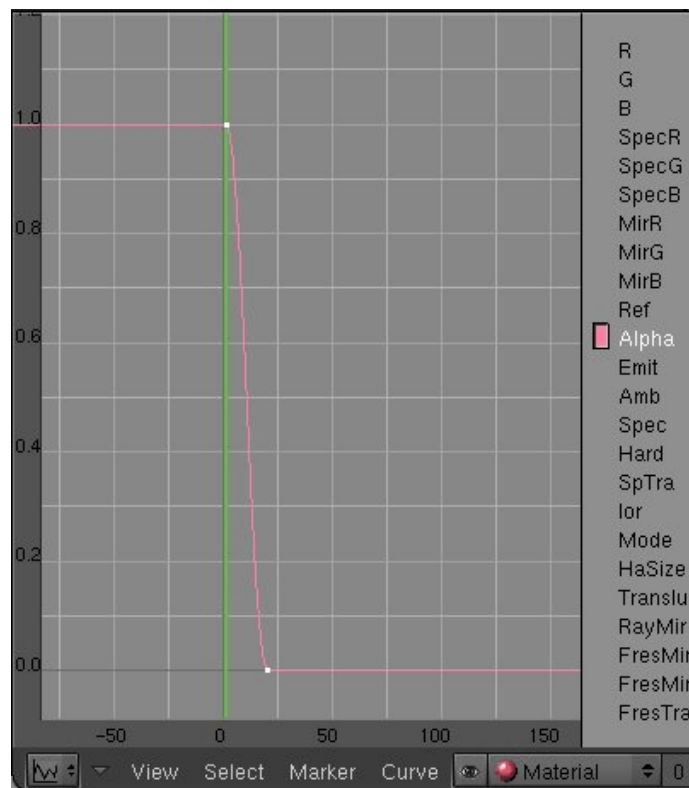
Get yourself to **frame1** and with your mouse over the shading panel, press **i**. This will open up a menu, select **Alpha** to insert an **Alpha key**. Skip all the way to **frame20** and insert another **Alpha key**. On **this frame (frame20)**, get yourself into the **animation layout (SR:1-Animation)(Ctrl+LeftArrow)** and pay attention to the **IPO Curve Editor**.



Select the pink **Alpha curve** and hit **tab**. This should allow you to edit the points. Select the **second point (the one at frame20)** and make sure that you have the entire point selected. **Three** points will be highlighted. It will look like this:



Now press **G** and then **Y** to lock the movement on the **Y-Axis**. Bring the point all the way down to 0 (hold down **Ctrl** to get locking). Hit **tab** to get out of **Edit Mode**. It will look like this:

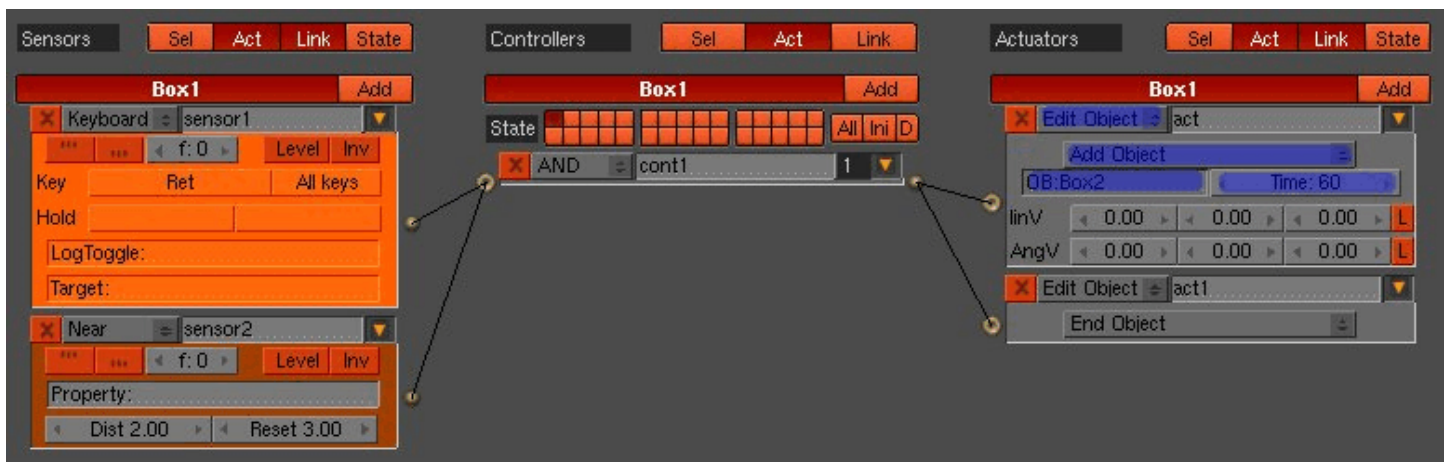


We have just animated the fading out of Box2.

Select **Box 2** and go to its **Logic (F4)**. Make it look like this:

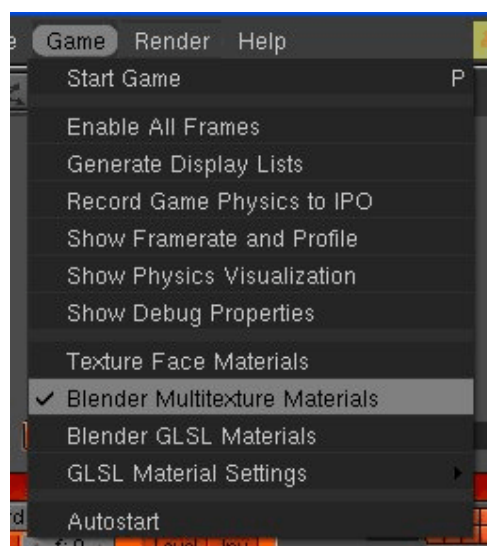


Then go to **Box1's Logic (F4)** and edit **Visibility** to this:



Basically, what we have just done is tell the computer to add Box2 right where Box1 was and for Box2 to last for 60, which is enough time for the fade out animation to play. After Box1 adds Box2, Box1 dies.

Now, before the final running of the game, **MAKE SURE** that you are in **texture mode (alt+z)**. Also be sure to set the game engine to **Blender Multitexture Materials** or the fade out animation will not work.



And that is all that's to it! I hope you found this tutorial insightful, but if you run into any problems or have any comments, you may contact me in the forums at [DarkScarab.com](http://DarkScarab.com).

This tutorial has hopefully given you insight on the use of properties. Numbers are dominant in logic and I hope you can use these properties to create great games in the BGE. You can use properties in other ways that accounting for the amount of bullets your gun has, but I shall leave that up to the reader.

I must give credit to Irascible One for letting me use the gun he made for other tutorials on DarkScarab. Credit goes to me, Dictator-for-Life, for all the rest of the work, tutorial and .blend file. Thanks to you for taking the time to read through this tutorial!