Part three of a series of tutorials for the Blender game engine by Smoking_mirror.

Document Description:
This is part three of a set of tutorials, it covers setting up a helicopter type character's weapons for use in the BGE*.
* Blender Game engine
Document Aim:
The aim of the tutorials is to introduce new users of the BGE to some of the concepts needed to produce a moderately complex game. The aim of this third tutorial is to show users how to set up their player character's weapons. A similar set up can be used for the enemy weapons.

Document Goals:
• Describe how to make a weapon's effects object.
• Show how to integrate the shooting script.
• Show how to set up the ground and enemies so they can be damaged and register hits.
• Show how to make the first steps of a basic HUD (heads up display, or UI, user interface)

Introduction:
We've already made a game model and set up its movement profile, now we are getting to the fun part. In this tutorial I'm going to show how to add weapons to your character. Again, I'm going to be supplying the scripts, all you have to do is set up the game objects as shown and plug in the scripts to your blend file by cutting and pasting.
Document Index:

Part 1 - Creating the weapon effects. - Page 4
Part 2 - Setting up the player logic and objects. - Page 15
Part 3 - Adding a HUD - page 20
Part 4 - Setting up the ground and enemies for collisions. - page 25

Afterword - page 27
Part One – Creating the weapon effects:

We are going to give our character 4 different kinds of weapons. The original Desert Strike only had 3, but we want to add more tactical challenge to the game by including anti-missile flares. The first weapon is the Machine gun, or 30mm cannon as it will be called in the HUD.

When creating the weapons effects we have two options. One is to use a billboard type object, a single plane which always faces the camera and looks the same from every angle. The other is to use 3 planes set up like the gun flash above. This has the advantage of being able to look different from different angles. You should choose which to use in each case. A missile tail should use 3 planes, while a small explosion might use the single billboard. There are also different types of transparency you can use. “Alpha clip” and “alpha blend” require an alpha channel on your image. “Add” just requires that the background, and any part you want to be invisible be black. For these ones we are going to use “add” type transparency. This looks best with the 3 plane type object.
So add a material to your weapon effect. Be sure to check the shadeless box and set the game settings. When you add a texture don't forget to set the UV mapping:

How you set up your materials and textures will depend on whether you are using the GLSL mode or not.
You should go ahead and make all the weapons effects. You can also use animated textures as I’ve done with the “big_bang” effect:

In the image window press “n” to open the option panel and set the game properties. When making an animated texture you should have a number of squares which can fit equally in to the size of the image. So if you have a 256x256 image, 4x4 animations work well, but 6x6 or 7x7 will give a strange effect.
We need the “gunflash” object as well as all the other pictured above. If the object is not going to have physics (set to “no collision”) you just need the mesh. If it is going to have collisions it's better to parent the graphics mesh to a collision mesh. This can be a simple cube or rectangle, set to “solid body” and with the invisible and ghost boxes checked. This allows us to better control the collision. See the missile above, we only want the missile, not its tail, to register collisions. The boxes at the top are empties set up to carry a sound.
You can set up the sound logic like this. Open the .wav file you've prepared, check the 3d sound box and set the volume to whatever sounds right in game. You need the following sound empties:

You can also add the helicopter blades sound to your player object. Just give it an always sensor and an AND controller.
You will need quite a few sounds for your project. [http://www.freesound.org](http://www.freesound.org) is a good place to look for sounds.

This list should be enough though we may need to add more. I will include these sounds with the Blend files, they will be packed along with the textures.
Going back to the weapon effects, we need to set up the actions for the explosions. Add key frames to set the scale of the explosion (see tutorial 2 for this info) and then go to the object display settings. You can set the object color here, which will be used in the action to change the color of the explosion, making it fade out over time.

Right click on the object color box to add a keyframe.

You need to make an action like this. It gets bigger and fades out over time.
Set up the logic like this.

For the rockets and missiles you will have to add two properties:

- `hit_type` means that this object will be detected by enemy checking to see if they have been hit. Hit type sets the damage level and makes the object be affected by flares (if hit type 2).
- Rockets should be hit_type 1 while missiles will be hit_type 2.
Your missiles will also need some logic to make them track to your target. Leave the tracking object box empty, it will be set by python.

You also need some bullet hole type objects. They should lay flat on the ground.
You need “m_hole”, “r_hole” and “b_hole”. They should be set to no collisions and they don't need any special logic. In other games we might give them an animation to make their appearance more natural, but here they are going to be too small to notice.

So the objects you need are as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Physics</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>“gunflash”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“player_rocket”</td>
<td>Weapon effect</td>
<td>Solid body, ghost</td>
<td>“hit_type” int=1, “p_bullet” bool=True</td>
</tr>
<tr>
<td>“player_missile”</td>
<td>Weapon effect</td>
<td>Solid body, ghost</td>
<td>“hit_type” int=2, “p_bullet” bool=True</td>
</tr>
<tr>
<td>“flare”</td>
<td>Weapon effect</td>
<td>Solid body, ghost</td>
<td>“flare” bool=True</td>
</tr>
<tr>
<td>“m_hole”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“r_hole”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“b_hole”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“small_bang”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“big_bang”</td>
<td>Weapon effect</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“mg_sound”</td>
<td>Weapon sound</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“rocket_sound”</td>
<td>Weapon sound</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“missile_sound”</td>
<td>Weapon sound</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“missile_hit”</td>
<td>Weapon sound</td>
<td>No collisions</td>
<td>None</td>
</tr>
<tr>
<td>“no_ammo”</td>
<td>Weapon sound</td>
<td>No collisions</td>
<td>None</td>
</tr>
</tbody>
</table>
You can set up the flare object to have animations, just like the explosions, but you need a slightly different set of animation curves:

This curve is set up for a material using ADD type alpha. The alpha channel stays at 1.0, while the colors fade out to black.
Part two – Setting up the player logic and objects:

We want to add some empties to our player objects, we need three empties called...

...rockets_right, rockets_left and m_gun. The empties are where the weapons are going to be fired from. Parent them to the player mesh, or to the roll action object (the circle).
We also need an empty set way out in front of the chopper called “null_target”.

This also needs a property called “null_target”.

It’s best to set his object a little lower than your chopper and parent it to the roll action object. This is what your guns will aim at if they have no enemy target, so you just want them to fire at the floor a little way in front of the helicopter.
Next we have to give our player object some new properties. Set them up just like in the picture above. “target_ob” and “enemy” are both strings, they can store any information here, in this case we are going to be storing references to objects in game. “target_ob” will be the “null_target” object we created above, while enemy will be the closest enemy to the player, if they are in the correct arc.
This is the logic set up. It has just two sensors and three actuators.

link the always sensor you use for the rotor rotation, this will engage the own[ini] routine when your player is created.
Be sure to add each actuator and sensor as shown. You can increase or reduce the number of logic ticks on the left mouse button sensor to get faster or slow firing weapons.

The python module is:

```
cobra.cobra_shooting
```

At this point you should get the new script from the place you got this tutorial. Just cut and paste it over the old cobra.py script in your text window. All the new code has been added to this script.

You may have already added the helicopter sound in the last section. Just connect it to the same always sensor as the helicopter blades. We will be changing that in future, but for now it's nice to hear the “whup-whup” sound as your chopper flies around the screen.
Part three – Adding a HUD:

There are several ways to handle adding a HUD or UI to your games. One of the most common is to add a overlay scene. However we are going to keep things simple and simply add some objects to the scene parented to our camera.

Add a simple plane to the scene and parent it to the camera so its rotation matches the camera.
Adjust the position in the view so it is in the bottom right hand corner.

Give the object a property called “amount”. Create a new action for the weapon display object to set the object color (like we did with the explosions and flare above).
Next we need to add some more properties to the camera:

And we also need to add some more actuators to the camera and display object:

Make sure you set the action to "property" not "play". The actions should be "display switch" and "color_code". We want the player to see if they are running out of ammo, but we don't want to get in to using text yet, so we are going to show ammo levels by color. "Amount" is the property which drives the color change of the weapon display object. It has to be on the object, not on the cam.
Create four objects on another layer, you will need a new texture for this, I'm using this one:
The arrow and power bar will be used later to show fuel, armor and nearest enemies and objectives.

Name the meshes of the objects as follows:

<table>
<thead>
<tr>
<th>Name:</th>
<th>Represents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>weapon_display_1</td>
<td>Machine gun</td>
</tr>
<tr>
<td>weapon_display_2</td>
<td>Rockets</td>
</tr>
<tr>
<td>weapon_display_3</td>
<td>Missiles</td>
</tr>
<tr>
<td>weapon_display_4</td>
<td>Flares</td>
</tr>
</tbody>
</table>

You should be careful how you set up the object color action. It should fade from pale green at 0 frames towards dark red at 100 frames and then black at 101 frames. This should give the following result:

When you are out of ammo the image goes black.

You should also create a blank mesh (move the verts on the UV image so they are covering a transparent part of the image) and call it “empty_display”. This will be displayed if your chopper has no weapons. If you did everything correctly you should now be able to fly around and fire your missiles, but they don’t explode. So how do we handle that?
Part four – Setting up the ground and enemies for collisions:

Add some cubes to your scene. Make them static and give them the following properties:

- **e_hit** (Boolean)
- **enemy** (Boolean)
- **health** (Integer: 600)

Next set up their logic like this:

The python module is:

```
cobra.player_weapon_hit
```

and the collision sensor should be checking for the property “p_bullet” (remember it's the property you gave the player's missiles and rockets earlier).
Do the same with the ground in your scene:

But don't give the ground the “health” property, instead it needs the “ground” property. You will need to give all enemies and buildings this set up if you want them to be shootable. Objects with the “ground” property will accept missile and rocket craters. You don't want to give this property to small objects such as flagpoles and people because they are smaller than the crater and it'll look like the crater is floating in mid air. Instead use it on things like large buildings.

Now when you fire your weapons the missiles and cannon should automatically track the enemy objects, and they should register explosions. If you cause enough damage you can destroy the enemy objects.

Later we are going to set up enemy so they explode when destroyed, but for now have fun firing your weapons.
I've added a new player model to the blend file that goes with this tutorial. If you change the player object adder to add “player_alt” you can enjoy flying around in a classic UH-1 Huey. This will be part of a later tutorial focusing on changing the player object to add more variety to missions (in the UH-1 you will be able to pick up passengers). Next time we will look at making enemies and giving them simple AI. Thanks for reading. If you have any questions please post them in the Blender artists thread.

Smoking Mirror AKA Pickledtezcat AKA John Topple.

30th June 2013
If you want to write your own tutorials to be included in the series you should use the following formatting conventions:

Main Title: Segoe Print 26 point, Centered.
Subtitles: Segoe Print 16 point, left aligned.
Main text: Segoe UI Semibold 16 point, left aligned.
Image captions: Segoe Print 16 point, Centered.

The document should be landscape, not portrait for better viewing on wide screen monitors without splitting pages, and images included should be of a maximum width of 512 pixels and should be centered. There should be a roughly 50/50 mix of text and images.

You can use the following image for title text background:

It's a simple gradient from pure white to “404040” gray on the right.

From now on all the tutorials can be found here:

[Tutorials by smoking Mirror](#)