AD41700 Computer Games
Prof. Fabian Winkler/Christian Barrett
Fall 2011

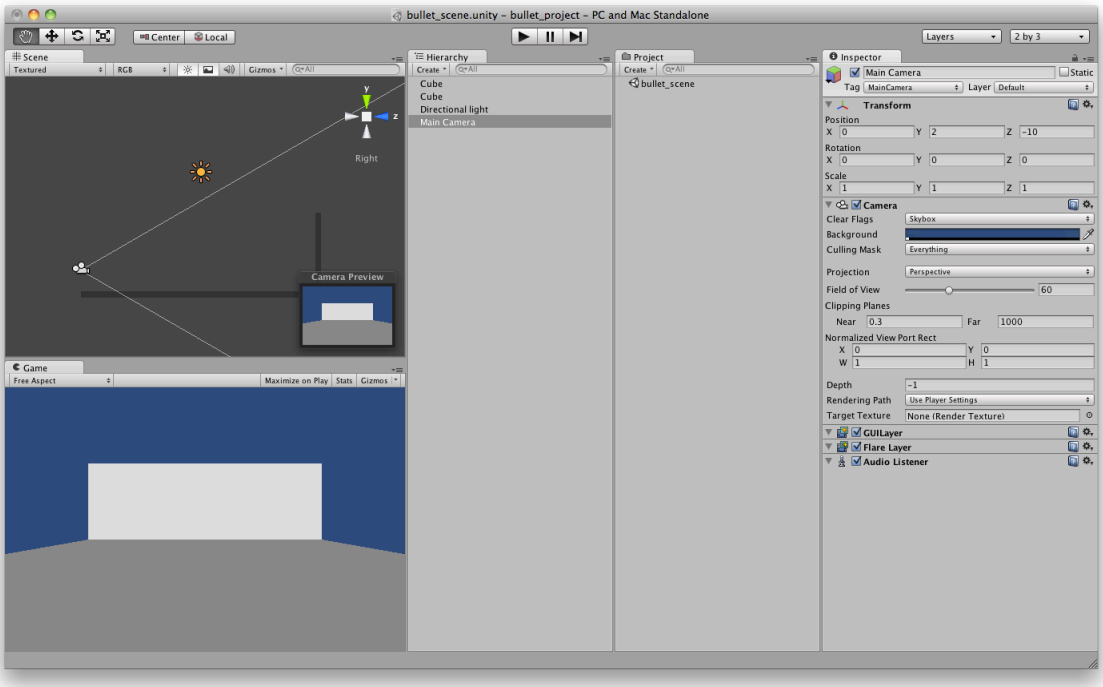
Creating Bullets

I am glad we have read Celia Pearce's Beyond Shoot Your Friends and discussed some creative ways of using the mechanisms of a first person shooter game, so I can talk in this workshop about: creating bullets (and shooting at things).

Start by creating a floor and a wall in a new scene in a new project:

File > New Project… and then: File > Save Scene As…

After setting up the project I create two cubes to function as a floor and a wall, your scene should look something like this:



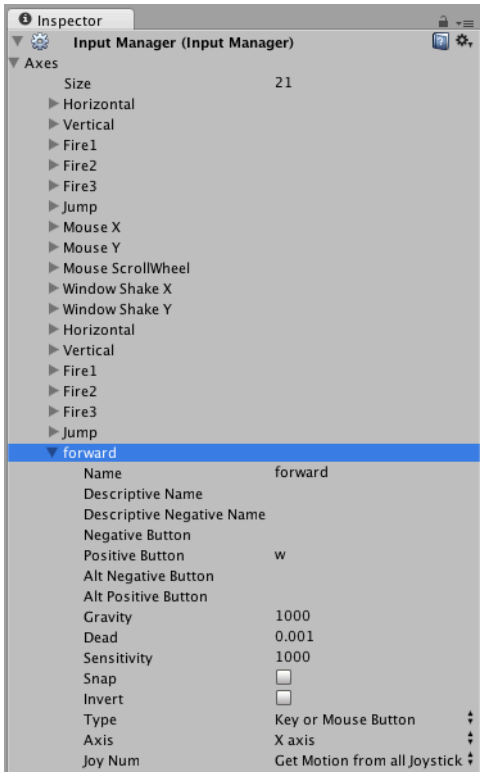
Simple Moving Script
Just in case we are having problems with the Standard Packages again today, here is a simple script that allows you to move the camera around.

```
var moveSpeed = 10.0;
var turnSpeed = 20.0;

function Update()
{

      if (Input.GetButton("forward"))
      {
      transform.position += transform.forward * moveSpeed *
      Time.deltaTime;
      }

      if (Input.GetButton("backward"))
      {
      transform.position += -transform.forward * moveSpeed *
      Time.deltaTime;
      }

      if (Input.GetButton("left"))
      {
      transform.eulerAngles.y += -turnSpeed * Time.deltaTime;
      }

      if (Input.GetButton("right"))
      {
      transform.eulerAngles.y += turnSpeed * Time.deltaTime;
      }


}
```

Now go to Edit > Project Settings > Input and in the Property window add 4 new input properties (increase the number under “size” from 17 to 21. By default these new properties are created as “jump” keys, just open up each of them and give them new names “forward”, “backward”, “left” and “right” and change their positive button value to the key you would like to assign to each of them.

Standard values would be:
Forward – key “w”
Backward – key “s”
Left – key “a”
Right – key “d”

You are now ready to navigate (in a very simple way) through your scene.



Using Prefabs

We will use the concept of Prefabs, reusable assets, for creating bullets on the fly.

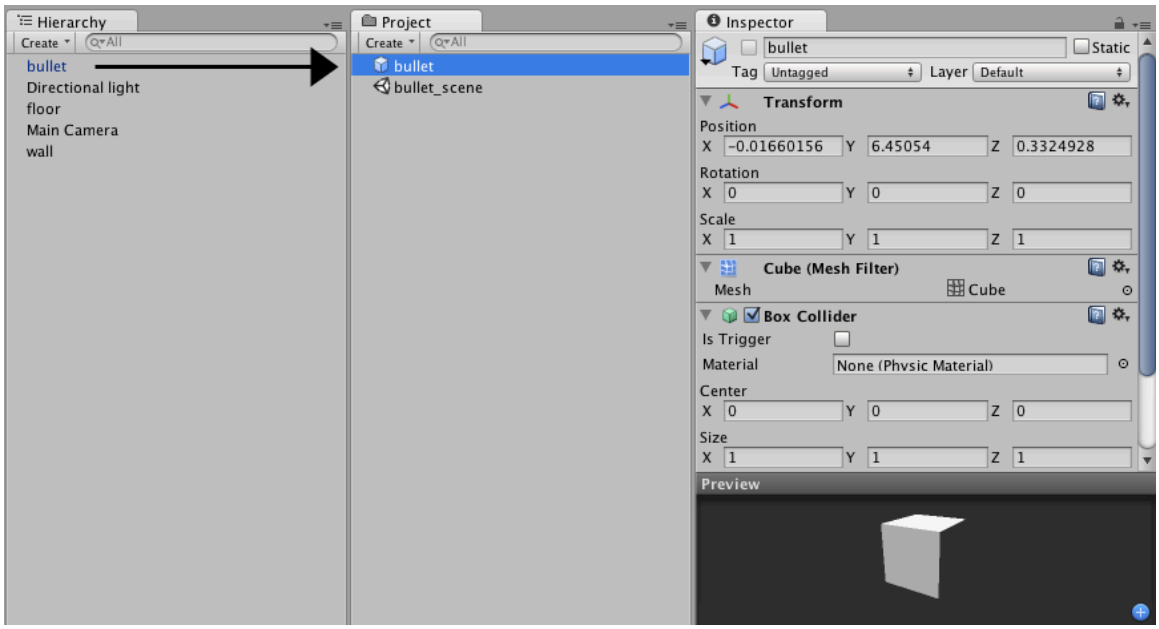
I create a simple cube (which we will use as the projectile/bullet later) and turn it into a prefab:

Game Object > Create Other > Cube…

Attach a rigid body component to the cube. While it is selected in the Hierarchy window got to: Component > Physics > RigidBody

I rename the cube game object in the Hierarchy window to “bullet” and I also create a Prefab called “bullet” (you can name these things whatever you want):
Asset > Create > Prefab

The Prefab shows up in the Project window. I just need to drag and drop the bullet game object from the Hierarchy window onto the bullet prefab in the Project window to assign it.

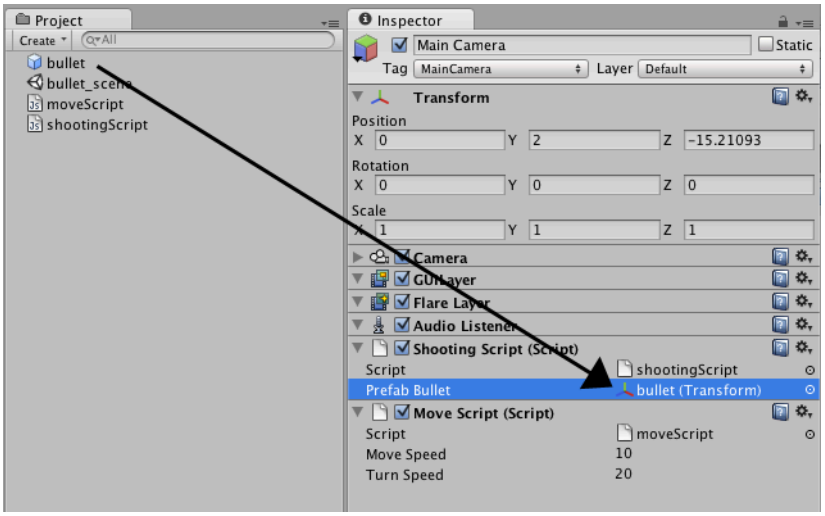


Since all the data about the bullet prefab is now in the bullet prefab in the Project window, you can delete the bullet game object in the Hierarchy window.

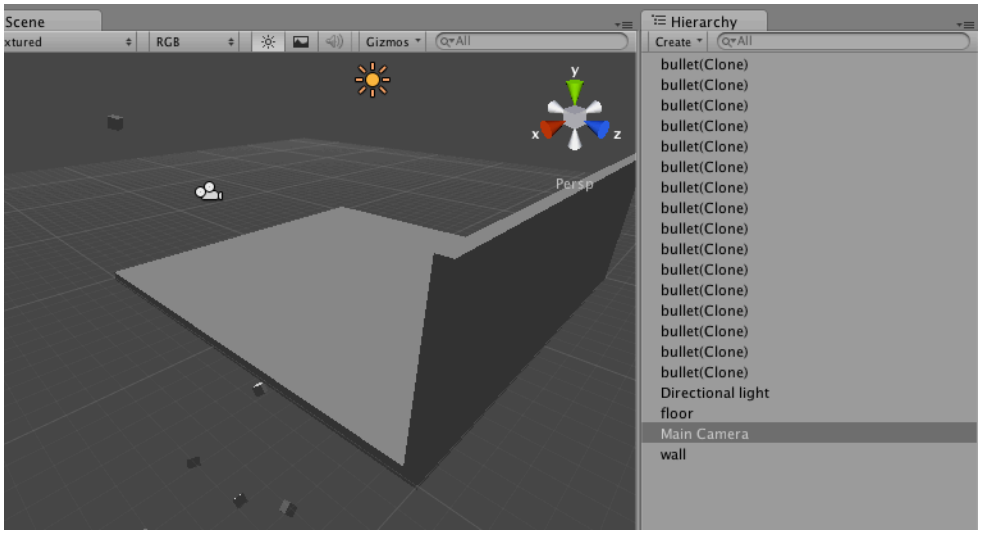
Now, I create a JavaScript which, upon being triggered, creates new instances of the bullet prefab:

```
var prefabBullet : Transform;

function Update()
{
      if (Input.GetButtonDown("Jump"))
      {
       var instanceBullet = Instantiate(prefabBullet,
       transform.position, Quaternion.identity);
      }
}
```

I attach this script to the camera and make sure that the prefab “bullet” is assigned to the prefabBullet variable in the script (I assign it in the Inspector): drag and drop the bullet prefab from the Project window onto the prefabBullet variable in the Inspector:



Now you can already try out the new script and you'll see that cubes are being created in the position of the camera. Because they have no forward pointing force attached to them they simply fall down:



You also see in the Hierarchy window how all the instances of the prefab bullet are listed as you create them.

Now we add the forward pointing force to the script:

```
var prefabBullet : Transform;
var forwardForce = 1000;

function Update()
{
 if (Input.GetButtonDown("Jump"))
 {
  var instanceBullet = Instantiate (prefabBullet, transform.position,
  Quaternion.identity);

  instanceBullet.rigidbody.AddForce(transform.forward *
  forwardForce);
 }
}
```

We are ready to try out the script and should see the cubes shooting out forward from the camera position rather than just dropping down.

Note: You can also attach this script to a First Person Controller. After importing the package (Asset > Import Package > Character Controller) and deleting the main camera from the scene you'll simply drag and drop the shootingScript onto the First Person Controller in the Hierarchy window.

If you would like to change the angle of the bullet's trajectory, play around with the vector of the AddForce function, like this:

```
var prefabBullet : Transform;
var forwardForce = 1000;
var forwardForce = 500;


function Update()
{
 if (Input.GetButtonDown("Jump"))
 {
  var instanceBullet = Instantiate (prefabBullet, transform.position,
  Quaternion.identity);

  instanceBullet.rigidbody.AddForce(0, upwardForce,
  forwardForce);
 }
}
```

Homework:
It would be relatively easy to setup another pair of keys that controls the camera looking up and down and using this vector information to interactively set the angle for the bullet's trajectory.