INSTITUTE OF TECHNOLOGY
BLANCHARDSTOWN

A Taster of Computing
[[VERSION – Unity 2D – C# language]]

Gravity Guy 3D (2014) - a little computer game... now in 3D

Part 2 - projectiles

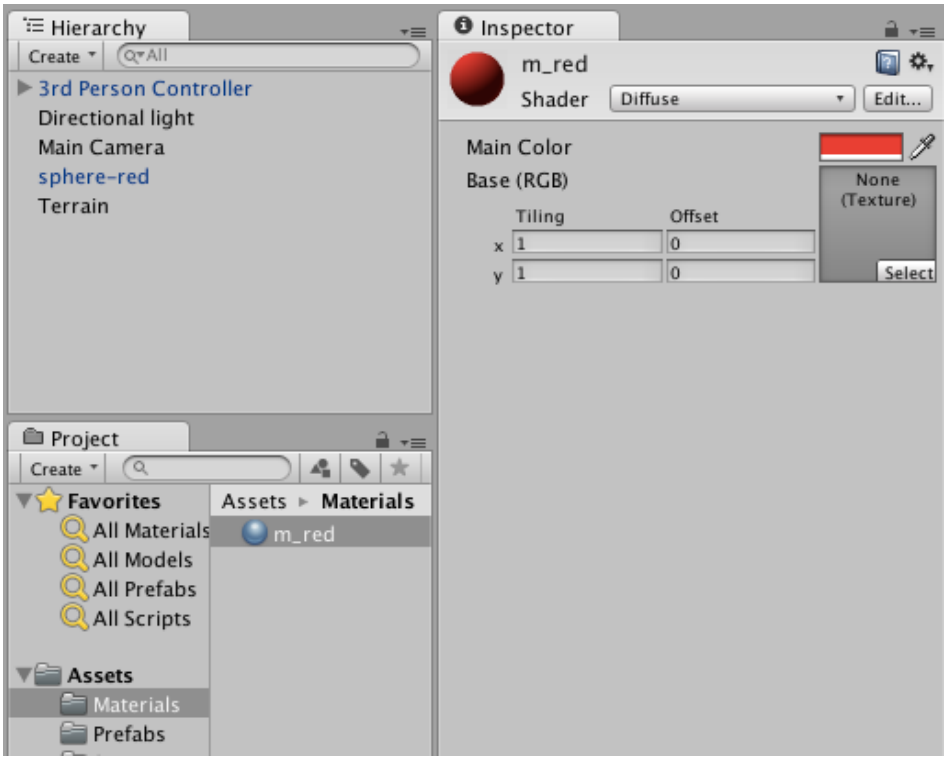
CONTENTS

1 Create a red-sphere projectile prefab (with a rigid body)

1.1 Create a red material

Do the following:

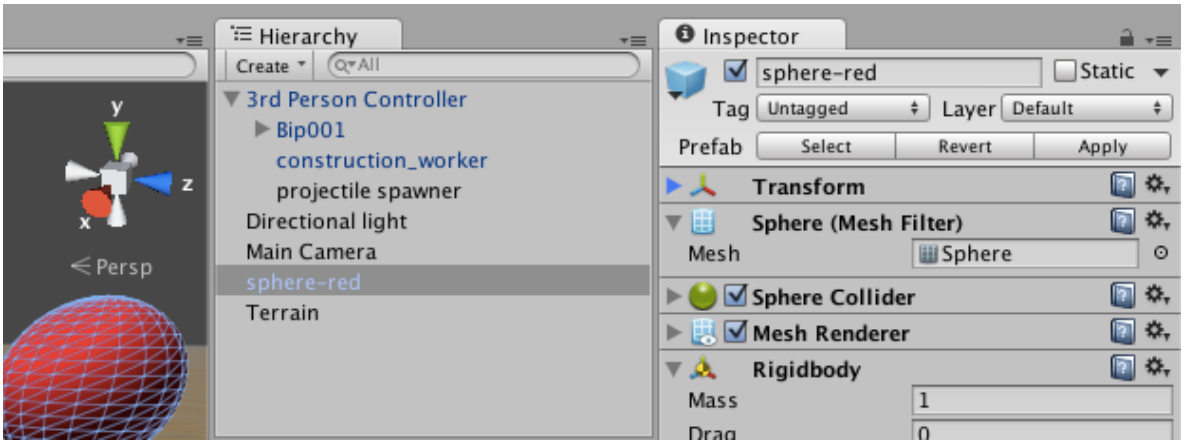
- Create a new folder Materials in the **Project** panel
- Create new material named “m_red”
- Select this material, and in the **Inspector** set the Main Color to red



1.2 Create your red sphere projectile

Do the following:

- In the **Hierarchy** add a new Sphere object, name it “Sphere – red”
- Add to the sphere a component: Physics > Rigidbody
- Drag your red material “m_red” from the **Project** panel over the “Sphere – red” gameObject



1.3 Create a red sphere ‘prefab’ based on your game object

Do the following:

- In the **Project** panel, create a new prefab named “Sphere-red-prefab”
- Drag your “Sphere-red” gameObject from the **Inspector** into your new prefab
  - The prefab should turn blue to show it has now been populated
- You can now delete Sphere-red from the **Hierarchy**

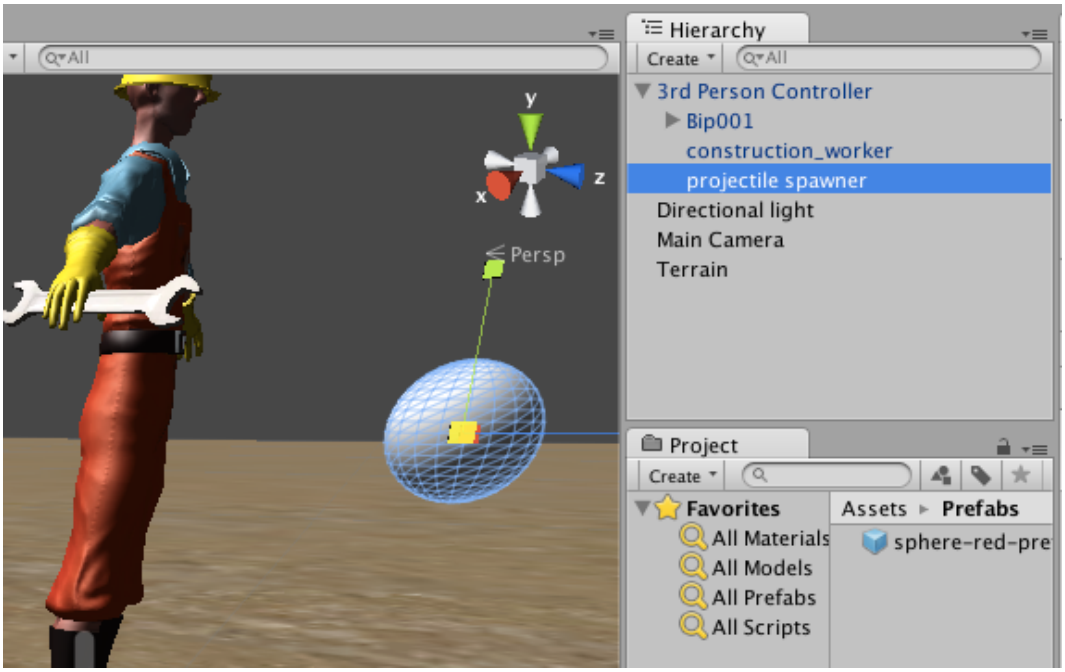
2 Create a projectile creator in 3rd Personal Controller GO

2.1 Add a small, named sphere as a child of your 3rd person controller

Let's create an object that will be the creator and point of origin for new projectiles

Do the following:

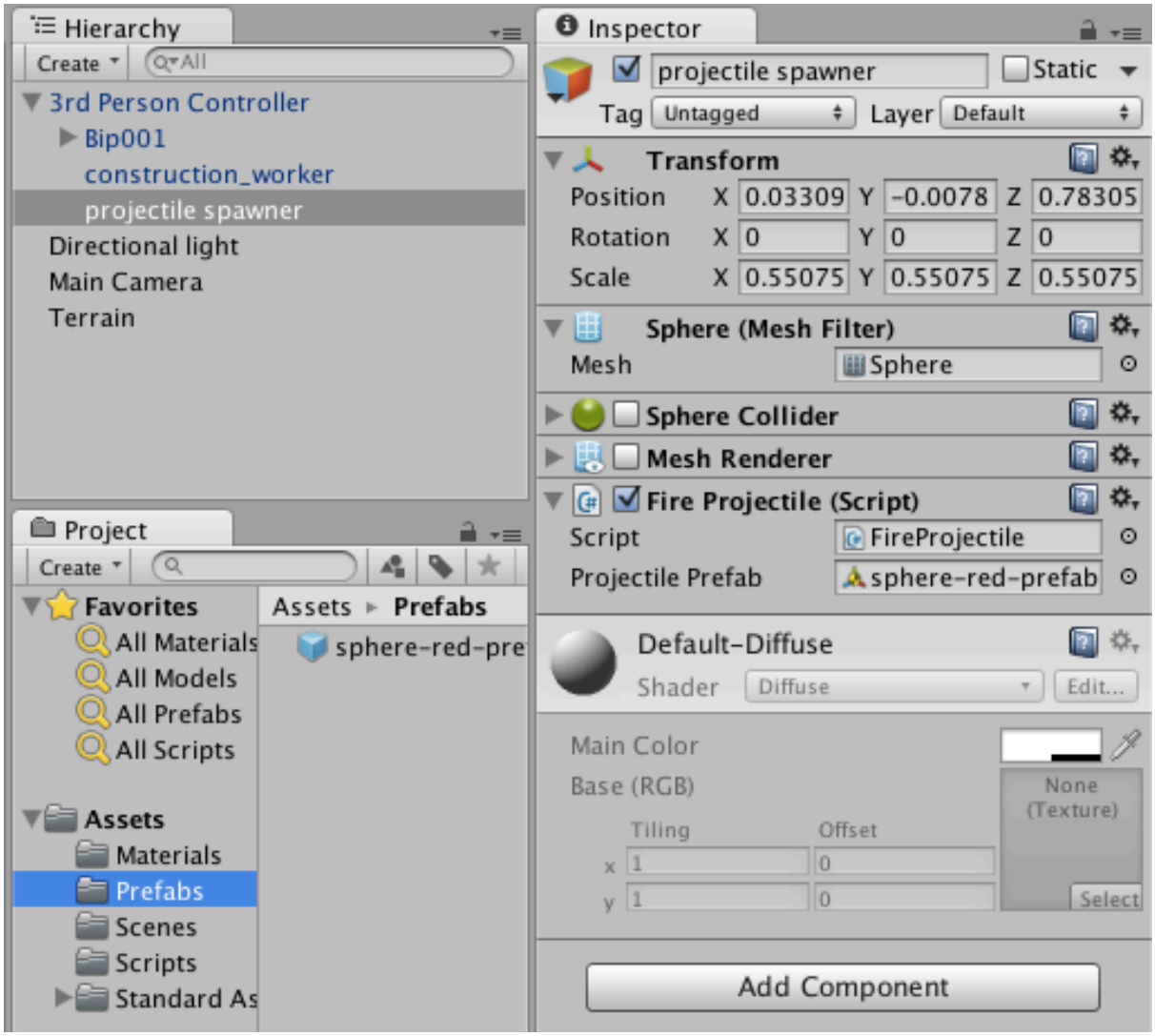
- Double click your 3rd Person Controller
  - And you may wish to zoom out a little
- Create a new sphere, located just in front of your 3rd Person controller
  - Name this sphere 'projectile-spawner'
  - Make it smaller
  - Disable the Sphere Collider component
  - Disable the Mesh render component
- In the **Inspector** drag 'projectile-spawner' into 3rd Person Controller, so that it becomes a 'child' of the 3rd Person Controller game object



2.2 Import the FireProjectile script from Matt's book

Do the following:

- Create folder Scripts
- Drag the FireProjectile script into this folder
- Add an instance of this script class as a component of your new **projectile-spawner** sphere inside your 3rd Person Controller
    - When the scripted component appears in the Inspector, drag your **Sphere-red-prefab** from your Prefabs folder into the public variable ProjectilePrefab



2.3 Playtest your game

Run the game, and fire red spheres by pressing the LEFT-CTRL key

2.4 Tweaking …

You might wish to 'tweak' the following:

- Change object being thrown to a different prefab
    - e.g. a 3D object like a knife, apple etc.
    - or a small cube, or smaller sphere etc.
- Make projectiles move faster
    - by increasing the value of projectileSpeed in script **FireProjectile.cs**
- Make the projectile fire upwards at at angle
    - By rotating the **projectile spawner object** inside 3rd Person Controller – projectiles will be fired in the direction of the **BLUE arrow**
    - So ROTATE the projectile spawner object and switch to MOVE mode to see your 3 xyz (red-green-blue) coordinate arrows
- Change the fire key
    - Menu **Edit | Project Settings | Input**, then change the Inspector value for Axis **Fire1**
- **Autodestroy** projectiles after 1.5 seconds
    - add this line at the end of method `CreateProjectile()`:
        - `Destroy( projectile.gameObject, 1.5f);`

3 C# code listing: FireProjectile.cs

```
// file: FireProjectile.cs
using UnityEngine;
using System.Collections;

public class FireProjectile : MonoBehaviour
{
	public Rigidbody projectilePrefab;
	private const float MIN_Y = -1;
	private float projectileSpeed = 15f;

	/** shortest time between firing */
	public const float FIRE_DELAY = 0.25f;
	private float nextFireTime = 0f;

	private void Update()
	{
		if( Time.time > nextFireTime )
			CheckFireKey();
	}

	private void CheckFireKey()
	{
		if( Input.GetButton("Fire1"))
		{
			CreateProjectile();

			// enssure a delay before next projectile can be fired
			nextFireTime = Time.time + FIRE_DELAY;
		}
	}

	private void CreateProjectile()
	{
	  Rigidbody projectile = (Rigidbody)Instantiate(projectilePrefab,
transform.position, transform.rotation);

	  // create and apply velocity
	  Vector3 projectileVelocity = (projectileSpeed * Vector3.forward);

	  // - need to use TransformDirection() so direction is
	  // releative to current direction camera is facing
	  projectile.velocity = transform.TransformDirection( projectileVelocity );

	}

}
```