**Project Light Game tutorial**

This is the second part of the light game tutorial to create the second important component.

**Task 1. Change to scene 2 and proceed with instructions.**

* Open Part 2 Scene.
* Click the LE\_RotaryObject and expand it, then click on the Light\_Emitter object in the hierarchy
* Double click on the “LightEmitter” script from the inspector

**Task 2. Set up variables.**

* Create two public game object variables named “projRefPoint” and “projectile”.

Should look like this:

public GameObject projRefPoint;

public GameObject projectile;

public Camera myCamera;

* Then create the final variable – a public float variable named particleVelocity.

This will act as your editable speed parameter for the projectile:

* public float particleVelocity;

**Task 3. Connect objects to components.**

* Click on the Light\_Emitter object (accessing its inspector).
* Whilst you have its inspector open, expand the Light\_Emitter object in the hierarchy to expose 4 objects.
* Click, hold and drag the “ProjRefPoint” empty game object into the ProjRefPoint GameObjects variable in the inspector to connect the Projectile’s transform and angling reference point.
* Then go to the Assets folder again and click “prefabs.
* Reclick on the Light\_Emitter object to gain access to its Inspector where the code is.
* Click hold and drag the “Light\_Particle” prefab to the “projectile” Game object variable inside the inspector code, this will activate the selection of the instantiated projectile once we get things working.
* Then look into the “LevelArea” object in the hierarchy, expand it and expand “CameraController” where you will see Camera.
* Drag “Camera” to Light\_Emitter’s “myCamera” component to connect the main camera.
* And lastly set up the speed of the Particle Velocity to 1000.

**Task 4. Code the projectile activated instantiation and force requirements.**

* Go back into the Light\_Emitter code and go down where it says update.
* Within update create an if statement called “Input. GetMouseButtonDown(0)) {} which will capture the player’s left mouse click.
* Within the {}, type “Raycast hit;” this will be your ray hit confirmation variable for later in the if statement variable to define when the mouse has been clicked on an object, the moment the ray hits the object and is registered.
* Then create the ray by going underneath and typing “Ray ray = myCamera.ScreenPointToRay(Input.mousePosition);” , that line simply just makes the screen’s clicks into a ray that the engine interprets as selection of chosen objects to be interacted with in a 3d space.
* Underneath within the first if statement, you need to make another if statement, this time it creating the physics aspect of the ray which will allow it to be used as a trigger to trigger an event which we need.
* Type inside the () “Physics.Raycast(ray, out hit, 100.0f) && hit transform.tag == “Emitter”) {}”. This is the point I explained above but in code.
* Before you continue, go back into unity, click the Light\_Emitter cube object again, click tag, click add tag and type in “Emitter” without the quotations.
* After that, go back into the LightEmitter code and enter inside the {} of the physics raycast section: “GameObject clonedProjectile = Instantiate(projectile, projRefPoint.transform.position, projRefPoint.transform.rotation) as GameObject;”
* This creates the desired light particle object by referencing its data and creating a clone of it, instantiating that clone at the transform’s position and uses the projRefPoint’s rotation to correct itself and treats it as a game object, this prevents it from breaking the original prefab.
* Underneath type “Rigidbody clonedProjectileRB;” to allow the code to have a rigidbody reference variable to the object. To know what to look for.
* Underneath that type “clonedProjectileRB = clonedProjectile.GetComponent<Rigidbody>();

clonedProjectileRB.AddForce(-transform.forward \* particleVelocity);”, this will be what the code will be looking for inside the particle prefab projectile to gain access to its rigidbody and to allow it to be pushed at a predetermined speed (being particeVelocity).

* Lastly, underneath type “Destroy(clonedProjectile, 10.0f);” to clean up cpu load.

This should all allow you to play the test level, click on the Light\_Emitter cube and fire a light projectile and it should fire in a straight line past the two cell objects when you click on it in play.

This is the end of tutorial 2, move to tutorial 3 like the previous steps.