**Project Light Game tutorial**

This is the third part of the light game tutorial to create the two important components allowing a chain reaction.

**Task 1. Change to scene 3 and proceed with instructions.**

* Open part 3 scene
* Go on the hierarchy, make sure all of the LR\_RotaryObject’s are expanded fully.
* Select Receiver and double click “ReceiveConjunction”

**Task 2. Set up variables.**

* Create two public gameobject variables called “theAirborneProjectile” and “sendConjunction”
* Then also create a Rigidbody variable called “rb”.

It should appear like this:

Rigidbody rb;

public GameObject theAirborneProjectile;

public GameObject sendConjunction;

* Open sender make a game object public variable called “RecCon”, a public script variable called “ReceiveConjunction”, two more public game objects called “projRefPoint” and “projectile” and lastly make another public float similar to the light emitter called “particleVelocity”.

It should appear like this:

public GameObject RecCon;

public ReceiveConjunction script;

public GameObject projRefPoint;

public GameObject projectile;

public float particleVelocity;

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

* Once that is set up, press save and go back to the hierarchy.
* On the first “LR\_RotaryObject”, look for the “Receiver” object and click it.
* Next go to the assets folder, reselect the first “Receiver” object and drag and drop the “Light\_Particle” into The Airborne Projectile slot.
* Next go back to the hierarchy, select “Receiver”, click and drag “Sender” into the Send Conjunction in the inspector.
* Repeat that for the next “LR\_RotaryObject ()’s below.
* Lastly expand Receiver and expand Sender. Then select Sender.
* Drag and drop “Receiver” from the hierarchy into the Rec Con slot.
* Repeat the drag and drop into the script slot.
* Then drag and drop FirePoint under sender into the “Proj Ref Point” slot.
* Open the assets folder and go into prefabs, drag and drop Light\_Particle into the “projectile” slot.
* Then change the particle velocity to 5000.

**Task 4. Code the messaging system between components for projectile activated instantiation and force requirements**

In “ReceiveConjunction” code:

* In Start(), type “rb = GetComponent<Rigidbody>();” to access the rigidbody of the Receiver.
* Below, create a private void OnTriggerEnter based on “Collider” using “other” as the code logic that focuses on the other game object.

That should appear like:

private void OnTriggerEnter(Collider other)

{

}

* Then create an if statement with “ other.gameObject.tag == “Particle” ” between () to allow the statement to target the Light\_Particle’s tag which you can check is correct by going back into the asset folder and checking the Light\_Particle’s tag.
* Next inside the {} you want to destroy the particle by typing “Destroy(other.gameObject);.
* To get console feedback you can add “Debug.Log(“Type whatever you want”);” inside of the {}.
* To actually activate an function from another piece of code remotely, use the sendConjunction variable by typing “sendConjunction.SendMessage(“ReceivedParticle”, 1); inside of the {}.

**Task 5 coding the Sender to finalize the firing process.**

In SendConjunction:

* In Start(), type “script = RecCon.GetComponentInChildren<ReceiveConjunction>();” to receive the script component of the ReceiveConjunction gameObject.
* Finally as a follow up, create a void function called “ReceivedParticle()” inside the () should be “int particleCount” to count the particles detected in an instance.
* In the {} type “Debug.LogFormat("Request Received particles, {0} continuing particle trajectory", particleCount);” to log number of particle received and the continuation of the particle’s trajectory.

Also type in “GameObject clonedProjectile = Instantiate(projectile, projRefPoint.transform.position, projRefPoint.transform.rotation) as GameObject;” and

“Rigidbody clonedProjectileRB;”

And these:

“clonedProjectileRB = clonedProjectile.GetComponent<Rigidbody>();”

“clonedProjectileRB.AddForce(projRefPoint.transform.forward \* particleVelocity);”

“Destroy(clonedProjectile, 10.0f);”

This was explained in much more detail in the Light\_Emitter tutorial 2 as the process allowing an instantiated object to spawn with force like firing a bullet. This is just repeating the same process.

Repeat this process of connecting the components for the other receivers and senders.