**Unity 2019 Common Pitfalls:**

**Best Practice For Prefabs:**

If a prefab will interact with another, have a failsafe to automatically assign references if public fields have not been manually filled in.

For example, the playerHUD prefab has a public gameobject called “Player.” Because packaging UI and 3D Gameobjects causes problems when adjusting the position of objects later on, the HUD is separated from the “Player” and does not maintain that reference.

In our void Start() statement, we can include the following lines:

If (Player == null)

{

Player = GameObject.FindWithTag(“Player”);

}

This is a safe default state, since realistically only the player’s ship will be tagged “Player”, and this line will not overwrite the “Player” GameObject that we manually assign in the event that we want this prefab to work in some other context.

**Performance Pitfalls:**

Cost of Raycast: While relatively computationally expensive, I have found that most modern systems can run several hundred raycasts per frame without any performance penalty. Raycasts are allowed, but make sure you know exactly how often your raycasts are called, and make sure raycasts stop as soon as they’re no longer needed. Above 50,000 raycasts per frame, frame rate will be below 10fps. Don’t ask me how I know this.

InvokeRepeating() Explained: InvokeRepeating() will start a new repeating process every time it is called. This means that it MUST be controlled by a dedicated Boolean if it is to be triggered in code. For example, if we want the player to take damage every 2 seconds while within a trigger collider, DO NOT DO THIS:

Void OnTriggerStay(Collider other)

{

InvokeRepeating (“DamagePlayer”, 0f, 2f);

}

OnTriggerStay is called in every physical frame in which the player is within the trigger collider. Unity’s physics engine updates every 0.02 seconds. This means that every 0.02s a new repeating instance of “DamagePlayer” is called. Instead of steadily damaging the player, the damage will ramp up exponentially as more instances of “DamagePlayer” are called and the player will die in a second or two. If something more computationally expensive like a raycast was called instead, frame will deteriorate exponentially.

In the above example, you can use OnTriggerEnter() to start the InvokeRepeating(), and OnTriggerExit() with CancelInvoke() to cancel the script.

**Importing Materials and UV’s from Blender 2.8:**

Remove Blender’s default “Light” and “Camera”: Do this either in Blender 2.8, or delete it after the prefab is made. Extra cameras are especially dangerous since Unity will sometimes try to render with them.

Do not split UV maps: Blender 2.8 allows assignment of many UV maps to a model. However, Unity will only assign the first 2: UV0 and UV1. This means that your models should be unwrapped as a single map. The good news is that Blender 2.8 has streamlined its material assignment process, and materials will automatically be baked for direct import into Unity 2019.

Scripting materials: Despite what the Unity 2019 documentation implies, Renderer.materials cannot be modified directly using the list operator “Renderer.materials[index] = material”. In order to set a new material, the original materials list must be written out as a distinct object within the code. This copy of the list can then be modified, and finally “Renderer.materials = modified list” can be used to switch the material. This is used for the interior of the animated engines of Jockey02. Note, if you create public slots for your desired materials, you need not worry about instantiating our loading.