Unity notes

Alexander Lisborg

2025-01-24

This is just notes written while I've been learning Unity. Concepts are covered in the order I researched them.

1 URP

The Universal Render Pipeline is a standard render pipeline that can laterbe exported to different systems. The URP can also be used to customize the rendering of the scene. More info needed.

1.1 Setting up the URP

This section covers how to set up the URP in a 3D project and how to convert existing materials to URP compatible materials.

1. Import the URP package to your project. This is done via the packet manager. (Window => Packet Manager => Unity Registry => Search for "Universal RP" => Install)
2. Create a URP asset. In the Unity file manager, right click in a folder => Create => Rendering => URP Asset
3. Set the Asset as renderer in the project settings. (Edit => Project Settings => Graphics => Default Render Pipeline)
4. Built in materials should now display as pink, this step will fix that. To convert existing materials to URP compatible materials, select the materials to convert. Convert them with Unity's built in conversion tool. (Edit => Rendering => Materials => Convert Selected Built-in Materials to URP)

2 Volumes

Volumes are part of the URP. With volumes, one can set global or local contributions to camera settings. "At run time, URP looks at all of the enabled Volumes attached to active GameObjects in the Scene and determines each Volume's contribution to the final Scene settings". Local volumes uses a hitbox component to determine the area which should be affected. Moving a camera into this area will override the selected effects in the camera's "settings" enabling transitions between different camera effects in different locations in the game. For example a cave could include a local volume that makes the player see a fog when entering the cave.

Properties:

Mode Either global or local. Local mode requires a hitbox.

Blend distance The distance from the volume that URP starts transitioning.

Weight The weight of the settings. Used to create blends.

Priority Layering option for Volumes. Higher priority volumes are prioritized.

Profile No idea.

3 Texture vs Material vs Shader

This section includes discussion about the differences between the three teminoligies ”texture”, ”material” and ”shader” which can be confusing when reading about game development.

3.1 Texture

A texture is an image applied to a 3D model without adding any geometry. Textures don’t include lighting information. It is the simplest of the three.

3.2 Material

A material is a broader category that defines the overall appearance of a surface of a model including it’s color, reflectivity, transparency and more. Materials include more rendering information than just an image as with the texture.

3.3 Shader

Shaders are small programs that tells the renderer how to render each pixel of the material. Shaders controll how materials are visually represented on the screen. Manipulating lightning calculations, surface shading and postprocessing effects. Shaders are the programs that take the information from the material / texture and determine how to actually render it on the screen. Even basic texture rendering need shaders to display the texture.

4 Multiplayer

This section goes through the different concepts used in multiplayer game development with Unity, how to set up a basic multiplayer environment.

4.1 Packages

Authentication com.unity.services.authentication

Multiplayer Center com.unity.multiplayer.center

Multiplayer Center Quickstart Content com.unity.multiplayer.center.quickstart ⋆ May not be avaliable in package manager, it can be installed through the Multiplayer Center when following the guide.

Multiplayer Play Mode com.unity.multiplayer.playmode

Multiplayer Services com.unity.services.multiplayer

Multiplayer Tools com.unity.multiplayer.tools

Multiplayer Widgets com.unity.multiplayer.widgets

Netcode For GameObjects com.unity.netcode.gameobjects

4.2 Demo

Building a network application is complex, Unity hides a loooot of the underlying complexity of building a multiplayer game. Since Unity is catered towards users that don’t neccesarily have networking experience, they have developed the multiplayer center as a guide for how different type of game projects should be set up.

4.2.1 Determine the relevant network architecture

- Open the multiplayer center (Window => Multiplayer => Multiplayer Center)
- Enter your game details into the tool
- Install recommended packages through the tool with the ”Install Packages” button at the bottom right. Keep the Multiplayer Center window open for the next section.

4.2.2 Netcode Demo

The rest of this demo follows the netcode solution. If Unity suggested another solution in the previous step, consider looking up information about that solution from another source as only the netcode solution is covered here. In the Multiplayer Center window, navigate to the ”Quickstart” tab. (Located at the top of the window) There should be a button to install the quickstart package, click it and install it. Navigate to the ”Netcode for GameObjects” part and click the ”Create and open scene with netcode setup” button. This creates a scene in a new folder which contains a basic demo of the netcode for GameObjects. When working with netcode GameObjects, you will work with the NetworkManager script. First of all, it includes a ”player” prefab. Every client that joins will automatically spawn a ”player”. The ”player” represents the GameObject that the client controlls. The ”player” prefab created by quickstart currently (Unity 6000.0.34f1) include:

Network Object Idk what this does, but seems to handle connection options.

Client Network Transform Manages the position of the gameobject over the network, this is used for synchronization so that when a client updates its position it is communicated to the server in a proper manner. When using this, be sure to uncheck any synching that is not needed, for example if roatation is not an option for the player, synching the rotation information just uses uneccessary bandwidth. Use unreliable Deltas option can be checked to save bandwidth and reduce lag. This makes it so that if packets are lost during transmission, they are just dropped rather than the game trying to resend the lost packages and checking if all packages have arrived. Unreliable deltas is just faster.

Client Authoritative Movement Seems to be a simple movement script. Just for demo purposes.

5 C# refrences

5.1 Singleton

```
public sealed class Singleton
{
    private Singleton()
    {
    }

    public static Singleton Instance { get { return Nested.instance; } }

    private class Nested
    {
        // Explicit static constructor to tell C# compiler
        // not to mark type as beforefieldinit
        static Nested()
        {
        }

        internal static readonly Singleton instance = new Singleton();
    }
}
```