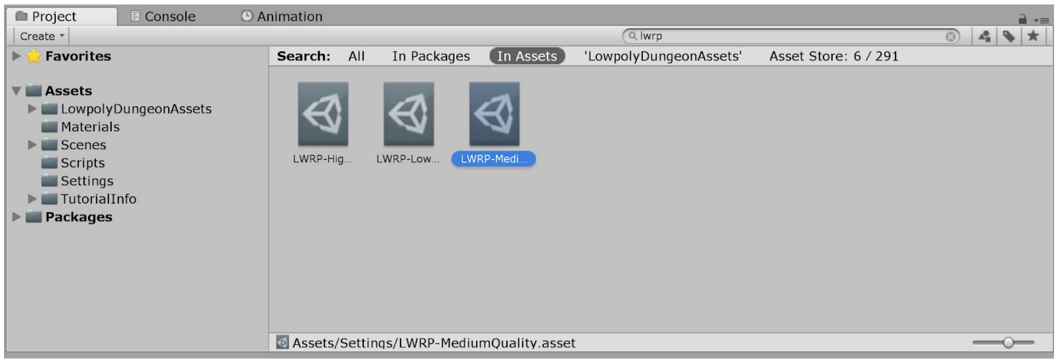
CSE 490j
Unity Introduction to Lighting and Post Processing

Lighting

Basic lighting in Unity is pretty straightforward. To start setting up, first we are going to change some of the lighting settings.

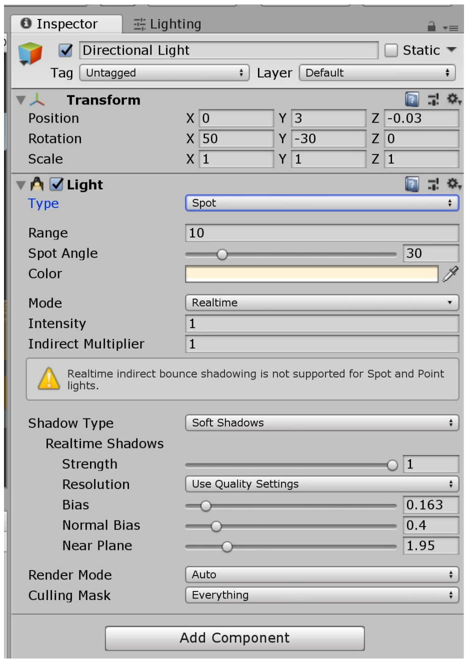
- In your project tab, in the search bar, search for “LWRP-MediumQuality.asset” and click on it



Assets/Settings/LWRP-MediumQuality.asset

- Look at it’s settings in the inspector, and set it’s “Pixel Lights” to 4.

This means, for every object in your scene, it is allowed to have 4 lights on it. If it has more than that, then other lights may not render on the object. If you want to increase this number, you can do it in the quality settings, but keep in mind that depending what platform you are using (computer, console, vr) it may hurt your frame rate (since it takes more calculations for more lights).



Directional Light Static
Tag Untagged Layer Default
Position X 0 Y 3 Z -0.03
Rotation X 50 Y -30 Z 0
Scale X 1 Y 1 Z 1
Type Spot
Range 10
Spot Angle 30
Color
Mode Realtime
Intensity 1
Indirect Multiplier 1
Realtime indirect bounce shadowing is not supported for Spot and Point lights.
Shadow Type Soft Shadows
Realtime Shadows
Strength 1
Resolution Use Quality Settings
Bias 0.163
Normal Bias 0.4
Near Plane 1.95
Render Mode Auto
Culling Mask Everything

Creating a Light

To Unity, a light is just another component on a gameobject. So let’s add one.

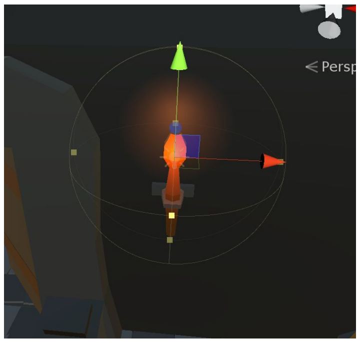
- Create a new game object called “light”
- Go to the inspector > add component > light

Take a look at some of the settings, and mess around with them a bit and see how they affect the scene. From your scene view, you can also see the are that the light affects as well as the result of the light immediately.

There are a handful of different types of lights you can use, but for the most part, there settings are the same.

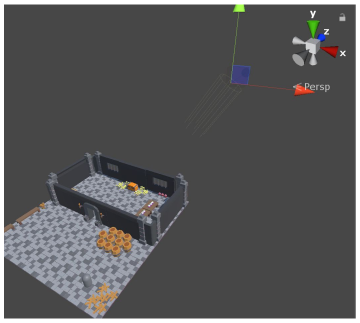
Types of Light

Point



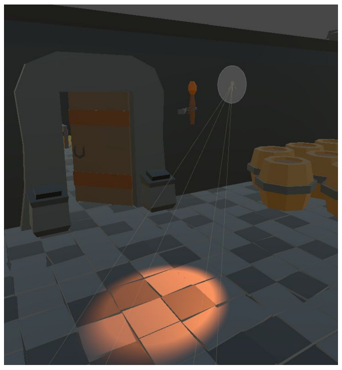
Point lights emit light from a single point in all directions.

Spot



Emits light like a spotlight, it's more cone shaped

Directional



Directional lights act like the sun. They emit light in a single direction, but across the whole scene.

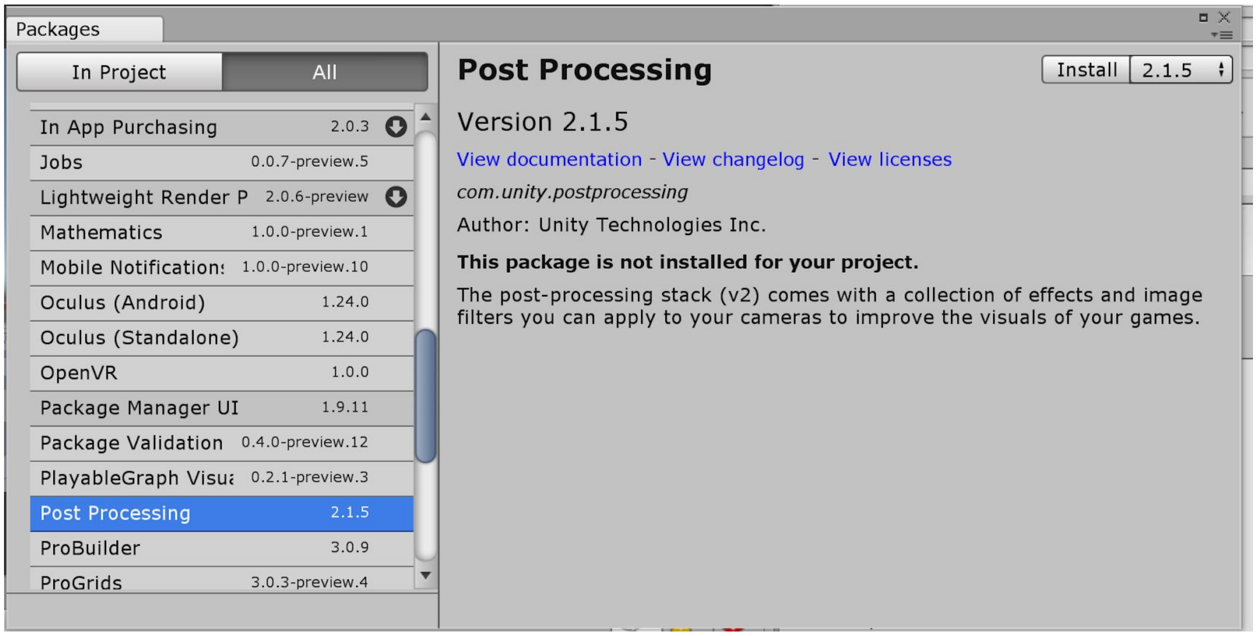
What is Post Processing

Post Processing effect that happens after everything is rendered to your screen (this isn't entirely true, but for now let's just say that it is). Think of it as a filter you can add over your scene.

Setting up Post-Processing

First we need to download the post-processing package.

- go to Window->Packages->All->Post-Processing



Packages

In Project | All

In App Purchasing	2.0.3
Jobs	0.0.7-preview.5
Lightweight Render P	2.0.6-preview
Mathematics	1.0.0-preview.1
Mobile Notification:	1.0.0-preview.10
Oculus (Android)	1.24.0
Oculus (Standalone)	1.24.0
OpenVR	1.0.0
Package Manager UI	1.9.11
Package Validation	0.4.0-preview.12
PlayableGraph Visu:	0.2.1-preview.3
Post Processing	2.1.5
ProBuilder	3.0.9
ProGrids	3.0.3-preview.4

Post Processing Install 2.1.5

Version 2.1.5

View documentation - View changelog - View licenses

com.unity.postprocessing

Author: Unity Technologies Inc.

This package is not installed for your project.

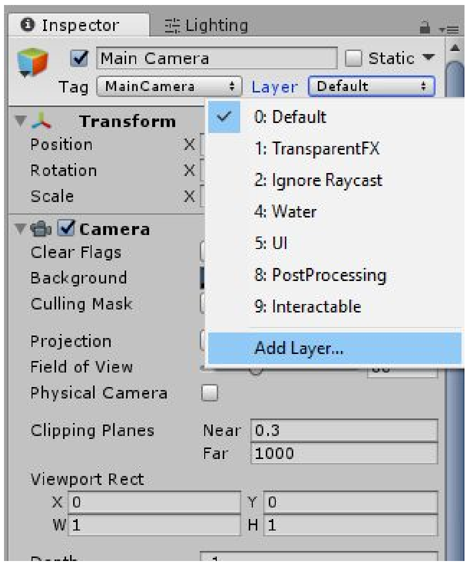
The post-processing stack (v2) comes with a collection of effects and image filters you can apply to your cameras to improve the visuals of your games.

- Click "Install"
    - Note: if at any point things things seem to be missing or something isn't lining up with what you see in the document, let a TA know. It might be an issue with which version o Post Processing you are using (as it updates pretty frequently)

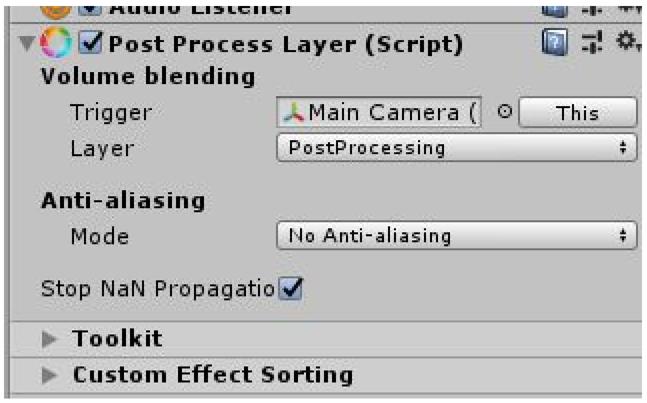
Using Post Processing Volumes

First, we want to create a layer dedicated to all our objects that will contain post processing data. Layers are a way Unity can categorize gameobjects in your scene for different purposes. So let's make one now:

- Click on a game object
- At the top you will see a dropdown labled "Layer"
- Click on it, and then click on "Add Layer…"



- On an empty “user Layer” name it PostProcessing



Audio Listener

Post Process Layer (Script)

Volume blending

Trigger	Main Camera (⊙ This
Layer	PostProcessing

Anti-aliasing

Mode	No Anti-aliasing

Stop NaN Propagatio ✓

▶ **Toolkit**

▶ **Custom Effect Sorting**

we need to create a Post Processing layer on our camera, so that it knows to look for post processing in our scene.

- Click on your camera
- Add the “Post Processing Layer” component
- In the “Layer” tab, select the PostProcessing layer you created earlier

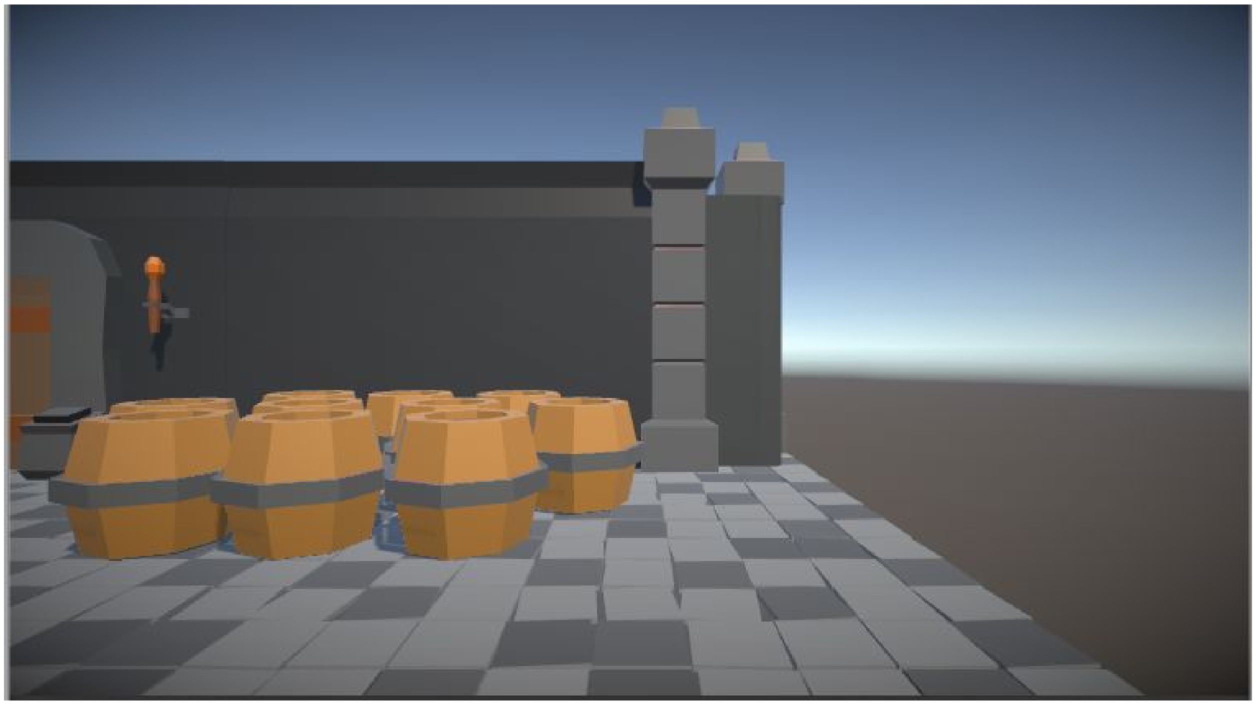
Now we have to use a post processing layer. This is where we will be actually placing our post processing effects.

- Create a new gameobject
- Name it “post processing”
- In the top corner of the Inspector, set its layer to PostProcessing
- Add the “Post Processing Volume” component to it
- For now, turn on “IsGlobal”
- Hit “new” next to the field labeled “Profile”.
- Now if you hit “add effect..” you will get a drop down menu of all the effect you can add to your scene.

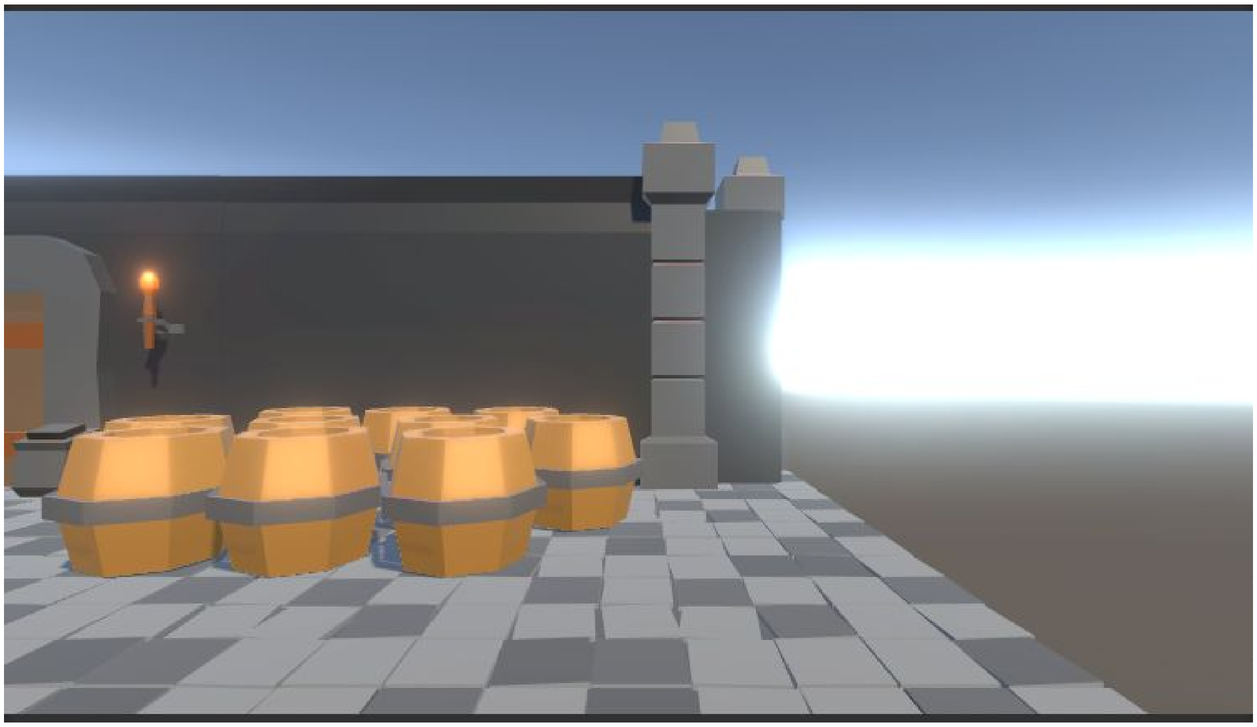
Post-Processing Effects

Post Processing, like many things in Unity, is just a matter of messing around with different effects and seeing how they work. To get you started, here are a few examples and what they do.

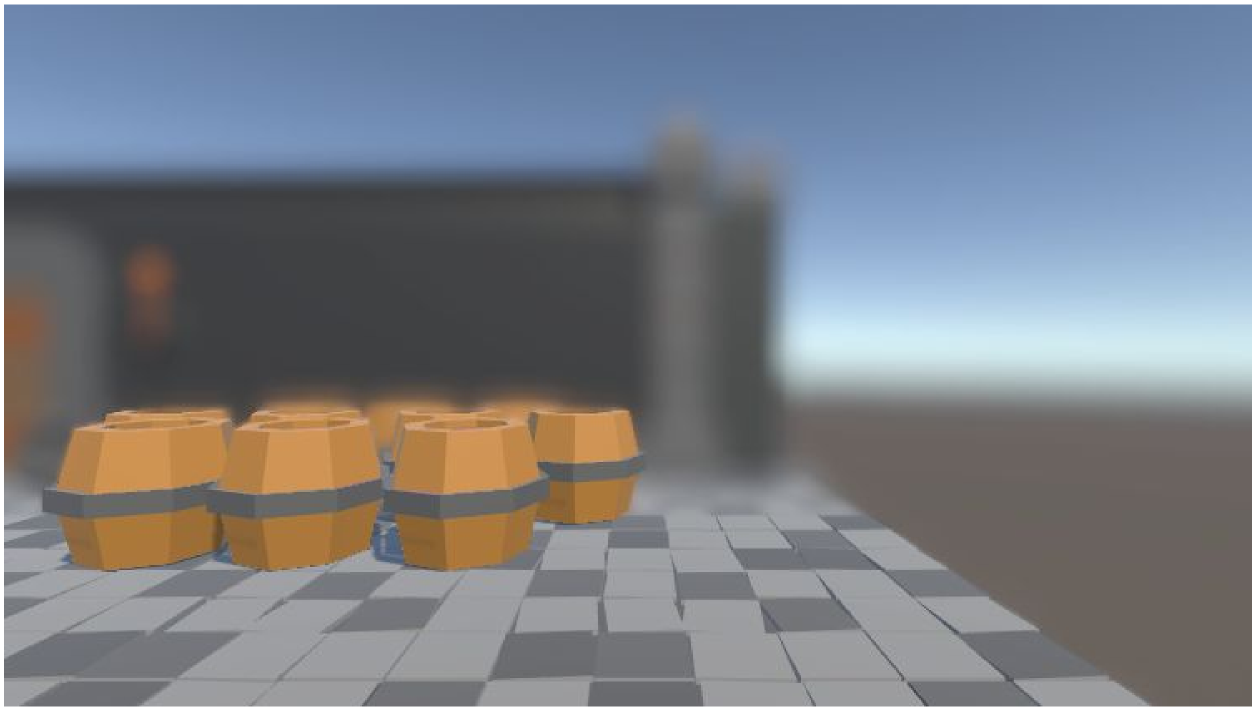
-Vignette: adds a nice faded border to the camera



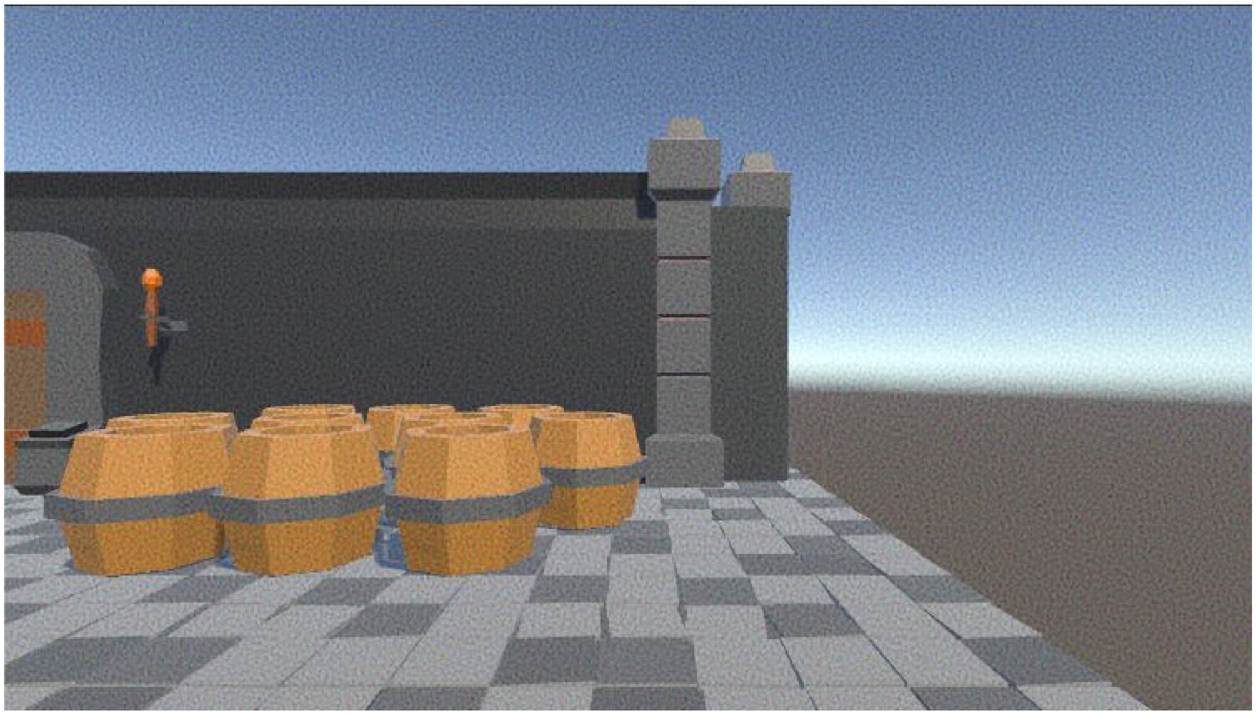
- Bloom: blows out lights and bright colors in the scene



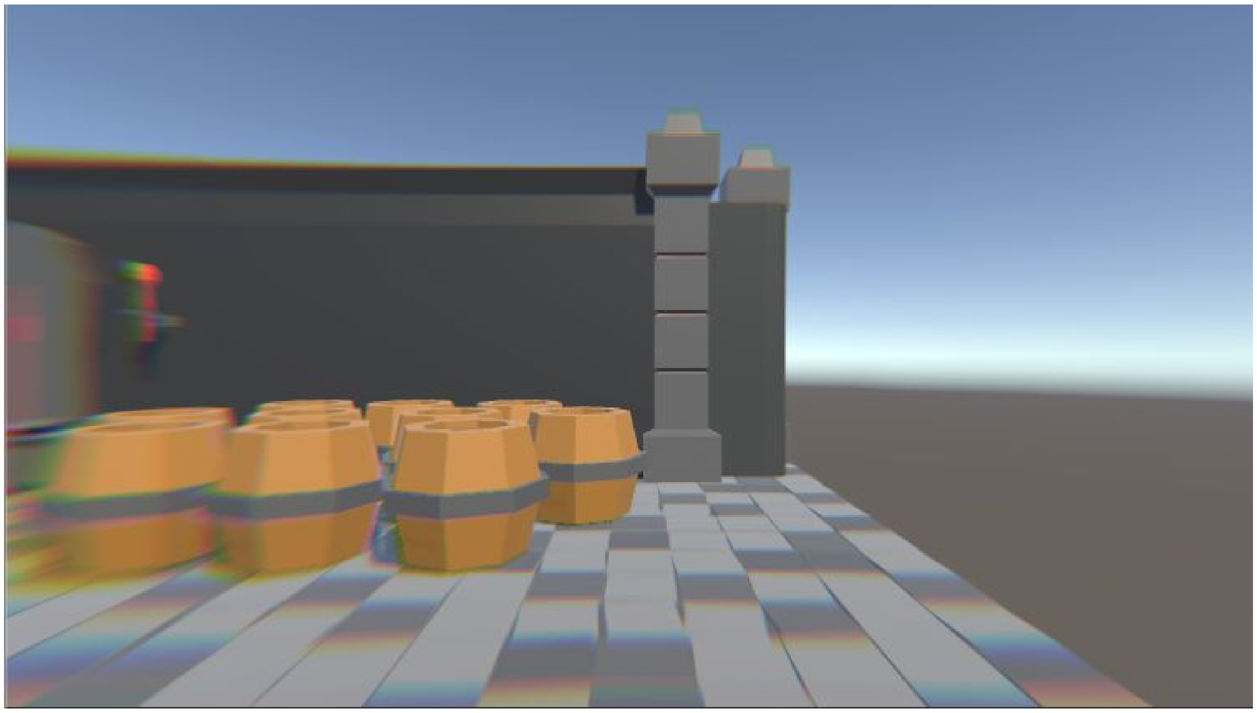
- Depth of field: can add a nice blur by distance effect



- Grain: adds grain



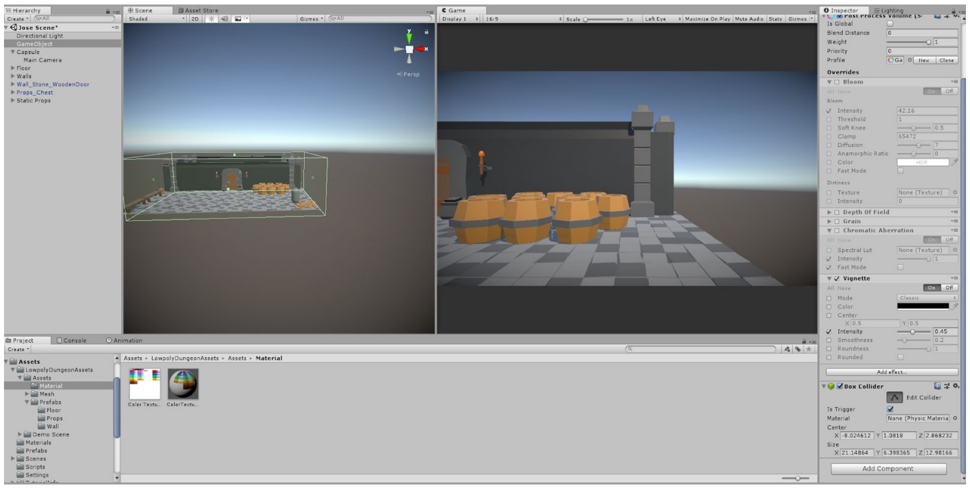
-Chromatic Aberration: I dunno how to describe this. It’s just a really cool distortion effect okay.



Post Processing Volumes

The other nice thing about Post Processing volumes is that you can have multiple volumes that can fade between each other based on the camera's position.

- Toggle off "is global" on your post processing volume
- Add a box collider
- Modify the box collider so that it fills all (or a portion of the area), as long as the player is inside it
- Set its "Is Trigger" to true



```
Post Process Volume
Is Global            [ ]
Blend Distance       0
Weight               1
Priority             0
Profile              Ga   New  Clone
Overrides
Bloom
  Intensity          42.16
  Threshold          1
  Soft Knee          0.5
  Clamp              65472
  Diffusion          7
  Anamorphic Ratio   0
  Color              HDR
  Fast Mode
  Dirtiness
  Texture            None (Texture)
  Intensity          0
Depth Of Field
Grain
Chromatic Aberration
  Spectral Lut       None (Texture)
  Intensity          1
  Fast Mode
Vignette
  Mode               Classic
  Color
  Center             X 0.5   Y 0.5
  Intensity          0.45
  Smoothness         0.2
  Roundness          1
  Rounded
Box Collider
  Is Trigger         [✓]
  Material           None (Physic Materia
  Center             X -8.024612   Y 1.0818     Z 2.868232
  Size               X 21.14864    Y 6.398365   Z 12.98166
```

So now, the post processing effect will only be applied if the camera is in the collider area.

If you were to make another post processing volume, with different settings, you can have 2 different post processing effects depending on the cameras position. You can even blend them somewhat using the blend distance setting.

Extra: Scripting for Post-Processing

Sometimes, you'll run into a situation where you need to change the Post-Processing at runtime. For example, let's say you only want to add vignette to your camera when the player is moving. To do this we will need to do a bit of scripting to modify a Post processing volume. Here is the general structure of how to do that

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

// 1. MAKE SURE TO USE THE RENDERING POST PROESSING STACK
using UnityEngine.Rendering.PostProcessing;

public class AnimatePost : MonoBehaviour
{
    // THIS IS YOUR REFERENCE TO THE POST PRCOCESSING VOLUME
    PostProcessVolume volume;
    // PARAMETER TO HOLD THE SPECIFIC POST PROCESSING EFFECT YOU WANT TO
MANAGE
    ColorGrading colorGrade;

    public float duration = 2.0f;

    float startValue;
    public float endValue;
    float t = 0;

    void Start()
    {
        volume = GetComponent<PostProcessVolume>();

        // THIS WILL TRY TO GRAB THE SPECIFIC EFFECT YOU
        // ARE LOOKING FOR IN THE POST PROECSS VOLUME
        volume.profile.TryGetSettings(out colorGrade);

        startValue = colorGrade.saturation.value;
    }
    void Update()
    {
        // WE CHECK TO MAKE SURE THE EFFECT IS PRESENT IN THE POST PROCESS
LAYER
        // TO AVOID ERRORS
        if (colorGrade != null)
        {
            t += Time.deltaTime / (duration);
            // YOU CAN GRAB A SPECIFIC VALUES FROM YOUR EFFECT LIKE THIS
            colorGrade.saturation.value = Mathf.Lerp(startValue, endValue,
t);

            Debug.Log(colorGrade.saturation.value);
        }
    }
}
```

Mood Assignment

Now that you've learned Post-Processing and lighting, I want you to make two different moods or in different parts of your scene. Use post processing volumes to split these two atmospheres. Take a snapshot of the mood you create and put it in your repository.

Extra: If you want to make a sky that matches your scene bette.