

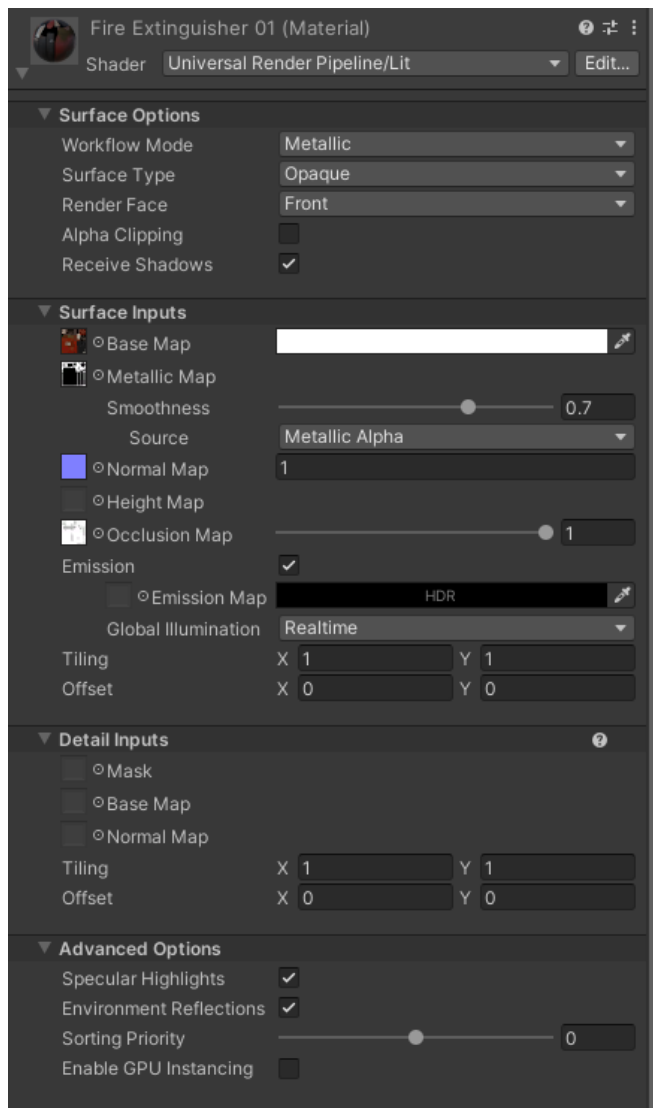
# CA1 – Rendering Foundations

Brona Keevers-Roux

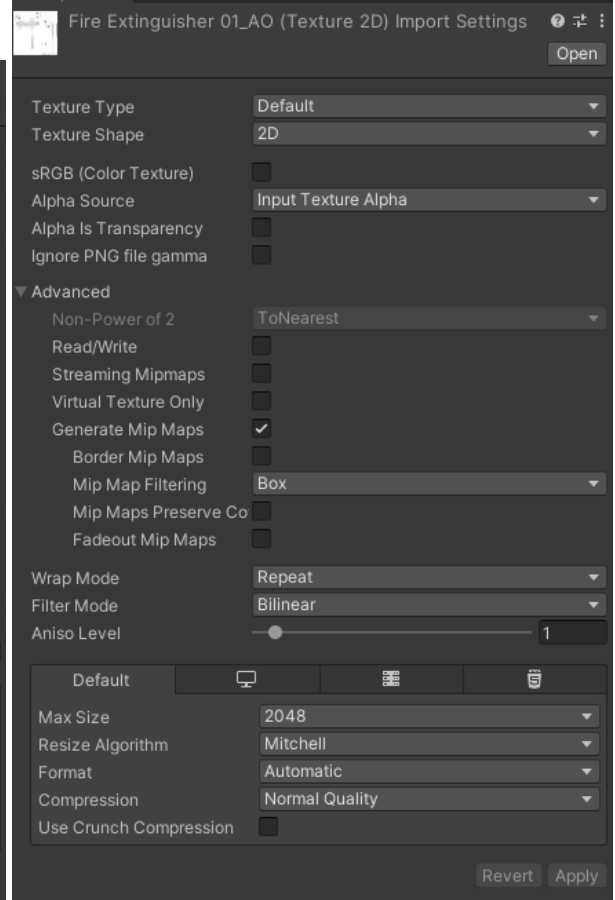
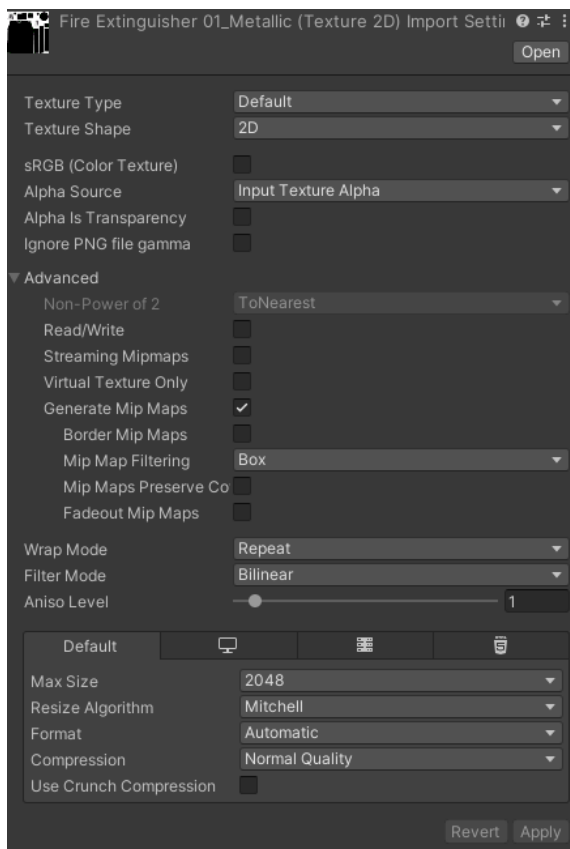
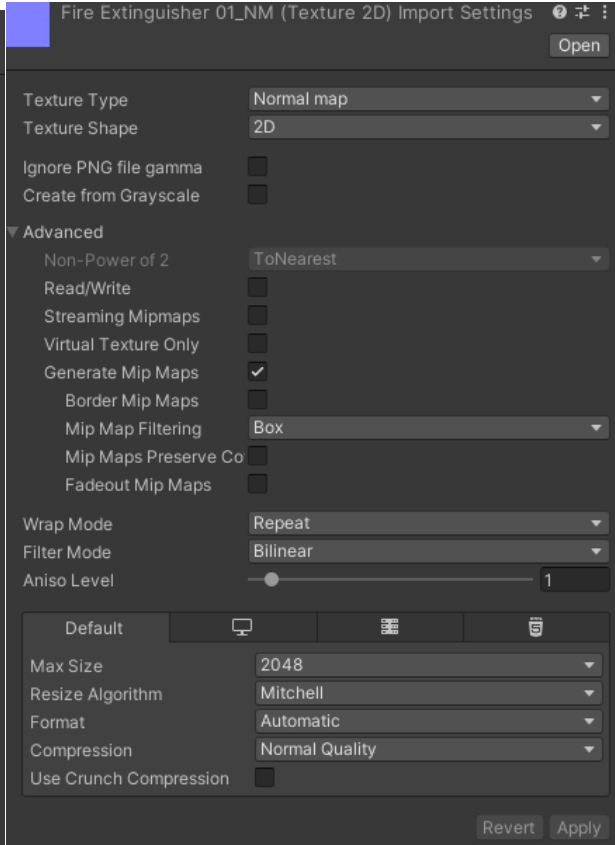
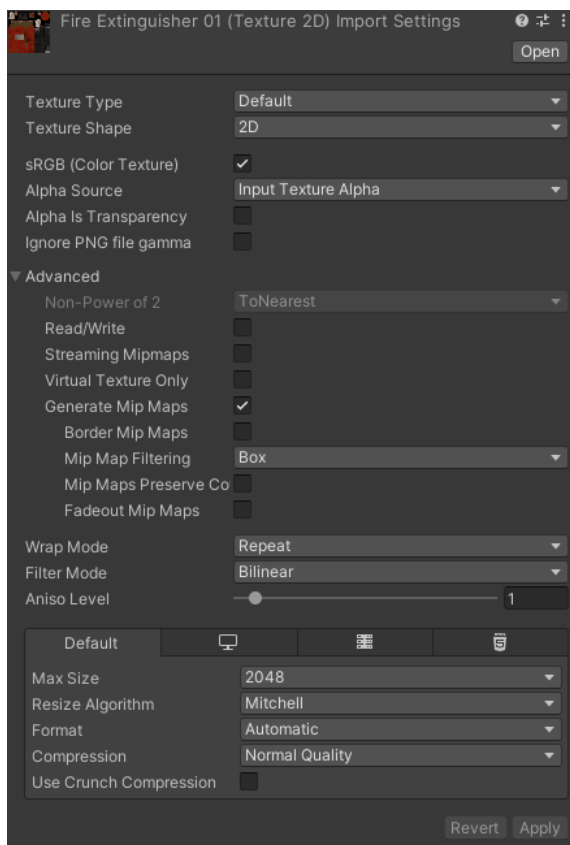
Github: <https://github.com/hillyleopard133/4th-year-game-dev/tree/main>

Video:

## PBR textures



In physically based rendering, Base Colour textures use sRGB because they represent visible colour and need to be converted properly for accurate lighting. This ensures the material looks correct under different light conditions. Data textures such as Normal, Metallic, Roughness, and Ambient Occlusion are different because they store surface information rather than colour. These must have sRGB disabled so Unity reads their values exactly as they are, without changing them. Normal maps also use the Normal Map texture type so Unity interprets them correctly as surface direction data. This ensures the lighting behaves correctly and the material appears physically accurate.

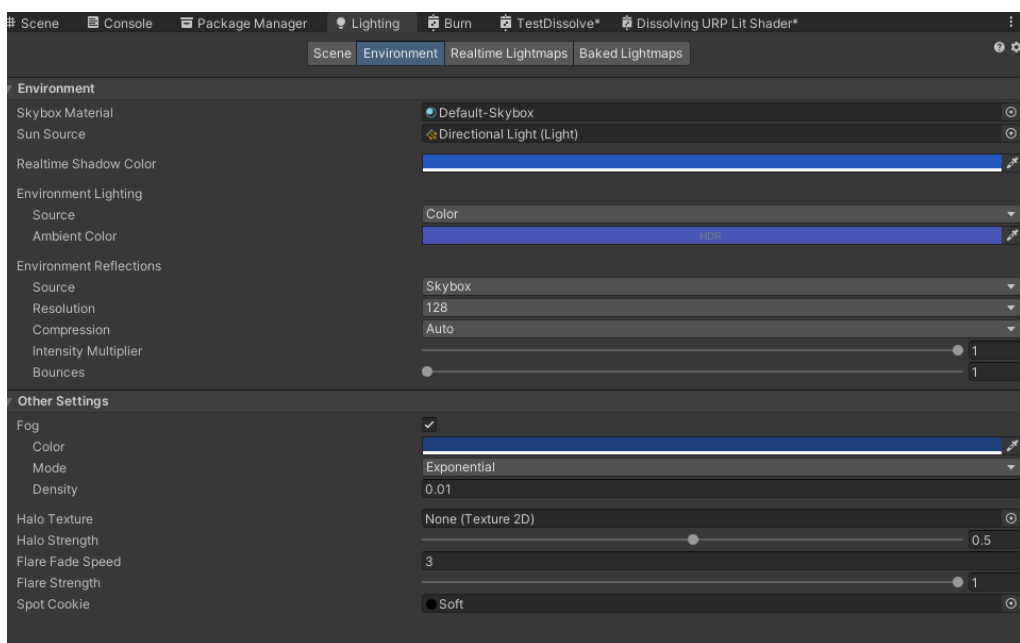
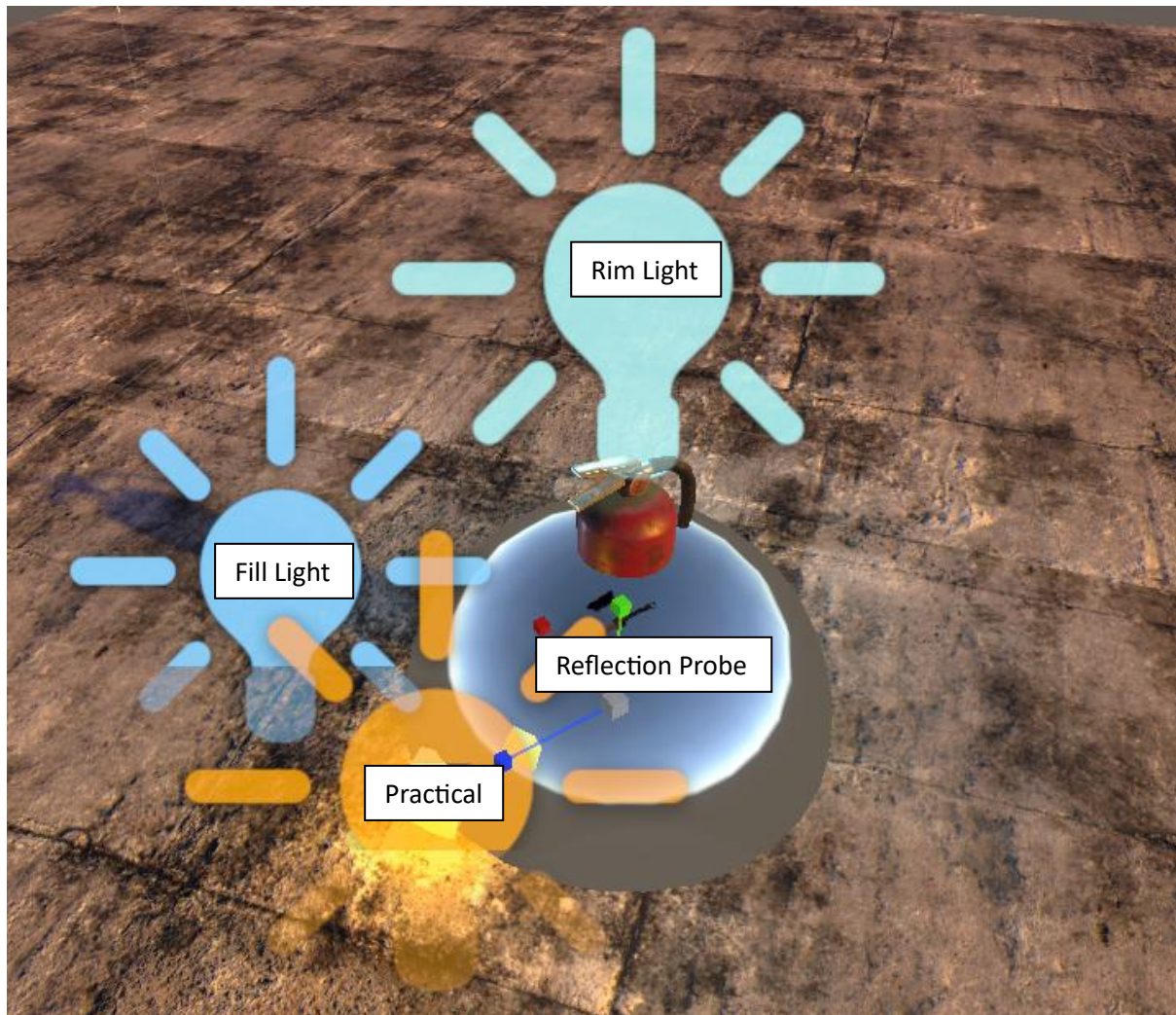


## **Shader Graph**

The burn edge is computed by creating a threshold window around the dissolve frontier using the Smoothstep node. Areas where the noise value falls within the threshold become the glowing burn, while other regions remain unchanged. The EdgeWidth controls how wide the burn band appears, and EdgeColor / EdgeIntensity allow artistic control over the color and brightness of the burn. DissolveAmount drives the animation of the effect, which is controlled via a C# script that updates the material at runtime, ensuring the burn moves naturally across the surface.

**Having technical issues with unity and currently unable to open the project, will troubleshoot and hopefully be able to take the screenshots and video tonight anyway. It will be too late for the moodle submission but I will add it to github if I can fix it. I have the shader made, all I had left to do was get the images and video. Shader is committed to the repo already anyway.**

## Scene Lighting



### **Directional Light**

The directional light represents the low sun at dusk and acts as the primary light source. It uses a warm colour and low angle to simulate sunset lighting and create long shadows. This establishes the overall time of day and provides the main directional shading for the scene.

### **Fill Light - Shadow Side**

The fill light provides soft illumination on the shadowed side of the hero asset to improve visibility and prevent the object from becoming too dark. A cool colour and low intensity were used to maintain the dusk atmosphere while improving readability. The light is limited to the hero asset using layer culling to avoid flattening the environment lighting.

### **Rim Light**

The rim light is placed behind the hero asset to create a highlight along its edges, improving silhouette separation from the background. This helps the asset stand out clearly, especially in low-light dusk conditions. A slightly cool colour was used to match the ambient sky lighting.

### **Practical Light - cube**

An emissive cube was used to represent a practical light source within the scene. The cube uses an emissive material to simulate a visible glowing object, and a point light was placed at the same position to provide actual illumination. This creates a motivated light source that improves realism and helps illuminate the hero asset and surrounding area. A warm colour was chosen to contrast with the cool ambient dusk lighting and reinforce the scene's atmosphere.

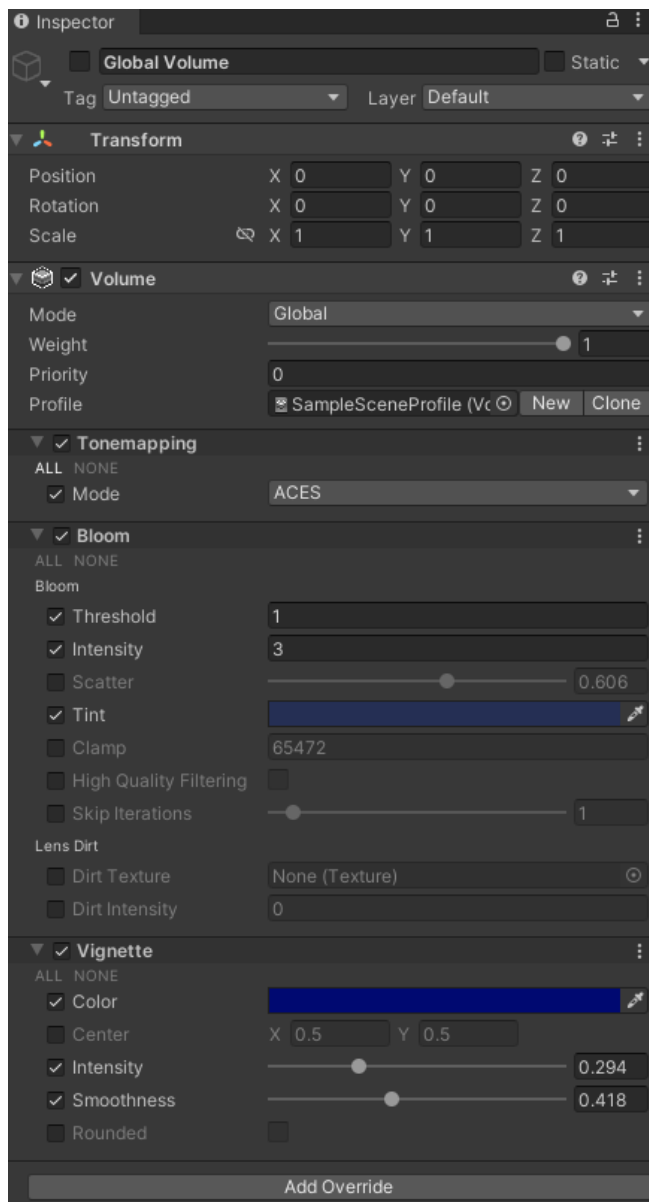
### **Reflection Probe**

The reflection probe captures the surrounding environment and provides accurate reflection data for physically based materials. This improves material realism, particularly for reflective surfaces. A baked reflection probe was used to reduce runtime performance cost.

### **Environment / Ambient Lighting**

The environment lighting was adjusted to create an open dusk ("blue hour") atmosphere. The ambient light colour and realtime shadow colour were set to a soft blue to simulate the cool fill of indirect light at dusk. A subtle blue fog was enabled to enhance depth and mood in the scene. These settings complement the warm directional sunlight and helper lights, maintaining readability while reinforcing the overall dusk lighting style.

## Post Processing



A Global Volume was added to the scene with ACES tonemapping, a subtle vignette, and moderate bloom. ACES tonemapping ensures physically correct colour grading and preserves highlight detail under the dusk lighting. The vignette darkens the edges slightly, helping to focus the viewer's attention on the hero asset without being intrusive. Bloom adds a gentle glow to bright areas, particularly the emissive cube and the directional sunlight, enhancing realism and supporting the dusk atmosphere. A before/after comparison shows improved contrast, more visually pleasing highlights, and a clearer mood when post-processing is enabled.



Before



After



## **Unity vs Unreal Material and Workflow Comparison**

### **Texture Mapping and PBR Inputs**

One key difference is Unreal expects Roughness as a 0–1 map, whereas Unity's URP may interpret Smoothness instead; this required minor inversion adjustments to maintain consistent surface response.

### **Dissolve/Burn-Out Effect**

Unreal's Material Editor terminology differs: "Opacity Mask" corresponds to Unity's AlphaClipThreshold, and "Emissive Colour" corresponds to URP's Emission input. Functionally, the visual result is similar, but Unreal requires all opacity-controlled materials to have Blend Mode set to Masked to achieve the hard dissolve effect.

### **Lighting Considerations**

For minimal dusk lighting in Unreal, a Directional Light was placed at a low angle with warm tint, and a Sky Light provided cooler ambient fill, mirroring the Unity setup. Reflection captures were added to evaluate the hero asset's PBR response. Both engines support real-time shadows, but defaults differ: Unreal tends to produce softer shadow bias by default, whereas Unity's URP requires explicit tweaking of shadow strength and colour for dusk ambience.

### **Key Differences and Pitfalls**

The most noticeable differences lie in terminology and default material behaviour. Unity separates "Smoothness" and "Metallic" inputs, while Unreal often combines them in a metallic-roughness workflow. Normal maps must be imported as Linear in Unity, whereas Unreal automatically interprets them. Shader Graph allows procedural nodes directly in the graph, whereas Unreal uses Material Functions or nodes like Noise and Lerp, which may require more connections to replicate the same effect.

### **Pipeline Standardization**

To streamline a production workflow across both engines, it would be best to maintain consistent PBR texture naming and import settings, clearly document how each map is interpreted, and standardize procedural effect parameters (e.g., dissolve thresholds, edge colour/intensity). Overall, the exercise highlights that while both engines share the same physically-based rendering principles, terminology, default material behaviour, and procedural workflows differ.