**Post-processing Assignment report**

|  |  |
| --- | --- |
| Basic requirements | Checkmark with solid fill |
| Multiple effects at the same time in any order (full-screen and polygon) | Checkmark with solid fill |
| HSL gradient | Checkmark with solid fill |
| Two-pass Gaussian blur | Checkmark with solid fill |
| Feedback in blur effect | Checkmark with solid fill |
| Polygon post-processing | Checkmark with solid fill |
| Retro game mode | Checkmark with solid fill |
| Bloom | Checkmark with solid fill |
| Lens stars | Close with solid fill |
| imGUI | Checkmark with solid fill |
| 7 extra shaders | Checkmark with solid fill |
| Depth of field | Close with solid fill |

**1. The use of post-processing in a graphics application**

**2. Specific techniques used**

**2.1 Gradient tint with changing hue**

First, two RGB colours are sent over to the GPU side. In the **Tint\_pp** shader they are converted into HSL colour space using functions. Then their hue is gradually increased by the frame time. It is also made sure that the hue doesn’t go out of the 0-360 range. Then it is converted back to RGB colour space and the gradient is calculated using the lerp function.

Finally, the scene texture is sampled and it is multiplicatively blended with the gradient colour.

**2.2 Box blur**

A simple one-pass blur effect can be implemented using the box blur algorithm. The surrounding neighbouring pixels and the pixel itself are averaged. It is unweighted, so all pixels contribute the same amount to the final image unlike in Gaussian blur.

A motion blur effect was also added to this shader (**Blur\_pp**). This technique blends the previous frames together using alpha blending. When motion blur is enabled, the alpha return value in the shader is set to 0.1. This is how much of the new frame is blended onto the existing one.

The motion blur doesn’t work the best if there are other post-processes on. It works well with most full-screen post-processes (except bloom), but it looks a bit strange with polygon and area post-processing. It is rendered before all the other post-processes. It can be switched on and off.

**2.3 Underwater**

This shader (**Water\_pp**) uses both of the previous techniques and also wobbles the screen.

First, the UVs are manipulated using sin and the frame time constant that was sent over to the GPU. Then, the image is blurred using box blur, and multiplicatively blended with a bluish greenish RGB colour sent over.

**2.4 Multiple effects**

**2.5 Gaussian blur**

**2.6 Retro mode (Pixelated)**

This shader (**Pixelated\_pp**) divides the screen into rectangles. The size of the sides of the rectangles are decided by the passed value gPixelSize, and dy and dx. Then, these rectangles are filled with one colour that is sampled from their lower left corners.

Text

Description automatically generated

**2.7 Bloom**

**2.8 Polygon post-processing**

**2.9 Additional shaders**

### 2.9.1 Edge detection and Neon

### 2.9.2 Chromatic Aberration

### 2.9.3 Paint effect

### 2.9.4 Frost effect

### 2.9.5 Posterization

### 2.9.6 Negative

**3. Improvements and extensions**

Bigger kernel size for the Gaussian blur. Variable blur strength, by increasing and decreasing the kernel. The kernel can be dynamically calculated by using the Gaussian formula.

Implementing depth of field and screen space ambient occlusion.

Implementing a realistic watercolour effect.

Being able to change the area post-process too.

More sophisticated user interface.

RGB to HSL conversion in C++