**Maths and Technology Assignment Evaluation**

Using post-processing in graphics applications can considerably improve the aesthetic and visual appeal of an application without needing to improve the quality of models or textures, lighting systems or other potentially complex elements, either by adding to the visuals (adding film grain, bloom, vignetting, etc.) or correcting/adjusting them (colour correction or upscaling).

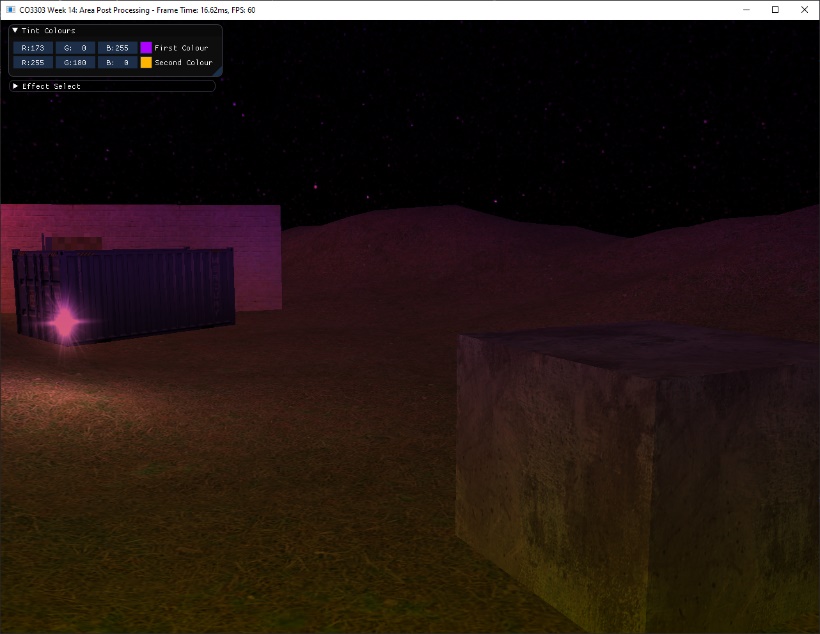
This application features a number of different post-processes that manipulate the scene in various ways. For example, the blur post-process samples the scene and offsets it horizontally and vertically before outputting this offset to the screen, creating a Gaussian blur effect that would be difficult to replicate through asset use alone - though it does this in two passes, it is still considerably less work.

Figure 2 – Two Colour Tint

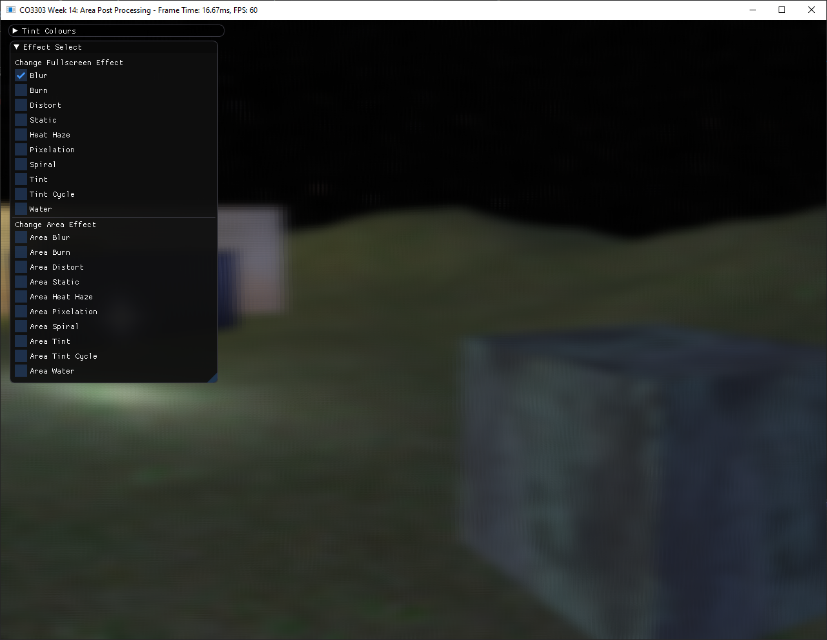
Similarly, the two colour tint affect also samples the scene texture and alters it slightly before outputting it. A selected colour is multiplied with a sampled pixel from the scene texture, resulting in the scene displaying various shades of the selected colour depending on the pixel’s location on screen, based on the texture’s original colour, (the grey wall displays a lighter shade of pink than the dark blue crate).

Figure 1 - Two-pass Gaussian Blur

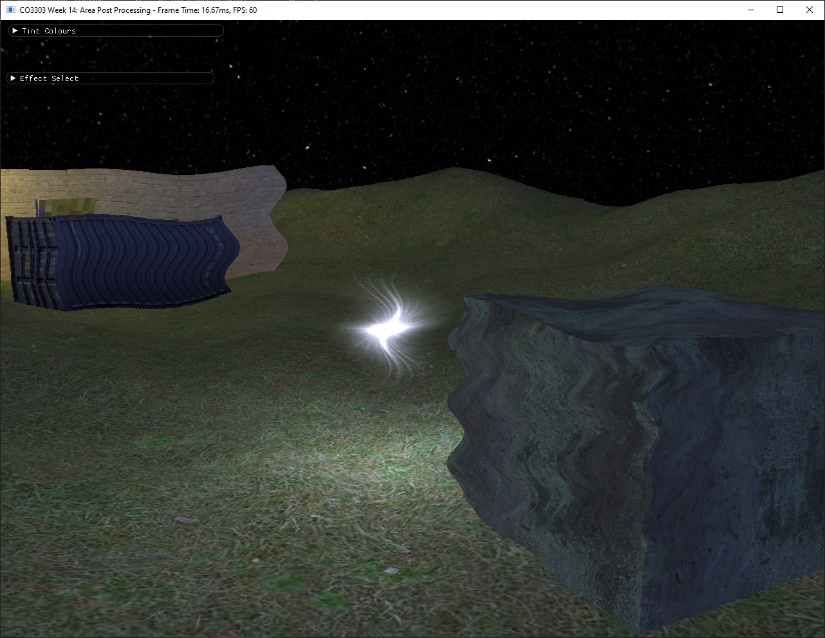
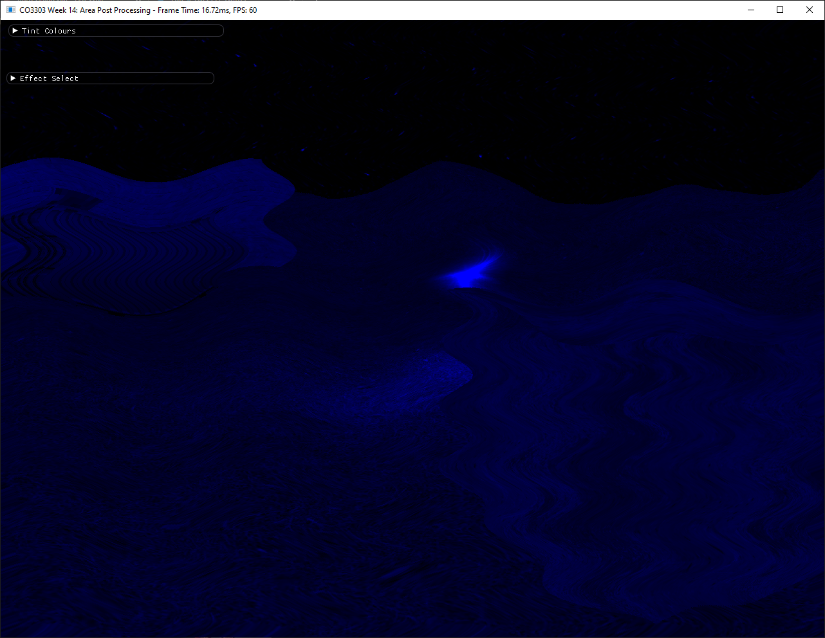
Some post-process require distortion of the scene to produce an effect. For example, the heat haze post-process and the water post-process both use sine waves to offset the scene to produce a continuous effect. By manipulating the scene in only the blue channel, a vaguely underwater look is produced.

Figure 4 - Water

Figure 3 – Heat Haze

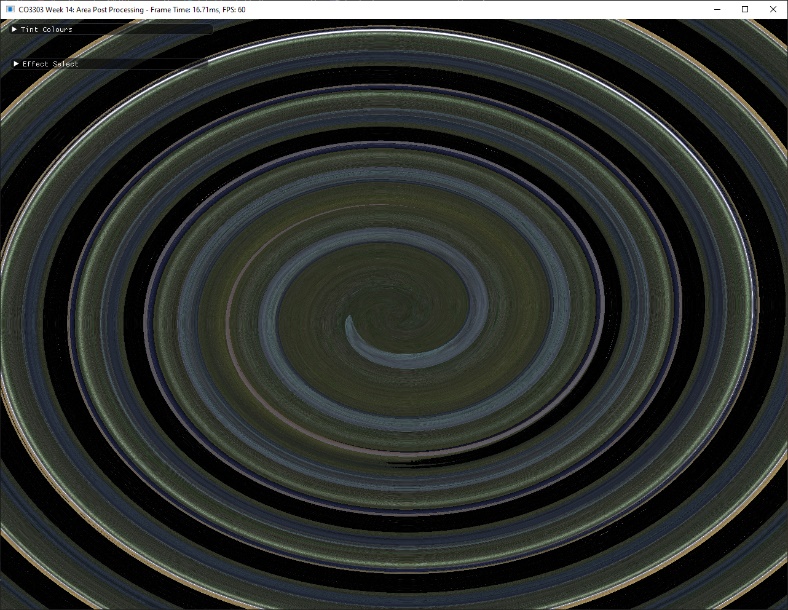
The spiral effect is also produced by manipulating the original scene texture, however it is more complex than looping sine waves. The vector from the centre of the area being manipulated, the sine and cosine of the produced spiral and a rotation matrix applied to the original vector are used to create a moving spiral effect that affects the entire area - whether it is used in the full screen, area or polygon spaces, the spiral effect will always originate in the centre.

Figure 6 – Area Spiral

Figure 5 – Full-screen Spiral

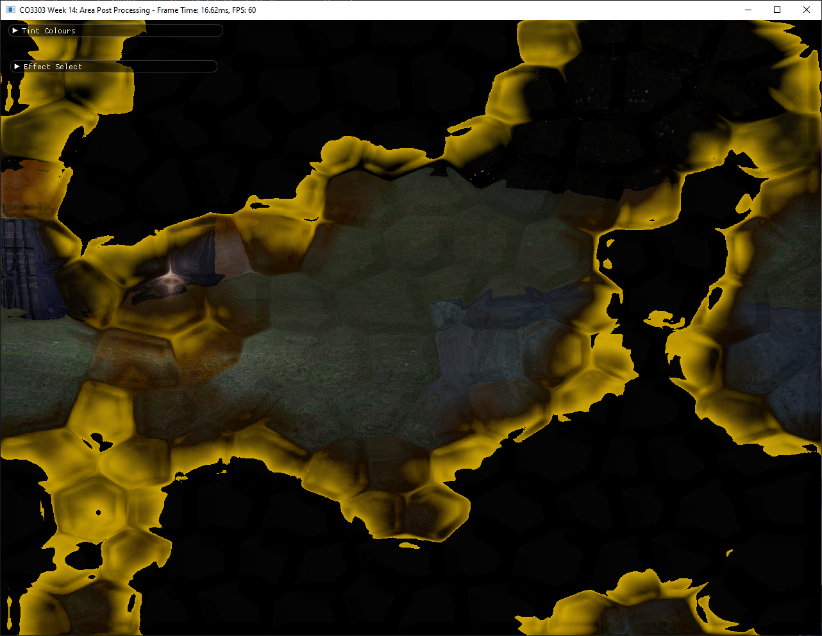
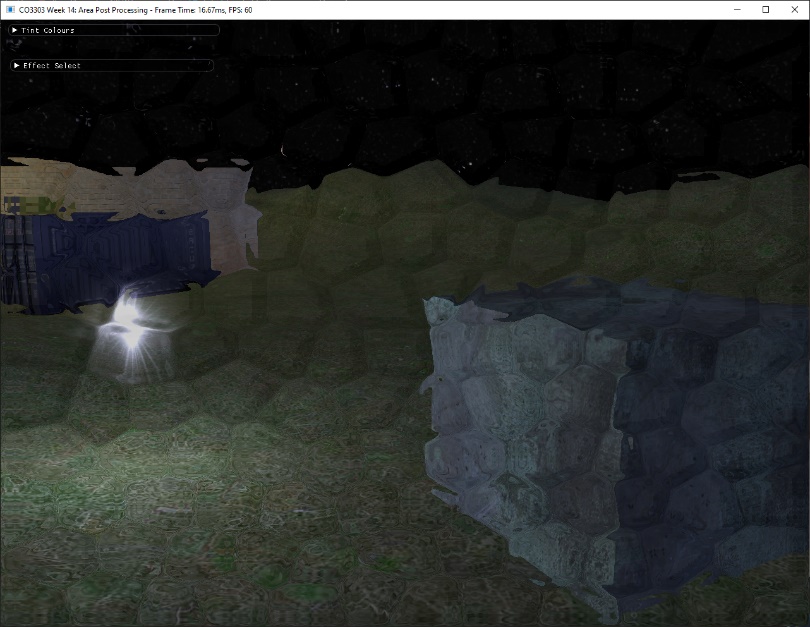
So far, all the effects mentioned have relied entirely on code to manipulate the scene. However, the burn and the distort effects rely on other textures to create their effects. The burn map and distort map are sampled in a similar way to the scene’s original texture. They are added to the outputted colour and produce effects that would be difficult to display via code manipulation alone.

Figure 8 – Burn, layered with Distort

Figure 7 – Distort



The static effect is a mix of code manipulation and external maps. The initial scene sample is averaged out to greyscale it, and this greyscale is added to the noise map to produce TV static - the ‘animation’ is a result of randomising the offset every frame.

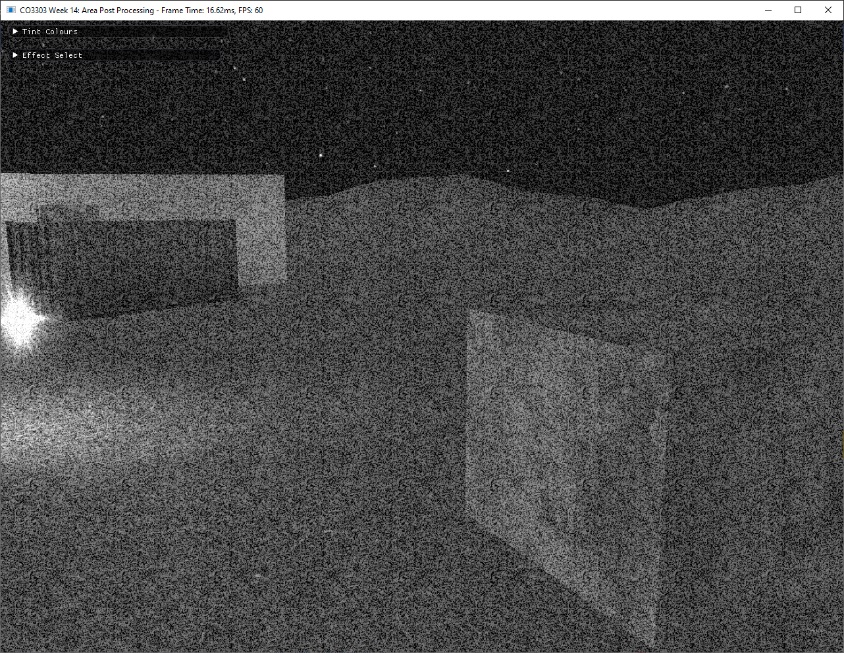
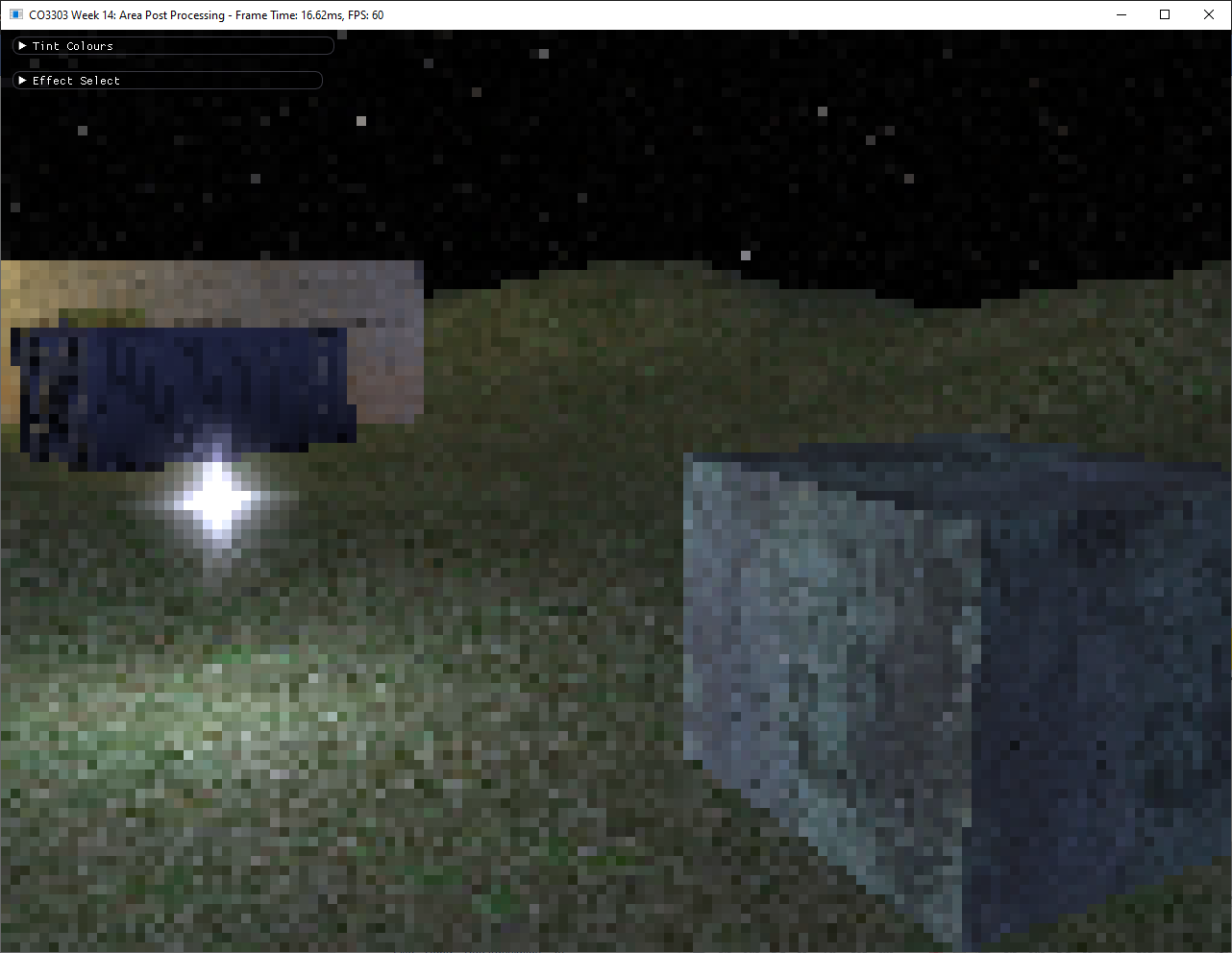
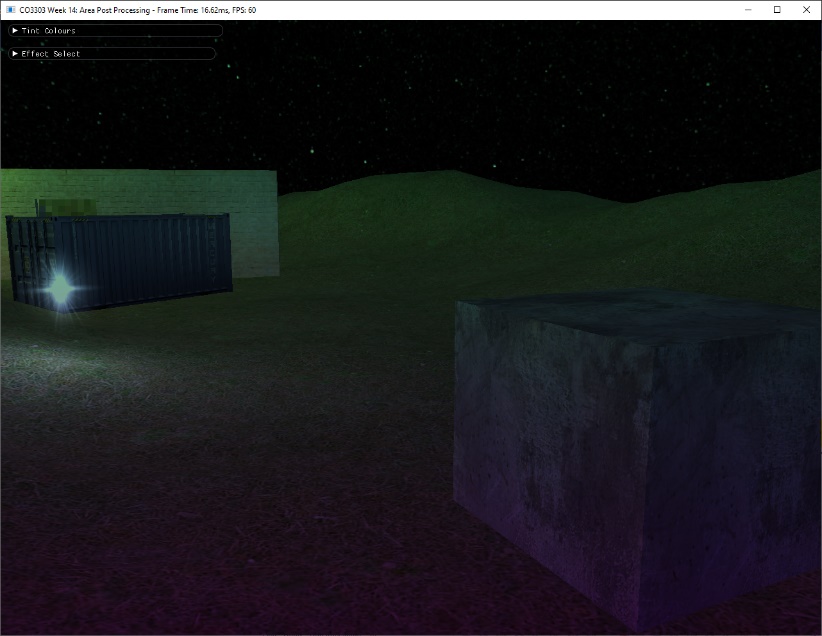
The pixelation effect takes a sample of the scene at determined coordinates and expands the colour to fill a square. The x and y coordinates are determined by a combination of the sceneUV and the viewport’s width and height respectively. Using the floor function, (which returns the largest possible integer less than or equal to an argument) the size of the ‘pixels’ is determined, and the result looks like a low-quality version of the initial scene.

Figure 10 – Pixelation

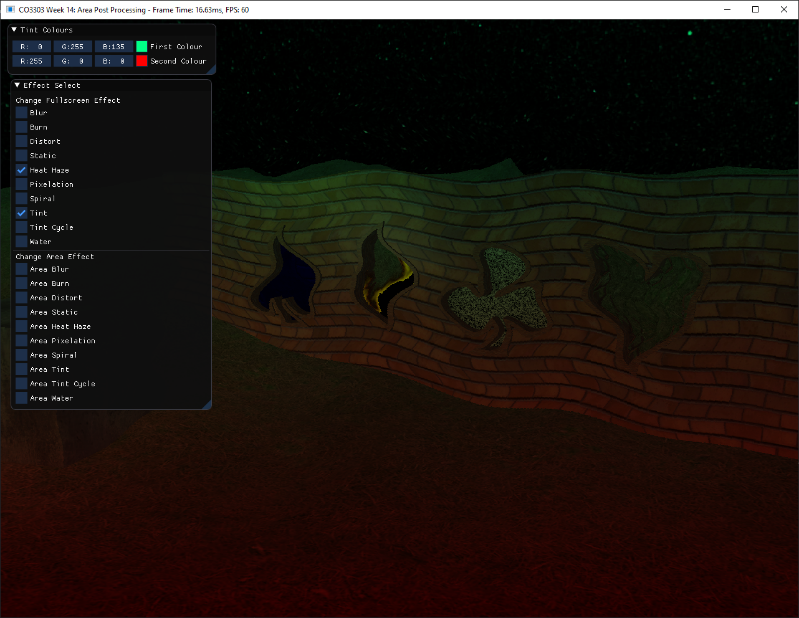
Figure 9 – Static

Finally, the tint cycle effect works the same as the two-tint effect, in that it multiplies the scene with a colour to produce what looks like a colour filter. However, this cycle constantly changes and its colours are selected via conversion from HSL to RGB. The initial colours are established in code, but the colours that follow are a result of a constant lerp between these colours as affected by a modifier.

Figures 11 and 12 – Tint Cycle at different times

Every effect Is able to run at the same time, whether it is an area effect or a full-screen effect. This is due to various lists holding applied post-processes, with listF holding full-screen effects, and listA holding area effects. The combine list ensures every effect applied before the latest effect is displayed. Effects can also be removed in any order – not in order of when they were applied.

This is because the lists are cleared and repopulated with whichever effects are set to ‘true’ or selected via the UI. If the list was not cleared, effects would be added continuously in every frame. The ImGUI allows the user to select their desired effects and manipulate the tint colours. Every main option is set to a Boolean that determines whether a post-process should be shown or not – if the Boolean is set to true, the current post-process is set to that selected option and it Is added to its relevant list.

The polygonal effects are constantly active and are visible in various in-scene windows. Their selected post-process can be changed quickly via code and they are added to the combine list, so they will be visible with other effects layered over them.

Figures 13 and 14 – Polygon effects with and without additional effects.

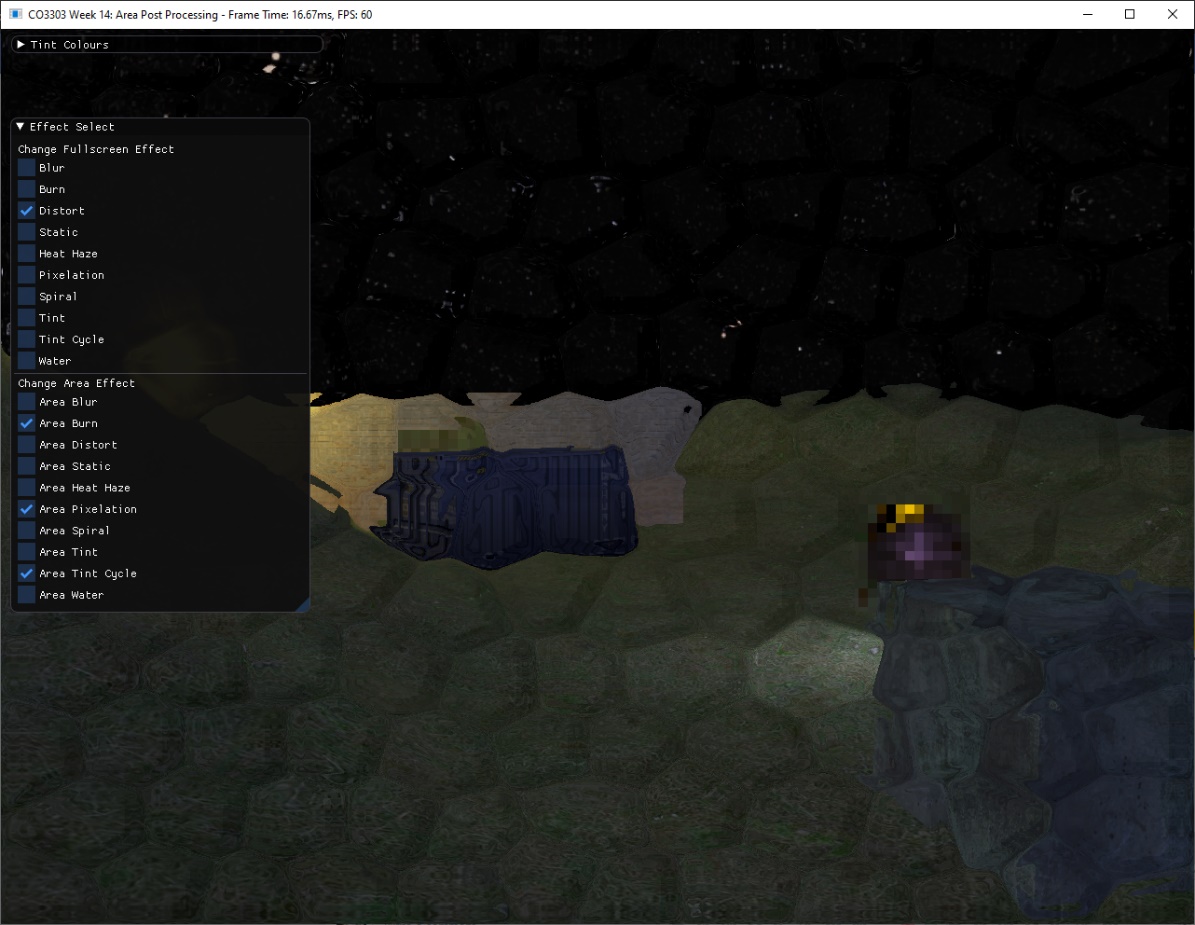


Figure 15 – Full-screen, area, and polygonal effects working at the same time.

To improve, the polygonal effects could also be changed via user input and more visual effects could have been added. Plus, variables that affect the post-process could also be decided by the user in a similar way to tint colour, for example the size of the pixels in the pixelation shader, or the speed of the heat haze effect.