A Report on Additional and Post-Processing Techniques used in a Computer-Generated Scene

Matthew Jenkinson *
Edinburgh Napier University
Computer Graphics (SET08116)

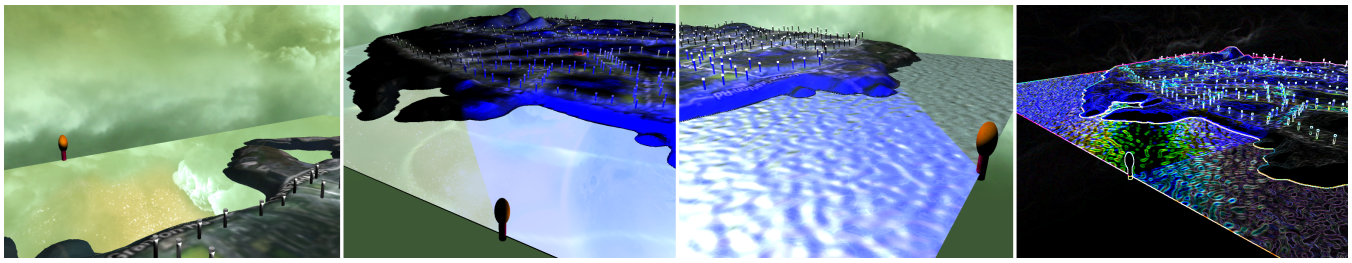

Figure 1: *From left to right: An image displaying the reflection of the sky-box in the water, an image displaying the same polygon with a texture, an image showing the sea with a normal-mapped texture, and the Sobel post-processing technique on the whole scene.*

Abstract

The aim of this report is to demonstrate post-processing techniques and some of the more advanced methods used when creating a 3D computer-generated scene. It shall explain how the techniques are used and how they are created. The audience this article is aimed at is quite broad, from those wanting to learn about these techniques and implement them themselves, to those who are just curious about the short-cuts used in creating video games; that make them look so good and run so well.

Keywords: post-processing, blur, grey-scale, edge-detection, Sobel, negative, invert, frustum, cull, reflection, sky-box, OpenGL, normal, mapping, bump, shader, optimization

1 Introduction

Including post-processing effects in one's scene makes it more visually appealing and often more realistic. In games this must be efficient as it will need to be rendered in real-time. This includes using textures for height information but not increasing the number of polygons. The effects used are as follows:

- Blur
- Negative
- Grey-Scaling
- Sobel
- Using a sky-box
- Frustum culling
- Normal mapping
- Reflection

When using techniques such as these one must be careful of the performance, so this means prioritizing the shader (GPU) over the CPU and reducing the number of calls to the shader within each frame. Decreasing the quantity of the geometry also helps, geometry can be manipulated to look as if it is more detailed than the number of polygons it has. The level of detail in the scene is worth considering as well, if there are a lot of objects that are unimportant then there is no need to make them detailed, however in some games this is an option that can be toggled.

Implementing shader-based techniques can be difficult as they cannot be debugged without the correct tools. Using methods found online may also be problematic as they are not always well-documented.

The performance of some effects are limited by the user's hardware so optimization is very important. For example using an AMD Radeon R9 390 GPU one can get frames per second of over 100, using an NVidia NVS 510 display adapter one can achieve a frame-rate rendering the same scene of around 10. However most modern graphics cards will give a stable 60 FPS rendering this scene.

2 Overview of Techniques

- **Blur:** Sampling the colours of pixels after the scene has been rendered on the first pass then averaging these colours and changing the surrounding colours. A process known as box filtering.
- **Negative:** Another post-processing technique that requires the scene to be rendered at least once before applying it. Simply put, the resultant colour is one minus the original colour, i.e: (1 - red-component, 1 - green-component, 1 - blue-component).
- **Grey-Scaling:** This takes the average of the 3 components of the colour of each pixel and sets that colour to be that average.

```
float avg = (out_colour.x + out_colour.y +
             out_colour.z) / 3.0;
vec4 grey = vec4(avg, avg, avg, 1.0);
```

- **Sobel:** This technique finds the edges of shapes within the 2D image that makes up the geometry and highlights them. It is a type of edge detection algorithm [Overvoorde 2015]
- **Sky-Box:** Essentially this is a large polygon that only uses 12 triangles. It is typically larger than the scene and moves with the camera so that it never reaches it. [dev]

*e-mail:40163650@napier.ac.uk

Figure 2: The 6 images used to create a sky-box arranged in a net.

- **Frustum Culling:** In an effort to improve efficiency and performance in one's program one may use frustum culling, this is used to detect whether or not a point (or box around an object) is within the camera's field of view, if it is not then no calculations (lighting or otherwise) are performed on it and it is not rendered. [ARF 2011] Typically the dot product is used to test whether a point is in front of or behind a plane.

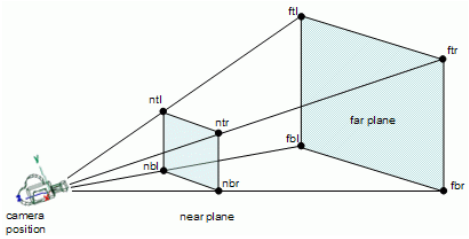

Figure 3: An image of the frustum created by the camera.

- **Normal Mapping:** These RGB textures are used to calculate how the surface is lit, giving the illusion of depth. This is done by setting the normals of the surface with this texture. [Kenwright 2014]

- **Reflection:** This is where at least one texture is reflected in another's surface. In OpenGL in a GLSL shader it can be calculated like so (this is for a reflection of a sky-box in water):

```
vec3 R = reflect(view_dir, normalize(normal));
vec4 reflect_colour = texture(skybox, R);
reflect_colour.w = 1.0;
```

In this case R is the texture coordinate of the location of the pixel that is seen in the reflection. Unfortunately this technique is inaccurate but it is fast and it looks the part. In physics the reflection is calculated as follows:

$$angleOfReflection = 180 - angleOfIncidence$$

In the scene however, the rays are not emitted from the light source but from the camera and so the calculation is a little different. We also use the normal of the texture so that we can create a specular effect. Also if we wish to be accurate we would need to calculate the light's refraction which would greatly affect the performance.

3 Where the techniques are used

Normal mapping and reflections are used in the water to give it more depth and realism, it looks especially good with specular lighting.

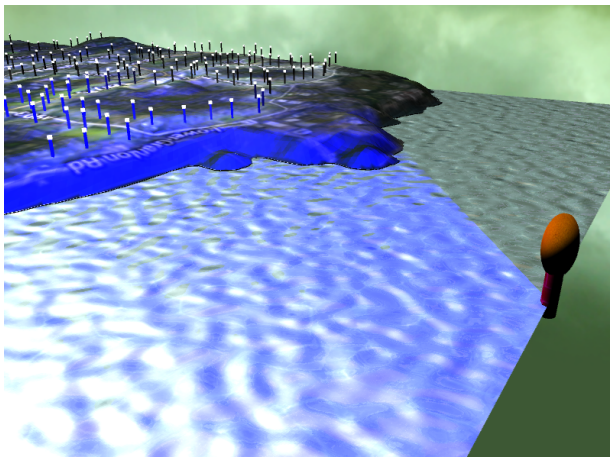

Figure 4: An image of the sea with the normal map applied. Some of the texture and reflection is lost with the bump map.

The sky-box surrounds the entire scene and moves with the camera but does not rotate with it.

Frustum culling is only applied to the light-house as the technique is only an approximation and would not give that much of an improvement to performance if applied to the entire scene. The reason for this is that all of the street lights are contained in one mesh as an optimization and if the frustum culling affected one of them then it would affect all of them.

All of the other techniques are applied to the whole scene one at a time but the current technique can be changed in real-time.

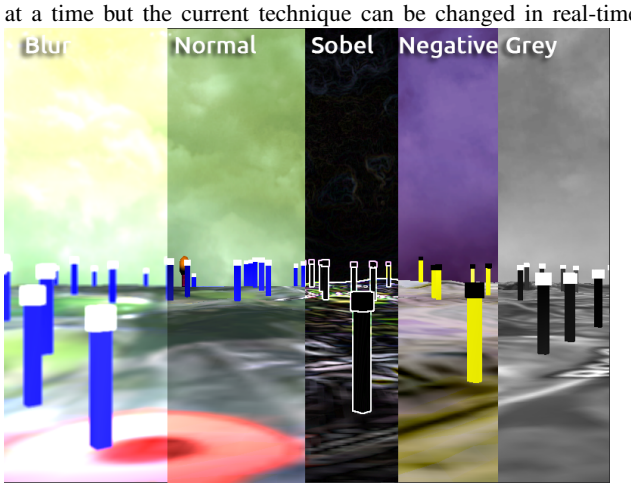

Figure 5: An image demonstrating the different effects at the same point within the scene.

The scene was also optimized in two more ways, the water - which was originally part of the mesh that made up the terrain - consisted of many triangles, now it is not generated with the rest of the terrain and only consists of two polygons. The street-lights were originally cylindrical and had many vertices in all planes, at first they were changed to not have more than one triangle in the y-plane around their edges, this improved performance greatly, at this it was decided to decrease their number of polygons further by making them into cuboids and therefore increase the performance further.

4 Conclusion

What one should take away from this is that these techniques aren't that complicated and are not too resource intensive for the graphics card to achieve. Some techniques are more useful than others, for example normal mapping is beneficial to every scene wanting to achieve some sort of realism but grey-scaling is only useful to those wanting an artistic look to their scene. There are other techniques out there that do different things, lens flare and depth of field can be good to mimic an actual camera for example. What is most important is that whatever you create must have some artistic coherence and integrity, know what effects to use and know when to use them. Use these effects wisely.

References

ARF, 2011. Geometric approach [to frustum culling] - extracting the planes. 2

DEV, O. Tutorial 25: Skybox. 1

KENWRIGHT, B., 2014. Bump mapping. 2

OVERVOORDE, A., 2015. Opengl framebuffers. 1