Real-Time Approximation of Displacement using Parallax Mapping

Shader Programming

Calum McManus
Computer Games Programming
De Montfort University, DMU
Leicester, Leicestershire
P15211272@my365.dmu.ac.uk

***Abstract*—Increasing realism in 3D applications can be difficult due to the limitations of modern graphics hardware. Parallax mapping is an extension of bump and normal mapping that can be used simulate depth and extra detail in surfaces that would otherwise require the mesh to contain it through additional polygons. This study will investigate the theory and implementation of two forms of parallax mapping along with the variables that affect them.**

Keywords— OpenGL, GLSL, Graphics, Parallax, Displacement

I. Theory of Parallax Mapping

Parallax mapping is an extension of traditional bump mapping that simulates additional dimensions and details to a surface (Sherrod, 2008, Kaneko et al. 2001). This can be implemented using one of two greyscale bump textures, height map or depth map (inverse height map), which are used to calculate an offset in the coordinates that are used to sample from the other textures, such as diffuse, specular and normal, that are associated with a mesh (Sherrod, 2008).

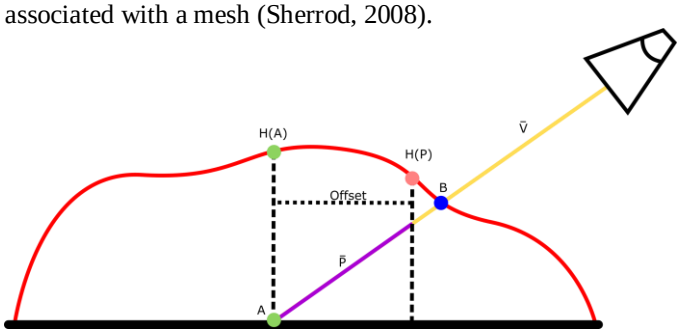

Figure 1. Calculating coordinate offset with a height map

When using a height map, the effect will look as if parts of the mesh are being extruded out of the surface. Figure 1 shows how the offset of the coordinates are calculated, where the black line represents a height value of zero and $\overline{V}$ is the view direction. Normally the point at A would be the part of the texture that is seen, but in the case of this example the point on the black line directly below $H(P)$ is sampled instead. This is calculated by scaling $\overline{V}$ by the height value at $H(A)$ which reduces the length of $\overline{V}$ by $\overline{P}$. The offset is now the different between A and the the point below the end of the new $\overline{V}$.

Depth maps work in a similar way, but instead of extruding detail outwards from the surface, it simulates parts being extruded inwards. Unlike a height map, the black line at the base now represents a *depth* value of 1.

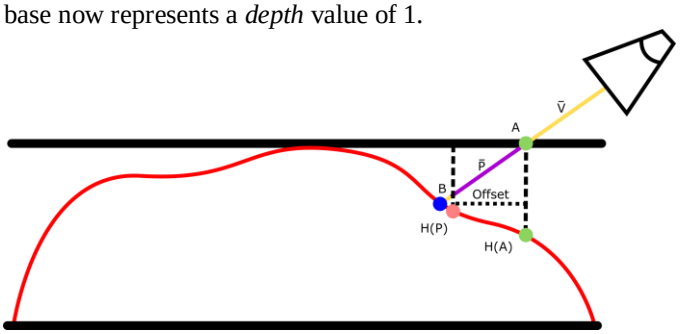

Figure 2. Calculating coordinate offset with a depth map

As in Figure 2, the offsets are now calculated using the black line at the top which represents the surface instead of the line at the bottom. This time $\overline{P}$ is calculated by *subtracting* $\overline{V}$ from the coordinates at A, and then the offset is obtained from the difference between A and the point *above* the end of the extended $\overline{V}$.

```
//*********************************************//
mat3 normalmatrix = mat3(transpose(inverse(View*Model)));
vec3 tangent = normalize(normalmatrix * VertexTangent.xyz);
vec3 normal = normalize(normalmatrix * VertexNormal);
tangent = normalize(tangent - dot(tangent, normal)*normal);
vec3 bitangent =  cross(tangent, normal) * VertexTangent.w;
mat3 TBN = mat3(tangent, bitangent, normal);
//*********************************************//
```

Figure 3. Creating a TBN matrix in GLSL

If either of these methods are followed, the result would be incorrect as the calculated offset is not lined up with the texture coordinates. To avoid this, the calculations must be done in tangent space. Similar to normal mapping, a TBN (tangent, bi-tangent and normal) matrix can be used to convert the view directions from model space to tangent space (Lengyel, 2001). The TBN matrix by its self will transform values from tangent

space to model space, but this can be swapped around using the transpose() function in GLSL.

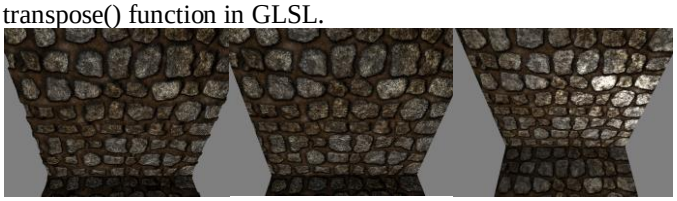

Figure 4. Parallax vs Normal vs Diffuse Specular

Now the calculations have been done in the correct space, the result seen in figure 4 can be produce which clearly has more depth and detail and normal mapping alone.

II. Theory of Steep Parallax Mapping

Steep parallax mapping is a further extension to the standard parallax mapping algorithm. Instead of taking a single sample, it takes multiple at equal intervals to better estimate the values on a sloped section of the bump map (McGuire and McGuire, 2005).

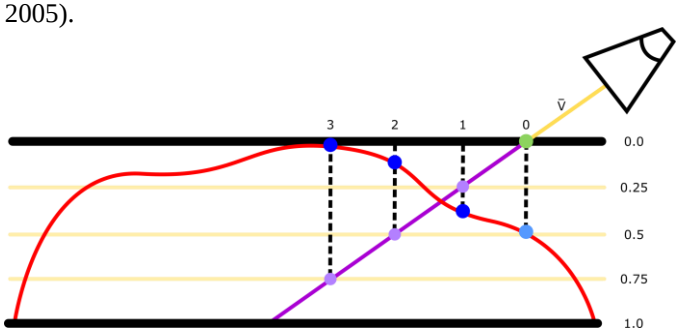

Figure 5. Sampling depth with multiple layers

Each layer is sampled starting at the top and working down, if the layer's depth is less than the depthmap's value then $\overline{P}$ does not intersect with the surface. This process is repeated until a point that is intersecting with the depthmap is found. In the example figure 5 is showing, point 1 does not intersect, so it continues to point 2 which does, the coordinate offset is now the difference between point 0 and point 2.

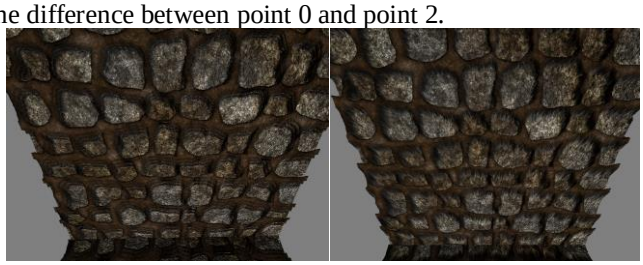

Figure 6. Steep Parallax Mapping, 4 layers vs 20 layers

Increasing the number of layers used in Steep Parallax Mapping can have huge impact of the visual effect, with the fewer layers becoming very visible as seen in Figure 6. Theoretically the number of layers used could be increased exponentially to get a better result, but instead the sampled values of the depthmap could be interpolated across multiple layers to smooth it. This is called interpolation enhancement or Parallax Occlusion Mapping (DeVries, n.d.).

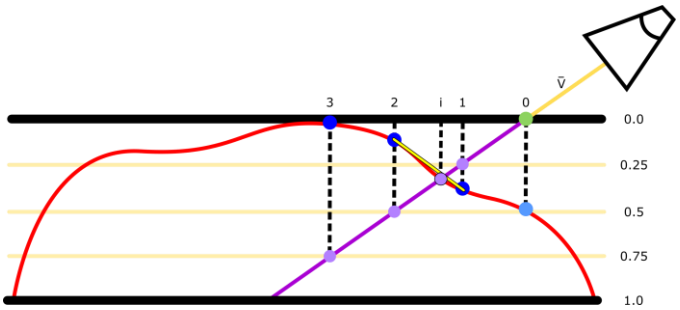

Figure 7. Linear Interpolation of depth values

Like in figure 5, the example is figure 7 shows that the offset would be between point 0 and point 2, but this time there is an extra point on the $\overline{P}$ vector labeled i. Point i is the closest point on $\overline{P}$ that intersects with the depthmap between points 1 and 2. This value is now sampled to give a significant more accurate approximation compared to the normal steep parallax algorithm.

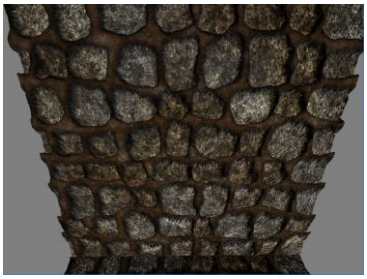

Figure 8. Parallax Occlusion Mapping with 8 layers

With this in place, the same steep parallax effect can be achieved, like in figure 8, with much smoother slopes and with fewer layers.

III. Viewing Angle and Scale

There are two main variables that affect parallax mapping, the viewing angle and the height/depth scale. The depth or height scale work by multiplying the sampled values by this variable to reduce or increase the effect map (McGuire and McGuire, 2005).

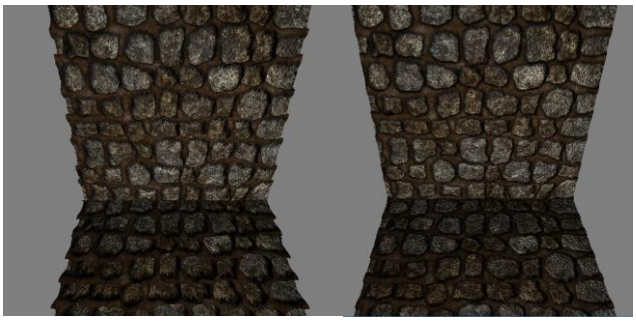

Figure 9. Steep parallax mapping, scale 0.08 vs 0.02

Figure 9 demonstrates the effect the scale can have as well as how small the scale variable can be. The scale value will often

be small as the values sampled from the height or depth maps will normally be too big and cause an undesired result.

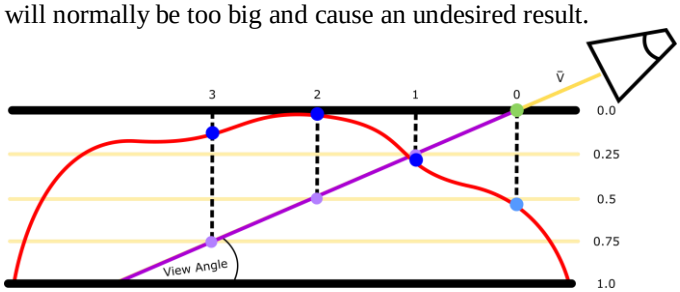

Figure 10, Steep Parallax Mapping with lower viewing angle

The view angle refers to the angle between $\overline{V}$ and the surface. It is important to account for this angle when calculating any form of parallax mapping due to the potential increase in the offset. In figure 10, the camera has been moved down to decrease the viewing angle which has increased the offset compared to figure 5 quite substantially. The larger the offset the more inaccurate the result can end up being. In the case of steep parallax mapping, this can be reduced by again increasing the number of layers but that also increases the number of GPU (graphic processing unit) cycles used. At any time, the number of layers needs to be the minimum it can be while still giving the same result, this can be done using the view angle to increase and decrease the number of layers dynamically (DeVries, n.d.).

```
const float minLayers = 8.0;
const float maxLayers = 32.0;
float numLayers = mix(maxLayers, minLayers,
    abs(dot(vec3(0.0, 1.0, 0.0), viewDir)));
```

Figure 11. Layer calculation in GLSL

Getting the dot product between the view direction and the surface normal, the value can be used along with the GLSL mix function to clamp the number of between a minimum and maximum value, as seen in figure 11. In this example there will always be at least 8 layers to ensure a good result when looking from above the surface, but it will be increased up to 32 layers as the value increase (as the view angle from figure 10 gets smaller).

IV. COMPARISON AND TESTING

Using the different types of parallax mapping each come with their own pros and cons. Regular parallax mapping is easy to render and won't have a huge impact on performance, but the effect is not as noticeable as steep. Steep gives a better result but only as a result of many more computations per frame.

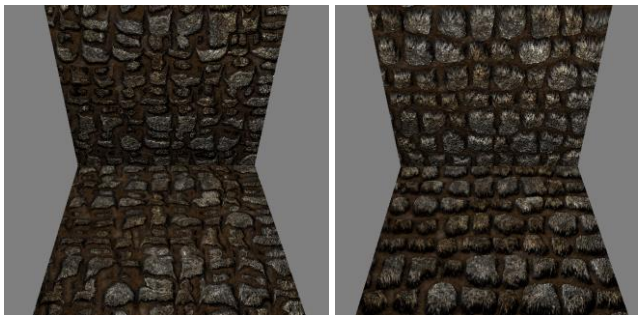
Figure 12. Not Steep vs Steep, 0.1 scale

When identifying which technique to use, the purpose of it as well as the texture being used need to be taken into consideration. As in figure 12, regular parallax mapping struggles with larger scales, but for surfaces with only small and shallow detail, this be a great addition to a 3D environment.

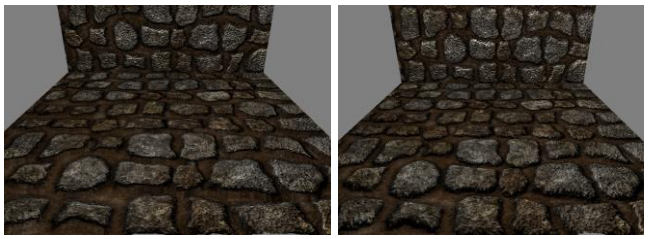
Figure 13. Low view angle, Not Steep vs Steep, 0.08 scale

The way each type reacts to different view directions is also different. At a lower angle, as in figure 13, non-steep mapping begins to stretch and there are noticeable graphical artifacts. On the other hand, steep mapping has in some places looks more realistic at the low angle. There is no noticeable stretching or artifacting either. However, in both of these cases it is clear that the low angles start to lessen the effects and the surface begins to look flatter as the camera gets closer to being on level with the plane.

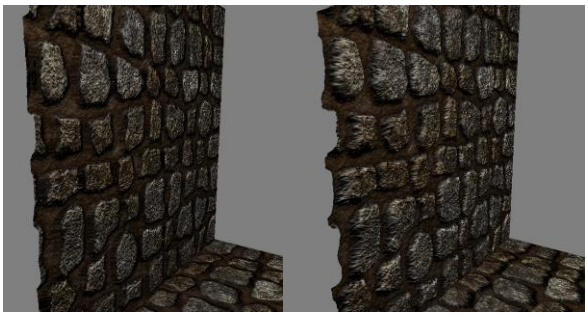
Figure 14. Side view angle, Not Steep vs Steep, 0.08 scale

Viewing this from the side gives a similar effect but near the edge, both methods display increased stretching of the texture. The stretching in steep, although more noticeable, is a better representation of the actual side of the stone compared to the standard parallax implementation.

V. Using the Demo Project

Open and run the project in Visual Studio 2015 and supports Debug and Release in both 32 and 64 Bit. The program will display two planes, a wall and floor, with a cobble texture. Normal parallax mapping will be used by default and pressing the P key will swap between non-steep and steep. The A and D keys can be used to adjust the scale, which when set to 0 will display without parallax mapping and pressing N will disable/enable the use of the normal map.

VI. Conclusion

Parallax mapping can be used to enchance the effect of the other textures by altering what part of them can be seen depending on the direction and position they are being viewed from. The way it is implemented allows it to be pluged into other projects easly and make use of the same TBN matrix used for normal mapping. There is also some support that can be implemented for self shadows that would dramtically improve the realism of the final result.

The shader used into the examples could be improved by correctly calculating the view direction with the mix of both the camera rotation and position. Additionally, there is potential for parallax mapping to be used in combination with other displacement mapping and tessilation technqiues, to improve upon opimiations in real-time 3D applications through more advanced level of detail systems (Wolff, 2013, Sellers et al. 2016).

VII. Bibliography

DeVries, J. (n.d.). *Parallax Mapping*. [online] Learnopengl.com. Available at: https://learnopengl.com/Advanced-Lighting/Parallax-Mapping [Accessed 22 Apr. 2018].

Kaneko, T. Takahei, T. Inami, M. Kawakami, N. Yanagida, Y. Maeda, T. and Tachi, S. (2001). *Detailed shape representation with parallax mapping*. In Proceedings of ICAT(Vol. 2001, pp. 205-208).

Lengyel, E. (2001). *Computing Tangent Space Basis Vectors for an Arbitrary Mesh*. [online] Terathon.com. Available at: http://www.terathon.com/code/tangent.html [Accessed 23 Apr. 2018].

McGuire, M. and McGuire, M.. (2005). *Steep parallax mapping*. I3D 2005 Poster, pp.23-24.

Sellers, G., Wright, R. and Haemel, N. (2016). *OpenGL superbible*. 7th ed. New York [etc.]: Addison-Wesley.

Sherrod, A. (2008). *Game graphics programming*. 1st ed. Boston, MA: Course Technology.

Wolff, D. (2013). *OpenGL 4 shading language cookbook*. 2nd ed. Packt Publishing Ltd.