Relief Texture Mapping

Philip West

Supervised by: Professor Shaun Bangay,

Adele Lobb and Professor George Wells

Department of Computer Science

Rhodes University

Abstract

The project is based on the texture mapping technique described in the paper "Relief Texture Mapping" by Manuel M. Oliveira, Gary Bishop and David McAllister. The technique [Oliveira et al, 2000] makes use of textures with depth to increase realism in three-dimensional scenes while maintaining interactive rates of display. Pre-warp equations are used to modify the textures so as to support the representation of three dimensional surface details as well as view motion parallax. Conventional texture mapping is then used to convert from texture coordinates to screen coordinates (Figure 1).

(a) **(b)** **(c)**

Figure 1. (a) The source image with associated height values per texel . (b) The pre-warped image. (c) The final view after the image in (b) has been applied to a surface using conventional texture mapping.

Introduction

Standard two-dimensional texture mapping is commonly used to add realism to scenes. A stored image file is superimposed onto a smooth surface in order to add colour detail to a scene [Bangay, 2003, pg 99], for example, a picture of bricks can be added to a polygon to give the appearance of a wall. This technique is restricted in the sense that it does not support the representation of three dimensional surfaces. When viewed from certain angles the absence of parallax may reveal that the surface is flat [Oliveira et al, 2000, pg 359,].

Relief texture mapping is an extension to conventional texture mapping that allows for the representation of three dimensional objects. [Oliveira et al, 2000] describes a technique that uses a two-pass pre-warp forward transform process followed by conventional texture mapping. The pre-warp equations are used to manipulate relief textures, textures with a per texel displacement value. This process handles the parallax effects that result from viewing three dimensional objects from differing angles. Conventional texture mapping is then used to convert the texture coordinates to screen coordinates, as well as handling scaling, rotation and the remaining perspective transformation [Oliveira, 2000, pg 360].

This project aims to use a one-pass approach, for simplicity, while still using the pre-warp equations described by [Oliveira et al, 2000]. The goal is to implement a relief texture mapping system and then extend this system to handle lighting and shading effects by texel shading using hardware.

Background: Image Based Rendering

Computer graphics has conventionally involved synthesizing images from geometric object models. These models are assigned surface descriptions detailing such properties as reflectivity. In this type of system the interaction of light with a scene is calculated. Image based rendering is a technique that generates synthesized images, which represent three dimensional objects, directly from other images [McMillan, 1997, pg 1-2]. In this way the need for geometric models of objects is removed. The technique involves a projective mapping of texels from a source image to their correct position in the destination image [McMillan, 1997, pg 4].

According to [McMillan, 1997, pg 6] it is possible to synthesize images using image-based methods instead of geometry-based methods. These synthesized images can be generated using fewer computational operations than images that are generated based on geometric models, and “are at least as realistic as those images synthesized from geometric models”. Image-based rendering can simplify the process of building and representing models.

There are various image-based methods which take varying approaches. In some cases images are used to represent approximations of a scene. Databases of images have been created and these are then queried when a desired view is required. This project deals with a method that is used to synthesize entire scenes and models from images.

Project Phases

The project has been divided into five phases, with each phase building on the previous phase. The final system should be able to create any complex object, which can be correctly lit and shaded, using at most six images. The user should be able to view the object, translate and rotate it, all at an interactive rate. The system is being developed in the C++ programming language, making use of OpenGL. Red Hat Linux is the operating system being used. The texel shading will be done using the nvidia quadro4 900 xgl graphics accelerator card.

Phase 1: Creating Relief Textures

Relief textures are textures which have a height value associated with each texel. [Oliveira, 2000-2] describes an approach where the height values are stored in the alpha channel of an image. This project makes use of two images (figure 2), one image is used to extract the colour values (figure 2 a) and the other is a greyscale image, known as a height map, which is used to store the displacement values of each texel (figure 2 b). The lighter the colour of a texel in the height map, the greater the height value of that texel.

(a) **(b)**

Figure 2. (a) A texture giving colour values. (b) A height map giving height values. Black (0,0,0) represents zero height and white (1,1,1) represents maximum height.

A problem, which stems from using height maps, is maximum value. Here white (1,1,1) would represent the maximum height possible, but currently this has not been standardised and thus the scale used to obtain height values from colour values must be adjusted for different texture maps. Another problem with height values is deciding on a unit of measurement. As yet this issue has not been solved and needs to be considered further.

Phase 2: Pre-warping equations

The pre-warping equations described by [Oliveira, 2000-2] are derived by factoring the three dimensional image warping equation defined in [McMillan, 1997] into a serial warp followed by conventional texture mapping. Hence the three dimensional warp is reduced to a two dimensional problem. The pre-warping equations are as follows:

$$\mathbf{u_i} = \frac{u_s + k_1 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)}$$

$$\mathbf{v_i} = \frac{v_s + k_2 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)}$$

Where (u_s,v_s) is the position of a texel on the source plane and (u_i,v_i) is the destination point of the corresponding texel. u represents the horizontal axis and v the vertical axis. $displ(u_s,v_s)$ gives the height value of the texel. k_1, k_2 and k_3 are constants for the configuration of the source and target view points. The equations used to calculate these values are given below:

$$\mathbf{k_1} = \frac{\bar{f}.(\bar{b} \times \bar{c})}{\bar{a}.(\bar{b} \times \bar{c})}$$

$$\mathbf{k_2} = \frac{\bar{f}.(\bar{c}\times\bar{a})}{\bar{a}.(\bar{b}\times\bar{c})}$$

$$\mathbf{k_3} = \frac{1}{\bar{c}.\bar{f}}$$

Vectors a and b are the basis of the plane of the source image. f is a unit vector perpendicular to the source plane. C_s is the origin of the source image plane. C_t is the target centre of projection (COP) and c is a vector from C_t to C_s. In OpenGL the COP is always at the origin (0,0). All these vectors and points are illustrated in figure 3.

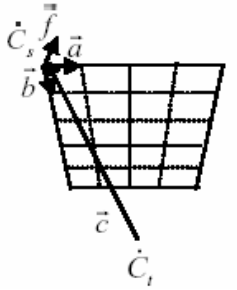


Figure 3. Image view plane with the values required for the pre-warping equations [Oliveira, 2000-2].

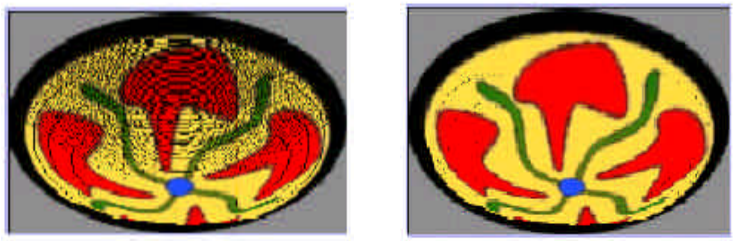
The pre-warping equations have been implemented using a forward mapping, one-pass approach. For each texel in the source image, the destination position is calculated and the texel is moved to that position.

Since texels are being warped to new positions on an image, it is possible for two texels to be warped to the same position. To ensure that the correct texel is converted to the correct position, [Oliveira, 2000-2] suggests using a painter's algorithm, which is described in [McMillan, 1997]. The algorithm describes the correct order to warp the texels in so as to ensure correct visibility. A depth buffer has been used in this project to achieve the same outcome. The buffer stores height values of texels at every position, when two texels are warped to the same position the buffer stores the texel with a greater height value and simply discards the other.

Phase 3: Hole-Filling

Texels cover the source image in an even distribution. Holes arise because texels covering a three dimensional object have an uneven distribution. When the source image is warped to create the final image the resulting texels are unevenly distributed. This means that holes will occur if only one texel is drawn on the destination image

for every texel on the source image [Parilov, 2002-2]. Figure 4 shows the source image with an even distribution of texels and the resulting warped image with holes.



(a) **(b)** **(c)**

Figure 4. (a) Source image with an even distribution of texels. (b) Destination image with holes resulting from an uneven distribution of texels. (c) Image with holes filled, not yet completely filled.

To solve this problem we use linear interpolation. In this method we determine, for each texel, if a hole is created between it and the previous texel. The texel directly above it must also be considered. When a hole needs to be filled the difference between the colours of the two texels is calculated this difference is averaged over the hole. This process is illustrated in figure 5.

(a) **(b)** **(c)**

Figure 5. (a) Texels X and Y are adjacent to one another in the source image. (b) In the destination image X and Y have been warped apart leaving a hole. (c) Linear interpolation is used to fill the hole between the texels.

An alternative method that can be used to eliminate holes is to use an inverse transform [Parilov, 2002-2]. When using a forward transform each texel in the source image is moved to a point on the destination image. An inverse transform finds a source texel for each desired image texel. Splatting is another possibility that can be used to fill holes when using a forward transform approach. Here a cloud of texels is created on the destination image instead of a single texel. The cloud is opaque in the centre and becomes increasingly transparent as its size increases. In the final destination image the splats will over lay one another giving the correct colour at each texel. We chose to use linear interpolation owing to its simplicity, but are currently investigating splatting as the current method is not perfect.

Phase 4: Over Flow

It is possible for a texel to be warped to an area that is outside the view plane of the destination image. This problem is known as overflow (figure 6). To correct this effect [Oliveira, 2000] attaches auxiliary polygons perpendicular to the view plane to accumulate these texels. Although this method requires rendering extra polygons a maximum of two polygons is needed for any view. This is the phase of the project that is currently being implemented.

(a) **(b)** **(c)**

Figure 6. (a) Parts of image warped outside the view plane. (b) Auxiliary polygons attached to retrieve data. (c) Image displaying loss of data resulting from over flow.

[Parilov, pg 3, 2002] suggests an alternative approach where the size of the destination view plane is dynamically resized to allow the entire destination image to be displayed. This method does not require additional polygons, but does rely on a more complex algorithm.

Phase 5: Optimization in Hardware

Relief texture mapping involves two dimensional images. Although the resulting images appear three dimensional, they incorrectly display the effects of lighting when using conventional lighting techniques. Lighting and shading effects can be created by using texel shaders in hardware. We will attempt to create a system capable of achieving this using the nvidia quadro4 900 xgl graphics accelerator card. Such a system has been implemented by [Fujita, 2002].

Conclusion

Image based rendering is a faster method of rendering complex three dimensional objects than traditional geometry based methods. [Oliveira et al, 2000] achieved an average frame rate of 9.42 frames per second whereas we have only managed to

achieve just below 2 frames per second. This rate is expected to increase with optimization of the code and implementation in hardware. Relief texture mapping has been used by [Oliveira et al, 2000] and various other authors to render three dimensional objects from arbitrary viewpoints while greatly reducing the processing time required to achieve this when using geometry based methods.

References

[Bangay, 2003] Bangay, S. *"Computer Graphics"*. Course Module. Department of Computer Science, Rhodes University Grahamstown, Version 3.10, April 2003.

[Fujita, 2002] Fujita, M. and Kanai, T. *"Hardware-Assisted Relief Texture Mapping"*. Eurographics 2002 short paper presentation (Saarbrucken, Germany, 2-6 September 2002), pp.257-262, 2002.

[McMillan, 1997] McMillan, L. *"An Image-based Approach to Three-dimensional Computer Graphics"*. Ph.D. Dissertation. Department of Computer Science, University of North Carolina at Chapel Hill, 1997.

[Oliveira et al, 1999] Oliveira, Manuel M. and Bishop, G. *"Factoring 3-D Image Warping Equations into a Pre-Warp Followed by Conventional Texture Mapping"*. Department of Computer Science, University of North Carolina at Chapel Hill, Technical Report TR99-002, January 15, 1999.

[Oliveira et al, 1999-2] Oliveira, Manuel M. and Bishop, G. *"Relief Textures"*. Department of Computer Science, University of North Carolina at Chapel Hill, Technical Report TR99-015, March 29, 1999.

[Oliveira et al, 1999-3] Oliveira, Manuel M. and Bishop, G. *"Image-based Objects"*. Department of Computer Science, University of North Carolina at Chapel Hill, Proceedings of 1999 ACM Symposium on Interactive 3D Graphics (Atlanta, GA), pp. 191-198, April 26-28, 1999.

[Oliveira et al, 2000] Oliveira, Manuel M., Bishop, G. and McAllister, M.

"Relief Texture Mapping". (New Orleans, La), pp. 359-368, July 23-28, 2000.

[Oliveira et al, 2000-2] Oliveira, Manuel M. *"Relief Texture Mapping"*. Ph.D. Dissertation. UNC Computer Science Technical Report TR00-009, Department of Computer Science, University of North Carolina at Chapel Hill, 2000.

[Parilov, 2002] Parilov, S. and Stuerzlinger, W. *"Layered Relief Textures"*. *Journal of WSCG*, vol. 10, no. 2, ISSN 1213-6972, pp. 357-364, February 2002.

[Parilov, 2002-2] Parilov, S. "*How to use Image-based Rendering in Computer Games*". Available http://www.cs.yorku.ca/~parilov/ibr/ , 2002.

[Peam, 2000] Peam, O. "*Faster Rendering using Relief Textures*?". Available http://members.ozemail.com.au/~owenp/relief_textures.htm , 27 January 2000.