Real-time Terrain deformation using bump and displacement mapping

Literature Synthesis

Justin Crause

1. Abstract

This paper examines current techniques that could be used in a real-time terrain deformation system. The focus will be on bump and displacement mapping techniques. Bump mapping uses textures to add additional detail but does not alter the surface geometry. Displacement mapping also makes use of textures to perturb the surface which can be used as a means of modelling.

Normal mapping is a crude technique that produces better results than straight texture mapping. It produces acceptable results for small scale or shallow detail. True displacement mapping was chosen as another technique appropriate for terrain deformation. A level of detail system will be used that combines the above mentioned techniques in order to achieve the best results and the best performance.

The other techniques described were found to be inadequate due to higher complexity which requires additional processing. These techniques were introduced many years ago. Current hardware uses specialised shaders to facilitate these functions.

2. Introduction

This paper investigates current computer graphics techniques that are used to add detail to simplified three dimensional objects. Particular attention will be paid to bump mapping and displacement mapping which is the primary focus of this work. A definition for each of these will be presented along with a brief evaluation and comparison. Alternative methods will be defined as well as the limiting factors which make them less desirable given the focus of this paper.

Bump and displacement mapping techniques have been around for many years in computer graphics. Bump mapping was first introduced by James F. Blinn in 1978 [1] and displacement mapping by Robert L. Cook in 1984 [2]. The techniques presented in this paper cover topics from many years ago and as such, limitations [3] that existed at the time those papers were written may not exist currently. For example, in 1978 the time taken to render a simple sphere with a bump map would amount to several minutes [1] but with the aid of modern computer graphics hardware, this has been reduced to a fraction of a second.

Texture mapping is a well known and widely used method to enhance the realism of computer generated content [3]. Being able to wrap a 3D surface in a two-dimensional texture to provide vivid and accurate detail is an integral part in modern computer graphics. The techniques discussed below all utilise textures in some form to provide the additional data required. Texture maps not only store colour information but can also be set to store normal or height values. This capability plays a fundamental role in the techniques described below.

The primary focus will be to achieve real-time deformation of a terrain surface by exploiting texture-trickery through the use of bump and displacement maps.

3. Bump Mapping Techniques

Smooth, computer generated surfaces are not very realistic and proved difficult to make the content believable to people. Blinn [1] proposed a technique to simulate surface irregularities that are inherent in real objects. A pre-computed image of a wrinkled or bumpy surface is used to compute surface irregularities. The bump mapping technique is applied as a stage in the lighting calculations applied to the scene. On current hardware this occurs in the vertex and fragment shaders of the GPU [4].

Bump mapping works by using additional data that is stored in texture maps. This data is used by

the various bump mapping techniques that make use of simplified proxy geometry [3] in place of the original object and then add back the higher detail. This is an important advantage of bump mapping but the simplified geometry is still unrealistic. This can be solved through the use of displacement mapping which is covered in section 4.

3.1. Normal Mapping

Normal mapping is the first of the bump mapping techniques being considered. Unlike displacement mapping, normal mapping works by perturbation of the normals only and is purely illusory as it does not alter vertex positions. It also does not alter the smooth silhouette edges of the object [1] as well as not accounting for self-occlusion of the surface features.

In a typical normal map each texel encodes the 3D normal of the corresponding surface point [3]. Bump maps can be useful to render high-frequency geometric detail on a given object. A small normal map can be applied over a large object to add repetitive detail. A common example would be an orange skin or brick wall [3]. An example of normal mapping on a stone texture is shown below.

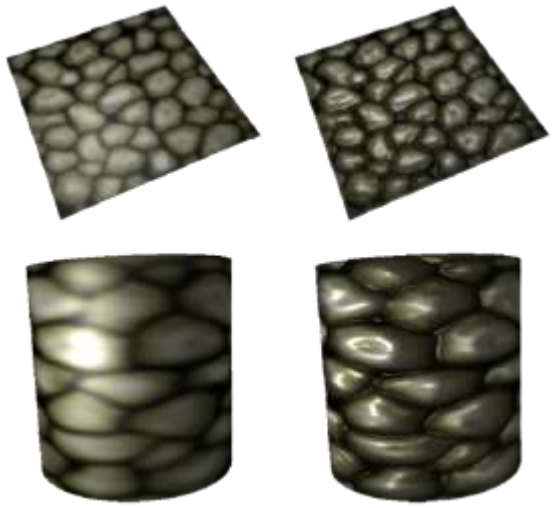

Figure 1: Standard texture mapping (left). Added normal map (right) [5]

Normal mapped surfaces do not provide adequate enough detail close up and would not be suitable for standalone implementation in the project. They could, however, be incorporated in a level of detail system that switched from higher accuracy displacement maps to normal mapping for more distant geometry.

3.2. Relief Mapping

Relief mapping is form of bump mapping that maps relief textures onto polygonal surfaces in real-time. It operates in tangent space, and as such supports self-occlusions, shadowing and per-pixel lighting [9]. Relief mapping works by utilising a root-finding approach on a height map texture [5, 7]. The viewing ray is transformed into tangent space. A linear search is performed to locate the surface intersections then a binary search is done to find the closest intersection. Following this, the shading is performed using the calculated attributes. The linear search has a fixed time step which means the user has to trade accuracy for performance as finer detail needs smaller step sizes.

A separate function that operates on the object's silhouette needs to be implemented [5] to add finer detail to the object boundary. This approach yields significantly better results than bump mapping but still lacks the geometric detail that displacement mapping achieves.

3.3. Parallax Mapping

Parallax mapping is a simple way to augment bump mapping to include parallax effects [7]. It uses per-pixel texture coordinate look ups to achieve high rendering quality [10]. Parallax mapping works by reading the height offset stored in the parallax map at the specified point where the view ray intersects the geometry. Once this height value is obtained, the point that the view ray intersects this height plane is used for the final texture lookup [5]. The diagram below illustrates this technique.

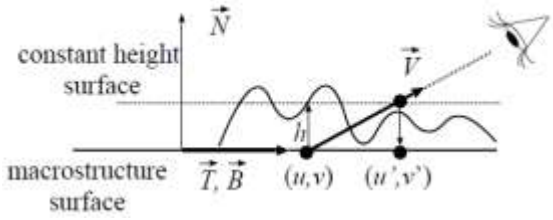

Figure 2: Parallax Mapping Technique [5]

This is a crude technique that provides acceptable results [7] provided a certain set of limitations are adhered to. For instance, this is only valid for smoothly varying height maps; it cannot handle high-frequency features, large displacements and self-occlusion. Offset limiting [5] is also required to render geometry in the

distance because when the view angle becomes too oblique, the offset approaches infinity, causing random data being obtained and thus producing undesirable results. Parallax mapping has one advantage, that being able to implement it in a couple of lines in the fragment shader [7]. However, it does not provide the realism or quality required for the project.

3.4. Inverse Displacement Mapping

Inverse displacement mapping is a variation of displacement mapping but still does not alter the underlying surface geometry [11]. It is a texture mapping technique and as such is included in the bump mapping section.

Inverse displacement mapping uses ray tracing and complex equations to produce the results that true displacement mapping does, however, vertices are not displaced in the process. This means that the silhouette problem noticeable in previously mentioned techniques is accounted for which comes closest to true displacement mapping. Self-shadowing and occlusion are also accounted for in this process. This process is the most authentic one but because of its complexity it would not be a feasible option for this project. Current hardware supports geometry and tessellation shaders which are designed to handle the displacement mapping process, whereas this technique was introduced in 1991 [11] where such hardware did not exist.

4. Displacement Mapping

In 1984 Cook [2] introduced an extension to bump mapping called displacement mapping. Because the location of a vertex is used in the final appearance it is possible to move the location of the vertex as well as perturbing the normal. Displacement mapping was originally designed to solve the issues that come from the use of bump maps, discussed above; mainly the silhouette effect. Apart from this, displacement maps have proved to be useful in other aspects of computer graphics.

Displacement mapping adds real surface detail to objects [6] by shifting the positions of the vertices along the normal to the surface by a distance that is extracted from sampling the displacement map. This differs from bump mapping which only affects the shading of the surfaces [7]. This allows for more advanced effects, not possible with even advanced implementations of bump mapping. These can include surface features that occlude parts as well as self shadowing.

Since displacement maps alter the position of a surface and its normal, these textures can be seen as a means of modelling [8]. Because of this, displacement mapping can be seen as an ideal method for terrain modelling [6]. This makes displacement mapping a technique of particular interest in this paper.

Displacement mapping works by perturbing the surface locations by an amount stored in a texel of the displacement map. This process typically takes place in the vertex shaders on GPU's but methods have been proposed that use solely the fragment shaders [6]. There are, however, some limitations when using fragment shaders only, such as the inability to render large surfaces. Mathematical distance functions are used to smoothly displace vertices based on the samplers input [7]. These functions can be computationally expensive but on modern hardware they can be computed relatively quickly. The process is slower than that of bump mapping; however it produces more realistic results. This is where a level of detail system could be implemented to choose between the two methods. Level of detail will be discussed later on in the paper.

True displacement mapping works not only by perturbing the vertices already present on the surface geometry but also by adding new ones. This works by tessellating [6] the surface to provide more points to produce smoother displacements. In current hardware the use of the geometry shader handles such creation of additional vertex information. Due to the additional overhead of adding geometry on the fly this paper will exclude this feature of displacement mapping. Instead a pre-tessellated surface with a set number of vertices will be utilised. As such there will be a limit to the amount of detail present on the final result after displacement mapping has been applied. High-frequency features that fall between adjacent vertices will be lost and as such a combination of techniques will be applied. This is covered in section 5 about the level of detail system.

Figure 3: Stone surface; normal mapped (top), displacement mapped (bottom) [7]

5. Chosen Implementation

Based on the techniques listed in sections 3 and 4, the following decisions have been made that will produce the most realistic results and still keep the system operable in real-time. Normal mapping was chosen because of its acceptable results and simple implementation. The other techniques may provide better results but with a greater cost in performance. Displacement mapping was chosen for finer detail as it provides the best results even though it has a high computation cost. A mixture of the two will be exploited to provide good results in real-time.

A level of detail (LOD) system is introduced to utilise simpler techniques for more distant geometry. In order to allow for this the terrain will consist of a tile based system with square shaped surfaces. These surfaces will have various degrees of tessellation from high to no tessellation.

Normal mapping provides a fast and more realistic effect than plain texture mapping; however, it has quite a number of limitations. These limitations will prevent it from being a viable solution for high detail scenarios. It does, however, make sense to incorporate it for more distant geometry where higher detail is not needed. This makes it a prime candidate to handle the final level in the LOD system. This would simply apply the normal mapping process onto a simple four point plane representing the tile.

For the closest objects in the scene, the highest amount of detail is necessary, which is why displacement mapping is most suitable. The accuracy of the displacement mapping process relies on the degree of tessellation of the surface geometry; as such, some very fine detail may be lost even in the closest tile. In order to correct this normal mapping must be applied over the displacement mapping process to add in detail that may have been missed.

Levels two and three of the LOD system make use of increasing simplification of the surface geometry. Since these tiles are further away, higher accuracy is not needed. Level two has feature tiles with a quarter the number of tessellations and features normal mapping. Level three features a quarter the tessellations found in level two and does not feature normal mapping.

The LOD system allows for a good balance between visual quality and performance. Because most of the content is generated on the fly, fast algorithms and data structures are utilised to keep the performance high and the application running in real-time.

6. Conclusion

From the various techniques detailed in this paper, the most suitable ones for real-time terrain deformation were normal and displacement mapping. These two techniques will be used in conjunction with a level of detail system to provide the most realistic look while allowing the system to run in real time. As an extension to this paper, further research on these two techniques is indicated and changes may be implemented. Appropriate tests on mainstream hardware will be conducted in order to obtain true performance data.

7. References

1. Blinn, J. F. 1978. Simulation of wrinkled surfaces. *SIGGRAPH Computer Graphics*. 12, 3 (Aug. 1978), 286-292.

2. Cook, R. L. 1984. Shade trees. *In Proceedings of the 11th Annual Conference on Computer Graphics and interactive Techniques H. Christiansen, Ed. SIGGRAPH '84. ACM*, New York, NY, 223-231.

3. Tarini, M., Cignoni, P., Rocchini, C., and Scopigno, R. 2000. Real Time, Accurate, Multi-Featured Rendering of Bump Mapped Surfaces. *Computer Graphics Forum*. 19, 3, 119-130.

4. Wang, J. and Sun, J. 2004. Real-time bump mapped texture shading based-on hardware acceleration. *In Proceedings of the 2004 ACM SIGGRAPH international Conference on Virtual Reality Continuum and Its Applications in industry (Singapore, June 16 - 18, 2004). VRCAI '04. ACM*, New York, NY, 206-209.

5. Szirmay-Kalos, L., and Umenhoffer, T. 2008. Displacement mapping on the GPU - State of the Art. *Computer Graphics Forum.* 27, 1.

6. Hirche, J., Ehlert, A., Guthe, S., and Doggett, M. 2004. Hardware accelerated per-pixel displacement mapping. In *Proceedings of Graphics interface 2004* (London, Ontario, Canada, May 17 - 19, 2004). ACM International Conference Proceeding Series, vol. 62. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, 153-158.

7. Donnelly, W. 2005. Per-Pixel Displacement Mapping with Distance Functions. *In GPU Gems 2*, M. Pharr, Ed., Addison-Wesley, pp. 123 -136.

8. Cook, R. L., Carpenter, L., and Catmull, E. 1987. The Reyes image rendering architecture. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 95-102.

9. Policarpo, F., Oliveira, M. M., and Comba, J. L. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of the 2005 Symposium on interactive 3D Graphics and Games* (Washington, District of Columbia, April 03 - 06, 2005). I3D '05. ACM, New York, NY, 155-162.

10. 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 the ICAT 2001*, 205-208.

11. Patterson, J., Hoggar, S., and Logie, J. 1991. Inverse displacement mapping. *Comp. Graphics Forum*. 10, 2, 129-139.