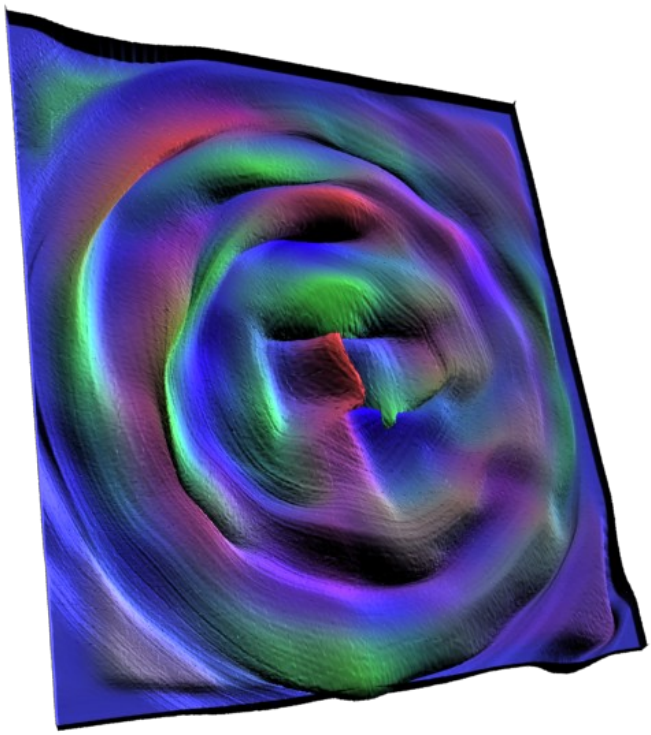
Investigation of Surface Mapping Techniques & Development of Non-Linear Displacement

Neil West

National Centre for Computer Animation
Bournemouth University
Innovations Project Report

2007



Abstract

Texture mapping is an important part of surface representation within computer graphics. Real-time applications especially benefit from the manipulation of surfaces to 'fake' detail that does not exist in the true geometric model. This project is a study of current real-time and non real-time texture mapping techniques, focusing on mapping that disturbs a surface to add detail. Research culminates in the development of an artefact that encompasses previous techniques and builds on them to create something that explores new areas of mapping.

Keywords

Texture mapping, bump mapping, displacement mapping, normal mapping, parallax mapping, relief mapping, RenderMan, surface details, non-linear displacement.

Content

1.0 Introduction

Computer graphics is still a relatively new industry and area of research. Although in development for many years, recently it has surged with rendering of images reaching a level of quality high enough to be used for live action visual effects and full animated features. Rendering of surface images to depict specific effects is especially important in the process. **Texture mapping** [Catmull, 1974] [1] plays a pivotal role in the rendering of quality surface details and many techniques have arisen in recent years. It is an ever-expanding area and research is on-going allowing for further innovation.

1.1 Motivation

Initially I had trouble deciding on a final project brief and it has constantly been changing throughout the project. I knew that I wanted to do some research into the field of rendering and pick one area to focus on more than others. I began by thinking about photo-realistic rendering and its application inside Mental Ray, but this presented a rather generic project with no real innovation potential.

For some time I have been interested in texture mapping and have used it commonly in the past on projects. At a basic level it is the process of shading a surface with an image so the renderer can correctly colour each point on the surface. As well as colour surfaces, texture maps can also be used in disturbing a surface to give the illusion of bumps. This can be done with a greyscale map or RGB map, for example.

Therefore I decided that this would be an interesting area to look into as it is an effective and computationally cheap way to add high quality surface detail. It can be used within real-time systems like games where it is not possible to have high detail meshes. It can also be used in non-real-time systems to add geometric detail at rendering time to surfaces to avoid having a scene heavy on actual geometry.

I have always been interested in non-real-time applications as I feel real-time can be constraining and I enjoy the freedom that a final rendered image can allow. This means I will tend towards research and development of a system to be used not in real-time but as a tool to add to a statically rendered image.

My final motivation for undertaking a rendering based Innovations project was to learn something more about a different renderer. In the past I have mostly used the AutoDesk Maya rendering interface, the ‘hypershade’ and options available within it, and Rendering with Maya Software or Mental Ray. I have never used a RenderMan based renderer and therefore it would be beneficial to obtain grounding in this industry standard type of rendering. As well as the actual renderer, I have never written shaders from scratch, only using the ‘hypershade’. It would be good experience to see a shader from the code aspect.

[1] Catmull, E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah, Salt Lake City, Utah.

1.2 Aims

Initially a great deal of research was needed into existing techniques. As there were so many, I aimed to constrain the research down to the main methods that seem to stand out over others. Hopefully the research work would give me a good understanding and grounding in the principles, so I could then take those principles and adapt them to my own innovative work.

In addition to researching the techniques, the aim was to cover some implementation. The intention was that a selection of the older techniques would be implemented to show the difference between them and to allow myself to get to grips with shader writing. This would also be a good implementation build-up to something originally developed by myself.

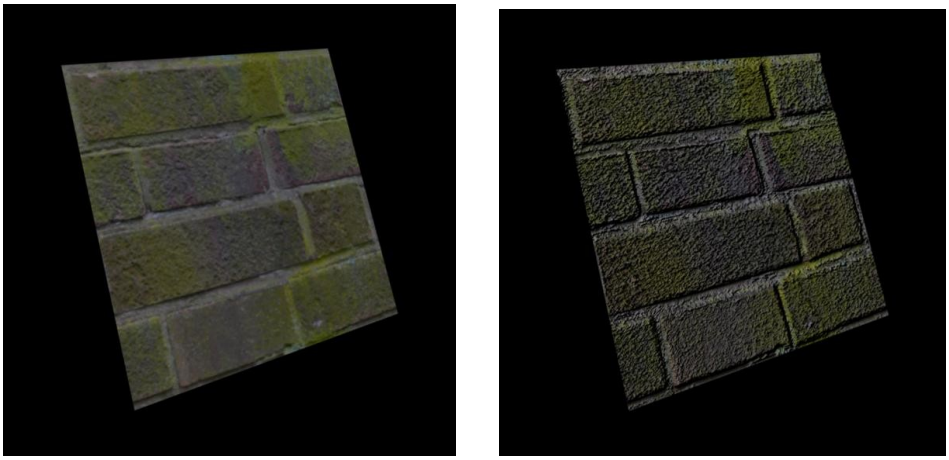
I planned to visualise all of the results within a RenderMan compliant renderer, allowing me to learn a new rendering platform and get some experience in writing shaders from scratch, in code. At first glance Mental Ray could have been an option, but it seems there is just not enough supporting material for development and this would be a whole project in itself to learn. It was also important to give myself enough knowledge to be able to implement the research I was doing on the different mapping techniques.

Finally it was aimed to explore and research into a new technique or develop on an already established one and so produce something that was innovative in the field as much as possible.

2.0 Research

The research began mostly with previous papers on the subject of mapping. The ACM SigGraph database is a good resource for papers covering computer graphics and allowed me to find information about all the different techniques. Papers can be a good source of information as they cover many of the newest innovations in the industry, but some can be confusing and over complicated. Online can be a great resource as well, but rather unpredictable in the results it will feed back to the user. For tutorials, online is the place to go to find existing work and code and there are forums on certain websites where feedback on problems can be acquired. Books are the last resource that could be used, but information can be out of date and there only seemed to be books covering the basic or older techniques in mapping. For RenderMan however, following through a book could be a good way to learn, step by step.

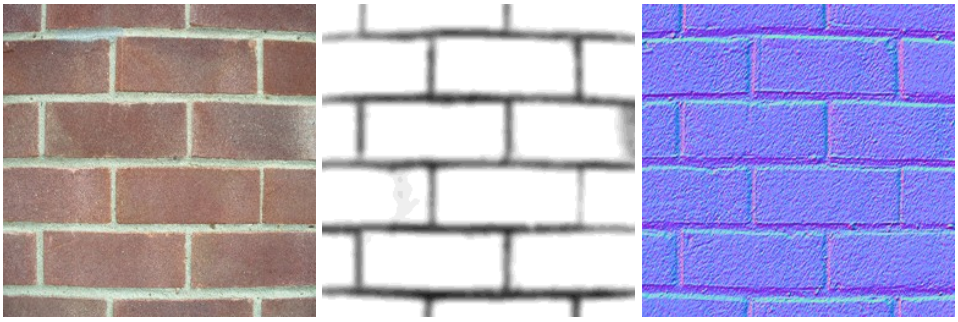
Actual research began with the basic techniques and learning the theory behind them and the ways they can be implemented. **Bump mapping** [Blinn 1978] [2] was the first technique that created extra detail on a surface; it works by perturbing the surface normal so that surface shading is effected to give the illusions of disruptions in the surface. This can be cheap to calculate and effective for quick results, as it is based on a greyscale height map that is used to adapt the normal to the current surface point height and so giving the illusion of depth. However, this technique only perturbs the surface normal and so does not change any actual geometry of the mesh. This means that there is no self-occlusion and no silhouette change for the object. Games use this technique a great deal because it does not add extra geometry to the scene and it can be calculated in real-time. All real-time techniques stem from this initial Bump mapping in some way. **Fig 1** shows the effect bump mapping can have on a surface.



[Fig 1] An example of simple colour mapping on a surface versus bump mapping.
(Left) is a simple colour-mapped surface.
(Right) is the colour-mapped surface with the addition of a bump map, no change in geometric silhouette

Normal mapping [Peercy 1997] [3] uses XYZ coordinates established from a colour RGB map to create a new normal. This is used to create more surface detail and has more flexibility than a bump map that only uses a two-way greyscale map. This is often used on game characters to add details to faces or clothes where more geometry would not be suitable. **Fig 2** shows a comparison between different maps and illustrates an RGB encoded normal map.

[2] Blinn, J.F. 1978. Simulation of wrinkled surfaces. In *Proceedingsof the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, 286-292
[3] Peercy, M., Airey, J., AND Cabral, B. 1997. Efficient bump mapping hardware. In *SIGGRAPH '97,* 303-306.



[Fig 2] A comparison.
(Left) is just a colour map of a brick wall.
(Middle) is a bump/displacement map encompassing height information.
(Right) is a normal map containing surface normal information in RGB values.

Parallax mapping [Kaneko 2001] [4] uses a principle of moving the UV coordinates of the texture map inline with the viewing vector from the camera and therefore gives the appearance of depth. It uses both a diffuse colour map with alpha channel and a normal map with an alpha channel. It uses the depth from the alpha channel and the viewing vector to shift the UV coordinates to a new place. The term parallax mapping comes from the definition :

> "the apparent shift of an object against a background caused by a change in observer position" [5]

This technique has been used in modern games such as F.E.A.R. as shown in **Fig 3** [6]. The technique can be used at very low cost but the basic method shows no support for self-shadowing of textures.

[Fig 3] An example of Parallax mapped bullet holes in F.E.A.R.. The texture coordinates of the bullet hole are manipulated with the camera view to give the illusion of depth.
(Left) shows the view from front on.
(Right) shows the view from the side.

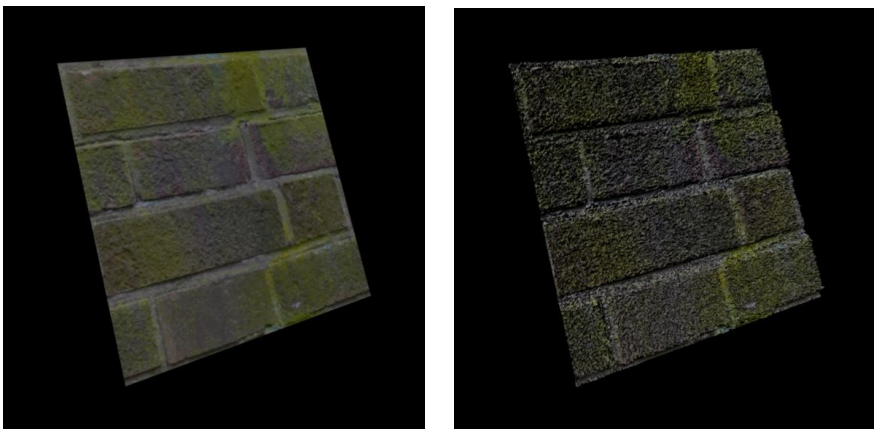
[4] Kaneko, T., Takahei, T., Inami, M., Kawakami, N., Yanagida, Y., Maeda, T., AND Tachi, S. 2001. Detailed shape representation with parallax mapping. In *Proceedings of the ICAT 2001*. 205-208.
[5] Wikipedia.org, 2005. *Parallax*. Available from : *http://en.wikipedia.org/wiki/Parallax*.
[6] Answers.com, 2007. *Parallax Mapping*. Available from : *http://www.answers.com/topic/parallax-mapping*.

Relief mapping [Oliveira 2000] [7] is linked with parallax mapping in that it uses a normal map and a depth map (usually in the alpha channel) to give things a more realistic look. Also by reading the light ray and the depth map, the parts of the image that are self-shadowed can be calculated. This produces better results, but takes longer than the other techniques, which can be detrimental to real-time processing. **Fig 4** illustrates Relief mapping results.

For all of these techniques the focus was on learning the initial theory behind each one, how it is implemented and the relative results of each. They are all fairly different techniques in their approaches, so this gave a good grounding to formulate an idea for a innovative method of my own. The methods gave a good foundation in real-time operations, but the desire had been to produce something that could be used for non-real-time work, as previously stated.

[Fig 4] An example showing the difference of Relief mapping.
(Left) shows Parallax mapping with no self shadowing.
(Right) shows Relief mapping exhibiting self-shadows and a greater look of depth.

From this real-time background, the research took a natural progression to **Displacement mapping** [Cook 1984] [8]. This form of mapping actually deforms surfaces at rendering time and produces more geometry on the model; this means that self-shadowing and silhouettes of models are no longer a problem as geometry is created. **Fig 5** illustrates this principle. However, displacement is currently completed with a greyscale map only and therefore points can only be displaced along the current surface normal. This limits the dynamics of the displacement and constrains it to only moving points along one direction. The process is simple as the actual point is moved along the normal instead of just the normal being changed, as in bump mapping.



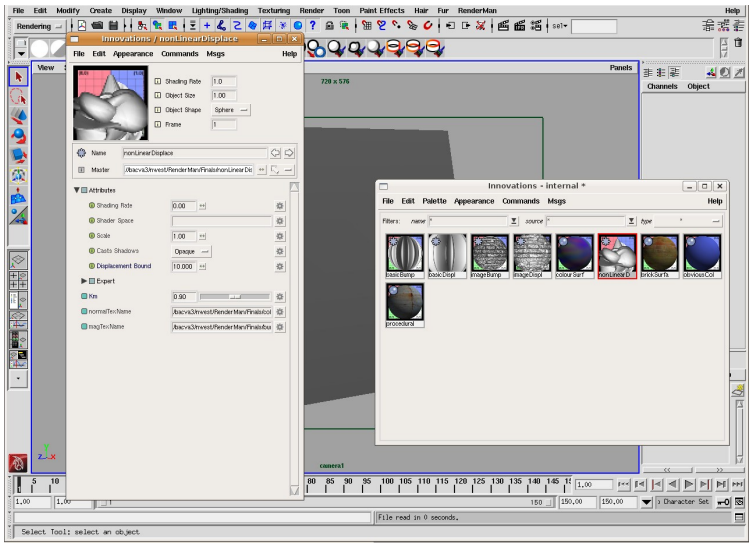
[Fig 5] An example of simple colour mapping on a surface versus displacement mapping.
(Left) shows a simple colour mapped surface.
(Right) Shows the introduction of a displacement map exhibiting real geometric depth and a change in object silhouette.

[7] Oliveira, M. M., Bishop, G. , AND McAllister, D. 2000. Relief texture mapping. In *Siggraph 2000, Computer Graphics Proceedings*, 359-368.
[8] Cook, R. L. 1984. Shade Trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. ACM Press. 223-231.

After researching all the methods it was decided something would be done with displacement, as this is a well-established non-real-time surface manipulation technique. I was interested in making the displacement less linear and restricted.

Looking into the many forms of mapping, I felt I had a good grounding in the subject matter. I found some of the maths behind the theories challenging, but once understood, each method could be defined as a series of simple steps. These steps would be the foundations for the eventual shaders that would be produced.

The next logical step was to move to the actual visualisation of the research and the method for doing this. ***(Essential RenderMan Fast, Stephenson, I. [9])***, was a good starting point for learning RenderMan at a basic level. Learning began with the creation of some basic RIB files for scene structure and object surface control. It was found that while this was useful in general terms for learning the RenderMan structure it was not directly related to the project that had been undertaken. Therefore after the creation of many basic RIB files it was decided to look into using the RenderMan Maya interface **MTOR** [10]. **Fig 6** shows the MTOR interface in use. This allows simple creation of scenes within AutoDesk Maya together with the creation and positioning of objects and lights. As the AutoDesk Maya interface was already very familiar, this made scene set-up a great deal simpler for me. Once the scene has been created, materials can be imported into a palette and applied to surfaces. These materials can include simple colour surface shaders, displacement shaders or any materials RenderMan can support. Although the MTOR interface could be improved, it still made development easier than writing RIB files from scratch.



[Fig 6] This shows the MTOR interface in use within AutoDesk Maya. Illustrating the palette containing imported shaders.

[9] Stephenson, I., 2003. *Essential RenderMan Fast*. London: Springer-Verlag.
[10] Pixar Animation Studios., 2007. *MTOR, The Maya to RenderMan Converter*. Available from : *http://renderman.pixar.com/products/tools/mtor.html*.

'Essential RenderMan Fast' was also useful for shader writing and it gave a good starting point for the creation of the displacement shaders that would eventually be needed. There are several essential elements that are required for the creation of any shader and once these were established it gave the basis knowledge needed for any development in the future. ***(Advanced RenderMan, Apodaca, A. A)*** [11] and ***(Texturing & Modelling A Procedural Approach, Ebert, D. S)*** [12] were also good reference sources for shader writing. Internet resources provided example code that could be adapted to my development purposes.

2.1 Research Conclusions

Coming to the end of the research, I began to see ways to develop previous ideas further and during research on the Internet a relatively un-documented technique called Non-Linear Displacement had come up. Displacement of points is usually a method of moving points along the already established surface normal based on a greyscale height map and this is by nature and linear movement. Non-Linear Displacement is a process based around the same idea, moving already established points to new locations. However, the difference is that it can manipulate points in full three dimensions, which allows creation of concave surfaces from an originally flat mesh. For example, a flat plane mesh could produce a displaced cola bottle shape by moving points to the correct positions. This was a very interesting idea, so this became the focus of development of the project. By combination of some real-time ideas and classic displacement, a shader and map could be produced that performed the functions as just described.

3.0 Implementation & Development

Having a good bed of research served as a good foundation for implementation and development of ideas. Knowledge of all techniques allowed for the development of the proposed innovative displacement process.

Through research, many ideas for surface manipulation had been discovered, but I decided only a selection would be actually implemented, as the final goal was now to produce the new Non-Linear Displacement shader. It was important to develop the basic techniques to show the base difference between Bump and Displacement. Normal mapping was viewed as an important technique as well, but its actual implementation was tougher than first thought and this will be detailed later in this section. To visualise the research it was felt it was important to show surface manipulation in both procedural generated form and through the use of image maps, as this would show a good range of realisation possibilities.

Bump mapping was a first logical step in development. This began with procedural generation, using a Mathematical function to give the surface the illusion of bump high-lights and low-lights. As illustrated here :

```
/*Calculate the magnitude of the bump displacement*/
float mag = sin(s*10*2*PI);
```

This produced successful results and, building on a basic bump map structure, was simple to implement. Using the same procedural formula to build a simple displacement shader was a sensible progression, showing the core differences between the two techniques. The first example showing shading changes, but no geometric silhouette manipulation; the second showing actual geometric mesh changes.

Using maps made the shaders slightly more complicated to develop as this included loading in texture maps and sampling the surface colour. A float is used to sample each point of the greyscale map to get a value between 0 and 1 and this value is used to dictate the amount of

[11] Apodaca, A. A., Gritz, L. 2000. *Advanced RenderMan*. USA : Morgan Kaufmann Publishers.
[12] Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., Worley, S. 2003. *Texturing & Modelling A Procedural Approach*. Third Edition. USA : Morgan Kaufmann Publishers.

displacement of points. Through the use of simple maps both methods and the differences could be easily displayed as with the procedural development. To add a more visually interesting aspect to the work, a brick texture was used to show a practical use for both techniques.

Comparatively, Normal mapping and Bump/Displacement mapping are very different techniques. As Normal mapping uses an RGB colour map to manipulate surface normals, it requires a different kind of shader implementation. Also there are different types, Tangent Space Normal maps and Object Space Normal maps. These both operate slightly differently. The basic theory of sampling a colour map to get a new normal coordinate value was relatively simple to implement and the creation of a shader to do this was viewed as not difficult. The concept of RenderMan geometric spaces was found to be a confusing concept in the context of mapping and after some time in development I chose to drop the actual final implementation of the Normal mapping technique. Perhaps this was due to my lack of experience within RenderMan or the difficulty of converting the mathematics of Normal mapping into shader realisation. Due to not including the Normal mapping technique, I chose not to implement the other more complex processes of Parallax mapping and Relief mapping. As these would only show an existing technique, and at their core are only an extension of non-geometric silhouette implementation, it was seen that time would be better spent implementing a Non-Linear shader artefact.

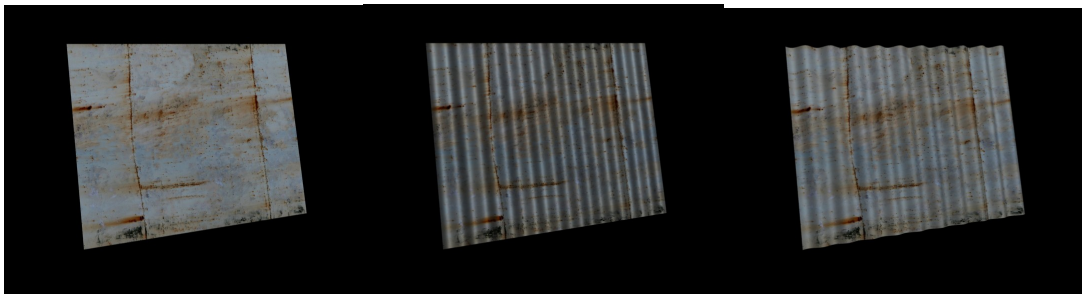
Even though Normal, Parallax and Relief mapping implementation had been dropped, the theories were still core to the development of Non-Linear displacement. The use of an RGB map was essential, as three-dimensional movements of points were required for non-linear work. Shader development began with a Normal mapping structure, sampling an RGB colour from a map, which translated into a normal vector to be used to place points. After tests with just an RGB map it was obvious that there was going to be a need for another element. The problem encountered was the lack of magnitude control; the direction could be established, but not the magnitude so the images looked too flat. Using the original theory of Displacement mapping, a greyscale map was introduced to control the magnitude of point displacement. This allowed for the movement of points in any direction and of any magnitude.

After this introduction the basic building blocks were in place for the Non-Linear Displacement process to work. This presented new challenges however; as even though the shader produced the correct results, the image maps were integral to this result. Presently there is no map generator that can produce an RGB image of an object with concave deformation in a surface like the ones generated by the Non-Linear shader. Normal mapping works with an RGB map and there is a selection of generators available through AutoDesk Maya and Adobe PhotoShop, listing two examples. After several trials with these generators to produce a good result, they were dropped as they only produced results that were similar to Normal map results and not showing the capabilities of the shader. For the purpose of this project a simple colour map was produced and blurred to create a random displacement effect on a sphere as can be seen in the final artefact video. This shows the Non-Linear displacement of points on the surface in a three dimensional environment.

4.0 Results

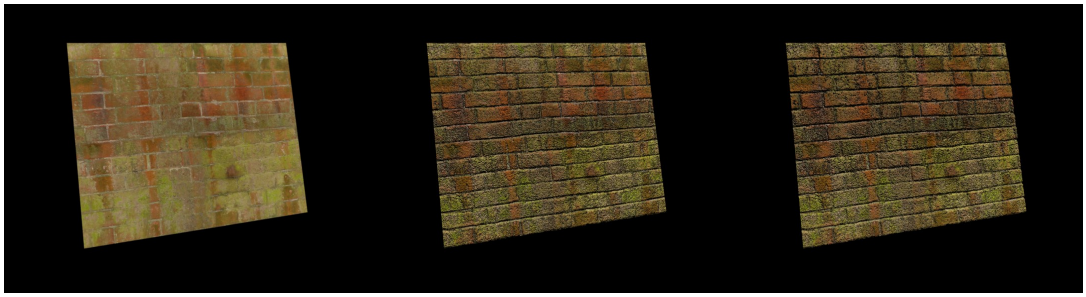
Here are the image results from the different methods that were implemented. A full shader code listing can be found in **Appendix A**.

RenderMan procedural bump and displacement



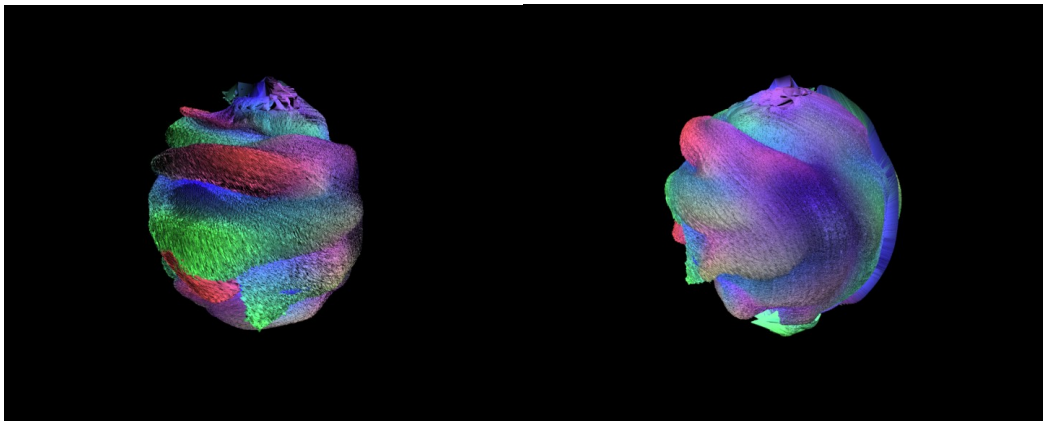
[Fig 7] Procedural bump and displacement shaders. Produced with mathematic formula.
(Left) is a simple colour map.
(Middle) is a bump shader showing fake depth but no change in geometric silhouette.
(Right) is a displacement shader showing real depth and self-occlusion.

RenderMan image based bump and displacement



[Fig 8] Image based bump and displacement maps. Produced with greyscale height maps.
(Left) A simple colour map of a brick wall.
(Middle) The colour map with a greyscale height bump map allowing the appearance of depth in the surface. No change in geometric silhouette.
(Right) The colour map with a greyscale height displacement map allowing real depth to be produced by displacing points. Geometric silhouette is changed along with the surface.

RenderMan non-linear displacement



[Fig 9] Non-Linear Displacement mapping. Images show two different angles of a sphere with a Non-Linear Displacement shader attached. Uses a colour RGB map for direction of displacement and a corresponding greyscale map for height of displacement. There are a few minor problems, but the technique is still very much in development.

The project produced good final results, the original Bump and Displacement methods were successful at showing the differences. With more time the final Non-Linear Displacement shader could have been shown in a more user-friendly manner.

5.0 Project Review

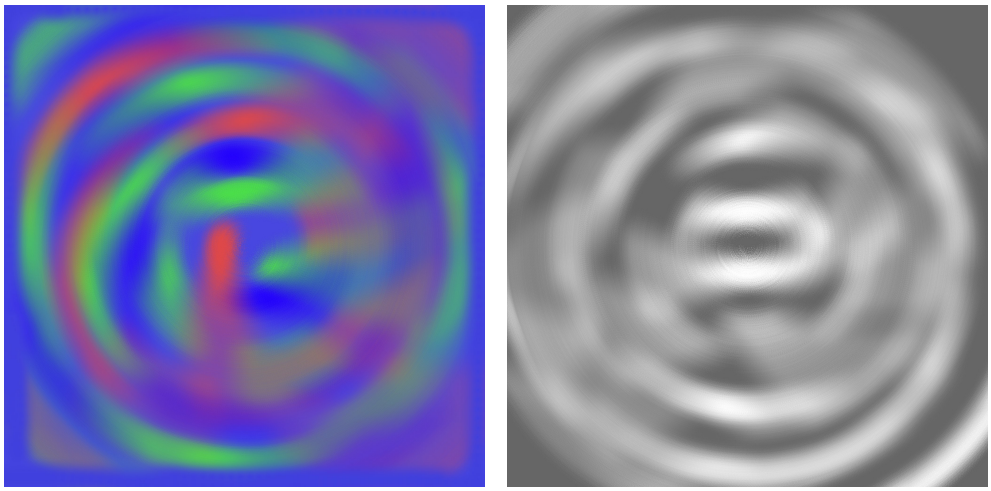
5.1 Problems

Although the research phase was very successful in gaining knowledge in the subject and preparing for the implementation phase, there were a few problems in the actual implementation phase itself.

The Normal mapping was frustrating as much effort was put into its development and in my view it was important to have in the project. Unfortunately, the amount of effort and time it was eventually taking was viewed as too much and it was regretfully dropped. However, even though the actual realisation of this method was not achieved, the code developed was not wasted and the basic theory of the process was encapsulated inside it.

Because of the lack of Normal map implementation, this translated over to the other more specialist real-time based methods. From the beginning, Parallax and Relief mapping were always going to take a development backseat compared with more established methods such as Bump and Displacement, but there was still a desire to include them. As the project evolved over development time it was clear that the implementation of these two more specialised methods was not necessary, but the theories behind them were essential to further development. Relief mapping, for example, uses the same base maps as the Non-Linear Displacement shader.

There was a problem with visualising the Non-Linear Displacement, the difficulty was the generation of the correct RGB and greyscale image maps to do whatever displacement task a user required. To displace a specific object out of another simple object it would require a painfully painted map and corresponding greyscale height map. Therefore in order to visualise the shader, it was decided to use a randomly generated RGB map and greyscale equivalent as these still showed the shaders capabilities without spending too much time on map creation. **Fig 10** shows these maps.



[Fig 10] The colour RGB map and greyscale height map used in the implementation of the Non-Linear Displacement shader.

Another problem was stretching of points in the model; this is partly a problem with the balance of the RGB map and the greyscale map. Without balance the map can cause stretching in an existing model. Another solution would be to change the UV's of the model to make the texture stretching less obvious to the eye. Finally there seems to be a problem

where the UV texture seam is present and this would require further work to produce maps that scaled over objects without showing an obvious seam.

5.2 Successes

The research part of the project was a great success, exploring sources of research in a much greater depth than on previous projects. Published papers were extremely useful for information gathering. This formed the basis for the majority of my research as mapping has been quite a popular research area in past years. Books and Internet provided most of my RenderMan research, finding tutorials and example code the greatest resource.

The research that was undertaken allowed a much greater understanding of the mathematics of surface mapping as well as basic vector and normal mathematics. This was essential knowledge for the project I undertook, but moreover it has uses far beyond this in future projects and development. As a result I feel far more confident in the area of mathematics and can now see a real application of elements of my last two years of study at University.

The project was successfully realised within a new renderer for me. (RenderMan complaint **PRMan** [13]). Initially this was found to be challenging but through work and practice of simple implementations enough was learned to be able to effectively show my project and its final results. The use of MTOR was also good experience as this bridged the gap between my original AutoDesk Maya work and the new RenderMan coding aspects.

The successful implementation of a selection of basic shaders allowed for comparison of the different results as well as gave a basic framework from which, I could visualise my own idea. The objects chosen to represent these techniques also successfully illustrate the differences and provide a good showcase for the shaders.

Implementation of the final Non-Linear Displacement shader was a success in that it shows the process working with the manipulation of points of a sphere. Despite the imperfections in the surface, especially around the UV seam, it is believed the technique has still produced a pleasing shape that can be showcased in the final artefact video.

6.0 Audience

As a means to an end, this project could have uses for all digital artists if taken to full development as a way of using basic geometry in a scene to form complex shapes at render time, and therefore saving on large scene files.The project in its current state is really for developers of future tools to take the techniques described and use them in a practically useful tool for the future.

7.0 Further Work

This project encompassed a good deal of research into various techniques and this allowed for the development into an innovative area of displacement mapping. The shader that was produced needs polishing however and a proper pipeline to be set-up. The development of a tool that will generate an appropriate map is essential; the tool should be able to take a detailed mesh and follow down the model producing an appropriate RGB mix on a colour map and magnitude in a greyscale map. The development of this tool however, would be another project entirely and not within the scope of this one.

[13] Pixar Animation Studios, 2007. *Pixar's RenderMan*. Available from : *https://renderman.pixar.com*.

8.0 Conclusion

Looking back at my original aims I believe the project has been successful overall. From the beginning, the content of the project was not set as it was structured more as a research study and not a definite development project. Through the research my final deliverable became clear and this deliverable built on the research theories I had investigated. Additionally, the final product is not a finished, polished artefact; the application of the theory could be taken down a number of development roads.

Personally the project was rewarding, as a great deal of knowledge was gained from the process that can be taken into ongoing or future projects. However I found that at times the research process could be frustrating and a good mix of practical application and research is better suited to my personal way of working.

9.0 References

Papers

Blinn, J.F. 1978. Simulation of wrinkled surfaces. In *Proceedingsof the 5th annual conference on Computer graphics and interactive techniques*, ACM Press, 286-292

Catmull, E. 1974. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, University of Utah, Salt Lake City, Utah.

Cook, R. L. 1984. Shade Trees. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. ACM Press. 223-231.

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.

Oliveira, M. M., Bishop, G. , AND McAllister, D. 2000. Relief texture mapping. In *Siggraph 2000, Computer Graphics Proceedings*, 359-368.

Peercy, M., Airey, J., AND Cabral, B. 1997. Efficient bump mapping hardware. In *SIGGRAPH '97,* 303-306.

Books

Apodaca, A. A., Gritz, L. 2000. *Advanced RenderMan*. USA : Morgan Kaufmann Publishers.

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., Worley, S. 2003. *Texturing & Modelling A Procedural Approach*. Third Edition. USA : Morgan Kaufmann Publishers.

Stephenson, I., 2003. *Essential RenderMan Fast*. London: Springer-Verlag.

Websites

Parallax Mapping [online]. *Answers.com*. Available from : *http://www.answers.com/topic/parallax-mapping.*
[Accessed 6 February 2007]

MTOR, The Maya to RenderMan Converter [online]. *Pixar Animation Studios*. Available from : *http://renderman.pixar.com/products/tools/mtor.html*.
[Accessed 15 February 2007]

Pixar's RenderMan [online]. *Pixar Animation Studios*. Available from : *https://renderman.pixar.com*.
[Accessed 28 January 2007]

Parallax [online]. 2005. *Wikipedia.org*. Available from *: http://en.wikipedia.org/wiki/Parallax.*
[Accessed 6 February 2007]

10.0 Further Reading

Papers

Oliveira, M. M., AND Poicarpo, F. 2005. An efficient representation for surface details.UFRGS Technical Report RP-351.

Polocarop, F., Oliveira, M. M., AND Comba, J. 2005. Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2005*, ACM Press, 155-162.

Websites

Bump Mapping. [online]. *Wikipedia.org*. Available from :
http://en.wikipedia.org/wiki/Bump_mapping.
[Accessed 5 February 2007]

Displacement Mapping. [online]. *Wikipedia, the free enclopedia*. Available at :
http://en.wikipedia.org/wiki/Displacement_mapping.
[Accessed 10 February 2007]

Normal Mapping. [online]. *Wikipedia.org*. Available from :
http://en.wikipedia.org/wiki/Normap_mapping.
[Accessed 8 February 2007]

Parallax Mapped Bullet Holes. [online]. *Cowboy Programming*. Available at :
http://cowboyprogramming.com/2007/01/05/parallax_mapped_bullet_holes.
[Accessed 10 February 2007]

11.0 Acknowledgements

Ian Stephenson, for inspiration and encouragement as project tutor.

Stephen Bell, for excellent overall project guidance.

Will Alexander, Luke Titley, James Leaning, for project assistance and many interesting late night discussions.

12.0 Appendix A

```
/*****************************************************************/
/*Basic Bump*/
/*****************************************************************/

/*A bump shader using procedural generation*/

displacement basicBump (
         float Km = 0.2 )
{
         /*Calculate the magnitude of the bump displacement*/
         float mag = sin(s*10*2*PI);
         /*Make a new normal and point to calculate the bump*/
         normal NN = normalize(N);
         point PP;

         /*Calculate the bump value for the new point*/
         PP = P + NN * mag * Km;
         /*Calculate the main normal based on the new point declared*/
         N = calculatenormal(PP);
}

/*****************************************************************/
/*Basic Displacement*/
/*****************************************************************/

/*A shader using a procedural displacement value on a surface*/

displacement basicDisplace (
         float Km = 0.2; )
{
         /*Calculating a displacement magnitude*/
         float mag = sin(s*10*2*PI);
         /*Make a new normal from the existing normal and normalise it*/
         normal NN = normalize(N);

         /*Change the point by displacing it along the newly declared normal*/
         P = P - NN * mag * Km;
         /*Calculate the new actual normal*/
         N = calculatenormal(P);
}

/*****************************************************************/
/*Image Bump*/
/*****************************************************************/

/*A shader using a bump map from a greyscale image file*/

displacement imageBump (
         float Km = 0.2;
         string texname = "ENTER_TEXTURE_PATH_NAME_HERE" )
{
         /*Calculate the magnitude of the bump displacement*/
         float mag = float texture(texname, s, t);
         /*Make a new normal and point to calculate the bump*/
         normal NN = normalize(N);
         point PP;

         /*Calculate the bump value for the new point*/
         PP = P + NN * mag * Km;
         /*Calculate the main normal based on the new point declared*/
         N = calculatenormal(PP);
}
```

```
/*************************************************************/
/*Image Displacement*/
/*************************************************************/

/*A shader based on a grayscale texture map to displace a surface*/

displacement imageDisplace (
         float Km = 0.2;
         string texname = "ENTER_TEXTURE_PATH_NAME_HERE" )
{
         /*Calculate the displacement magnitude from a grayscale map*/
         float mag = float texture(texname, s, t);
         /*Make a new normal from the existing normal and normalise it*/
         normal NN = normalize(N);

         /*Change the point by displacing it along the newly declared normal*/
         P = P + NN * mag * Km;
         /*Calculate the new actual normal*/
         N = calculatenormal(P);
}

/*************************************************************/
/*Non Linear Displacement*/
/*************************************************************/

/*A shader using an RGB normal map and a greyscale height map to create more complex displaced detail*/

displacement nonLinearDisplace (
         float Km = 0.2;
         string normalTexName = "ENTER_RGB_TEXTURE_PATH_HERE";
         string magTexName = "ENTER_GREYSCALE_TEXTURE_PATH_HERE" )
{
                  /*Declaring four vectors to store the appropriate information*/
         vector sVector,                    /*Vector for the point s direction*/
                tVector,                    /*Vector for the point t direction*/
                upVector,           /*Vector perpendicular to s and t*/
                mapVector;          /*Vector to store the color information from the RGB map*/
                  /*A new vector showing the direction to displace the new point*/
         vector newVector = vector(0,0,0);

                  /*Float to store the displacement magnitude from the height map*/
         float mag = float texture(magTexName, s, t);

                  /*Sampling the colour from each point on the RGB map*/
         color sampleColor = color texture(normalTexName, s, t);

                  /*Convert the colour into a vector and change the range from -1.0 to 1.0*/
         mapVector = vector(sampleColor);

                  /*Get the derivative of each point for s and t, making a vector for that position*/
         sVector = Deriv(P,s);
         tVector = Deriv(P,t);
         upVector = tVector ^ sVector;

                  /*Normalise all the vectors*/
         upVector = normalize(upVector);
         sVector = normalize(sVector);
         tVector = normalize(tVector);

                  /*Calculating the new direction vector*/
         newVector = vector(comp(mapVector,0) * comp(sVector,0)
                           + comp(mapVector,1) * comp(tVector,0)
                           + comp(mapVector,2) * comp(upVector,0),
                           comp(mapVector,0) * comp(sVector,1)
                           + comp(mapVector,1) * comp(tVector,1)
```

```
                        + comp(mapVector,2) * comp(upVector,1),
                        comp(mapVector,0) * comp(sVector,2)
                        + comp(mapVector,1) * comp(tVector,2)
                        + comp(mapVector,2) * comp(upVector,2));

                /*Calculating the new point using the new normal, a magnitude and a constant*/
        P = P + normalize(newVector) * mag * Km;
                /*Calculate the new normal using the new point*/
        N = calculatenormal(P);
}
```