COM6503 Assignment 2 Report

Eylul Lara Cikis
acp22elc
elcikis1@sheffield.ac.uk

/* I declare that this work is my own. Author: Eylul Lara Cikis, elcikis1@sheffield.ac.uk */

1 Introduction

In 3D graphics texturing is the process of putting a pattern or color on a surface. This act can be used in a multitude of ways to get a desired effect. These range from adding simple surface details to a model to make it more realistic to being able to fake effects such as holes, grooves and bumps. We can even have these textures respond to lighting to make them incredibly realistic such that we can make the user believe those effects are really there geometrically instead.

These texturing techniques can add an exceptional amount of realism to a scene while also being more efficient than their alternatives.

2 Texturing

There are many texturing approaches used in 3D graphics, each of them having their own advantages and downsides, and being useful in different situations:

2.1 2D texture mapping

First and most commonly used approach is 2D texture mapping. Also sometimes called UV mapping, it involves pasting a 2D texture onto a 3D object's surface. Generally square images with the resolution of powers of 2 is used as that makes it even more efficient for GPU's to work with these textures. This approach of course has its downsides, it can result in distortion and stretching of textures especially around curved surfaces. It can also have troubles with aliasing which we will touch on later.

An alternative to that approach would be 2 stage mapping, this method is generally used as an alternative when a polygon is too complex to parameterise the u, v values for. Instead uses an intermediate object that is easy to parameterise, usually a cylinder or rectangle. In stage two this intermediate object is mapped to the object we want to texture.

2.2 3d texture mapping

A third approach is 3D texture mapping which places an object inside a 3D pattern. In this case the 3D texture is usually represented as a mathematical algorithm or a procedure. This allows for the programmer to generate infinite variations of textures. It is especially useful for material textures such as marble or wood.

Figure 1: Example of 3D texture mapping for the wood and specular lighting for the metal

Just like seen from the example in figure 1, 3D texture mapping would be very useful to replicate a texture similar to that of a door like in that photograph

This approach is also used for simulations where a bunch of small particles can interact with the environment according to a function. It allows for

more realistic textures and makes it easy to generate varied textures without much effort. Unfortunately, it is more difficult to use than other techniques as it is computationally expensive and difficult to program.

3 Texturing Techniques

3.1 Why use these texturing techniques?

Some texturing techniques and approaches allow us to get the same effect we want on an object only using a fraction of the computational power. They allows us to add visual detail to an object that would be incredibly expensive to replicate using modeling techniques. Examples include adding bumps or wrinkles to a model, adding a reflection to a shiny surface without simulating it and more.

3.2 Bump Mapping

Bump mapping is used to add an illusion of detail to a 3D object. The way it does this is using a grayscale image to represent the height of a surface on a model. These height values are used to change to surface normals of the model, which adds the illusion of bumps by interacting with the light differently.

This is a simple and efficient way to add this detail. Doing so with modeling would be incredibly computationally expensive and time consuming and increase the polygon count too much.

This technique can especially be extra efficient since most modern GPU's have efficiency to use the technique built into them for now decades and have become very efficient at it. [1]

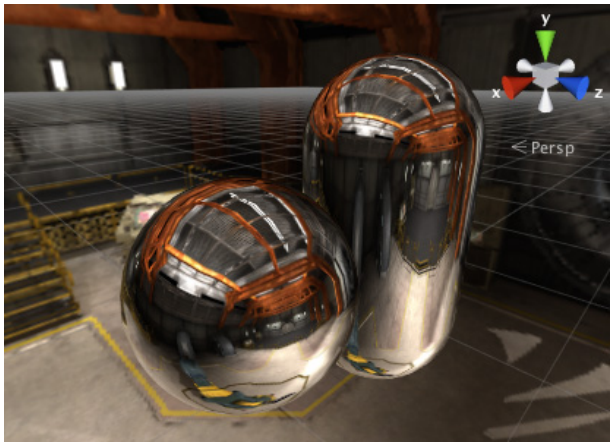
However it does have its downsides primarily that it doesn't change the geometry of the object so it will have unrealistic silhouettes.

3.3 Environment Mapping

Environment mapping allows us to make reflective surfaces in a very simple and cheap way. Compared to ray tracing which is computationally expensive to use in real time applications, environment mapping stores the surrounding world in a "surrounding texture" that is associated with the object. An example of what environment mapping is trying to replicate in real world scenarios can be seen on the metal handle in Figure 1.

This can also allow us to place an object we want in the real world by taking photographs of a real scene from the position of the object we want to insert.

Figure 2: Example of environment mapping in unity using their cubemap system (image from Unity User manual under CC BY-NC-ND



There are of course disadvantages to using this technique compared to ray tracing, firstly the reflection is incorrect reflective objects will not interact with each other as seen in Figure 2 while being good enough to fool, especially from a distance.

Figure 3: Example of two mirrors reflecting light from each other

Since the reflection is not a real simulation, instead a fake replication, this will unfortunately not be able to replicate the effect seen in Figure 3. To do that with just a texturing technique is out of practicality, we would have to use more advanced rendering methods like ray tracing. [2]

3.4 Normal Mapping

Normal mapping like bump mapping allows us to add surface detail to an object without actually altering the geometry. However where normal mapping has an even more advantage and impressive results is, it uses a 2D image called normal map, where it stores x,y,z components of a normal as RGB values. We can then use this stored information about the direction of the surface to simulate the lighting of the surface.

This allows for some incredible optimisations by only using a simplified mesh of an object instead of all the polygons and a normal map is added on top to trick the eye. This can reduce the amount of polygons needed to to show a detailed object hundred, even thousand fold.

3.5 Displacement Mapping

Displacement mapping can produce highly detailed textures and realistic looking objects, while not as expensive as straight modeling, it can also be computationally expensive and requires a high-resolution model.

This method can be challenging to work with since creating a high-resolution displacement map might not always be the best approach. This approach unlike the previous ones also actually alters the geometry of the object. This also fixes the silhouette issue of bump mapping because of the same reason.

3.6 Ambient Occlusion

If we want to achieve more complex shadows and soft shadows without using ray tracing there is also ambient occlusion. It is a way to simulate the way light is blocked by objects thus creating areas of shadow and relative darkness. This effect, while not as expensive as ray tracing, is still computationally demanding, but it can add an impressive amount of depth and realism to a scene. There are many techniques of ambient occlusion each with their own positives and negatives of performance, realism and accuracy.

3.7 Subsurface Scattering

There are also techniques to create semi-translucent objects with subsurface scattering. Or realistic acting reflections and lighting through ray tracing. The main common point between almost every technique listed is the closer a scene gets to being realistic, the more computation it will require and harder it will be to do so in real time.

While an offline render by for example Pixar can afford to use all of these techniques and even more complicated ones, real time applications such as games might choose to swap high polygon counts and ray tracing for normal mapping and environment mapping.

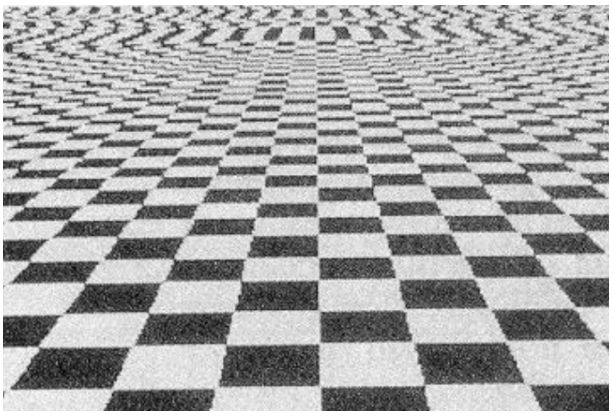
4 Practicalities and Limitations

Since the most commonly used technique is mapping a 2D image on a 3D object and scene and projecting that scene back to 2D screen there will arise many problems. And people have over the years have come up with practical solutions to these problems.

4.1 Aliasing and Anti Aliasing

Aliasing is visual artifacts that occur when a high-frequency signal is sampled at a lower frequency than its original. For graphics this can occur due to: Low resolution; if the resolution of the 3D scene is not high enough there will be errors at the edges of an object and it will be inaccurate. This will cause objects that are far away in a scene to appear with jagged edges. Sampling rate; While rendering a scene we will have to sample each for each pixel at discrete points in order to create a 2D image from a 3D scene. If this sampling rate is not high enough the image will not represent the edges in detail and edges of objects will be inaccurate. This can even lead to parts of the texture to go skip, or go missing. Movement of the object or the camera can also cause aliasing issues where it leads to non smooth edges while moving.

Figure 4: Example of chequerboard problem



All of these aliasing issues will cause the scene to look messy or “not done”. This is why anti

aliasing techniques are used to combat them. Anti-Aliasing works by averaging the colors of pixels around an edge of the object to essentially smooth them out. There are several techniques of anti aliasing.

Multi sampling anti aliasing(MSAA): this technique works by increasing the number of samples taken per pixel thus reducing the sampling rate problem of aliasing

Supersampling anti aliasing(SSAA): this technique renders the scene at a higher resolution then downsamples the image, thus reducing the low resolution aliasing problem

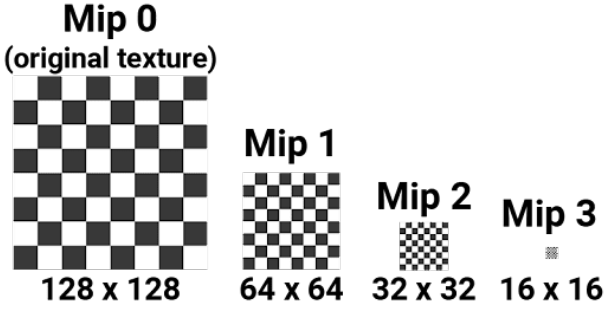
Temporal anti aliasing(TAA): with this technique multiple samples are taken over time to reduce temporal aliasing during motion

These techniques are often used in tandem with each other to create a better looking image. But as always with everything in 3D graphics there is the tradeoff of reduced performance.

4.2 Mipmapping

We don't always need to display incredibly detailed textures when an object is so far away from the screen that in screen space it will take maybe 4 pixels of the end 2D image of the scene. Thus is the idea behind mip-mapping. The further away from the camera an object is, the less screen space it will take therefore requiring less detailed textures. Instead of using a single, high-res texture for an object, mip-mapping creates many resolutions for the same texture and allows us to use a lower-resolution one that better matches the object's size on the screen. This technique can improve performance dramatically and reduce the amount of memory required.

Figure 5: An example mipmap taken from Unity Manual (released under CC BY-NC-ND)



4.3 Bilboarding and Imposters

Billboards use a 2D image on a plane to represent a 3D object. Typically they represent trees and other complex foliage. A billboard always faces the camera, so they are very simple to render efficiently, even when the user or camera is moving. We can also use multiple images that are layered rotated from each other to represent an object. Each image taken from a different viewpoint/angle. Thus the need to rotate with the camera can be eliminated which can produce an unsettling effect.

This effect by itself can reduce the amount of rendering required to represent complex landscapes with many trees and foliage like a forest mountain many times over. [3]

4.4 Texture Atlases

Another efficiency that can be used is to use a texture atlas instead of multiple smaller images to load our textures to the program. A Texture atlas is a large image that contains multiple smaller images, the multiple different textures don't necessarily need to be in sequential order in the atlas, thus using the space more efficiently especially in unwrapped complex objects. This technique provides a multitude of benefits.

Improved GPU performance, when we use a texture atlas we need to switch the texture file less times while texturing multiple objects, which reduces the amount of memory and cycles used by the GPU, increasing performance.

Reduced memory usage, because texture atlas is a single image, they use less overhead and can use space more efficiently than using multiple smaller files.

Easier management for programmers, by making a bunch of smaller files into one big one, the programmers are required to work with less of them which can make it easier to organize and manage textures.

Better compression, bigger images in general are compressed better than compressing multiple smaller images.

Reduced wasted space, when a complex object is unwrapped there will inevitably be wasted space in a texture file, texture atlases can fill this space in an easier way since there will be more opportunities to fit stuff together.

4.5 Texture Compression

Algorithms can be used to compress the textures we need to use ahead of time, and then decompressing it when it is needed for rendering. The biggest advantage of doing this will be the reduced memory usage which can be significant, it can also improve performance and reduce storage costs. A trade-off between storage space and quality will have to be made when choosing a compression algorithm.

4.6 Filtering

Filtering like mip-mapping is concerned with how a texture is shown at many different sizes. It is primarily concerned with improving the quality of texture when they are displayed at a lower resolution than the intended one. If filtering is not used unintended artifacting can happen with resolution changes. There are multiple techniques of filtering that are used:

Point sampling: This is a very rudimentary form of filtering, each pixel in the texture is mapped to a single pixel on the screen. This will produce a pixelated result when viewed from afar.

Bilinear filtering: By using the average color of surrounding pixels to calculate a color of the pixel we are looking for, this type of filtering will result in a smoother appearance. Similar to aliasing this will blur the end image.

Trilinear filtering: An extension of bilinear filtering, average of surrounding pixels are taken in both the current texture map and the next lower resolution in the mipmap. This will result in smoother and more detailed results than bilinear filtering when a lower resolution texture is needed.

Anisotropic filtering: Using a combination of point sampling and bilinear filtering, anisotropic filtering improves the texture quality when the object is viewed at an angle. This produces a more detailed appearance.

5 Rendering Approaches

None of the texturing techniques would matter, without a very crucial step of creating 3D graphics which is rendering. Rendering involves simulating lighting behavior in a scene and generating the shown image that represents that lighting.

However not all of the texturing approaches and techniques can be used with all of the rendering techniques, and they shouldn't have to since each rendering approach provides its own downsides and upsides.

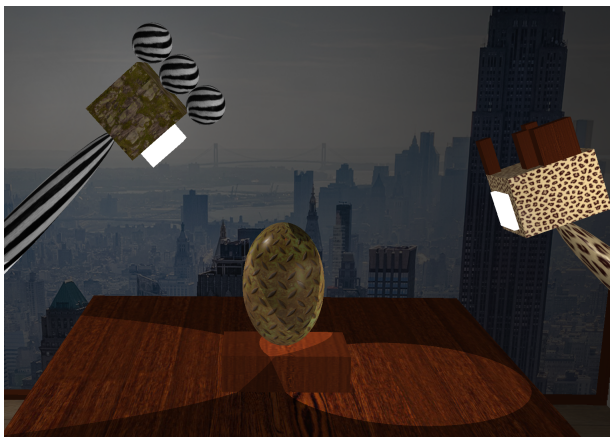
5.1 Phong and Blinn Phong

Phong and Blinn-Phong are simple lighting models that simulate and approximate the way light reflects off of a surface. They do this by calculating diffuse, specular and ambient components of light. By default this approach does not produce reflections, soft shadows or other lighting effects, this makes it very suitable for all of the texturing techniques we have looked at so far as most of them were techniques to help these shortcomings of this rendering approach.

Since this has been a tried and true method used for many years people have of course improved upon this model over time. There are approaches that use physically based Blinn-Phong models that produce very impresive results in real time. [4]

Blinn-Phong model produces more realistic specular highlights and is faster than the Phong model thus is used primarily instead. An example of what these specular highlights are trying to replicate in real world scenarios can be seen on the metal handle in Figure 1.

Figure 6: Example of specular highlights from assignment 1



A replication of specular highlight using Blinn-Phong rendering technique can be scene in Figure 6, This figure also shows us using 2d texture mapping to texture wooden and metal surfaces. A transparency effect can also be seen used, on the window in the back.

Pros:

- Very simple lighting models and are very easy to implement
- It is computationally inexpensive
- Well suited for real-time rendering

Cons:

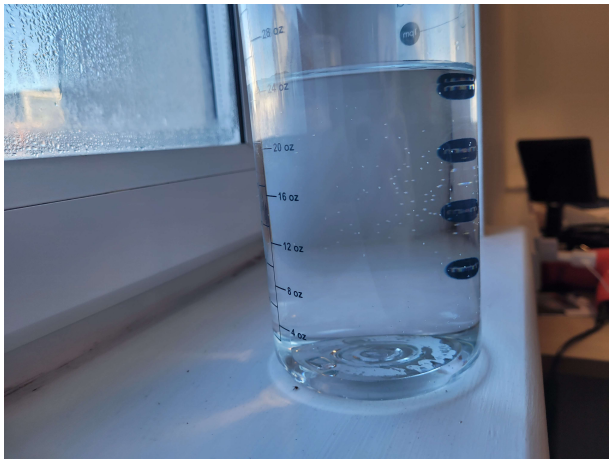
- No natural shadows or reflections without additions to the model
- Does not produce very realistic images.
- Does not take into account the way light interacts with surfaces
- Is only an approximation, especially ambient lighting is just a guess the developer makes

5.2 Raytracing

While the previous method approximates global illumination with ambient light, ray tracing technique simulates the behavior of light in a scene. It does this by producing a ray that is traced from the camera, through each pixel in the output image and to the scene. This has the advantage of naturally interacting with things like reflective surfaces, transparent surfaces by working recursively.

This allows this method to have effects like shades, shadows and reflections without relying on texturing techniques like environment mapping. This method is very powerful and versatile. Ray tracing will be able to reproduce the mirroring effect seen in Figure 3 especially with a high enough recursion depth.

Figure 7: Example of light reflecting from the shiny bottle also refracting going through the water



Producing something complex like the example shown in Figure 7 is just not practically possible without the help of ray tracing, texturing techniques like subsurface scattering can only go so far.[2]

Ray tracing can produce highly realistic images, however it is very expensive to do so computationally and can take a long time to render an image. In recent years it is slowly becoming possible to achieve results in real-time however this is still very difficult and nice.

Pros:

- High visual quality, can produce very realistic images
- Results are very accurate in terms of lighting
- Reflections, refractions, global illumination and ambient lighting can be simulated not just approximated

Cons:

- Very expensive to do so computationally
- Real-time performance requires specialised equipment
- Can be more complex to implement
- Cost can get out of hand exponentially when image size increases

5.3 Physically-based rendering (PBR)

PBR is based on principles of physics. It is a mimicry of the way that light behaves in the real world very closely. The most important thing about PBR is that it needs to use physically-based bidirectional reflective distribution functions BRDF for short. These mathematical functions calculate factors like reflectivity, roughness, albedo.

This method is much more complex than simple rendering methods like Blinn-Phong. It produces very realistic images like ray tracing by simulating the way that light gets scattered. It is possible to use PBR in real-time rendering which makes it an attractive option to use in games where performance is important.

Pros:

- Realistic images
- Can be used in low light conditions
- Can be used in real-time rendering applications

Cons:

- Much more complex method than Phong to implement
- Requires more information about the objects and surfaces in a scene which will take more time and effort
- There are higher conditions that need to be met to be able to use this method

	Bump Mapping	Environment Mapping	Normal Mapping	Displacement Mapping	Ambient Occlusion	Subsurface Scattering
Blinn-Phong	Can be used	Can be used	Can be used	Can be used	Can be used	Can be used
Ray tracing	Not needed since lighting is simulated	Can't be used since reflections are real	Not needed since lighting is simulated	Can be used	Ray-traced ambient occlusion is part of the process [5]	Reflections and refractions are real
Physically-based	Not needed because of PBR shaders	Can be used	Normal Map is part of the PBR maps	Can be used	Is used as part of PBR process	Refraction Map is part of the PBR maps

Table 1: Availability of different texturing techniques with different rendering approaches

6 Conclusions

There have been over the years many techniques, shortcuts and practicalities created in terms of texturing and rendering in 3D graphics. These have had spectacular results in making scenes more realistic and impressive, while also reducing the amount of computational power needed to render a scene at the same time.

While impressive in the recent years technological advancements like real-time raytracing have made some of these obsolete in terms of realism and simulation. However as the realism improves in scenes other techniques have become much more important to the rendering and texturing processes.

References

[1] M. J. Kilgard, "A practical and robust bump-mapping technique for today's gpus," in *Game Developers Conference 2000*, pp. 1–39, 2000.

[2] L. Szirmay-Kalos, B. Aszódi, I. Lazányi, and M. Premecz, "Approximate ray-tracing on the gpu with distance impostors," in *Computer graphics forum*, vol. 24, pp. 695–704, Citeseer, 2005.

[3] S. Behrendt, C. Colditz, O. Franzke, J. Kopf, and O. Deussen, "Realistic real-time rendering of landscapes using billboard clouds," in *Eurographics*, vol. 24, 2005.

[4] Y. Gotanda, "Beyond a simple physically based blinn-phong model in real-time," in *ACM SIGGRAPH*, p. 10, 2012.

[5] P. Gautron, "Real-time ray-traced ambient occlusion of complex scenes using spatial hashing," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*, pp. 1–2, 2020.