TEESSIDE UNIVERSITY

Real-Time Graphics

Light Up My World

Matthew Neale

5/5/2010

TABLE OF CONTENTS

INTRODUCTION

I have been tasked with designing and implementing a 3D graphics demo that must run in real-time and produce high quality imagery of the given scene.

The next section looks at the ICA brief and establishes the most appropriate rendering techniques for this project. Also discussed are the performance testing measures employed when profiling the application and any changes to the design of the application due to the results of the profiling.

In the Evaluation Section the Decisions made concerning Which Graphics API, Shader models and Shading Techniques to use will be assessed. The Application as a whole is also assessed against other similar applications, for its visual quality, performance and other factors such as scalability, and complexity.

ANALYSIS AND DESIGN

ANALYSIS OF THE BRIEF

The Geometry the application is required to use is a model of Slovakia's Sponza palace. As this is the case it is assumed that the style of the shading should be realistic as opposed to overly stylized/cartoon shaded etc.

The brief states that the scene to be displayed is a small enclosed environment this means, for example, techniques like fogging will add little to the scene. This also implies objects will occlude each other and affect each other's shading significantly.

The camera system must move but the geometry does not have to, this allows optimisations to be performed, for example, if the geometry is assumed to be constant, the GBuffer (deferred shading is used in this application as discussed below) only needs updating when the camera moves.

Only an objects base colour can be used as the diffuse colouring in the application this means extra attention should be made to the parts of 3D scenes that add realism beyond high detailed textures, such as, normal perturbing methods e.g. normal, parallax and relief mapping.

Some shading methods specific to the Sponza geometry can be utilised as this is the only scene being used.

The Sponza model includes objects marked with metal, fabric, leaf and other material types this knowledge can be used to home the shading to the specific scene making much more of an impact than a generic shading model, this, however makes the Application less reusable in other situations but for the purposes set out in the brief this is the best method for getting good results.

The Brief requires many light sources to light the scene, deferred shading lends its self to scenes with multiple light sources as the geometry's positions, colours and normals only need computing once no matter how many light sources are in the scene. Differed shading is also useful with the optimisations stated earlier.

As the brief states all lights must move optimisation's like the one mentioned earlier cannot be transferred to the lighting system.

The application must run in full screen mode at the resolution of 1280 by 800 at frame rate of at least 30 fps. Direct3D Offers good shader support and profiling opportunities over Open GL. PIX is an extremely useful tool for debugging Shaders Open GL has BuGLe but that only works on Linux. Perf Hud is only available for direct3D although the Perf SDK is available for OpenGL users. OpenGL has gDebugger which unfortunately costs money and glslDevil which seems like a good application, but due to the fully rounded support set up for direct3D and my existing DirectX knowledge I created the application with the DirectX SDK.

OVERVIEW OF MAIN TECHNIQUES

- Deferred Shading – this allows for multiple lights with a greater efficiency than forward shading but an overhead of a GBuffer pass.
- Shadow Mapping – A quick way to add shadows to a scene, I create a shadow map, splat to the scene white where the scene should be lit, then blur this image so that when it comes to shade the area the intensity of the shading can be multiplied by the intensity of the image creating a nice fall off even for far away shadows.
- Ambient Occlusion – if light is not being cast on a surface it should be completely black. This nearly never happens in real life as objects are almost always slightly reflective and therefore some light will almost always get to a surface, ambient occlusion is a relatively inexpensive way of adding some realism to a scene.
- HDR Rendering and Tone Mapping – this technique coupled with carful variable selection can really add to the overall realism of the scene, by bringing the scenes colour range back into the range of the monitor the viewer can see more detail in the scene, by choosing the value best to be seen we can add levels of realism by, for example, making the overall scene darker when you stare at a light for too long.
- Reflections – the data files provided of the scene describe different types of surface. E.g. metal. By homing Shaders to specific surface types, realism and authenticity is added as objects in the scene act as is expected of them.
- TV filter – the final process of Rendering is the TV pass it adds subtle noise to the image to help hide any banding left after the Tone Mapping pass.

SCREEN SPACE REFLECTIONS

The Reflections In the Application are not the traditional reflection implementation; texture look ups into a cube map for each reflecting object. my method approximates the reflected colour based on the screen space refection of the normal. This will obviously never give the correct result but it requires lookups from one texture instead of 6 and the texture does not need to be created especially for the reflections, it already exists. this technique is extremely fast when compared to cube mapping, and although cube mapping gives a better result and allows for off screen data to be taken into account, the issues with my method are negligible when used on bumped or warped surfaces where the viewer gets a sense of reflection without needing the real reflections.

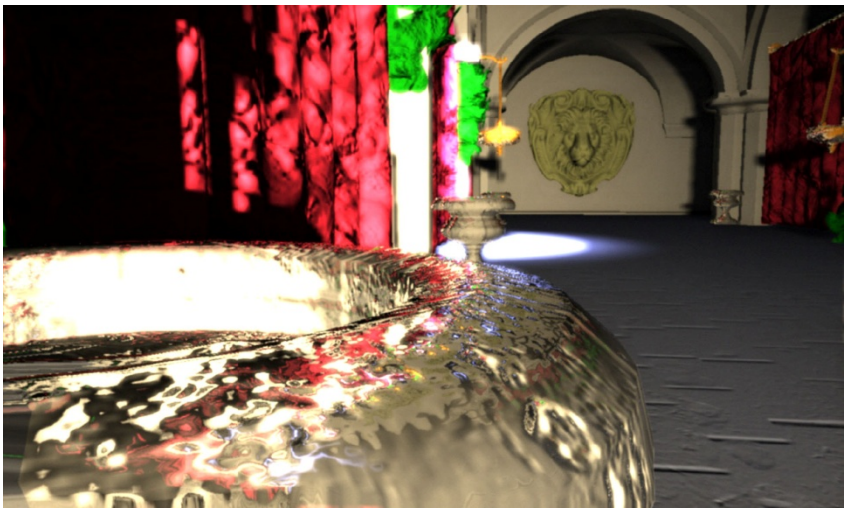


Figure 1 Screen Space Reflections

PERFORMANCE TESTING AND PROFILING

Using PerfHUD I was able to perform advanced profiling of my application finding where bottlenecks occurred. My Application used to slow at the shadow mapping stage of rendering in the pixel shader. I was drawing a full screen quad for each light source for both the splatting of white to the scene, the blurring and the shading of the scene. I removed the blur and shaded the scene in one pass instead of two. This sped the application up, but significantly lowered the shadow quality to a standard I was not happy with. Instead of rendering a full screen quad I render a cone representing the lights path this means when a light is not on screen the pixel shader is never called. The next bottleneck now is creating the GBuffer. I have yet to resolve this issue, but I have some inefficiencies that a can remove. For a start I store the world positions in a D3DFMT_A16B16G16R16F buffer I could get the Screen space positions from the depth buffer saving write operations to an extra buffer and saving a whole Render Target of memory. I also store the normals in a D3DFMT_A16B16G16R16F buffer which I initially used to for storing other information (shininess for specular) this has been left in for ease but should be changed as normals could be stored as two components as the third can be recreated as normals are of unit length.

My tone mapping is partly done on the CPU , this allows me greater control over how to use the Tone mapping technique to affect the image ,but it is a GPU and a CPU stall. This is made more noticeable as it is an intermittent stall so the flow of the application gets staggered. This problem could be helped by choosing a lower mip-map level to pass to the CPU. The reason I didn't do this is I had an idea to increase the ambience of the scene by adding non shadowing spot lights to the scene in regions of high intensity light these lights would face in the reflected direction of the light bounce with the regions colour as its colour and intensity. I ran out of time before implementing this and so haven't seen how this would turn out , but I hope it would add an adequate approximation of colour bleeding and global illumination. Alternatively I could do the tone mapping on the GPU, solving the stall ,as no blocking operations would occur, but losing some of the flexibility of the technique.

EVALUATION

When a project such as this is taken on by a company within the computer graphics industry the project would have a team of programmers and artists and a longer time period in which to produce the artefact. With this in mind, my bench mark for evaluating the success of the project is a games engine of a small games company in mid production.

I feel the design of the engine allows for great scalability as it is modular in nature and I can quickly add or remove parts of the program without affecting the flow of data.

VISUAL QUALITY

The Application has some nice features like the light correction that occurs when moving from areas of different luminance values; this mimics the human eye's adaption to different light environments.

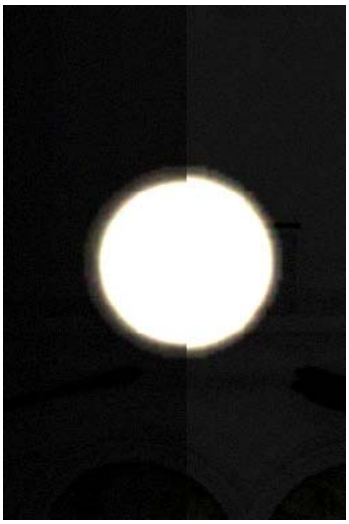


Figure 2 shows the contrast between first looking at a dark area and giving your eyes enough time to adjust to the darkness

The metal effect coupled with the bloom from my tone mapping gives good results compared to the complexity of the effect. The effect is simple and in no way graphically correct but it really enhances the scene.

Figure 3 screen capture of my metal effect with bloom

If I had more time I would focus more on this sort of effect adding semi-transparency and dust in the light and other subtle effects that add to the overall scene.

I am happy with the visual quality of the product compared to a mid production game engine, it is in no way finished but it's a good start.

PERFORMANCE

The Application runs between 40 and 60 frames a second, this is an adequate frame rate for a game engine as my work is mainly GPU bound if this application was to be taken forward as a game engine I would move the DirectX Present function to be called after the game code, as this function is one of the only blocking functions I use for the Graphics Engine, this would allow the GPU to process the scene while game code is running, hopefully maintaining the frame rate.

TECHNICAL ACHIEVEMENT

My application does not have any advanced features, this is the main issue in my opinion with my application, I have used very basic techniques and perhaps not implemented them as efficiently as I could have, but this is what would be expected mid way through production on a graphics engine. I have definite inefficiencies in the code as I have built upon ideas and experimented during production. These should be found and changed during testing and refinement stages of production.

CONCLUSION

I am missing techniques that are considered standard in commercial Graphics Engines, I am also being inefficient in a number of ways, I also feel my knowledge of 3D graphics and the Graphics pipeline could and should be better than it currently is. However my determination and ability to critically evaluate Graphical problems is good, and the techniques I have implemented I fully understand and can build upon. This Demo has Normal mapping, shadow mapping, Tone mapping Ambient Occlusion and approximate reflections and under most situations looks good, if I had more time I would have improved my shadows as these are the most noticeable issues with the demo changing to variant shadow maps would be a good start after which I could assess their effectiveness.

USAGE INSTRUCTIONS

the input keys are loaded via xml from the file "xml\input\move.xml" you can edit this to whatever key setup you want, this application also works using the Xbox controller and the ps3 controller, Guitar hero guitar, bass and drums but I suggest just using the keyboard and mouse as drumming around Sponza can get tiring.

Key	action
z	Adds a light at your position in the camera direction
Controller left analogue stick up/ w	Move forward
Controller left antilog stick down/ s	Move backward
Controller left analogue stick left/ a	Strafe left
Controller left antilog stick right/ d	Strafe right
Mouse move/ Controller right analogue stick move	Rotate camera
Escape/start button	quit

a full list of keys is available in the Appendices.

the frame rate for the current session is stored in an xml file after the program has exited. the file can be found in the "xml\statistics\" folder.

SPECIFICATION

- deferred shading
- normal mapping
- Ambient occlusion - this technique needs work on it but it does add a sense of indirect lighting
- Shadow mapping - this would need to change for lights that are far away from the viewer but shadows are fully functional and look reasonable.
- reflections - as stated earlier, reflections do not reflect in a 'real' way nor are they meant to, this is an approximation of a reflection meant to give the feel of true reflections, this should only be used on warped or un even surfaces where true reflections are not necessary.
- HDR rendering - the scene is rendered using floating point colours with an effectively infinite intensity range.
- Tone Mapping - before presenting the scene to the screen I find an appropriate range for the lighting levels and change the image to the light range, I have strived to make this as near to how the human eyes work as possible for example I cap the maximum intensity of light that you can see and made the adjustment the speed it takes human eyes to adjust to different light levels.
- I have various anti-aliasing techniques within the application, the Shadow mapped area of the screen is blurred giving soft shadow edges and less shadow artefacts. There is a TV static post possess which hides banding. Also the HDR lessens banding as does the tone mapping as the change in colours is subtler.

BIBLIOGRAPHY

Bavoil, S. (2008). *Screen Space Ambient Occlusion.* Nvidia.

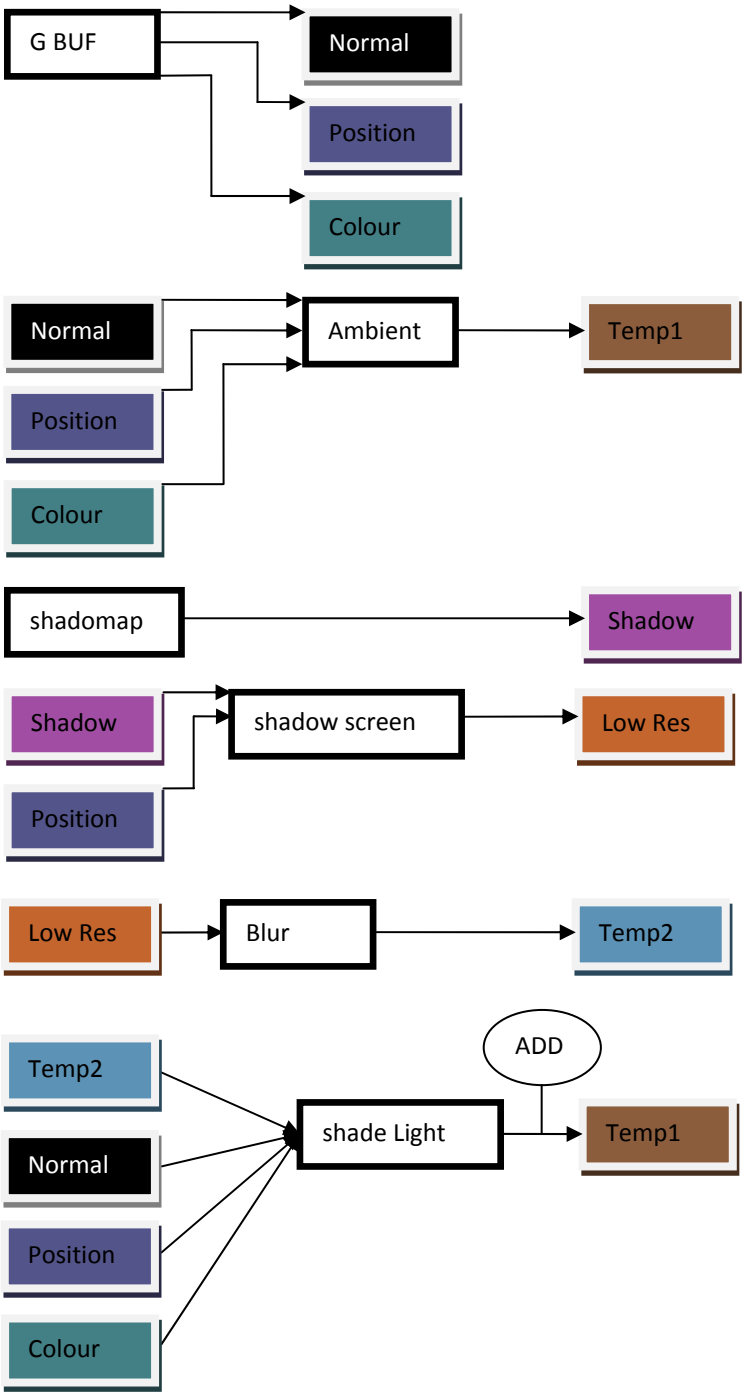
Mittring. (2007). *Finding Next Gen – CryEngine 2.*

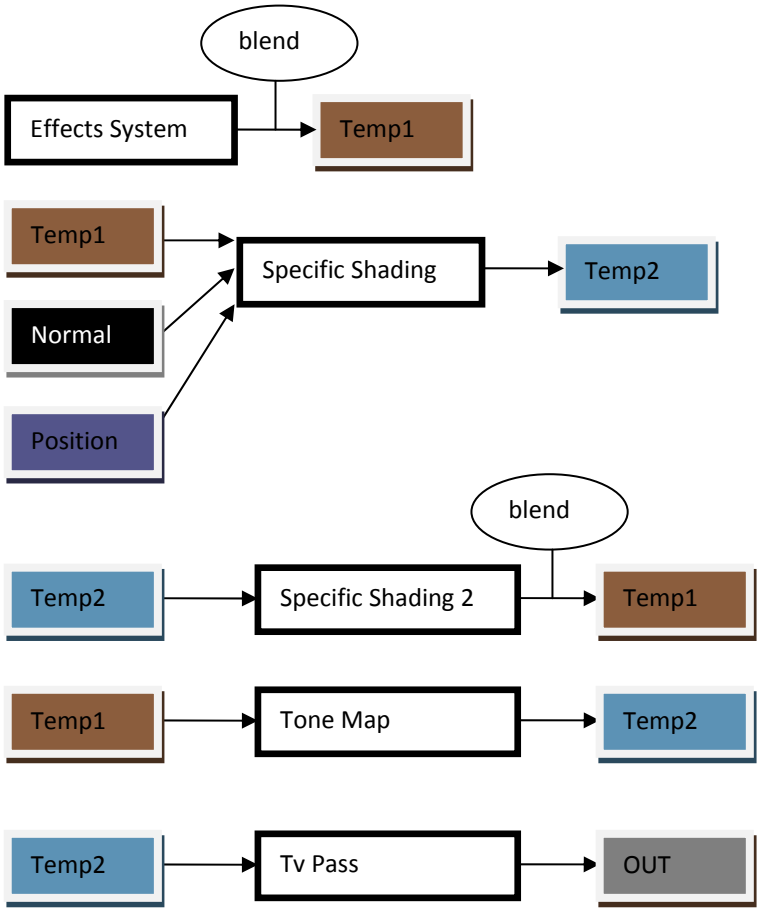
Reinhard, S. (n.d.). Photographic Tone Reproduction for Digital Images.

Szirmay-Kalos, U. (2009). *Volumetric Ambient Occlusion.*

APPENDICIES

DEFERRED RENDER TARGET FLOW




blend
Effects System
Temp1
Temp1
Specific Shading
Temp2
Normal
Position
blend
Temp2
Specific Shading 2
Temp1
Temp1
Tone Map
Temp2
Temp2
Tv Pass
OUT

READ ME

some of this engine is used in other projects and some was written during my placement year. all the graphics code was written specifically for this module. the user input, XML parser and sound components were written for my own personal projects and should not be assessed.

MOVE.XML

```
<?xml version="1.0" encoding="utf-8" ?>
- <actionmap>
  <inputtype>all</inputtype>
- <action>
  <key ascii="">p</key>
  <message>switch render</message>
    </action>
- <action>
  <key ascii="">z</key>
  <message>switch cam</message>
    </action>
- <action>
  <key>left analog stick: up</key>
  <message>move forward</message>
    </action>
- <action>
  <key>left analog stick: down</key>
  <message>move backward</message>
    </action>
- <action>
  <key>left analog stick: left</key>
  <message>move left</message>
    </action>
- <action>
  <key>left analog stick: right</key>
  <message>move right</message>
    </action>
- <action>
```

```
<key ascii="">W</key>
<message>move forward</message>
  </action>
- <action>
<key ascii="">S</key>
<message>move backward</message>
  </action>
- <action>
<key ascii="">a</key>
<message>move left</message>
  </action>
- <action>
<key ascii="">d</key>
<message>move right</message>
  </action>
- <action>
<key>up</key>
<message>turn up</message>
  </action>
- <action>
<key>down</key>
<message>turn down</message>
  </action>
- <action>
<key>left</key>
<message>turn left</message>
  </action>
- <action>
<key>right</key>
<message>turn right</message>
  </action>
- <action>
<key>right analog stick: up</key>
<message>turn up</message>
  </action>
- <action>
```

```
<key>right analog stick: down</key>
<message>turn down</message>
  </action>
- <action>
<key>right analog stick: left</key>
<message>turn left</message>
  </action>
- <action>
<key>right analog stick: right</key>
<message>turn right</message>
  </action>
- <action>
<key>start</key>
<message>quit</message>
  </action>
- <action>
<key>escape</key>
<message>quit</message>
  </action>
  </actionmap>
```

SCREEN SHOTS