“MAXIS-mizing” Darkspore*:
A Case Study of Graphic Analysis and Optimizations in Maxis’ Deferred Renderer

Released in early 2011, the 2nd Generation Intel® Core™ processors (codenamed “Sandy Bridge”) have fundamentally changed the PC gaming landscape, with processor and graphics merged into a single piece of silicon. The tighter integration of graphics into the processor has revolutionized gaming performance and made your favorite games run and look great on mainstream machines. As a developer, the market for your games has opened up immensely.

Targeting the mainstream is possible and Intel provides you the tools to help achieve success in this market. Intel® Graphics Performance Analyzers (GPA) 4.0 was released at GDC 2011 with full support for this next generation of processors. Game developers including the Darkspore team at Maxis have taken full advantage of Intel® GPA 4.0 features to make sure their game fully utilizes the new processor graphics.

A New Gaming Experience Made Possible With Processor Graphics

Figure 1: PC Volumes Dwarf Consoles through 2014

Figure 1: PC Volumes Dwarf Consoles through 2014
Darkspore is Designed for Everyone

Darkspore is best described as an online action RPG where you control a squad of 3 heroes to fight the Darkspore, an evil infestation of creatures. The game includes a version of the award winning Spore Editor where you can fully customize not just basic features of the characters like color, height, textures, but vertices in the characters’ meshes. With Darkspore, Maxis is targeting mainstream to high end graphics hardware. At the Game Developers Conference 2011, David Lee Swenson, the Lead Engineer on Darkspore, presented how he used Intel® GPA 4.0 to help achieve this goal.

Darkspore Rendering

For Darkspore, Maxis implemented a variation of the light pre-pass renderer described by [Engel 2008]. One main advantage of using a light pre-pass renderer is to decouple lighting from the scene geometry. In a forward renderer, lighting is computed at the time the object is drawn. In a light pre-pass renderer, light is accumulated in an off screen buffer and either applied in a post pass or sampled in a final pass per object. The Darkspore renderer is composed of three main passes: deferred pass, lighting pass, and final pass. The deferred pass saves the material data for opaque objects in the scene, the light pass saves all lighting calculations into a light buffer, and the final pass uses the results of the previous passes to create the final frame. The deferred pass uses 2 render targets. The first render target is composed of world normals with a gloss term: Normals (RGB) + Gloss (A).

Figure 2: First render target = world space normals (RGB) + gloss (A)

Depth ([R*256]G), Specular Power (B), and a Toon ID (A) for a toon-style character outline are rendered to the second target of the deferred pass.

Figure 3: Second render target = Depth ([R*256]G) + Specular Power (B) + ToonID (A)

Following the deferred pass, there’s a lighting pass that renders up to 6 parallel lights as well as cloud shadows and the main shadow. Then, each point or spot light are rendered with gobos and/or shadows to a 16F light buffer: Diffuse (RGB) + Specular (A).
During the final pass, each object is rendered again sampling the light buffer. Values for areas intended to glow are written to a second target. Then, the glow target is downsampling and blurred back together with particles and post-processing effects like fog, distortion, and the death effect.

At this point all it needs is the UI and the final frame is complete.

**Darkspore and Intel® GPA**

Before Intel® GPA, the Darkspore team at Maxis was using a long list of tests and settings to understand and debug game performance. With Intel® GPA, the game can be run with the applicable command line options. Once a frame is captured, it’s just a matter of setting the X and Y axis to the appropriate metrics to get a useful profile of performance.
Since Darkspore was known to be pixel shader bound, the Darkspore team typically would set the X axis to the “PS Duration” metric and the Y axis to “GPU Duration”. With these metrics, the taller the bar, the more GPU time it is taking and the wider the bar the more pixel shader time.

The four calls labeled above (A, B, C, D) are the most expensive in this frame capture. Calls A and C correspond to the deferred and final pass for the blood decals. Call B corresponds to the parallel lights, cloud shadows, and shadows. Call D runs the edge detection that was originally a bigger part of the game but has since been moved to the high spec configuration. Looking at the shaders for the blood decal calls (A and C), a few issues were found and fixed by the Darkspore team:

- Tiling of decals was supported but never used
- Vectors were normalized that were used only for a cubemap lookup
- A Fresnel term was adding very little to the scene, given the fixed camera angle
- Alpha test was implemented with both a clip instruction and blending
- Normal calculation was overly complex with values that could be moved to the vertex shader
After optimizing the decal shader and moving character shading to the high spec, re-profiling Darkspore only shows Call B that corresponds to parallel lights and shadows. Given the amount of work done in Call B, it is expected to be expensive but grouping all these tasks into one call amortizes the cost of picking up the normal and depth values and recovering the position.

**Hot Loading Shaders in Frame Analyzer**

The Darkspore team used Frame Analyzer to verify the changes made to the decal shaders had the positive impact they expected as shown above. But, they also took advantage of the hot loading shaders feature of Frame Analyzer to test changes on the fly. Frame Analyzer allows for replacement of shaders in HLSL or shader assembly for selected calls. By hot loading shaders in Frame Analyzer, you can immediately see the performance difference of the changes without having to capture another frame and hopefully recreate enough of the same events in the scene to make it comparable.

Using this hot loading functionality, the Darkspore team was able to rapidly test changes made to the decal shaders. After loading the modified decal pixel shader for calls A and C, the deferred and final pass draw calls, Frame Analyzer showed a 30% and 24% improvement respectively for these calls.

**They Drew First (Deferred) Blood**

Looking further at these two calls that draw the blood decals in Frame Analyzer, we can see their volumes if we set the render state to wireframe. The blood decals are effectively writing all of the pixels in the volume because there is no test in place to early out in the pixel pipeline. The stencil buffer can be used to kill pixels in the final pass.
The Darkspore team made several miscellaneous optimizations to the trees, terrain, character detail, and render system. The forest levels in the game had lots of trees. Maxis found all the trees’ models had roots below the ground that were being rendered, adding unnecessary polygons that no one ever saw. These were promptly removed saving processing time.

Other Optimizations for Darkspore

The Darkspore team made several miscellaneous optimizations to the trees, terrain, character detail, and render system. The forest levels in the game had lots of trees. Maxis found all the trees’ models had roots below the ground that were being rendered, adding unnecessary polygons that no one ever saw. These were promptly removed saving processing time.

After making the change to the final pass and doing a simple frame rate check at the beginning of the level, there was a noticeable frame rate improvement of about 2 FPS. In Frame Analyzer, this new stencil write test can be set on the decal calls to fully understand the effect:

1. Select both calls A and C (deferred and final pass) and setting STENCILENABLE to true.
2. Then for call A, set STENCILFUNC to D3DCMP_ALWAYS, the STENCILPASS to D3DSTENOILOP_REPLACE, and STENCILREF to 2.
3. For call C, set STENCILFUNC to D3DCMP_EQUAL and the STENCILREF to 2.

Figure 13: Dense tree root geometry was drawn but not visible

Figure 12: Blood decals now being stenciled correctly
In Darkspore, the terrain mixes 4 textures together per pass. The artists have control of which textures are mixed per vertex. Large sections were found to only need one texture instead of mixing four. These triangles that had only one texture were instead rendered with a simpler material. This was a big win in some cases and smaller in others, but always a win.

With the Spore Editor, the player can customize their squad creatures with countless combinations of parts and equipment. This results in some very detailed but also very polygon heavy creatures. The wireframe of the high LOD (level of detail) for a player creature shows the high polygon density of the creatures’ models. NPCs were found to be at least this dense. Surprisingly, when these creatures got reduced to the one triangle per pixel level, they were hurting both pixel and vertex shader performance because of the way most graphics parts allocate 4 pixels to a “quad” and a minimum of one quad per triangle. This resulted in a waste of 3/4ths of the pixel shader performance.
As part of the light pre-pass renderer, normals were originally saved in view space. The view space normals only required two channels but weren’t worth the cost of pixel shader instructions to recover the normal. Also, a world space normal was needed for reflective/refractive objects which resulted in a transform in the pixel shader, which would be a really bad place for an extra matrix transform. In the end, it was worth an extra channel to keep the normal in world space and avoid adding the extra matrix transform to the pixel shader.

**Conclusion**

Using the various features available in Intel® GPA 4.0, the Darkspore team was able to discover and fix bottlenecks in their graphics pipeline. At the end of the day, many of the optimizations made for mainstream graphics improved the overall gaming experience. With these optimizations in place, Darkspore runs at well over 30 FPS on the 2nd Generation Intel® Core™ processors (codenamed “Sandy Bridge”).

**References**


**Authors**

David Lee Swenson (Electronic Arts) is a 20 year software industry veteran. He’s Maxis lead engineer on new rendering architecture and art pipelines for Darkspore. Previously, David was responsible for the Spore environment rendering and terraforming systems. He has also worked on console and PC titles at LucasArts, 3DO, Sierra On-Line, Orion, and MediaFactory.

Omar A Rodriguez (Intel Corporation) a graphics software engineer in the Intel Software and Services Group, where he supports Intel graphics solutions in the Visual Computing Software Division. He holds a B.S. in Computer Science from Arizona State University. Omar is not the lead guitarist for the Mars Volta.