Credits:

Fredrik Mäkeläinen - Programming

Originally from Åsarna, Sweden, Fredrik is currently located in Skellefteå, Sweden, persuing a BSc degree in games programming at gsCEPT. His love for graphics and game programming started early with the “demoscene” and he feels that (almost) nothing beats the visual feedback generated from graphics programming. With a constant hunger for new technology and with a past experience from the ordinary programming industry at MAP Skandinaviska AB, he felt that why have game programming as a hobby when he could make It his living.
For more information, contact him at fredrik.makelainen@home.se!

Oskar Holmstrand - Graphics

Originally from Stockholm, Sweden, Oskar is currently located in Skellefteå, Sweden, persuing a bachelor degree in computer graphics at gsCEPT. A vivid imagination, a keen eye, the will to create.
Combining skill with interest, it’s no wonder that it’s game graphics that Oskar wants to do. Past experience include work on a procedural terrain visualization engine, Blueberry 3d(tm) at Sjöland & Thyselius Virtual Reality Systems, and work on a massive multiplayer persistant universe space-trading game at Redeem Interactive Studios.
For more information, check out www.oskarholmstrand.com!

LULEÅ UNIVERSITY OF TECHNOLOGY

Shades of Divinity

3rd year specialization project, 22.5 ECTS

Shades of Divinty was developed by two students during a 30 week half-time period. The project consisted of 3 separate parts each spanning 7.5 ECTS - planning, research and execution.

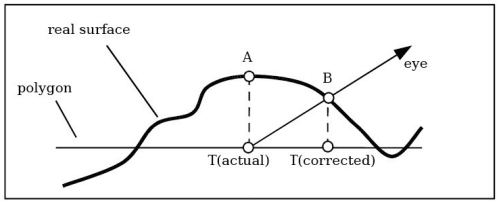
The goal was to test what can be accomplished in realtime graphics utilizing the newest hardware. Inspiration came from state-of-the-art engines, like the Unreal 3 engine and the Half-life 2 engine.

Shades of Divinty can be downloaded from http://www.oskarholmstrand.com/sod

Programming:

Maya Exporter. Because Oskar is used to working in Maya and It's the tool the school has to offer I, started this project by writing an exporter for it. The exporter is coded as a plugin for Maya 6.0 in C++ using VC.2003 and the Maya C++ API. It exports one mesh, bound skeleton including bindpose, triangulates polygons, splits vertices depending on smoothing groups and UV-seams, calculates tangents and binormals used for the normal mapping, rotational and translational keyframes and the vertex skin weights for smooth skinning.

Shading input. A lot of the shading data is sampled from textures to give Oskar detailed control of the rendering. Shading parameters coming from textures include diffuse color, specular color, shininess, normal, reflectivity, height, attenuation, light color.



Ground. Is rendered using normal mapped per pixel diffuse and specular lightning according to the phong model. It features parallax mapping for greater 3D feel and receives soft shadows.

Angel. Is rendered using normal mapped per pixel diffuse and specular lightning according to the phong model. It features environment reflections and generates / receives soft shadows. The morphing affect all textures (diffuse color, specular color, normal, shininess, reflectivity). Is smooth skinned using Kochanek Bartels hermite spline for translational interpolation and SLERP (Spherical Linear Interpolation) using quaternions for rotational interpolation between keyframes.

Ambient lightning. Implemented as in half life 2 so detail from the normal map is visible even when shadowed, traditional ambient lightning just tend to flatten everything. It works by specifying an ambient cube which measures the color of the light flowing from that direction in space. Each pixel rendered samples and blends between 3 of the ambient cube sides depending on It's normal.

Spotlight. A projected texture is used for the color, and an attenuation map to control the falloff. By using textures for these parameters instead of a mathematical equation makes it easier to create good looking lightning.

Sky. Rendered as an ordinary skybox at "infinite" distance from the viewer.

Shadows. The technique used is called shadow mapping and a 1024x1024 map is used in SoD. Five 2x2 PCF (Percentage Closer Filtering) samples are taken per pixel to give "soft" shadow edges.

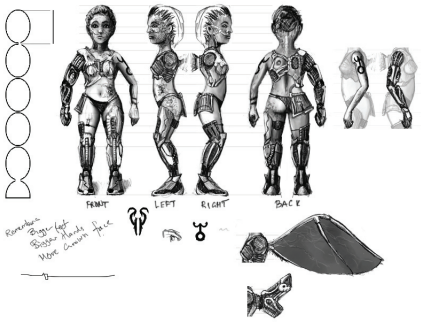
Volumetric light and fog. Is lit according to the lights projected texture, receives shadows from blocking objects. The fog is generated using three octaves of noise from a 16x16x16 texture and is rendered at a much lower resolution then the screen but via smart filtering scaled to minimize aliasing.

Morphing. The morphing is done per pixel and is an ordinary linear morph between the source and target textures.

Demo engine. Is coded using C++ and Direct3D 9.0 in VC.2003. All shaders are written in HLSL and some of them makes use of pixels shader 3.0 because of their length and branching requirements. The demo is 100% fillrate bound and the only thing that's done on the CPU is the smooth skinning, everything else is purely executed on the GPU.

Graphics:

Concept. The first task set upon me was to design a character which didn't differ in general shape, but greatly differed in style. After thinking about my options I decided that evil/good approach would be good. Also affecting the choice of character was the fact that there is only one animation, effectivly excluding any character combination which would require differing animation. (ex. Zombie). After reading up on angelology and sketching some on regular paper, I did a the final sketches with the artpad in photoshop. By using fading layers, I could test my morphs as early as in the concept phase.



Modelling. Since both models use the same low-polygon mesh, it was only natural to do it first. The modelling required knowledge of how normalmaps are generated, since this greatly affects the topology. The final lowpoly model landed on 3070 triangles. For the highpoly versions, I divided the character in logical parts anywhere there would be hard edges, for easier accessibillity and overview. The good highpoly version was created in plain Maya. Additional detail was then added by converting a bumpmap to a normalmap. The evil high-poly was created in Zbrush, as this eased up achieving the disgusting look I was after. The highpoly versions summed up together was perhaps 3 million triangles.

UV-mapping. When creating the UV-map, I had many considerations. The most important was to create a mapping that made it easy to paint. Secondary consideration included placement of seams, placement of UV-elements for easier access, mipmap generation and UV-space utilization. If you optimize the UV-space totally, you may end up with a map that is very tiresome to paint, and thus inevitably lowers the quality of the resulting map.

Texturing. Mainly using the normalmaps as guides, I created the textures in photoshop. I started by roughly blocking out just the basic colours to make sure the transition was adequately smooth, and that the composition worked. All texturing was done from scratch with the artpad. This may not be optimal for a production environment, but I figured I would take the time to hone my skills while still in school. By placing different materials (eg. rust, red plastic etc) on different photoshop layers, I eased up the creation of the other maps. Now I could simply desaturate and adjust the brightness of a layer to adjust its attributes.

Rigging. Not being a very strong rigger, I decided to try to keep the rig as simple as possible. Reverse foot setup, IK for the hands, expression driven spine. A thing to constantly consider when rigging was that everything should be able to be baked into rotations except the root-joint which also had translations.

Animation. I started by sketching out a few key poses as line drawings. These poses was what I wanted, everything between them would be on-the-fly improvisation and logical thinking. I considered using motion capture, but rejected the idea due to the fact that I wanted the wings to be an integrated part of the animation, not just some loosly attached items on her back. I also had a session with a live actor, which helped me decide more clearly what I wanted.

Final words. As always, knowledge is increased when producing content. Many mistakes I made I could go back and correct, as my production pipeline was pretty flexible. For other mistakes I had to do tedious work-arounds to solve.