Submission protocol 1

Group swarm

GIANI Carlos Rafael, e0325834, E 033 932, e0325834@student.tuwien.ac.at
ZENDEL Oliver, e0125096, E 786 881, e0125096@student.tuwien.ac.at

1. Idea/Storyline

Within a factory/foundry scenery a ball of metal trash is crashing into the ground. The remaining shards suddenly jerk to motion and arrise as a quasi-inteligent swarm. From now on the camera is fixed behind the swarm following it. The swarm itself seems to follow a path through the foundry while each individual shard is moving more erratically to stay within the swarm.

As the swarm passes over a glowing molten metal path to shards begin to melt as well and all the ugly spikes and cuts are dripping away leaving smooth nearly insect like metal objects.

A shimmering light enters the scene and the swarm starts to follow the light instead of the old path. The light seems to try to get away and flee from the swarm, but the swarm keeps up and finally touches the light. It explodes and the explosion wave knocks the whole swarm onto the ground. While falling the pieces are electrified and turn into real insects that keep lying on the floor.

After a short break the individual insects awake and format the swarm again. They fly through parts of the factory to arrive at a large open shippment hall. The camera keeps back while the swarm flies into freedom

2. Effects

2.1. Deferred Rendering

We are using the classical approach to deferred rendering, which is: store material properties into a G-Buffer, and in a second pass, light areas using the data from the G-Buffer. Variants such as Light-Indexed Deferred Rendering [1] and Deferred Lighting [2] have been evaluated. Their drawbacks outweigh the benefits, especially for the short duration of this course. To combat the problem with the lack of material diversity, a 3D lookup texture is used [4].

We are using a G-Buffer with the following components:

(a) R10G10B10A2 : normal encoded in RG using the spheremap transform [3] , material type ID encoded in G [4], alpha is unused as of yet

(b) RGBA8: albedo encoded in RGB, alpha is unused as of yet

(c) R32F: linear depth

(d) RGBA8: screen-space distortion encoded in RG, glossiness encoded in B, specular exponent encoded in A (the range 0..1 gets mapped to 1..127 in the shader)

In case we do not find any use for the alpha channel of (b), we will drop the spheremap transform in (a), encode normals as plain XYZ vector in its RGB channels, and move the material type ID to (b)'s alpha channel instead.

Translucent objects will be forward-rendered. Since we do not expect many translucent parts, and none with complex lighting, this approach is sufficient.

Deferred rendering is already implemented in the first version of the demo.

2.2. Shadow Mapping

Shadowing will be done by using shadow maps, as presented in the course. Shadow maps will be used for omnidirectional lights (point lights), spot lights, and directional lights.

Directional lights will use one shadow map, projected on the scene. We expect few directional lights to be in use in the final scene, most likely only one, the sun. We will first try to implement the directional light shadow map in its straightforward form. In case shadow quality does not suffice, we will attempt to use Cascaded Shadow Maps [5].

Point and spot lights will share code paths; spot lights will be considered a special case of a point light. In both cases, Variance Shadow Maps [6] will be used to get soft shadows. Should light bleeding become a severe problem, we will look into Layered Variance Shadow Maps [7].

Shadow maping is a must for the final version. Shadows add a significant amount of visual quality to the demo, and affect the overall result to great extents.

2.3. High Dynamic Range

The rendering will make use of HDR effects, more specifically, bloom and tone mapping. For blooming, a texture pyramid similar to a mipmap chain is first produced (using a brightness filter zeroing out the parts that shall not exhibit the bloom effect). Then the individual textures are blurred with a gaussian kernel. The result is additively blended to the framebuffer. This approach is explained by [8].

We will use a very simple exponential tone mapping operator, as shown by [9].

HDR is partially implemented already in the first version of the demo. The deferred renderer stores the lighting results in a R11FG11FB10F buffer. This gives us a dynamic range which we consider big enough for our purposes. Tone mapping still needs adjustments for the final version.

2.4. Boids

The protagonists in the scene will be a swarm, controlled by the Boids algorithm [10]. We will make use of geometry instancing in case the impact of amount of draw calls caused by the swarm is too severe. The swarm will follow a path, implemented as a piecewise cubic bezier spline.

This effect is a must for the second version.

2.5. Heat/Haze effects

To visualize heat, haze, and distortions in general, we take a screen-space approach. It follows the explanations given by [11]. Distortions will occur in screen-space, using XY offsets stored in a distortion map. Texture fetches will then be affected by these distortions. This approach fits naturally with deferred rendering, which already uses a screen-space pass.

This effect is nice to have for the second version, but we will sacrifice it if we run out of time.

2.6. Texture animation

Animating the boids may be too time consuming for this course, so we will try to use texture animations instead as much as possible. Textures will simply be swapped, in a loop. For example, to simulate wings flapping, we will try to model the wings in all their stages; if there are two wings, and we want to show three animation stages, we model 6 wings, give the first pair texture set #1, the second pair gets set #2, and the third pair gets set #3. In the next animation step, the first pair gets set #3, the second one set #1, the third one set #2. If the objects are small and plentiful enough, we believe the result will we visually acceptable.

This effect is also nice to have, and will be sacrificed if necessary.

2.7. Explosions

For the explosions, we will use particles systems. We take advantage of the capabilities of today's GPU generation, and animate the particles fully within the GPU [12].

This effect is nice to have, and we will consider simpler substitutes if necessary.

3. Sources

All web pages were retrieved on 24.11.2011.

[1] Trebilco, D. *Light Indexed Deferred Rendering.* Retrieved from http://code.google.com/p/lightindexed-deferredrender/

[2] *Deferred Lighting approaches.* Retrieved from http://www.realtimerendering.com/blog/deferred-lighting-approaches/

[3] Pranckevičius, A. *Compact Normal Storage for small G-Buffers.* Retrieved from http://aras-p.info/texts/CompactNormalStorage.html

[4] *Multiple Materials.* Retrieved from http://www.catalinzima.com/tutorials/deferred-rendering-in-xna/multiple-materials/

[5] Engel, Woflgang F. Section 4. *Cascaded Shadow Maps.* ShaderX5 , Advanced Rendering Techniques, Wolfgang F. Engel, Ed. Charles River Media, Boston, Massachusetts. 2006. pp. 197–206.

[6] Lauritzen, A. *Variance Shadow Maps.* Retrieved from http://www.punkuser.net/vsm/

[7] Lauritzen, A. and McCool, M. 2008. *Layered variance shadow maps.* In Proceedings of graphics interface 2008 (GI '08). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 139-146.

[8] *How to do good bloom for HDR rendering.* Retrieved from http://kalogirou.net/2006/05/20/how-to-do-good-bloom-for-hdr-rendering/

[9] *MegaPOV manual.* Retrieved from http://megapov.inetart.net/manual-1.2/global_settings.html

[10] Reynolds, C. 1987. *Flocks, herds and schools: A distributed behavioral model.* SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques (Association for Computing Machinery): 25–34, doi:10.1145/37401.37406, ISBN 0-89791-227-6

[11] Oat, C. *Real-Time 3D Scene Postprocessing.* ATI Research. Retrieved from http://developer.amd.com/media/gpu_assets/Oat-ScenePostprocessing.pdf

[12] Latta, L. *Building a Million Particle System.* Game Developers Conference 2004. Retrieved from http://www.2ld.de/gdc2004/