

# Demoname: Brechpunkt

Gruppe 10

Thomas Koch - 01526232

Felix Kugler - 01526144

**Tested on AMAROK, NVIDIA GTX 1060.**

## Libraries

GLFW [1], GLEW [2], GLM [3] are used. Most of the geometry is loaded from obj files via tinyobjloader [4]. stb\_image [5] is used to read textures i.e. the normal texture of the ground plane. BASS [6] is used for music playback and for performing a FFT which is used to animate the colored cubes.

## Effects

### SSDO

Point lights placed at each cube surrounding the scene illuminate the scene. A simple phong shading model is used for this. Additionally, an environment map with all emitting meshes is rendered and blurred. Screen-Space Directional Occlusion [7], which samples the environment map, has been implemented. The depth buffer is scaled down to a fourth of its size on both dimension before being sampled to increase memory throughput. Sample selection uses a dithering pattern created with Halton sequences. By sampling the environment map during AO calculation, emitting meshes cause correct soft-shadows. This is especially visible between the eight blocks in the center of the scene. Afterwards, a selective gaussian blur is performed on the calculated lighting to remove noise.

### Depth of Field

The depth of field effect is separated into a horizontal and a vertical pass, and is thereby able to compute large radii efficiently. This is based on the approach by McIntosh et al. [8]. By using a signed circle of confusion, which has negative radii for objects in front of the focus, these objects correctly bleed into the focus.

### Normal Mapping

Normal textures have been created using GIMP and Blender. Textures from textures.com have been used. The values of the texture is transformed into view space using the surface normal, tangent and bitangent vectors [9]. Normal maps are used for the cracks on the floor tiles.

## Screen-Space Reflections

Screen-space reflections have been implemented similar to [10]. For each pixel a reflection vector is ray-marched using a fixed step size. The ray is tested for intersection with the depth buffer. Due to the fixed step size, small detail can be missed. Therefore, a binary search is executed in order to get a more accurate intersection point.

## Refractive (Glass) Objects

Refractive objects have been realized according to [11, 12]. In order to refract a ray through an object, the exiting position of the ray has to be approximated. To do this, depth peeling (culling the front face) is used. Additionally, the point on the mesh directly behind the face along its normal, the intersection point of a ray along the negative normal, is pre-calculated. The models are created using Blender and exported as VBO using an addon written in Python. When a model is exported using our addon, the pre-calculation takes place.

## Bloom

The bloom effect uses a simple separated gaussian blur. ~64 pixel are sampled horizontally and vertically. It is applied at half the resolution of the rendered scene. [13]

## GPU Particles

A compute shader performs very simple Newtonian physic simulation. Instanced rendering is used to draw them. Particles are spawned with an initial force and air resistance is simulated in the compute shader until the particles come to a stop. A ring buffer is used such that old particles are overwritten when new ones are added.

## Cube Mapping

Part of the scene is rendered into a cube map before the main forward pass. A geometry shader is used to copy each primitive into the 6 different sides of the cubemap to render all sides at once. [14]

## Anti-Aliasing

For performance and simplicity all shading, including ambient occlusion and reflections, is performed without anti-aliasing. A texture holding primitive ids is rendered with the main forward pass. After all shading and DOF has been performed a second forward pass mixes the non-anti-aliased color according to the primitive ids in a multisampled pass. Unlike FXAA this results in correctly filtered edges. As the geometry of the scene is simple this step is cheap. By performing this step after DOF, the silhouette of objects in focus is correctly filtered as well.

# Tools

Blender was used to create and position the cube objects and the floor tiles, as well as to edit and export the Stanford models. An addon for Blender has been developed in Python to export glass objects directly in the format we use in VBOs, as these need pre-calculated vertex attributes (the opposite position, see above).

# Models

The Stanford Dragon [15] and the Stanford Lucy [16] model have been used. All other models (the colorful cubes, the four center cubes and the ground plane) have been modeled in Blender.

# Controls / Arguments

Without any arguments, the demo starts in windowed mode with resolution 1920x1080.

- F1 toggles music playback.
- F2 toggles the free camera, which can be moved with the mouse and WASD keys.
- The left and right arrow keys can be used to rewind and fast-forward the animation respectively.
- `--fullscreen` to start the demo in fullscreen
- `--WIDTHxHEIGHT` to start the demo in windowed mode with specified resolution e.g. `--1920x1080`.

# References

[1] <https://www.glfw.org/>

[2] <http://glew.sourceforge.net/>

[3] <https://glm.g-truc.net/>

[4] <https://github.com/syoyo/tinyobjloader>

[5] [https://github.com/nothings/stb/blob/master/stb\\_image.h](https://github.com/nothings/stb/blob/master/stb_image.h)

[6] <http://www.un4seen.com/>

[7] Ritschel, Tobias, Thorsten Grosch, and Hans-Peter Seidel. "Approximating dynamic global illumination in image space." *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM, 2009.

[8] McIntosh L., Bernhard E. Riecke, and Steve DiPaola. "Efficiently Simulating the Bokeh of Polygonal Apertures in a Post-Process Depth of Field Shader." *Computer Graphics Forum*. Vol. 31. No. 6. Oxford, UK: Blackwell Publishing Ltd, 2012.

[9] Lecture slides 2018, Shading, pages 33-38

[10]

<https://sakibsaikia.github.io/graphics/2016/12/26/Screen-Space-Reflection-in-Killing-Floor-2.html>

- [11] Wyman, Chris. "An approximate image-space approach for interactive refraction." ACM transactions on graphics (TOG) 24.3 (2005): 1050-1053.
- [12] Wyman, Chris. "Interactive image-space refraction of nearby geometry." Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. ACM, 2005.
- [13] Lecture slides 2018, Screenspace effects, pages 7-11
- [14] Lecture slides 2018, Shading, pages 8-13
- [15] <http://www.mrbluesummers.com/3572/downloads/stanford-dragon-model>
- [16] <https://www.thingiverse.com/thing:41939>