Kitty Panic
Submission 2

1 Controls

W,A,S,D	Move Kitty forward, to the left, backward, to the right
Mouse	Rotate the camera
F1	Toggle Debug Informations (FPS, Culling Info, ...)
F2	Toggle Framerate
F3	Toggle Wireframe
F4	Toggle Texture-Sampling Quality
F5	Toggle Mipmap Quality
F6	Toggle Shadows
F8	Toggle View Frustrum Culling
F9	Toggle Transparency
Space	Jump

In the menu besides the W and S key, the UP and DOWN arrow keys might also be used to change selection. Also the RETURN key can be used besides the Space key to select/activate the current menu entry.

2 Settings

Resolution can be changed in the data/resolution.txt file.

3 Integrated Effects

3.1 Bone Animation

The characters (kitty and the enemies) are animated by bone animation. They file format is collada and assimp is used to load the file format. The vertices are multiplied with the bone matrix on the GPU. The bone matrices are computed on the CPU.

3.2 Toon Shading

The shading is quantized using a 1D texture to get a toon like shading effect. The black outlines are created using a sobel filter as post-processing step. The

filter is executed on the depth texture of the scene. This creates black edges especially on "Tiefenkanten". When using the color texture of the scene in our opinion too many edges will be colored black (just try to play around with the F-Keys).

We've also tried rendering the backfaces in wireframe mode (actually, this was our first version of the effect), but we like the sobel version on the depth buffer better.

3.3 Particles

Particles are rendered on the GPU using a geometry shader, but calculated on the CPU. Particle effects can be seen at the torch which is placed in some levels and when the kitty dies.

3.4 Shadow Mapping

We've followed the slides from shadow mapping and used Poisson Sampling (c.f. `http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/#header-13`) for shadow mapping. For performance reasons on our relatively weak GPUs we've disabled shadows per default. But on the presentation computer we've managed to get a relatively high FPS with shadows enabled.

4 Comments to Requirements

4.1 Gameplay

Gameplay is implemented.

You can get killed by enemies, collect rupees, win a level, die, enter the highscore and play through all the levels.

4.2 Complex Objects

Kitty and the enemies are non-trivial objects, loaded from collada/obj files. In level 3 there are some trees, which are also rather complex objects.

4.3 Animated Objects

Kitty and the enemies are animated using bone animation.

4.4 View Frustrum Culling

This is done using bounding boxes checkes against the view-frustrum-planes in the world coordinate system. Some heuristics are used when culling static objects (i.e. walls, floors) on the ground.

4.5 Transparency

Rupees and the mini-map on the bottom right part of the screen are transparent. If you press F9 you should see some subtle differences.

5 Additional Libraries

We are using the following libraries:

- assimp: for model loading
- glew: for OpenGL extensions, ...
- glfw: for opening the window, controls, ...
- glm: for vector and matrix math
- fmod: for sound
- tinyxml2: for xml parsing
- micropather: implementation of the A* algorithm

6 Used Tools

- Blender for 3D modelling
- Photoshop for some texture editing
- Visual Studio C++ for development and gDebugger for debugging.