186.831 – Computergraphik – 2016S
Group: Hamstfoo

Brief description

Gameplay: The hamster starts on one end of the street and has to reach the other. On the way, he encounters cats, which he has either to ditch or punch. Three punches are necessary to neutralize a cat. If the hamster is in a certain distance to the cat, the cat approaches and tries to punch the hamster. Three punches in total will kill the hamster and the game is lost. There are also some walls in the way, the hamster has to jump over them to reach his goal. Beer and Popcorn can be collected by running through them. By drinking beer the hamster gains health, Popcorn will increase your game points (which don't have any influence on the game, but collecting stuff is fun ;]).

Effects:

- Light Mapping + Separate Textures
- Spotlights
- Static LOD
- Normal Mapping
- Cel Shading

Complex Objects: We don't have trivial objects. All our meshes are our own creation and even the scene entity is made of one mesh, so it is pretty complex as well.

Animated Objects: Hamster and Cat have animations. They are able to run and punch.

View-Frustum-Culling: View Frustum Culling was implemented with the help of this tutorial: http://www.lighthouse3d.com/tutorials/view-frustum-culling/. The View Frustum Culling happens in FrustumG.cpp.

Transparency: Our Collectables (Popcorn and Beer) are rendered transparent. Their alpha channel is fixed to 0.5 for each pixel.

Experimenting with OpenGL: For all info about the key functionality press F1 in-game.

"Features" of the game

- Handmade (from scratch) 3D models
- Animals as main characters
- Beer and Popcorn
- Great Sounds
- Super-intuitive combat system (requires only one mouse button ;])

Illumination and Textures

All our entities are textured and static entities use a second texture (light_texture) to show lighting (➔ light mapping). Dynamic entities are first blinn-phong shaded and the result is rendered to a lower amount of possible colors (➔ cel shading). Collectables like Beer and Popcorn are shaded emissive (so if they are placed in an area without light, the get drawn with their non shaded texture color).

Additional libraries

Assimp – http://assimp.sourceforge.net/

DevIL – http://openil.sourceforge.net/

freeglut - http://freeglut.sourceforge.net/

glew – http://glew.sourceforge.net/

gflw – http://www.glfw.org/

Bullet – http://bulletphysics.org/wordpress/

glm - http://glm.g-truc.net/0.9.7/index.html

freetype - https://www.freetype.org/

Effects:

- Light Mapping + Separate Textures
  All static entities use a separate light texture to render their lighting. The textures get combined in the lightMapping.frag shader.
  http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-15-lightmaps/
- Spotlights
  There is a spotlight placed for each street lamp in the scene.
  http://www.tomdalling.com/blog/modern-opengl/08-even-more-lighting-directional-lights-spotlights-multiple-lights/
- Static LOD
  Cats are using static LOD, so if they are far away from the camera they are drawn with a low poly mesh. If you approach them, they get drawn more detailed with subdivided surfaces. Also the cat's animation is only used when the cat is in the specified high poly distance.
- Normal Mapping
  Hamster and Cat have their own normal texture which are used for normal mapping in the cel.frag shader. The shader checks if a normal map is present and is using the given normal map to render the entity. The texture vector has to be brought in the right space via a tangent space matrix.
  http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/
- Cel Shading
  Cel shading is used to render the surrounding area. As these areas are static and light mapped our Cel shading is applied to the light-vector given by our light mapping textures.
  http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/toon-shading/

Model Creation

Blender was the only tool used for the creation of all our models. All models are made from scratch.