Puzzle The Kugel

Kamper Raphael 1125579
Perkonigg Matthias 1129269

1. Controls

Key	Effect
W, A, S, D	Control sphere
Arrow Keys, E, Q	Control free moving camera, E (UP), Q(DOWN)
C	Switch cameras
Space	Manipulate Path
M	Music on/off
R	Reset
P	Move enemies on/off
F2	FPS on/off
F3	Wire frame on/off
F4	Texture sampling qulity: Nearest/Linear
F5	Mip Maping Quality: Off/Nearest/Linear
F8	Viewfrustum-Culling on/off
F9	Transparany on/off
ESC	Quit game

The keys ‚A‘ and ‚D‘ are rotating the sphere. ‚W’ applies force to the sphere in the current looking direction (‚S’ in opposite direction).

2. Gameplay

The goal of the game is to reach the finish line. As the sphere can’t jump you have to collect the red items. Those give you the power to invert the path so you the sphere can fall down onto another platform. If the sphere collides with a spider or you fall off the platforms and hit the ground the sphere is set back to start and you lose a life. If you have lost all your lives the game is lost. If you realize that you cannot go further anymore and are trapped you can reset the game by pressing “R” but lose a life.

3. Implemented Effects

1. **Normal Mapping**
   Normal Mapping is applied to all objects in the scene. The tangent and bitangent space is calculated during the loading process of the models [4] and then passed to shader [3]. Might be useful to stop the spider (press “P”) to see the effect on a complex object. The light source is moving along the z-axis.
2. **CPU Particle System (+ Instancing)**
   If you collect an item, it explodes before it’s no longer visible.
3. **Shadow Maps + Variance Shadow Maps instead of PCF**
   Using the hints from the lecture and the tutorials of the “OpenGL SuperBible” [1] and for variance shadow mapping the “OpenGL Development Cookbook” [5]
4. **Projected Textures**
   Almost the same like shadow mapping but with color attachment hat is bound to the FBO. Projected Texture can be seen at the start point and marks the finish. You’ll see it if you roll with the sphere over the places the texture is projected to.

4. Cameramodes

Free moveable camera (controlled by the arrow keys and ‘E’ and ‘Q’ to look up and down).

Third person camera automatically follows the sphere in the “looking direction” of the sphere within a fixed distance.

5. Objects

All objects are loaded via assimp from .obj files (wavefront format). And the textures are loaded via FreeImage. We basically followed the scheme from “learopengl.com”[1] but adapted it (FreeImage, DrawShadowMaps() basicMaterialLoading()), calculate tangent and bitangents[4].

Path is loaded from a .txt file (level.txt) where every entry represents the position of a cube. The cube model is loaded and textured once and then rendered (translated and “randomly” rotated).

Sphere simple sphere with texture.

Item is also a simple transparent sphere.

Enemy is a spider and the body is yet not added as and body to bullet.

Normal mapping is applied to all those objects.

6. Animated Objects

The legs of the spider are hierarchical animated. Therefore random values are generated to time how fast the legs are moving up and down.

7. Textures

All textures are loaded via the FreeImage library and specified in the .mtl file of the objects. For the text rendering we use the freetype2 library. All objects do have a diffuse and a normal map assigned to them in the .mtl file.

8. Lightning

There is a simple directional lightning by a global lightning source (moving along the z-axis). You can “see” this movement by watching the shadows.

9. Viewfrustum-Culling

Not applied to sphere, because the camera always looks at it.

10. Experimenting with OpenGL

FBO: used for shadow mapping and projected textures.
VAO: used for models.
VBO: used for models.
Transparency: the items are transparent.
Texture Sampling: control with F4.
MipMapping Quality: control with F5.

11. Additional Libraries

Assimp is used for model loading.
URL: http://assimp.sourceforge.net

Bullet for physic simulation and collision detection.
URL: http://bulletphysics.org/wordpress

Freeimage to load our textures.
URL: http://freeimage.sourceforge.net

Irrklang for sound.
URL: http://www.ambiera.com/irrklang/index.html

Freetype for text rendering.
URL: http://www.freetype.org

12. References

[1] http://www.learnopengl.com/#!Model-Loading/Model

[2] Richard S. Wright, “OpenGL® SuperBible: Comprehensive Tutorial and Reference, Fifth Edition.” Addison-Wesley Professional, 2010.

[3] http://www.learnopengl.com/#!Advanced-Lighting/Normal-Mapping

[4] http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

[5] Muhammad Mobeen Movania, “OpenGL Development Cookbook”, Pack Publishing Ltd., 2013.