Rescue mission – CG-Project report

Martin Peche, 01555052
Stefan Brandl, 01555842

20. June, 2018

1 Scene

In our scene we can see a horde of trolls resting around a campfire. Soon after they are detected by a wizard, illuminated in his mage-light. Starteled by this unexpected encounter the biggest troll (Stefan) panics and starts running in circles. The wizard defends itself by casting a ball of light towards the horde of trolls. Soon the light fades. A treasure hunter notices the distraction and tries his luck at the trolls chest. He succeeds and finds a heap of gold. Afterwards we leave this world full of magic, treasure hunters and trolls as the camera pans out into the distance.

2 Custom scene graph nodes

For our project, we implemented a bunch of custom scene graph nodes. The source of all of them is placed in the directory code/nodes/ .

Skybox node

For the sky in the background, a separate SkyboxSGNode was introduced. The node works by rendering a 2x2x2 cube around the camera, ranging from -1 to 1 in each direction x, y, z. While rendering, all depth checks are disabled, effectively making the skybox draw over anything rendered before

it on the screen. This is the reason why the cube does not need to bee bigger than the whole scene. Additionally, depth writing is disabled, meaning that any object rendered will overdraw the skybox, making it appear in the background every time.

Particle system

The particle system is implemented as a node on its own, enabling it to be transformed, rotated and scaled as any other object. More details on the implementation and the technical details particle system are in chapter 4.1.

Texture node

For applying textures, we took the TextureSGNode from the labs, and adapted. Each texture node increments a variable in the context determining the texture unit which should be used. If no one is set in the shader, texture unit 0 is used by default ([1])

Animation node

For realizing the animations, a special AnimationSGNode was introduced. It is basically a TransformationSGNode, but instead of using a static transformation matrix, it is supplied a function, which generates a arbitrary transformation matrix based on the current animation time [2].

Animation trigger node

A AnimationTriggerSGNode enables the definition of such-called "trigger point". A trigger point starts an animation as soon as the camera enters a defined radius around the origin of the trigger.

As long as the animation is not triggered yet, all children of this node receive a time of 0 through the context, effectively halting the animation of all

[1] provide code sample?
[2] Satz kürzen

AnimationSGNode s and other time using nodes. When the AnimationTriggerSGNode is triggered, the current time point is stored.

A AnimationTriggerSGNode can only be triggered once and can not be reset for now.

Spotlight node

The spotlight node is used to define a spotlight in the world. It inherits from the LightSGNode and provides two additional uniforms:

angle Specifies the angle of the cone in which the light illuminates

direction Defines the direction of the Spotlight. This is a direction vector

3 Requirements

This section will describe shortly *where* and *how* we implemented the requirements given by the assignment.

3.1 Scene Graph elements

Custom Model

For manually composed models we have

- Chest (which consists of multiple complex parts)
- Trolls
- Wizard
- Treasure Hunter

Separate Animations

- Running Troll with moving legs
- Wizard swinging his staff
- Treasure Hunter moving his hands separately

Render complex model

See section 3.7.

3.2 Materials

The wizard, moving lightball, gold (in the chest), trees and the whole fireplace (excluding the particle system) are materials.

3.3 Texturing

See section 3.7.

3.4 Illumination

Multiple light sources

Fire (static), wizard light (static), projectile from wizard (moving and spotlight).

Spotlight

See section 3.4.

Phong-lighting

Phong shading is applied to all objects in the scene with the exception of the particle system since it is transparent.

3.5 Transparency

The particle system uses semi-transparent textures.

3.6 Camera

E for upwards movement

Q for downwards movement

W forward

A left

S backward

D right

X to place an additional tree under the camera

3.7 Custom model

We had to define one model all by our self, and we chose to do this on the chest in the forest. Our goal was to provide a model which is separated into tow parts: the lower half of the chest and the upper part. This allows us to animate the hinge of the chest and let it be opened by a guy finding the treasure.

Figure 1 shows the chest as a whole model. The bottom part was mostly just a cube, but the upper half (the rounding) was rather complex. It is defined by around 40 vertices.

Figure 2 shows the complex part from the side. You can clearly see the rounded object.

Figure 1: The chest

3.8 Alpha texturing

We added the alpha texture as well alspha belding to our particle system. More details on the particle system and *alpha blending* are described in chapter 4.1.

4 Special effects

We combined both special effects into one *thing* (it was told form the lectures that this would be okay).

4.1 Particle System

Additional to the separate BillboardSGNode described in section 4.2 we implemented billboarding in the particle system. In figure 3 you can see our particle system used in the scene. We decided to set it up as a fire, but any texture would work.

The particle system uses basic quads to render the single fire particles. Each quad has an *starttime* and a *direction*. Using the starttime, the particle can determine its age and set itself to an appropiate position using the direction and the origin of the particle system itself.

Figure 2: The side view of the chest

The particle system takes 4 arguments:

- The texture which it should be applied to the quads (fire in our case)
- The frequency of particles, determining how many updates will pass between single spawns of particles
- The amount of particles spawned at once
- And the maximum amount of particles ever in the system. This is to prevent the infinite creation of particles and therefore crash the program

On every update, it is checked whether new particles should be spawned. If so, the oldest particles are "killed" and replaced with new ones with a random direction vector.

We tried to use a particle render method which is based on GL_POINTS, which worked for particle systems not close to the camera. But the size of the point was supplied in pixels, and also we used a approximation $\frac{1}{d}$, where

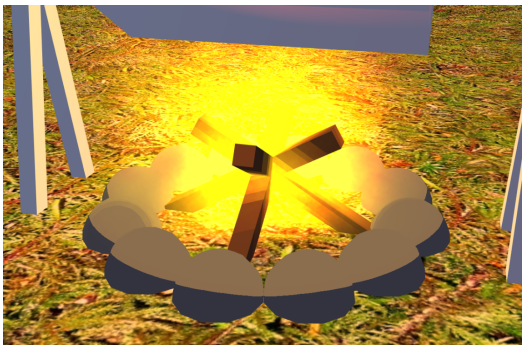

Figure 3: The fire particle system in action

d was the distance from the camera to the particle, which resulted in wrong approximations when coming close like in our scene.

4.2 Billboarding

The BillboardSGNode uses the inverse matrix of the view Matrix to cancel out the rotation. To leave the object at its world position, the translation parts of the new matrix are set to zero.