EZG18-Museum

Implementation

In our demo the camera flies through a museum room filled with antique artefacts. The only light source is a glowing ball of particles that is chased by the camera. As the ball comes near a wall a portal opens to another wall of the museum and the camera may fly through this portal. The sides of the portals are enhanced with another particle system. As the camera follows the glowing ball you can see parallax mapping on the walls and the floor of the museum. When the animation reaches its end, it automatically starts from the beginning again.

All the animations, including the camera, have been done in Blender and are imported with the assimp library using the collada file format. For the smooth camera movement Catmull- Rom splines are used for keyframe interpolation in our engine.

Libraries

For the object loading we are using assimp (http://www.assimp.org/).

For loading textures, we use FreeImage (http://freeimage.sourceforge.net/).

Since we use our own physics, we do not have a library for that.

Effects

We implemented parallax mapping, particle systems, skeletal animations and portals.

Parallax Mapping

Parallax Mapping was implemented according to the tutorial by LearnOpenGL:
https://learnopengl.com/Advanced-Lighting/Parallax-Mapping
The first step in parallax mapping is the calculation of the tangent space. For this we calculate the tangent vector for each vertex of our models once right after they were loaded. Since the bitangent vector is just the dot product between the vertex normal and the tangent vector, we calculate it in the vertex shader, so we don’t have to send it to the graphics card all the time.
In the fragment shader, we transform the necessary vectors to tangent space and calculate the texture offset according to the height map.

Particle System

We used the slides from the lecture as a basis to build our particle system. LectureSlides: Particle Systems with Compute & Geometry Shader. In addition we added controls for emitters and attractors in the compute shader to get more control and to make more complicated effects possible. For the portal particle system, the particles are emitted on the border line of an ellipse defining the portal border and are attracted to the center of the portal. They have a short lifetime so that they don’t make it all the way to the center of the portal. The glow ball particle system only has one attractor in the center and no emitters. The particles live forever and are only spawned once in the beginning. The initial spawning position of each particle is chosen based on three random samples of a Gaussian distribution using one sample for each dimension.

Skeletal Animation

We have implemented linear vertex skinning on the GPU as described by Kavan et. al https://dl.acm.org/citation.cfm?id=1230107. We have rigged the Buddha statue and made a simple skeletal animation for it in Blender. The skeleton and animation data is loaded with assimp. The final transformation for each bone is computed for every frame on the CPU recursively along the skeletal hierarchy. The transformations are then passed to the vertex shader via uniform matrices for rendering.

Portals

The portals have already been implemented before starting this course and no external sources (except from being inspired by playing the game Portal) have been used for this purpose. In a nutshell, the scene is rendered multiple times, once for each portal depth level. For this purpose, the camera's view matrix is computed for each depth level by transforming it according to how the portals are connected with each other. Then, the scene is rendered from back to front using the previously computed view matrices. This requires an additional depth map, as no geometry nearer than the portal of the current depth level should be rendered. To make the rendering more efficient, frustum culling has been implemented. For portal rendering, the frustum for each depth level is only a small fraction of the original view frustum and it is adjusted so that it tightly matches the portal from the previous depth level to avoid drawing unnecessary geometry.

Tools

In our scene, the room (walls, floor, ceiling) and the outside (gras, pebbles) have been created by us using Blender. We used textures from https://www.wildtextures.com/. All the museum artefacts are models we got from the following websites: https://free3d.com/ https://www.turbosquid.com/ https://www.cgtrader.com/free-3d-models. For parallax mapping we needed normal and height maps. These were generated using the online tool provided by http://cpetry.github.io/NormalMap-Online/ and adjusted by using Gimp https://www.gimp.org/.

Controls

1 – switch rendering of portals on/ off

2 – switch between parallax mapping and Blinn-Phong shader

Right Arrow – fast forward animation 1,5 seconds

Left Arrow – rewind animation 1,5 seconds

P – stop/ start camera animation

C – switch between animation/ debug camera

WASD + Mouse – movement of debug camera in the usual first person style

There is an additional configuration file (config.cfg) where the screen resolution, brightness and gamma correction can be adjusted.

Graphics card

We tested our project on NVIDIA.