Berne University of Applied Sciences (BFH)

Department Engineering and information technology

BZG1310 - Object-oriented geometry

Documentation Project 1

Subject: Spinning cube with spheres on its edges –
Classical approach using OpenGL and GLSL

Student: Sven Osterwalder (ostes2@bfh.ch)

Date: December 16, 2013

Professor: M. Stampfli

1 Management summary

This dissertation is part of a project in the context of the subject *BZG1310: Object-oriented geometry* at Bern University of Applied Sciences.

One goal of this project is the implementation of a rotating cube which has a sphere on each edge using classical approaches within the field of computer graphics (e.g. matrices or quaternions).

The goal could be reached and the task was implemented using C++, OpenGL and GLSL and matrices.

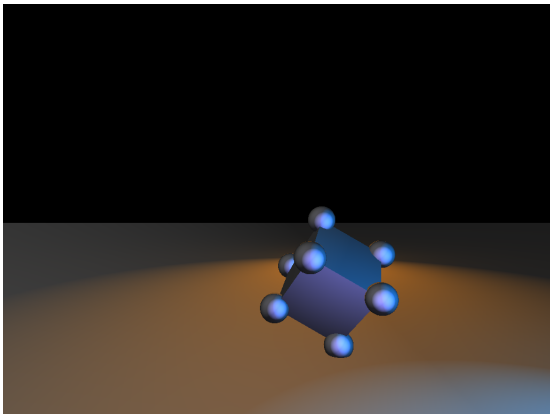

Figure 1: Screenshot of the running application

2 Implementation

The implementation was done using C++, OpenGL and the shader language GLSL.

2.1 Framework

To cover the main tasks such as providing minimal application logic, loading files (models, textures and shaders) or handling input, a minimal framework based upon GLEW, ASSIMP and GLFW was built.

The framework consists the following aspects:

- Main routine (initializing, running and finishing the application)
- Camera handling
- Helper class for mathematical functions (consisting vectors and matrices)
- Representation of light through shaders based on GLSL
- Model loading using ASSIMP library
- A pipeline for performing transformations
- Management of textures using ImageMagick
- Common utilities (e.g. for loading files into memory)

2.2 Main process

In general the following is done when executing the application:

- Initialization of the application
- Execution of the application until a quit event is received
- Termination of the application

2.3 Geometric operations

As the geometric operations are the main aspect of the course BZG1310, this section shows how this aspect was realized within the application.
To perform the needed translations, rotations and scaling of the objects, matrix multiplication was used.

As basis for the representation, perspective projection was used in form of matrices, which represent the world space, as well as a camera.

The positions as well as the rotations are represented resp. implemented using three dimensional vectors, the scaling is implemented using four dimensional matrices.

The procedure for calculating the positions and the rotation of objects is as follows:

- Setting the world position according to the current object:

$$worldTranslationtionMatrix :=$$

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & object.X \\ 0.0 & 1.0 & 0.0 & object.Y \\ 0.0 & 0.0 & 1.0 & object.Z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

- Setting the world rotation according to the current object:
  $worldRotationMatrix := rotationZ * rotationY * rotationX$

  $rotationX :=$

  $$\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & cos(object.X) & -sin(object.X) & 0.0 \\ 0.0 & sin(object.X) & cos(object.X) & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

  $rotationY :=$

  $$\begin{bmatrix} cos(object.Y) & 0.0 & -sin(object.Y) & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ sin(object.Y) & 0.0 & cos(object.Y) & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

  $rotationZ :=$

  $$\begin{bmatrix} cos(object.Z) & -sin(object.Z) & 0.0 & 0.0 \\ sin(object.Z) & cos(object.Z) & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

- Getting the world transformation:
  $worldTransformation := worldTranslationMatrix * worldRotationMatrix$

- Set the world matrix:
  $worldMatrix := worldTransformation * scalingOfTheCurrentObject$

- Repeat the steps above for the world perspective

As one can see, with the routine above the objects get first rotated (locally) and then translated to a specific point in space which concerns only the cube. Additionally the same procedure gets done for objects rotating around a so called pivot - a relative point in space. This is needed to keep the spheres performing the same rotation as the cube in addition to their own local rotation.

The only difference is hereby that the world transformation needs to be done respecting the provided point in form of a four dimensional matrix and by changing the order of the multiplication, which results in the following:

$$
\begin{aligned}
&worldTransformation := \\
&\quad providedPointMatrix * \\
&\quad worldRotationMatrix * \\
&\quad worldTranslationMatrix
\end{aligned}
$$

3 Conclusion

The project was a bit effortful but also great fun to do. I could sure have chosen a less effortful way to realize the project but I chose this way as the subject area is one of my main interests and I was concerning myself before with this topic. It especially helped me to get a deeper understanding about how to implement the theory into a practical application.

4 Glossary

ASSIMP stands for Open Asset Import Library and provides the possibility to load various well-known 3D model formats in a uniform manner. 2

GLEW stands for OpenGL Extension Wrangler and is a cross-platform C/C++ library that helps in querying and loading OpenGL extensions. 2

GLFW stands most probably for Graphics Library Framework and is a lightweight utility library for use with OpenGL. 2

GLSL is the short form for OpenGL Shading Language and is a programming language which allows to execute programs on the graphics processing unit. 2

ImageMagick is an open source software suite for displaying, converting, and editing raster image files. 3

OpenGL stands for Open Graphics Library and is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D computer graphics. 2