Belegarbeit - OpenGL

Tan Minh Ho
s82053

Prof. Dr.-Ing. habil. Wolfgang Oertel
Computergrafik I

January 12, 2022

Contents

1 Aufgabenstellung

Schreiben Sie ein Programm in C/C++, das unter Verwendung von OpenGL, Vertex- und Fragment-Shadern folgende Aufgaben realisiert.

1.1 Aufgabe 1

Geometrische Objeckte: Erzeugen Sie eine interaktive zeitlich animierte Szenze mit mehreren unterschiedlichen farblichen und texturierten dreidimensionalen geometrischen Objekten.

1.2 Aufgabe 2

Beleuchtung: Beleuchten Sie die Szenze mit verschiedenartigen Lichtquellen so, dass auf den Objekten unterschiedliche Beleuchtungseffekte sichtbar werden.

1.3 Aufgabe 3

Ansicht: Stellen Sie die Szenze gleichzeitig in verschiedenen Ansichten und Projektionen in merheren Viewports des Anzeigefensters dar.

1.4 Aufgabe 4

Programm: Stellen Sie das komplette Programm in Quelltextform als Visual-Studio-C/C++-Projekt und in ausführbarer Form als exe-File derart bereits, dass die Lauffähigkeit unter MS Windows gewährleistet ist.

1.5 Aufgabe 5

Dokumentation: Fertigen Sie eine Systemdokumentation in Form eines pdf-Dokumentes von etwa 10 Seiten an, die Deckblatt, Gliederung, Aufgabenbeschreibung, Lösungsansatz, Lösungsumsetzung, Installations- und Bedienugsanleitung, einige Bildschirm-Snapshots, Probleme, Literatur- und Quellenverzeichis enthält.

1.6 Aufgabe 6

Abgabe: Übergeben Sie die Ergebnisse der Aufgaben 4 und 5 zusammengefasst in einem Verzeichnis "$Name\_Vorname\_Bibliotheksnummer$" an den Lehrenden. Bei Bedarf kann sich eine Abnahme der Belegarbeit mit Demonstration der Lauffähigkeit erforderlich machen.

2 Installationsanleitung

Um die Quellcode zu compilieren, müssen OpenGL-Bibliotheken installiert werden. Sie sind

- GL
- GLU
- GLUT oder FreeGlut
- GLM oder FreeImage
- GLEW oder Glee

In Linux / MacOS kann man einfach **package manager** [1] verwenden. Nach der Installation lauft das Kommando **make**. Außerdem gibt es andere Kommandos: **make clean** und **make remove**.
In MS Windows verwendet das Programm Visual Studio (oder eine andere IDE) zu compilieren. Und alle oberen Bibliotheken müssen vorher gebunden sein. Beachten: Das Kommando **make** funktioniert in MS Window nicht.

Die Texturen und alle Quellcode sowie Headerfile und Shaders Dateien müssen in dem Verzeichnis "src" gespeichert werden. Headerfile findet unter "src/libs", Quellcode unter "src", Texturen "src/Texture" und Shaders "src/Shader".

3 Lösungsumsetzung

3.1 Headerfile

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <glm.hpp>
#include <gtx/transform.hpp>
#include <glew.h>
#include <freeglut.h>
#include <FreeImage.h>
#include <gtc/matrix_transform.hpp>
#define GLM_ENABLE_EXPERIMENTAL

enum VAO_IDs {Cube, Pyramid, NumVAOs};
enum Attrib_IDs {vPosition, vColor, vTexture, vNormal};

extern GLuint VAOs[NumVAOs];
extern GLuint VBO, EBO, Texture[4];

void generateCube();
void drawCube();
void generatePyramid();
void drawPyramid();
```

3.2 Cube

```
#include "libs/includes.h"

GLuint VAOs[NumVAOs];
GLuint VBO, EBO, Texture[4];

void generateCube(){
    glGenVertexArrays(NumVAOs,VAOs);
    glBindVertexArray(VAOs[Cube]);

    GLfloat CubeVertices[] = {0.7,-0.6,0.0,     1.0,1.0,     0.0,-1.0,0.0,
                              0.0,-0.6,0.0,     0.0,1.0,     0.0,-1.0,0.0,
                              0.0,-0.6,0.5,     1.0,0.0,     0.0,-1.0,0.0,
                              0.7,-0.6,0.5,     0.0,0.0,     0.0,-1.0,0.0,


                              0.0,-0.1,0.0,     1.0,0.0,     0.0,1.0,0.0,
                              0.7,-0.1,0.0,     0.0,0.0,     0.0,1.0,0.0,
                              0.7,-0.1,0.5,     1.0,1.0,     0.0,1.0,0.0,
                              0.0,-0.1,0.5,     0.0,1.0,     0.0,1.0,0.0,


                              0.0,-0.1,0.5,     0.0,1.0,     0.0,0.0,1.0,
                              0.7,-0.1,0.5,     1.0,1.0,     0.0,0.0,1.0,
                              0.7,-0.6,0.5,     1.0,0.0,     0.0,0.0,1.0,
                              0.0,-0.6,0.5,     0.0,0.0,     0.0,0.0,1.0,


                              0.7,-0.1,0.0,     1.0,0.0,     0.0,0.0,-1.0,
                              0.0,-0.1,0.0,     0.0,0.0,     0.0,0.0,-1.0,
```

```
                                0.0,-0.6,0.0,       0.0,1.0,        0.0,0.0,-1.0,
                                0.7,-0.6,0.0,       1.0,1.0,        0.0,0.0,-1.0,


                                0.7,-0.1,0.5,       1.0,1.0,        1.0,0.0,0.0,
                                0.7,-0.1,0.0,       0.0,0.0,        1.0,0.0,0.0,
                                0.7,-0.6,0.0,       0.0,1.0,        1.0,0.0,0.0,
                                0.7,-0.6,0.5,       1.0,0.0,        1.0,0.0,0.0,

                                0.0,-0.1,0.0,       0.0,1.0,        -1.0,0.0,0.0,
                                0.0,-0.1,0.5,       1.0,0.0,        -1.0,0.0,0.0,
                                0.0,-0.6,0.5,       1.0,1.0,        -1.0,0.0,0.0,
                                0.0,-0.6,0.0,       0.0,0.0         -1.0,0.0,0.0};

    GLushort CubeIndices[] = {0,1,2,3,
                              4,5,6,7,
                              8,9,10,11,
                              12,13,14,15,
                              16,17,18,19,
                              20,21,22,23};

    glGenTextures(1,Texture);
    glBindTexture(GL_TEXTURE_2D,Texture[1]);

    FreeImage_Initialise(TRUE);
    FIBITMAP *bitmapData;
    int imgH, imgW;
    BYTE *bitmapBits;
    FREE_IMAGE_FORMAT bitmapFormat=FIF_UNKNOWN;

    bitmapFormat    =FreeImage_GetFileType("src/Texture/2.jpeg");
    bitmapData      =FreeImage_Load(bitmapFormat,"src/Texture/2.jpeg");
    imgH            =FreeImage_GetHeight(bitmapData);
    imgW            =FreeImage_GetWidth(bitmapData);
    bitmapBits      =FreeImage_GetBits(bitmapData);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imgW, imgH, 0, GL_BGR,
    GL_UNSIGNED_BYTE, bitmapBits);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_WRAP_BORDER,GL_REPEAT);
    FreeImage_Unload(bitmapData);
    glBindTexture(GL_TEXTURE_2D,Texture[1]);

    glGenBuffers(1,&VBO);
    glBindBuffer(GL_ARRAY_BUFFER,VBO);
    glBufferData(GL_ARRAY_BUFFER,sizeof(CubeVertices),CubeVertices,
    GL_STATIC_DRAW);
    glGenBuffers(1,&EBO);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,EBO);
    glBufferData(GL_ELEMENT_ARRAY_BUFFER,sizeof(CubeIndices),CubeIndices,
    GL_STATIC_DRAW);

    glEnableVertexAttribArray(vPosition);
    glVertexAttribPointer(vPosition,3,GL_FLOAT,GL_FALSE,8*sizeof(float),0);
    glEnableVertexAttribArray(vTexture);
    glVertexAttribPointer(vTexture,2,GL_FLOAT,GL_FALSE,8*sizeof(float),(void*)
    (3*sizeof(float)));
    glEnableVertexAttribArray(vNormal);
```

```
85      glVertexAttribPointer(vNormal,3,GL_FLOAT,GL_FALSE,8*sizeof(float),(void*)
        (5*sizeof(float)));
86  }
87
88  void drawCube(){
89      generateCube();
90      glDrawElements(GL_QUADS, 24, GL_UNSIGNED_SHORT, 0);
91  }
```

3.3 Pyramid

```
1   #include "libs/includes.h"
2
3   void generatePyramid(){
4       glGenVertexArrays(NumVAOs,VAOs);
5       glBindVertexArray(VAOs[Pyramid]);
6
7       GLfloat PyramidVertices[]={ 0.5,0.0, -0.5,      0.0,0.0,        0.0,
        -1.0, 0.0,
8                                   -0.5,0.0,-0.5,      0.0,1.0,        0.0,
        -1.0, 0.0,
9                                   - 0.5,0.0,0.5,      1.0,1.0,        0.0,
        -1.0, 0.0,
10
11                                  -0.5,0.0,0.5,       0.0,1.0,        0.0,
        -1.0, 0.0,
12                                  0.5,0.0,0.5,        1.0,1.0,        0.0,
        -1.0, 0.0,
13                                  0.5,0.0,-0.5,       1.0,0.0,        0.0,
        -1.0, 0.0,
14
15                                  -0.5,0.0,-0.5,      1.0,0.0,
        0.0,0.5,-0.8,
16                                  0.5,0.0,-0.5,       0.0,0.0,
        0.0,0.5,-0.8,
17                                  0.0,0.8, 0.0,       0.5,1.0,
        0.0,0.5,-0.8,
18
19                                  0.5,0.0,-0.5,       0.0,0.0,
        0.8,0.5,0.0,
20                                  0.5,0.0, 0.5,       1.0,0.0,
        0.8,0.5,0.0,
21                                  0.0,0.8, 0.0,       0.5,1.0,
        0.8,0.5,0.0,
22
23                                  0.5,0.0,0.5,        1.0,0.0,
        0.0,0.5,0.8,
24                                  -0.5,0.0,0.5,       0.0,0.0,
        0.0,0.5,0.8,
25                                  0.0,0.8, 0.0,       0.5,1.0,
        0.0,0.5,0.8,
26
27                                  -0.5,0.0, 0.5,      0.0,0.0,
        -0.8,0.5,0.0,
28                                  -0.5,0.0,-0.5,      1.0,0.0,
        -0.8,0.5,0.0,
29                                  0.0,0.8, 0.0,       0.5,1.0,
        -0.8,0.5,0.0};
30
```

```
    GLushort PyramidIndices[]={0,1,2,
                               3,4,5,
                               6,7,8,
                               9,10,11,
                               12,13,14,
                               15,16,17};
    glGenTextures(1,Texture);
    glBindTexture(GL_TEXTURE_2D,Texture[0]);

    FreeImage_Initialise(TRUE);
    FIBITMAP *bitmapData;
    int imgH, imgW;
    BYTE *bitmapBits;
    FREE_IMAGE_FORMAT bitmapFormat=FIF_UNKNOWN;

    bitmapFormat    =FreeImage_GetFileType("src/Texture/1.jpeg");
    bitmapData      =FreeImage_Load(bitmapFormat,"src/Texture/1.jpeg");
    imgH            =FreeImage_GetHeight(bitmapData);
    imgW            =FreeImage_GetWidth(bitmapData);
    bitmapBits      =FreeImage_GetBits(bitmapData);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imgW, imgH, 0, GL_BGR,
    GL_UNSIGNED_BYTE, bitmapBits);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D,GL_WRAP_BORDER,GL_REPEAT);
    FreeImage_Unload(bitmapData);

    glGenBuffers(1,&VBO);
    glBindBuffer(GL_ARRAY_BUFFER,VBO);
    glBufferData(GL_ARRAY_BUFFER, sizeof(PyramidVertices),PyramidVertices,
    GL_STATIC_DRAW);
    glGenBuffers(1,&EBO);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER,EBO);
    glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(PyramidIndices),PyramidIndices
    ,GL_STATIC_DRAW);

    glEnableVertexAttribArray(vPosition);
    glVertexAttribPointer(vPosition,3,GL_FLOAT,GL_FALSE,8*sizeof(float),0);
    glEnableVertexAttribArray(vTexture);
    glVertexAttribPointer(vTexture,2,GL_FLOAT,GL_FALSE,8*sizeof(float),(void*)
    (3*sizeof(float)));
    glEnableVertexAttribArray(vNormal);
    glVertexAttribPointer(vNormal,3,GL_FLOAT,GL_FALSE,8*sizeof(float),(void*)
    (5*sizeof(float)));
}

void drawPyramid(){
    generatePyramid();
    glDrawElements(GL_TRIANGLES,18,GL_UNSIGNED_SHORT,0);
}
```

3.4 Main

```
#include "libs/includes.h"

using glm::mat4;
using glm::vec3;

```

```
GLint height,width;
GLuint program;
GLfloat depth = 3.0, high =0.0,side, lighpos, rota, angle, radius=10.0;
GLuint loadShaders(const char* vertexFilePath,
                   const char* fragmentFilePath,
                   const char* geometryFilePath,
                   const char* tesscontrolFilePath,
                   const char* tessevaluationFilePath,
                   const char* computeFilePath);

void init(){
    program  = loadShaders("src/Shader/Dreiecke.vs", "src/Shader/Dreiecke.fs",
    "", "", "", "");
    glUseProgram(program);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);
    glFrontFace(GL_CW);
    glCullFace(GL_BACK);
}

void display(){
    glClearColor(1.0,1.0,1.0,1.0);
    glClear(GL_COLOR_BUFFER_BIT  );
    glViewport(0, height/2, width/2, height/2);
    vec3 cameraPos = vec3(side, high, depth);
    vec3 cameraFront = vec3(rota,0.0,-1.0);
    mat4 Translation = glm::translate(mat4(1.0),vec3(0.15,0.25,0.6));
    mat4 Scale = glm::scale(vec3(0.5,0.5,0.5));
    mat4 Model = glm::rotate(Translation*Scale,angle,vec3(1.0,0.0,0.0));
    mat4 View  = glm::lookAt(cameraPos,cameraPos+cameraFront,vec3(0.0,1.0,0.0)
    );
    mat4 Projection = glm::perspective(120.0f,1.0f,0.1f,10.0f);
    mat4 ModelViewProjection = Projection * View * Model;
    GLuint locFinal = glGetUniformLocation(program,"ModelViewProjection");
    glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
    GLuint locModel = glGetUniformLocation(program,"Model");
    glUniformMatrix4fv(locModel,1,GL_FALSE,&Model[0][0]);
    vec3 lightPos = vec3(-1.5,0.0,0.0);
    vec3 lightColor = vec3(1.0,1.0,1.0);
    GLuint loclightPos = glGetUniformLocation(program,"lightPos");
    glUniform3fv(loclightPos,1,&lightPos[0]);
    GLuint loclightColor = glGetUniformLocation(program,"lightColor");
    glUniform3fv(loclightColor,1,&lightColor[0]);
    GLuint loccameraPos = glGetUniformLocation(program,"viewPos");
    glUniform3fv(loccameraPos,1,&cameraPos[0]);
    drawPyramid();
    Translation = glm::translate(mat4(1.0),vec3(0.15,0.15,0.6));
    Model = glm::rotate(Translation,angle,vec3(1.0,1.0,1.0));;
    ModelViewProjection = Projection * View * Model;
    glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
    drawCube();
    glViewport(width/2,height/2,width/2,height/2);
    View  = lookAt(vec3(0.0,0.0,3.0),vec3(0.0,0.0,2.0),vec3(0.0,1.0,0.0));
    Model = Scale;
    ModelViewProjection = Projection * View * Model;
    glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
    drawCube();
    drawPyramid();
    glViewport(0,0,width/2,height/2);
    View  = lookAt(vec3(0.0,3.0,0.0),vec3(0.0,2.0,0.0),vec3(1.0,0.0,0.0));
```

```
64      Model = Scale;
65      ModelViewProjection = Projection * View * Model;
66      glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
67      drawCube();
68      drawPyramid();
69      glViewport(width/2,0,width/2,height/2);
70      View  = lookAt(vec3(3.0,0.0,0.0),vec3(2.0,0.0,0.0),vec3(0.0,1.0,0.0));
71      Model = Scale;
72      ModelViewProjection = Projection * View * Model;
73      glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
74      drawCube();
75      drawPyramid();
76      glutSwapBuffers();
77      angle+=0.1;
78  }
79
80  void reshape(int w, int h){
81      glViewport(0,0,w,h);
82  }
83
84  void timer(int value){
85      glutPostRedisplay();
86      glutTimerFunc(10,timer,0);
87  }
88
89  void keyboard(unsigned char theKey, int mouseX, int mouseY){
90    switch (theKey){
91          case 'w':
92              depth-= 0.25;
93              break;
94          case 's':
95              depth+=0.25;
96              break;
97          case 'a':
98              side+=0.25;
99              break;
100         case 'd':
101             side-=0.25;
102             break;
103         case 'h':
104             high-=0.25;
105             break;
106         case 'l':
107             high+=0.25;
108             break;
109         case 'j':
110             rota-= 0.25;
111             break;
112         case 'k':
113             rota+= 0.25;
114             break;
115         case 'q':
116             exit(0);
117     }
118 }
119
120 int main(int argc, char** argv){
121     glutInit(&argc, argv);
122     glutInitDisplayMode(GLUT_RGBA);
123     height=glutGet(GLUT_SCREEN_HEIGHT);
```

```
124     width=glutGet(GLUT_SCREEN_WIDTH);
125     glutInitWindowSize(width,height);
126     glutInitContextVersion(4,5);
127     glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);
128     glutCreateWindow("NOT EXE");
129     glewExperimental=GL_TRUE;
130     if (glewInit()) printf("Error");
131     init();
132     glutKeyboardFunc(keyboard);
133     timer(0);
134     glutReshapeFunc(reshape);
135     glutDisplayFunc(display);
136     glutMainLoop();
137 }
```

3.5 Limitierungen und Verbesserungsvorschläge

- In tiling-Window-Manger zeigt das Programm nur einen Teil der Szenze. Im Moment gibt es keine Lösung.
- Das Programm ist nicht optimiert und kann unter MS Window wegen Bibliotheken nicht laufen.

4 Literatur und Quellenverzeichnis

- Pamcan -Rosetta : wiki.archlinux.org/title/Pacman/Rosetta [1]
- Prof. Dr. Wolfgang Oertel: Computergrafik I Vorlesungsmaterial
- OpenGL - Archwiki: wiki.archlinux.org/title/OpenGL