

Belegarbeit - OpenGL

Tan Minh Ho
s82053

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

January 12, 2022

Contents

1	Aufgabenstellung	3
1.1	Aufgabe 1	3
1.2	Aufgabe 2	3
1.3	Aufgabe 3	3
1.4	Aufgabe 4	3
1.5	Aufgabe 5	3
1.6	Aufgabe 6	3
2	Installationsanleitung	4
3	Lösungsumsetzung	5
3.1	Headerfile	5
3.2	Cube	5
3.3	Pyramid	7
3.4	Main	8
3.5	Limitierungen und Verbesserungsvorschläge	11
4	Literatur und Quellenverzeichnis	11

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 Objekte: Erzeugen Sie eine interaktive zeitlich animierte Szene mit mehreren unterschiedlichen farblichen und texturierten dreidimensionalen geometrischen Objekten.

1.2 Aufgabe 2

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

1.3 Aufgabe 3

Ansicht: Stellen Sie die Szene gleichzeitig in verschiedenen Ansichten und Projektionen in mehreren 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 Bedienungsanleitung, einige Bildschirm-Snapshots, Probleme, Literatur- und Quellenverzeichnis enthält.

1.6 Aufgabe 6

Abgabe: Übergeben Sie die Ergebnisse der Aufgaben 4 und 5 zusammengefasst in einem Verzeichnis "*NameVornameBibliotheksnnummer*" 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 läuft 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.

3 Lösungsumsetzung

3.1 Headerfile

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <glm.hpp>
6 #include <gtx/transform.hpp>
7 #include <glew.h>
8 #include <freeglut.h>
9 #include <FreeImage.h>
10 #include <gtc/matrix_transform.hpp>
11 #define GLM_ENABLE_EXPERIMENTAL
12
13 enum VAO_IDs {Cube, Pyramid, NumVAOs};
14 enum Attrib_IDs {vPosition, vColor, vTexture, vNormal};
15
16 extern GLuint VAOs[NumVAOs];
17 extern GLuint VBO, EBO, Texture[4];
18
19 void generateCube();
20 void drawCube();
21 void generatePyramid();
22 void drawPyramid();
```

3.2 Cube

```
1 #include "libs/includes.h"
2
3 GLuint VAOs[NumVAOs];
4 GLuint VBO, EBO, Texture[4];
5
6 void generateCube() {
7     glGenVertexArrays(NumVAOs, VAOs);
8     glBindVertexArray(VAOs[Cube]);
9
10     GLfloat CubeVertices[] = {0.7, -0.6, 0.0,      1.0, 1.0,      0.0, -1.0, 0.0,
11                               0.0, -0.6, 0.0,      0.0, 1.0,      0.0, -1.0, 0.0,
12                               0.0, -0.6, 0.5,      1.0, 0.0,      0.0, -1.0, 0.0,
13                               0.7, -0.6, 0.5,      0.0, 0.0,      0.0, -1.0, 0.0,
14
15                               0.0, -0.1, 0.0,      1.0, 0.0,      0.0, 1.0, 0.0,
16                               0.7, -0.1, 0.0,      0.0, 0.0,      0.0, 1.0, 0.0,
17                               0.7, -0.1, 0.5,      1.0, 1.0,      0.0, 1.0, 0.0,
18                               0.0, -0.1, 0.5,      0.0, 1.0,      0.0, 1.0, 0.0,
19
20                               0.0, -0.1, 0.5,      0.0, 1.0,      0.0, 0.0, 1.0,
21                               0.7, -0.1, 0.5,      1.0, 1.0,      0.0, 0.0, 1.0,
22                               0.7, -0.6, 0.5,      1.0, 0.0,      0.0, 0.0, 1.0,
23                               0.0, -0.6, 0.5,      0.0, 0.0,      0.0, 0.0, 1.0,
24
25                               0.7, -0.1, 0.0,      1.0, 0.0,      0.0, 0.0, -1.0,
26                               0.0, -0.1, 0.0,      0.0, 0.0,      0.0, 0.0, -1.0,
27
28                               0.7, -0.1, 0.0,      1.0, 0.0,      0.0, 0.0, -1.0,
29                               0.0, -0.1, 0.0,      0.0, 0.0,      0.0, 0.0, -1.0,
```

```

30         0.0, -0.6, 0.0,      0.0, 1.0,      0.0, 0.0, -1.0,
31         0.7, -0.6, 0.0,      1.0, 1.0,      0.0, 0.0, -1.0,
32
33
34         0.7, -0.1, 0.5,      1.0, 1.0,      1.0, 0.0, 0.0,
35         0.7, -0.1, 0.0,      0.0, 0.0,      1.0, 0.0, 0.0,
36         0.7, -0.6, 0.0,      0.0, 1.0,      1.0, 0.0, 0.0,
37         0.7, -0.6, 0.5,      1.0, 0.0,      1.0, 0.0, 0.0,
38
39         0.0, -0.1, 0.0,      0.0, 1.0,      -1.0, 0.0, 0.0,
40         0.0, -0.1, 0.5,      1.0, 0.0,      -1.0, 0.0, 0.0,
41         0.0, -0.6, 0.5,      1.0, 1.0,      -1.0, 0.0, 0.0,
42         0.0, -0.6, 0.0,      0.0, 0.0,      -1.0, 0.0, 0.0};
43
44     GLushort CubeIndices[] = {0,1,2,3,
45                               4,5,6,7,
46                               8,9,10,11,
47                               12,13,14,15,
48                               16,17,18,19,
49                               20,21,22,23};
50
51     glGenTextures(1, Texture);
52     glBindTexture(GL_TEXTURE_2D, Texture[1]);
53
54     FreeImage_Initialise(TRUE);
55     FIBITMAP *bitmapData;
56     int imgH, imgW;
57     BYTE *bitmapBits;
58     FREE_IMAGE_FORMAT bitmapFormat=FIF_UNKNOWN;
59
60     bitmapFormat    =FreeImage_GetFileType("src/Texture/2.jpeg");
61     bitmapData      =FreeImage_Load(bitmapFormat, "src/Texture/2.jpeg");
62     imgH            =FreeImage_GetHeight(bitmapData);
63     imgW            =FreeImage_GetWidth(bitmapData);
64     bitmapBits      =FreeImage_GetBits(bitmapData);
65
66     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imgW, imgH, 0, GL_BGR,
67     GL_UNSIGNED_BYTE, bitmapBits);
68     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
69     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
70     glTexParameteri(GL_TEXTURE_2D, GL_WRAP_BORDER, GL_REPEAT);
71     FreeImage_Unload(bitmapData);
72     glBindTexture(GL_TEXTURE_2D, Texture[1]);
73
74     glGenBuffers(1, &VBO);
75     glBindBuffer(GL_ARRAY_BUFFER, VBO);
76     glBufferData(GL_ARRAY_BUFFER, sizeof(CubeVertices), CubeVertices,
77     GL_STATIC_DRAW);
78     glGenBuffers(1, &EBO);
79     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
80     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(CubeIndices), CubeIndices,
81     GL_STATIC_DRAW);
82
83     glEnableVertexAttribArray(vPosition);
84     glVertexAttribPointer(vPosition, 3, GL_FLOAT, GL_FALSE, 8*sizeof(float), 0);
85     glEnableVertexAttribArray(vTexture);
86     glVertexAttribPointer(vTexture, 2, GL_FLOAT, GL_FALSE, 8*sizeof(float), (void*)(3*sizeof(float)));
87     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                                    -0.5, 0.0, 0.5,      0.0, 1.0,      0.0,
    -1.0, 0.0,
11                                    0.5, 0.0, 0.5,      1.0, 1.0,      0.0,
    -1.0, 0.0,
12                                    0.5, 0.0, -0.5,      1.0, 0.0,      0.0,
    -1.0, 0.0,
13                                    -0.5, 0.0, -0.5,      1.0, 0.0,
14
15     0.0, 0.5, -0.8,      -0.5, 0.0, -0.5,      1.0, 0.0,
16     0.0, 0.5, -0.8,      0.5, 0.0, -0.5,      0.0, 0.0,
17     0.0, 0.5, -0.8,      0.0, 0.8, 0.0,      0.5, 1.0,
18     0.0, 0.5, -0.8,
19     0.8, 0.5, 0.0,      0.5, 0.0, -0.5,      0.0, 0.0,
20     0.8, 0.5, 0.0,      0.5, 0.0, 0.5,      1.0, 0.0,
21     0.8, 0.5, 0.0,      0.0, 0.8, 0.0,      0.5, 1.0,
22     0.8, 0.5, 0.0,
23     0.0, 0.5, 0.8,      0.5, 0.0, 0.5,      1.0, 0.0,
24     0.0, 0.5, 0.8,      -0.5, 0.0, 0.5,      0.0, 0.0,
25     0.0, 0.5, 0.8,      0.0, 0.8, 0.0,      0.5, 1.0,
26     0.0, 0.5, 0.8,
27     -0.8, 0.5, 0.0,      -0.5, 0.0, 0.5,      0.0, 0.0,
28     -0.8, 0.5, 0.0,      -0.5, 0.0, -0.5,      1.0, 0.0,
29     -0.8, 0.5, 0.0,      0.0, 0.8, 0.0,      0.5, 1.0,
30     -0.8, 0.5, 0.0};

```

```

31     GLushort PyramidIndices[] = {0, 1, 2,
32                                   3, 4, 5,
33                                   6, 7, 8,
34                                   9, 10, 11,
35                                   12, 13, 14,
36                                   15, 16, 17};
37     glGenTextures(1, Texture);
38     glBindTexture(GL_TEXTURE_2D, Texture[0]);
39
40     FreeImage_Initialise(TRUE);
41     FIBITMAP *bitmapData;
42     int imgH, imgW;
43     BYTE *bitmapBits;
44     FREE_IMAGE_FORMAT bitmapFormat = FIF_UNKNOWN;
45
46     bitmapFormat = FreeImage_GetFileType("src/Texture/1.jpeg");
47     bitmapData = FreeImage_Load(bitmapFormat, "src/Texture/1.jpeg");
48     imgH = FreeImage_GetHeight(bitmapData);
49     imgW = FreeImage_GetWidth(bitmapData);
50     bitmapBits = FreeImage_GetBits(bitmapData);
51
52     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imgW, imgH, 0, GL_BGR,
53                 GL_UNSIGNED_BYTE, bitmapBits);
54     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
55     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
56     glTexParameteri(GL_TEXTURE_2D, GL_WRAP_BORDER, GL_REPEAT);
57     FreeImage_Unload(bitmapData);
58
59     glGenBuffers(1, &VBO);
60     glBindBuffer(GL_ARRAY_BUFFER, VBO);
61     glBufferData(GL_ARRAY_BUFFER, sizeof(PyramidVertices), PyramidVertices,
62                 GL_STATIC_DRAW);
63     glGenBuffers(1, &EBO);
64     glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, EBO);
65     glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(PyramidIndices), PyramidIndices,
66                 GL_STATIC_DRAW);
67
68     glEnableVertexAttribArray(vPosition);
69     glVertexAttribPointer(vPosition, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(float), 0);
70     glEnableVertexAttribArray(vTexture);
71     glVertexAttribPointer(vTexture, 2, GL_FLOAT, GL_FALSE, 8 * sizeof(float), (void *)
72                             (3 * sizeof(float)));
73     glEnableVertexAttribArray(vNormal);
74     glVertexAttribPointer(vNormal, 3, GL_FLOAT, GL_FALSE, 8 * sizeof(float), (void *)
75                             (5 * sizeof(float)));
76 }
77
78 void drawPyramid() {
79     generatePyramid();
80     glDrawElements(GL_TRIANGLES, 18, GL_UNSIGNED_SHORT, 0);
81 }

```

3.4 Main

```

1 #include "libs/includes.h"
2
3 using glm::mat4;
4 using glm::vec3;
5

```



```

6 GLint height,width;
7 GLuint program;
8 GLfloat depth = 3.0, high =0.0,side, lighpos, rota, angle, radius=10.0;
9 GLuint loadShaders(const char* vertexFilePath,
10                    const char* fragmentFilePath,
11                    const char* geometryFilePath,
12                    const char* tesscontrolFilePath,
13                    const char* tessevaluationFilePath,
14                    const char* computeFilePath);
15
16 void init(){
17     program = loadShaders("src/Shader/Dreiecke.vs", "src/Shader/Dreiecke.fs",
18                          "", "", "", "");
19     glUseProgram(program);
20     glEnable(GL_DEPTH_TEST);
21     glEnable(GL_CULL_FACE);
22     glFrontFace(GL_CW);
23     glCullFace(GL_BACK);
24 }
25 void display(){
26     glClearColor(1.0,1.0,1.0,1.0);
27     glClear(GL_COLOR_BUFFER_BIT);
28     glViewport(0, height/2, width/2, height/2);
29     vec3 cameraPos = vec3(side, high, depth);
30     vec3 cameraFront = vec3(rota,0.0,-1.0);
31     mat4 Translation = glm::translate(mat4(1.0),vec3(0.15,0.25,0.6));
32     mat4 Scale = glm::scale(vec3(0.5,0.5,0.5));
33     mat4 Model = glm::rotate(Translation*Scale,angle,vec3(1.0,0.0,0.0));
34     mat4 View = glm::lookAt(cameraPos,cameraPos+cameraFront,vec3(0.0,1.0,0.0));
35     mat4 Projection = glm::perspective(120.0f,1.0f,0.1f,10.0f);
36     mat4 ModelViewProjection = Projection * View * Model;
37     GLuint locFinal = glGetUniformLocation(program,"ModelViewProjection");
38     glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
39     GLuint locModel = glGetUniformLocation(program,"Model");
40     glUniformMatrix4fv(locModel,1,GL_FALSE,&Model[0][0]);
41     vec3 lightPos = vec3(-1.5,0.0,0.0);
42     vec3 lightColor = vec3(1.0,1.0,1.0);
43     GLuint loclightPos = glGetUniformLocation(program,"lightPos");
44     glUniform3fv(loclightPos,1,&lightPos[0]);
45     GLuint loclightColor = glGetUniformLocation(program,"lightColor");
46     glUniform3fv(loclightColor,1,&lightColor[0]);
47     GLuint loccameraPos = glGetUniformLocation(program,"viewPos");
48     glUniform3fv(loccameraPos,1,&cameraPos[0]);
49     drawPyramid();
50     Translation = glm::translate(mat4(1.0),vec3(0.15,0.15,0.6));
51     Model = glm::rotate(Translation,angle,vec3(1.0,1.0,1.0));
52     ModelViewProjection = Projection * View * Model;
53     glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
54     drawCube();
55     glViewport(width/2,height/2,width/2,height/2);
56     View = lookAt(vec3(0.0,0.0,3.0),vec3(0.0,0.0,2.0),vec3(0.0,1.0,0.0));
57     Model = Scale;
58     ModelViewProjection = Projection * View * Model;
59     glUniformMatrix4fv(locFinal,1,GL_FALSE,&ModelViewProjection[0][0]);
60     drawCube();
61     drawPyramid();
62     glViewport(0,0,width/2,height/2);
63     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 Szene. Im Moment gibt es keine Lösung.
- Das Programm ist nicht optimiert und kann unter MS Window wegen Bibliotheken nicht laufen.

4 Literatur und Quellenverzeichnis

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