4. Übungsblatt - C++

Henrik Gerdes, Manuel Eversmeyer

20. November 2018

```
/*
 *  MainWindow.hpp
 *
 *  Created on: Nov. 04 2018
 *      Author: Thomas Wiemann
 *
 *  Copyright (c) 2018 Thomas Wiemann.
 *  Restricted usage. Licensed for participants of the course "The C++ Programming Language" only.
 *  No unauthorized distribution.
 */

#ifndef __MAINWINDOW_HPP__
#define __MAINWINDOW_HPP__

#include <string>

#include <SDL2/SDL.h>

#define GL3_PROTOTYPES 1
#include <GL/glew.h>

#include "Model.hpp"
#include "Camera.hpp"

namespace asteroids
{

    class MainWindow
    {
    public:

        /**
         * @brief Construct a new Main Window object
         *
         * @param title    The title of the window
         * @param plyname  A .ply file to render
         * @param w        The window width
         * @param h        The window heigt
         */
        MainWindow(std::string title, std::string plyname, int w, int h);

        /**
         * @brief Start the window's main loop
         */
```

```
45          void execute();
46
47          /**
48           * @brief Destroys the Main Window object
49           *
50           */
51          ~MainWindow();
52
53      private:
54          /* Our SDL_Window ( just like with SDL2 wihout OpenGL) */
55          SDL_Window* mainWindow;
56
57          /* Our opengl context handle */
58          SDL_GLContext mainContext;
59
60          /* The Model to represent*/
61          Model* model;
62
63          /*Window width*/
64          int width;
65
66          /*Windows height*/
67          int height;
68
69          /*Camera*/
70          Camera* camera;
71
72
73      };
74
75  }
76
77  #endif
```

MainWindow.hpp

```
1   /*
2    *   MainWindow.cpp
3    *
4    *   Created on: Nov. 04 2018
5    *       Author: Thomas Wiemann
6    *
7    *   Copyright (c) 2018 Thomas Wiemann.
8    *   Restricted usage. Licensed for participants of the course "The C++
         Programming Language" only.
9    *  No unauthorized distribution.
10   */
11
12  #include "MainWindow.hpp"
13  #include "Camera.hpp"
14
15  #include <iostream>
16  #include <sys/types.h>
17  namespace asteroids
18  {
19
20  MainWindow::MainWindow(
21      std::string title,
```

```
22      std::string plyname, int w, int h)
23  {
24      height = h;
25      width = w;
26      /* Initialize SDL's Video subsystem */
27      if (SDL_Init(SDL_INIT_VIDEO) < 0)
28      {
29
30          printf("Failed to init SDL\n");
31          exit(EXIT_FAILURE);
32      }
33
34      /* Create our window centered at 512x512 resolution */
35      mainWindow = SDL_CreateWindow(title.c_str(), SDL_WINDOWPOS_CENTERED,
            SDL_WINDOWPOS_CENTERED,
36          width, height, SDL_WINDOW_OPENGL);
37
38      if (!mainWindow)
39      {
40          printf("Unable to create window\n");
41          exit(EXIT_FAILURE);
42      }
43
44      /* Create our opengl context and attach it to our window */
45      mainContext = SDL_GL_CreateContext(mainWindow);
46
47      /* Set our OpenGL version.
48         SDL_GL_CONTEXT_CORE gives us only the newer version, deprecated
               functions are disabled */
49      SDL_GL_SetAttribute(SDL_GL_CONTEXT_PROFILE_MASK,
            SDL_GL_CONTEXT_PROFILE_CORE);
50
51      /* 3.2 is part of the modern versions of OpenGL, but most video cards
            whould be able to run it */
52      SDL_GL_SetAttribute(SDL_GL_CONTEXT_MAJOR_VERSION, 3);
53      SDL_GL_SetAttribute(SDL_GL_CONTEXT_MINOR_VERSION, 2);
54
55      /* Turn on double buffering with a 24bit Z buffer.
56         You may need to change this to 16 or 32 for your system */
57      SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);
58
59      /* This makes our buffer swap syncronized with the monitor's vertical
            refresh */
60      SDL_GL_SetSwapInterval(1);
61
62      /* Init GLEW */
63      #ifndef __APPLE__
64      glewExperimental = GL_TRUE;
65      glewInit();
66      #endif
67
68      SDL_GL_SwapWindow(mainWindow);
69
70      /* Init OpenGL projection matrix */
71      glClearColor(0.0, 0.0, 0.0, 1.0);
72      float ratio = 1024 * 1.0 / 768;
73      glMatrixMode(GL_PROJECTION);
74      glLoadIdentity();
```

```
75      glViewport(0, 0, 1027, 768);
76      gluPerspective(45, ratio, 1, 10000);
77
78      /* Ender model view mode */
79
80      glMatrixMode(GL_MODELVIEW);
81
82      model = new Model(plyname);
83      camera = new Camera(Vector(-20.0,0.0,-40.0), 1.0, 5.0);
84  }
85
86  void MainWindow::execute()
87  {
88
89      bool loop = true;
90
91      /*TO DO FIX Camera. (applay)*/
92      /* Set camera position and direction */
93      glLoadIdentity();
94      //gluLookAt(-5.0, 0.0, -30.0, 20.0, -5.0, 1.0, 0.0, 1.0, 0.0);
95
96      while (loop)
97      {
98          glClear(GL_COLOR_BUFFER_BIT );
99
100         SDL_Event event;
101         while (SDL_PollEvent(&event))
102         {
103             /* Check if window has been closed */
104             /* Keybindings */
105             switch(event.type)
106             {
107                 case SDL_QUIT:
108                     loop = false;
109                     break;
110                 case SDL_KEYDOWN:
111                     switch(event.key.keysym.sym)
112                     {
113                         case SDLK_w:     camera->move(Camera::FORWARD);
                                break;
114                         case SDLK_s:     camera->move(Camera::BACKWARD);
                                break;
115                         case SDLK_a:     camera->move(Camera::LEFT); break;
116                         case SDLK_d:     camera->move(Camera::RIGHT); break;
117                         case SDLK_KP_4:  camera->turn(Camera::LEFT); break;
118                         case SDLK_KP_6:  camera->turn(Camera::RIGHT); break;
119                     }
120                     break;
121             }
122             camera->apply();
123             model->render();
124
125         }
126
127         /* Bring up back buffer */
128         SDL_GL_SwapWindow(mainWindow);
129     }
130
```

```
}

MainWindow::~MainWindow()
{
    if(model)
    {
        delete model;
    }
    if(camera)
    {
        delete camera;
    }

    /* Delete our OpengL context */
    SDL_GL_DeleteContext(mainContext);

    /* Destroy our window */
    SDL_DestroyWindow(mainWindow);

    /* Shutdown SDL 2 */
    SDL_Quit();

}

} // namespace asteroids
```

MainWindow.cpp

```
/*
 *  Camera.hpp
 *
 *  Created on: Nov. 04 2018
 *      Author: Thomas Wiemann
 *
 *  Copyright (c) 2018 Thomas Wiemann.
 *  Restricted usage. Licensed for participants of the course "The C++
    Programming Language" only.
 *  No unauthorized distribution.
 */

#ifndef __CAMERA_HPP__
#define __CAMERA_HPP__

#define PI 3.14159265
#define PH 1.57079632

namespace asteroids
{

/**
 * @brief Class to represent 3D vertices with float coordinates
 */
class Vector
{
public:
    /**
     * @brief Construct a new Vector object
     *
```

```
30      * @param ix     Initial x value (default: 0.0)
31      * @param iy     Initial y value (default: 0.0)
32      * @param iz     Initial z value (default: 0.0)
33      */
34     Vector(float ix = 0.0, float iy = 0.0, float iz = 0.0)
35         : x(ix), y(iy), z(iz) {}
36
37     float x;
38     float y;
39     float z;
40 };
41
42
43 /**
44  * @brief Class to represent a virtual camera using gluLookAt
45  *
46  */
47 class Camera
48 {
49 public:
50     /**
51      * @brief Enumeration to encode types of camera movements
52      */
53     enum CameraMovement
54     {
55         FORWARD,
56         BACKWARD,
57         LEFT,
58         RIGHT,
59         UP,
60         DOWN
61     };
62
63     /**
64      * @brief Construct a new Camera object at (0, 0, 0) with
65      *        upward orientation and lookAt at (0, 0, -1)
66      *
67      */
68     Camera();
69
70     /**
71      * @brief Construct a new Camera object with upward orientation
72      *        and lookAt at (0, 0, -1)
73      *
74      * @param position      Initial position
75      * @param turnSpeed     Turning speed in radians per call
76      * @param moveSpeed     Move speed in world units per call
77      */
78     Camera(Vector position, float turnSpeed, float moveSpeed);
79     /**
80      * @brief Destroys the Camera object
81      */
82     ~Camera();
83
84     /**
85      * @brief Moves the camera according to given direction
86      *
87      * @param dir           Moving direction
```

```
88          */
89         void move(CameraMovement dir);
90
91          /**
92          * @brief turns the camera according to given direction
93          *
94          * @param dir           Moving direction
95          */
96         void turn(CameraMovement dir);
97
98         /**
99          * @brief Calls gluLookAt with the internal parameters
100         *
101         */
102        void apply();
103
104        /**
105         * @brief Set the turn speed  of the camera
106         *
107         * @param speed         The new turn speed
108         */
109        void setTurnSpeed(float speed) { m_turnSpeed = speed;}
110
111        /**
112         * @brief Set the move speed  of the camera
113         *
114         * @param speed         The new move speed
115         */
116        void setMoveSpeed(float speed) { m_moveSpeed = speed;}
117
118    private:
119        /// View up vector
120        Vector  m_up;
121
122        /// Translation
123        Vector  m_trans;
124
125        /// Look at vector
126        Vector  m_l;
127
128        /// Rotation angles encoded in vector, i.e., x is the
129        /// rotation around the x-axis and so on
130        Vector  m_rot;
131
132        /// Initial position of the camera
133        Vector  m_initial;
134
135        /// Turn speed in radians per call
136        float   m_turnSpeed;
137
138        /// Move speed in world units per call
139        float   m_moveSpeed;
140    };
141
142    } // namespace asteroids
143
144    #endif
```

Camera.hpp

```
/*
 *  Camera.cpp
 *
 *  Created on: Nov. 04 2018
 *      Author: Thomas Wiemann
 *
 *  Copyright (c) 2018 Thomas Wiemann.
 *  Restricted usage. Licensed for participants of the course "The C++ Programming Language" only.
 *  No unauthorized distribution.
 */

#include <stdio.h>
#include <iostream>
#include <math.h>
#include <GL/glu.h>

#include "Camera.hpp"
namespace asteroids

{

    Camera::Camera()
    {
        m_initial = Vector(0,0,0);
        m_l = Vector(0,0,-1);

    }

    Camera::Camera(Vector position, float turnSpeed, float moveSpeed)
    {
        m_initial = position;
        m_l = Vector(0,0,-1);
        m_trans = Vector(0,0,0);
        m_rot = Vector(0,0,0);
        m_up = Vector(0,1,0);
        m_turnSpeed = turnSpeed;
        m_moveSpeed = moveSpeed;

    }

    void Camera::move(CameraMovement dir)
    {


        switch (dir)
        {
            case FORWARD:    m_trans.x += m_moveSpeed * sin(m_rot.y);
                             m_trans.z += m_moveSpeed * cos(m_rot.y);
                             break;

            case BACKWARD:   m_trans.x -= m_moveSpeed * sin(m_rot.y);
                             m_trans.z -= m_moveSpeed * cos(m_rot.y);
                             break;
            case LEFT:       m_trans.x -= m_moveSpeed * sin(m_rot.y + 90);
```

```
55                              m_trans.z -= m_moveSpeed * sin(m_rot.y );
56                              break;
57              case RIGHT:     m_trans.x += m_moveSpeed * sin(m_rot.y + 90);
58                              m_trans.z += m_moveSpeed * sin(m_rot.y );
59                              break;
60
61              case UP:        m_trans.y += m_moveSpeed;break;
62              case DOWN:      m_trans.y -= m_moveSpeed;break;
63
64              default:        std::cout << "Error: Undefined Label!" << std::
                    endl;
65                  break;
66          }
67      }
68
69      void Camera::turn(CameraMovement dir)
70      {
71          switch (dir)
72          {
73              case LEFT:  m_rot.y -= m_turnSpeed; break;
74              case RIGHT: m_rot.y -= m_turnSpeed; break;
75
76              default:    std::cout << "Error: Undefined Label!" << std::endl
                    ;
77                  break;
78          }
79
80      }
81
82      void Camera::apply()
83      {
84          /* Calc look at vector based on rotation state */
85          m_l.x = m_initial.x + m_trans.x + sin (m_rot.y);
86          m_l.z = -m_initial.z - m_trans.z - cos (m_rot.y );
87          m_l.y = m_initial.y + m_trans.y + sin (m_rot.x);
88
89          /* Clear matrix stack */
90          glLoadIdentity () ;
91
92          /* Apply transformation */
93          gluLookAt (m_initial.x + m_trans.x, m_initial.y + m_trans.y, -
                m_initial.z
94              -m_trans.z, m_l.x, m_l.y, m_l.z, m_up.x, m_up.y, m_up.z);
95
96
97      }
98
99      Camera::~Camera()
100     {
101         /*Nothing to do. Everything ist static */
102     }
103 }
```

Camera.cpp

```cpp
/*
 *  Main.cpp
 *
 *  Created on: Nov. 04 2018
 *      Author: Thomas Wiemann
 *
 *  Copyright (c) 2018 Thomas Wiemann.
 *  Restricted usage. Licensed for participants of the course "The C++ Programming Language" only.
 *  No unauthorized distribution.
 */

#include "MainWindow.hpp"
#include "Camera.hpp"

#include <iostream>

int main(int argc, char** argv)
{

    if (argc == 2)
    {
    std::string buffer = argv[1];
    //buffer = "../models/arrow.ply";
    asteroids::MainWindow* modelwindow = new asteroids::MainWindow("Model Render", buffer, 666,500);
    modelwindow->execute();
    delete modelwindow;

    }
    else
    {
        std::cout << "usage: asteroids <modelfile>" << std::endl;
    }
    return 0;
}
```

Main.cpp