HOCHSCHULE DARMSTADT

Graphische Datenverarbeitung

Dokumentation - Motorbike Game

Dokumentationsbericht vom 12.03.2021

Veranstaltung: Graphische Datenverarbeitung
Studiengang: INFORMATIK
Koordinatoren: Prof. Dr. Björn Frömmer
Studierende: Jan Niklas Blanquett (760515)
Loredana-Lavinia Mihut (759509)

Inhaltsverzeichnis

1 Einleitung **2**

2 Technische Details **2**

3 Spielbeschreibung **3**

4 Programmstart **4**

5 Modelle **6**

5.1 Driver.cpp Driver.h 6

5.2 Motorbike.cpp Motorbike.h 6

5.3 Landscape.cpp Landscape.h 8

5.4 Text.cpp Text.h 8

5.5 Cube.cpp Cube.h 8

6 Hauptprogramm **9**

6.1 draw() 9

6.2 building() 9

6.3 singlebuilding() 9

6.4 mainMenu() 10

6.5 subMenu() 11

6.6 Specialfunc() 11

6.7 environment() 13

6.8 resize() 14

6.9 backgroundcolor() 14

6.10 init() 15

6.11 prepareGametoDisplay() 16

6.12 MouseFunc() 16

6.13 key() 17

6.14 LIGHT 18

6.15 Toggle_fullscreen() 18

1 Einleitung

Diese Dokumentation dient als Anleitung / Erklärung des Spiels Motorbike Game für die alternative Prüfung (Rendering Competition) zum Thema Fahrzeuge im Fach Graphische Datenverarbeitung im WS2021.

In Rahmen dieses Projekts werden folgenede Anforderungen erfühlt:

- Kreativität
- einen komplexen, mehrfach verzweigten Szenegraph mit mind. 3 Ebenen
- sinnvolle, abhängige Animationen
- Userinput
- Licht / Beleuchtung
- Texturen

2 Technische Details

Entwicklungsumgebung: CLion
Programmiersprache: C++
Betriebssystem: Windows
Benötigte Bibliotheken: FreeGlut / Glut u. Soil (Texturen)
Ausführen des Projekts: um das Projekt in CLion ausführen zu können, benötigt man eine CMake-Datei CMakeList.txt, die folgeden Informationen beinhaltet:

```
add_executable(gdv_blanquettmihut
        main.cpp
        Cube.h Cube.cpp
        Landscape.cpp Landscape.h
        Motorbike.cpp Motorbike.h
        Text.cpp Text.h
        Driver.cpp Driver.h)

target_link_libraries(gdv_blanquettmihut -lSOIL -lOpenGL32 -lfreeGLUT -lglu32)
```

Figure0 CMakeList.txt

3 Spielbeschreibung

Das Motorbike-Game stellt ein einfaches 3D-Spiel dar, in dem ein Motorradfahrer versucht, möglichst viele BitCoins einzusammeln, ohne die roten Hindernisse, die Kegeln auf der Straße zu berühren, da in dem Fall das Spiel zu Ende ist.

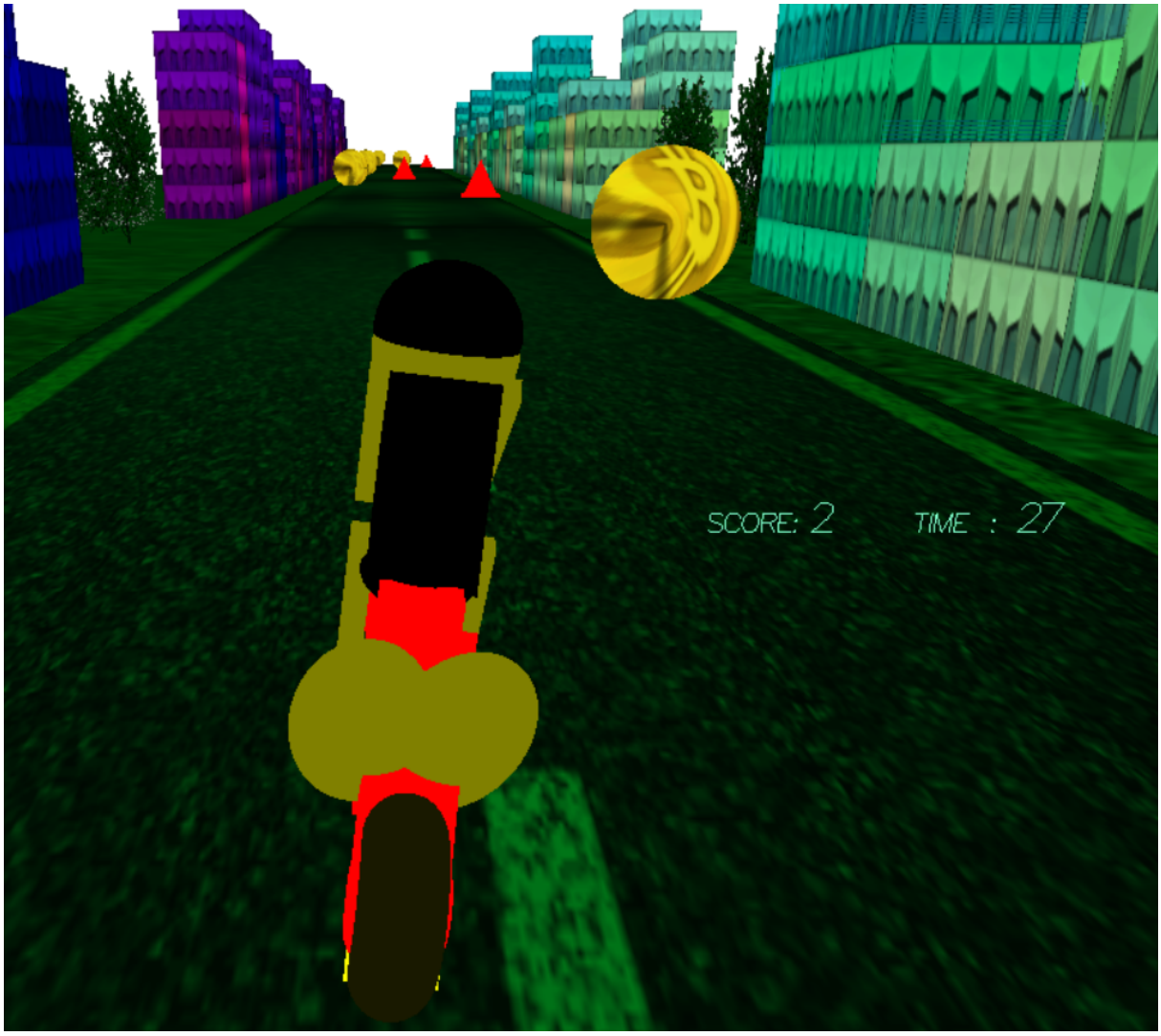

Figure1 Ausschnitt aus dem Motorbike Game

4 Programmstart

Beim Ausführen des Programms landet man im Hauptmenu, wo man aufgefordert wird, eine von zwei Optionen: L für Licht und G für Spielstart auszuwählen.

Figure2 Spielstart

Wird die Option L vom Spieler ausgewählt, so erscheint eine dunkle Szene, die vom Motorradlichtwerfer rundherum beleuchtet wird (siehe Figure2).

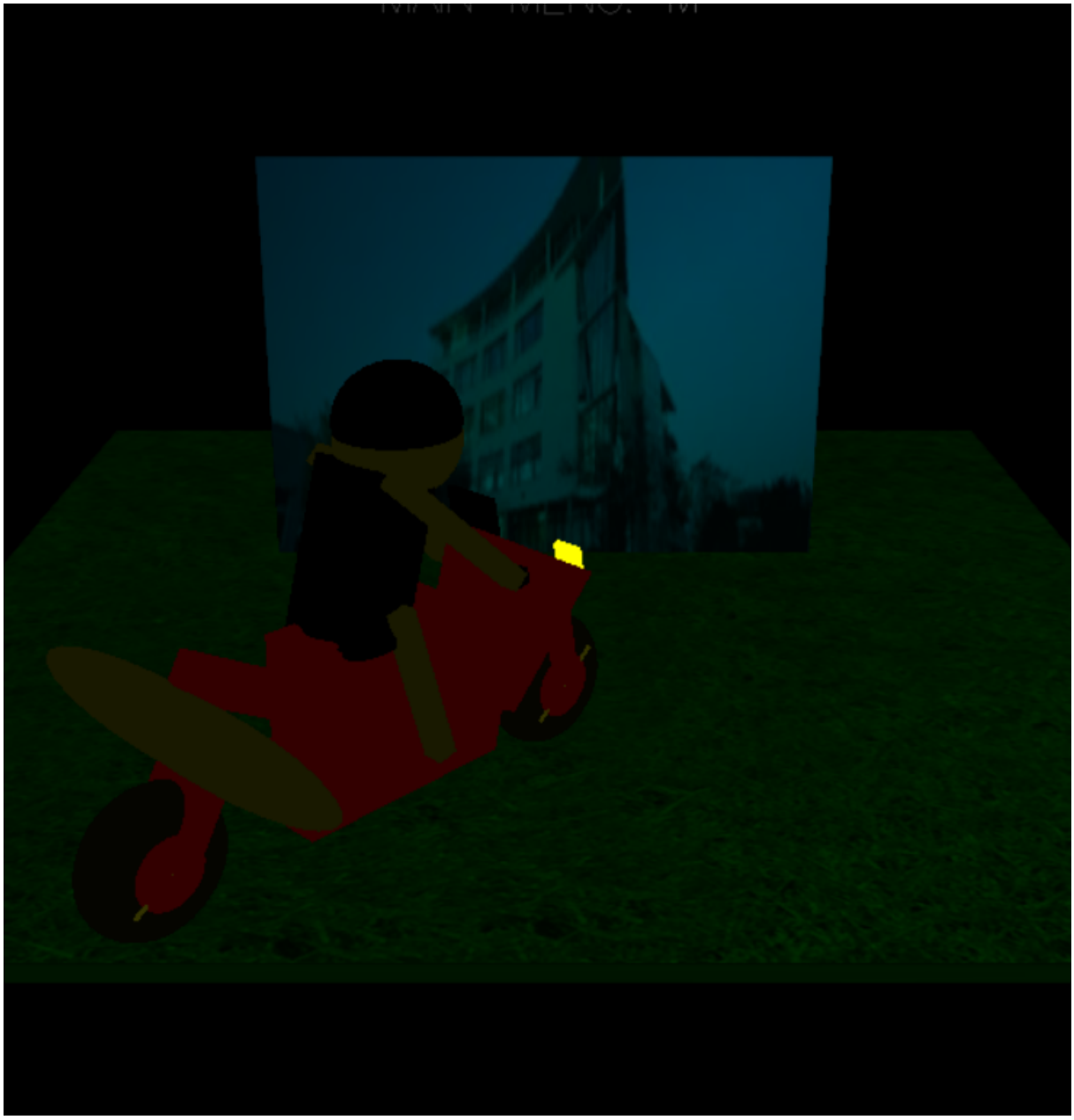

Figure3 Lichtzene

Beim Wählen der Option G (Start Game) wird das tatsächliche Spiel wie in der Spielbeschreibung angekündigt gestartet.

5 Modelle

5.1 Driver.cpp Driver.h

Die Klasse Driver wird dafür benutzt, um den Fahrer zu zeichnen. Der dazugehörige Szenegraph ist unten abgebildet:

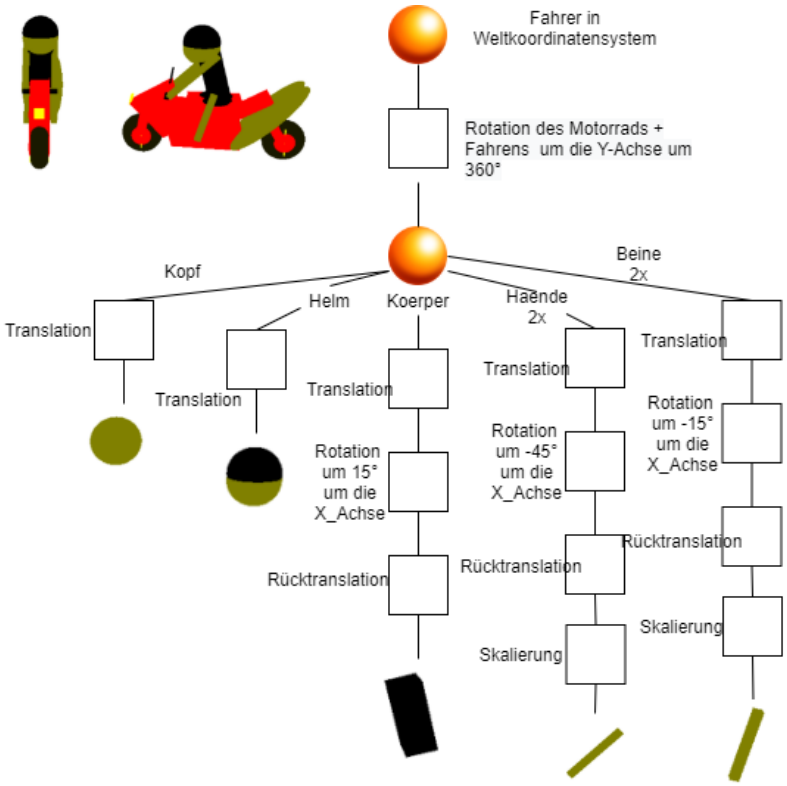

Figure4: Szenegraph Driver

5.2 Motorbike.cpp Motorbike.h

Das Motorrad wurde in der Klasse Motorbike.cpp erzeugt. Der Szenegraph ist unten abgebildet:

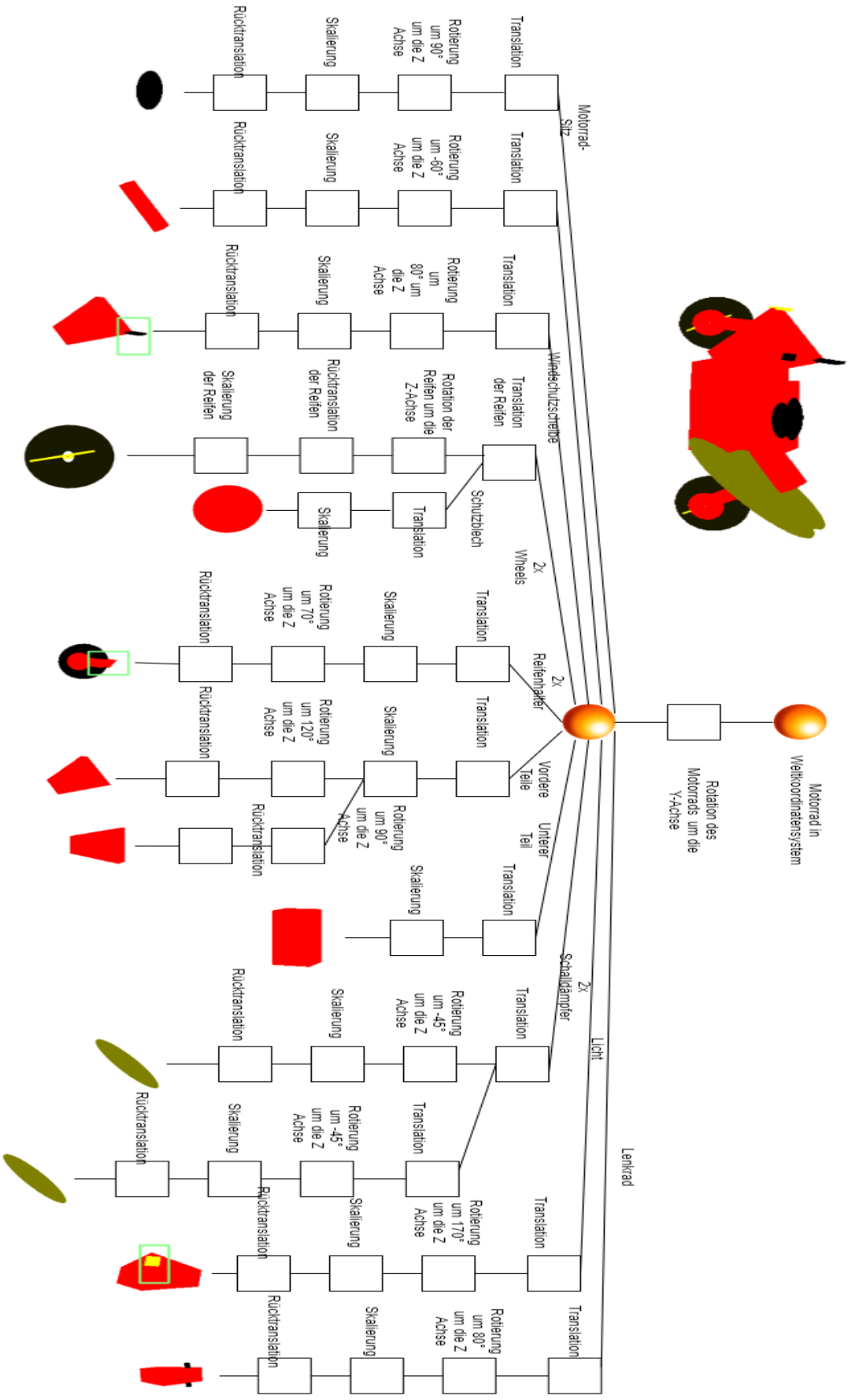

Figure5: Szenegraph Motorrad

5.3 Landscape.cpp Landscape.h

In dieser Klasse sind die Funktionen zu fineden, die mit dem Zeichen der Landschaft zu tun haben.

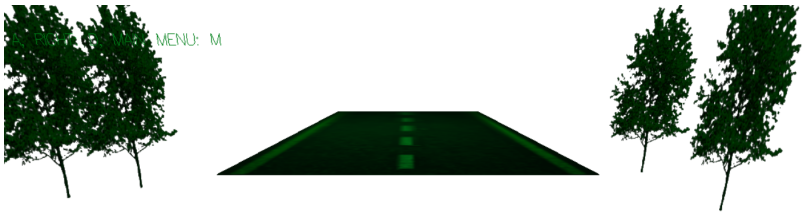

Figure6: Landschaft

5.4 Text.cpp Text.h

In dieser Klasse sind verschiedene Funktionen zu finden, die die Ausgabe von Text auf dem Bildschirm ermöglichen.

```
void drawChar(char c, float x, float y, float z)
{
    glPushMatrix();
    glTranslatef(x,  y: y+8,z);
    glScalef( x: 0.003f, y: 0.003f,z);
    glutStrokeCharacter( font: GLUT_STROKE_ROMAN , c);
    glPopMatrix();
}
```

Figure7: Landschaft

5.5 Cube.cpp Cube.h

Hier findet man eine Erweiterung der von Dozenten zur Verfügung gestellten Datei Wuerfel_mit_normalen. Diese beinhaltet Funktionen, die die Verwendung von Texturen zulässt.

6 Hauptprogramm

6.1 draw()

Die Funktion draw() dient dazu den nächsten Frame im Spiel als Bild darzustellen. Sie wird also fortwährend des Spielablaufes ständig aufgerufen. Einserseits wird das Motorrad für eine korrekte Position skaliert und anschließend mit der Funktion drawbike() gezeichnet. Außerdem sorgt sie dafür, dass sich das Motorrad sich nur zu einer gewissen Weise rotiert. Des Weiteren wird eine Spurbegrenzung des Motorrads überprüft. Es werden dort die Zeichnungen der Szene entlang der Ferne in Richtung der Z-Achse hinein festgelegt.Abschließend wird noch überprüft, ob das Motorrad einen Hinderniskegel oder einen entsprechenden Bitcoin getroffen hat.

6.2 building()

Die Funktion building() dient als Vorbereitung für die Funktion singlebuilding(). Mit ihrem Aufruf wird festgelegt, wie oft die Funktion singlebuilding() aufgerufen wird, also wie viele Häuser nacheinnander erstellt werden sollen.

6.3 singlebuilding()

Die Funktion singlebuildung() dient dazu die Erstellung eines einzelnenes Hauses vorzubereiten. Anfangs wird per Modulo eine Farbrichtung des Hauses ausgewählt. Mit dem Aufruf der Funktion Wuerfel-TexturBuilding() wird die Textur, die die Häuser haben sollen übergeben. Dort werden dann mit verschiedenen GL-POLYGON-Befehlen das konkrete Haus gezeichnet.

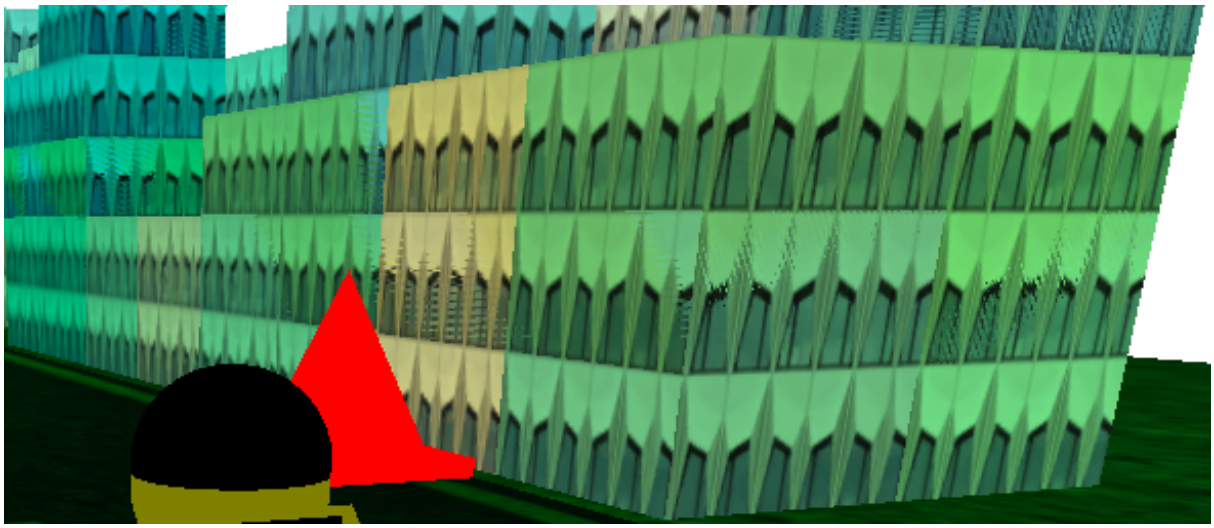

Figure8: Häuserabschnitt

6.4 mainMenu()

Die Funktion mainMenu() dient dazu die Optionen der Funktion subMenu() darzustellen. Sie wird mit einem Rechtsklick dargestellt und man erhält die Optionen Background Color und Exit zur Auswahl, wie unten dargestellt.

Figure9: Hauptmenü

6.5 subMenu()

Die Funktion subMenu() dient dazu die entsprechenden Optionen des Menüs, die sogenannten Labels die zum Untermenüpunkt BackgroundColor und des Hauptmenüs gehören zu definieren und zu erstellen. Unten zu sehen ist die Auswahl der Background Farben.

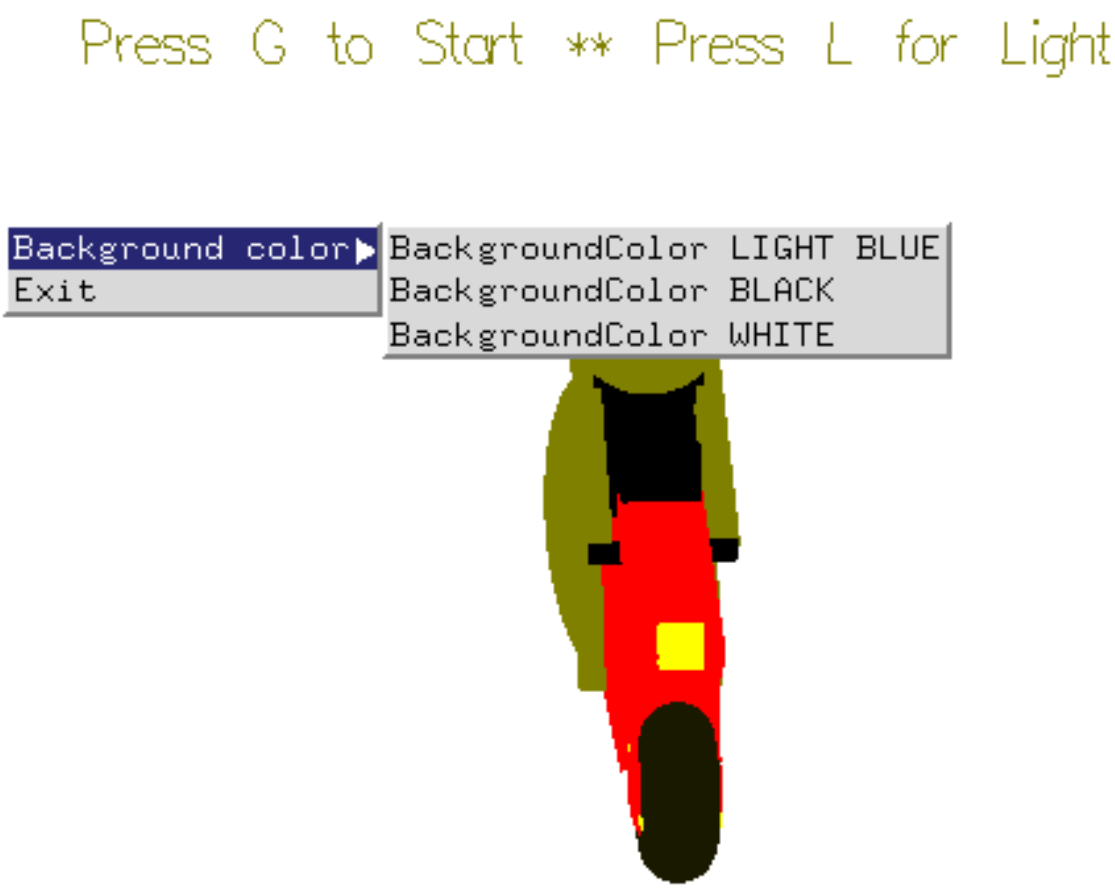

Figure10: Untermenü

6.6 Specialfunc()

Mit der Funktion Specialfunc() werden die Funktions- und Pfeiltasten abgefragt. Mit den Pfeiltasten Oben, Unten, Links und Rechts wird die Ansicht im Spiel geändert. Mit Links und Rechts verändert sich die Ansicht in X-Richtung. Mit Oben und Unten hingegen verändert sich die Ansicht hingegen in Y-Richtung. Unten ist eine Abbildung zu sehen mit einem maximalen X-Winkel und maximalen Y-Winkel.

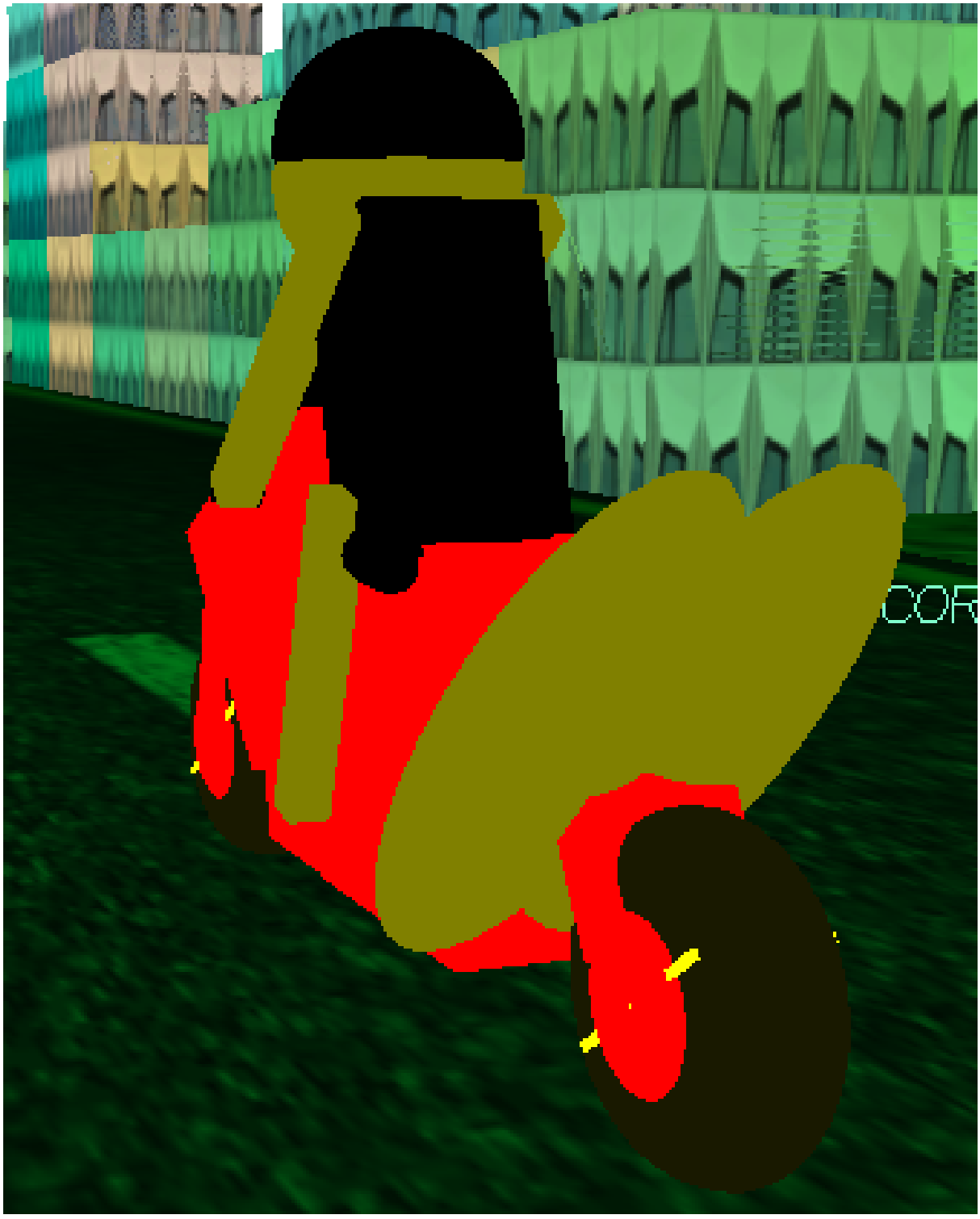

Figure11: Veränderte Ansicht des Spiels

6.7 environment()

In der Funktion environment() wird die Umgebung (Hochhäuser, Straße, Grassfläche, Bäume, Hindernisse, BitCoins) vorbereitet und angezeigt.

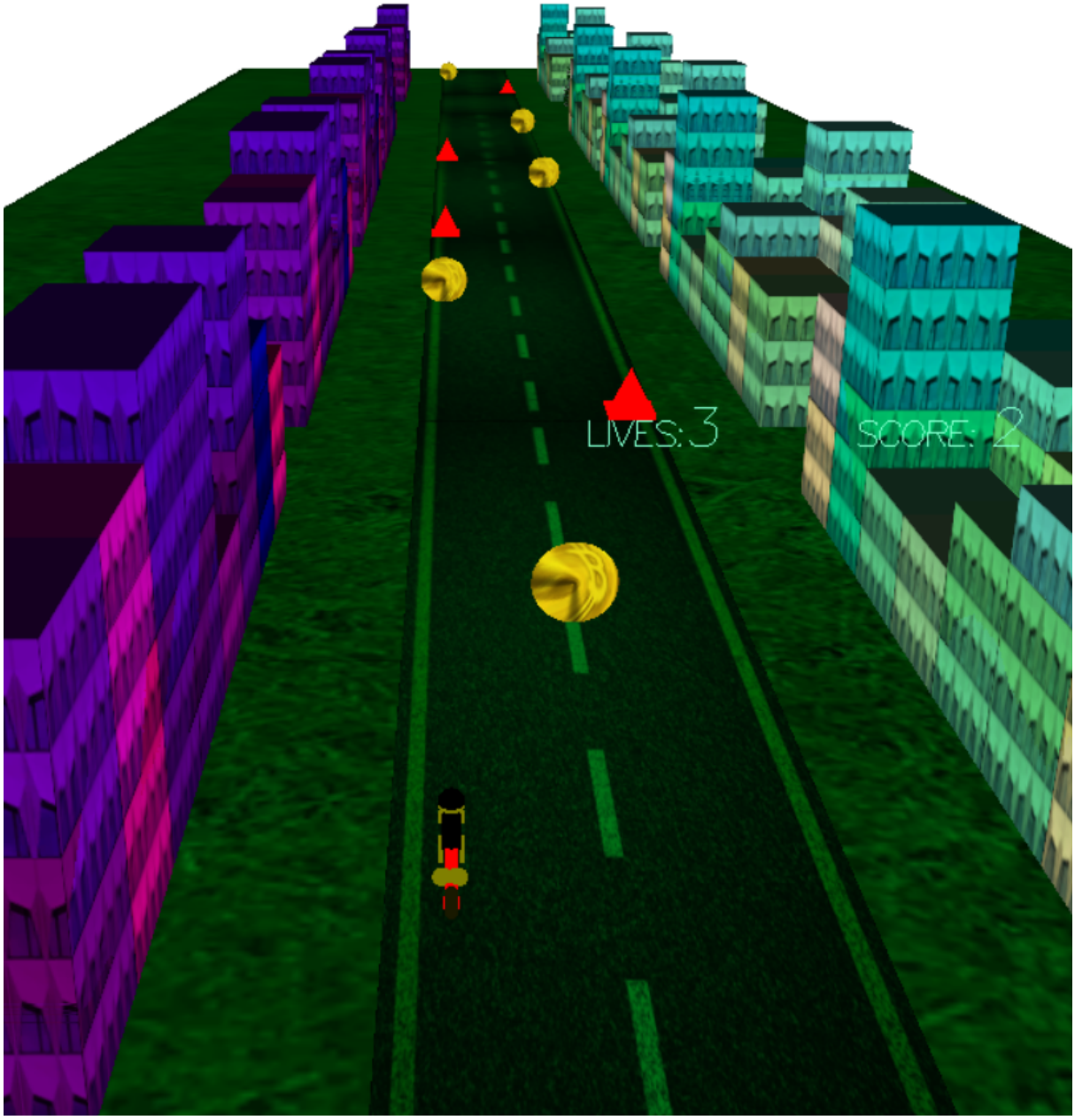

Figure12: Landschaft im Spiel

6.8 resize()

Damit die Größe der Objekte nicht explizit von der Fenstergröße abhängig gemacht wird, wurde in der Funktion resize() die Projektionsmatrix angepasst, wenn sich die Fenstergröße ändert.

```
static void resize(int width, int height)
{
    const float ar = (float) width / (float) height;
    glViewport( x: 0,  y: 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-ar, ar,  bottom: -1.0,  top: 1.0,  zNear: 2.0,  zFar: 1000.0);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

Figure13: Fenstrgröße anpassen

6.9 backgroundcolor()

Diese Funktion dient dazu, mittels Mouse-Interaktion die Hintergrundfarbe zu ändern. Dies wird in einem switch-case Block abgefangen: die Hintergrundfarbe wird gesetzt und zusätzlich wird einfach ein Flag gesetzt, das glut anweist, den Display-Rückruf bei der nächsten Schleifeniteration aufzurufen. Es stehen drei Optinen zur Verfügung: hell blau, schwarz und weiß.

Figure14: Hintergrundfarbe ändern

6.10 init()

Hier finden jene Aktionen statt, die zum Programmstart einmalig durchgeführt werden müssen, wie *gluOrtho2D* - Koordinaten für die linke und rechte vertikale Schnittebene - *glClearDepth* - gibt den Tiefenwert an, der von glClear zum Löschen des Tiefenpuffers verwendet wird - und das einmalige Laden von Texturen:

```
buildingTexture = SOIL_load_OGL_texture(buildingTexpath,
                                        force_channels: SOIL_LOAD_AUTO,
                                        reuse_texture_ID: SOIL_CREATE_NEW_ID,
                                        flags: SOIL_FLAG_MIPMAPS
                                        | SOIL_FLAG_INVERT_Y
                                        | SOIL_FLAG_NTSC_SAFE_RGB
                                        | SOIL_FLAG_COMPRESS_TO_DXT );
```

6.11 prepareGametoDisplay()

Mittels dieser Funktion wurde das Menu im Programm eingebaut. Der Benutzer hat die Möglichkeit zwischen Hauptmenu/Startseite, Lichtszene und Spiel hin und her zu wechseln. Der Wechsel erfolgt mittels Tastatureingaben:

- **M** - Startseite
- **G** - Spielstart
- **L** - Lichtszene

6.12 MouseFunc()

Durch rechten Mouse-Click kann der Spieler zwischen Hintergrundfarbe ändern oder Programm beenden auswählen.

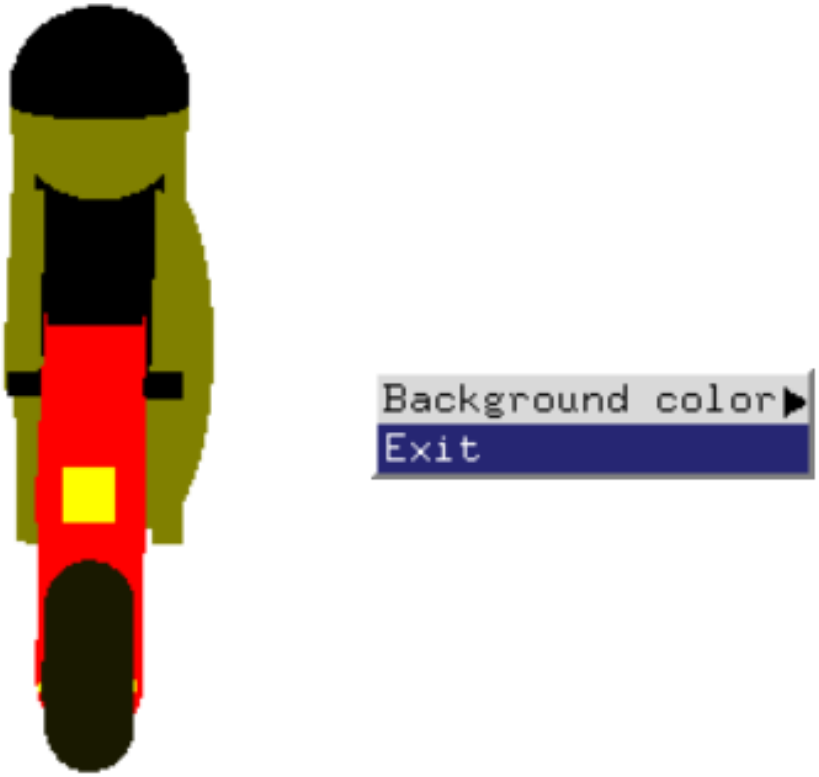

Figure15: Mouse Interaktion

6.13 key()

Der Userinput mittels Tastatureingaben wird wie folgt ermöglicht:

- **q:** das Programm wird sofort beendet
- **r:** Rotation um 360° um die ganze Szene herum
- **t:** Beenden der Rotation
- **z:** hineinzoomen
- **Z:** hinauszoomen
- **a:** Lenken des Motorrads nach links
- **d:** Lenken des Motorrads nach rechts
- **g:** Starten des Spiels
- **f:** Switchen zwischen Vollbild- und Normalbildmodus
- **m:** zurück zum Hauptmenüseite
- **l:** zur Lichtszene navigieren

6.14 LIGHT

Für die Beleuchtung der Szene wurden die folgenden konstanten Variablen erzeugt und initialisiert:

```
const GLfloat light_ambient[]  = { 0.0f, 0.0f, 0.0f, 1.0f };
const GLfloat light_diffuse[]  = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat_ambient[]    = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[]    = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[]   = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };
```

Figure16: Mouse Interaktion

Mittels den OpenGl-Funktionen werden die Lichter aktiviert *glEnable(GL_LIGHT0) u. glEnable(GL_LIGHTING)*. glLight setzt die verschiedenen Eigenschaften einer Lichtquelle. Die Lichtquelle wird über den Parameter light bestimmt und ist ein symbolischer Name der Form: GL_LIGHTi wobei 0 <= i < GL_MAX_LIGHTS ist.

Um die automatische Normalisierung zu aktivieren, wird *glEnable(GL_NORMALIZE)* verwendet. Wenn *glEnable(GL_COLOR_MATERIAL)* aktiviert ist, beinflussen die Eigenschaften, die über Mode angegeben wurden, der Seiten, die über Face gewählt wurden, die aktuelle Farbe über die ganze Zeit.

6.15 Toggle_fullscreen()

Wie der Name schon verrät, wird diese Funktion benutzt, um mithilfe der Drücktaste F zwischen einer Vollanzeige und normaler Bildschirmgröße hin- und herzuschalten.