Documentation zum VRML Projekt

Computergrafik WS21/22

Iúri Figueiredo Archer

November 30, 2021

Einleitung und Werkzeuge

Dieses Dokument dient sowohl als Dokumentation für die Abgabe wie auch als meine eigene Referenz um die Abgabe zu erleichtern. Desweiteren nutze ich die Gelegenheit auch um meine LaTeX Kenntnisse zu erfrischen.

Es wurden folgende Programme benutzt, um das VRML Projekt zu realizieren.

- *Visual Studio Code* für VRML und Python
- LaTeX für die Dokumentation
- *view3dscene* und *Cortona3D* für die Visualisierung der VRML-Welt
- *draw.io* wurde für das Erstellen der Diagramme und Skizzen genutzt und um diese als .pdf zu exportieren.

Cortona3D war leider notwendig, da *view3dscene* noch keine Skripte unterstützt (weder JS noch VRML-Script). Dies stellte sich als Herausforderung, da Cortona3D nur von Internet Explorer unterstützt wird und sonst keinem anderen Browser - was mir als Linux Nutzer besonders viel Spaß bereitet hat.

Dieses Projekt wurde mit Git erstellt und existiert als Repository auf Github unter `github.com/cyberme0w/VRMLProjekt`.

1 Aufgaben und Umfang der Bearbeitung

V1

a)

Die Szene wurde im VRML-Browser *view3dscene* geöffnet, da es leider für Linux keine vernünftige Alternative gibt.

b)

Das entsprechende Szenengraph sieht folgendermaßen aus:

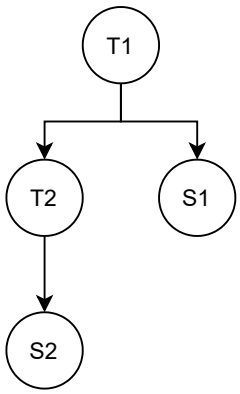

Figure 1: Szenengraph aus *Transform*- und *Shape*-Knoten von meinErstesVRML.wrl

c)

Ändert man die Rotation oder Translation von T1 (oberstes Transform), wirkt diese auf beide Figuren (Sphere und Box). Ändert man die untergeordnete Transforms, wirken diese nur auf die entsprechende Figur. Die Eigenschaft scaleOrientation gibt an, bezogen auf welche Axe die Skalierung durchgeführt wird. Da in der meinErstesVRML Welt keine Skalierungen durchgeführt werden, spielt die scaleOrientation keine Rolle.

d) (Zusatzaufgabe)

Man kann die *Shapes* einfärben, indem man eine der *Color* Eigenschaften des Material Knoten verändert.

```
#VRML V2.0 utf8
material Material{ diffuseColor 0   1 0 } # Green
material Material{ diffuseColor 1 65 0 } # Orange
```

e) (Zusatzaufgabe)

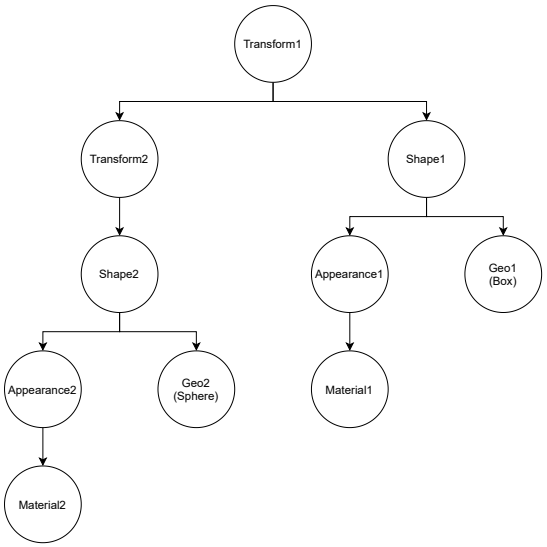

Figure 2: vollst. Szenengraphen von meinErstesVRML.wrl

V2

a)

Es wurde anhand der vorhandenen Punkte folgende Skizze erstellt:

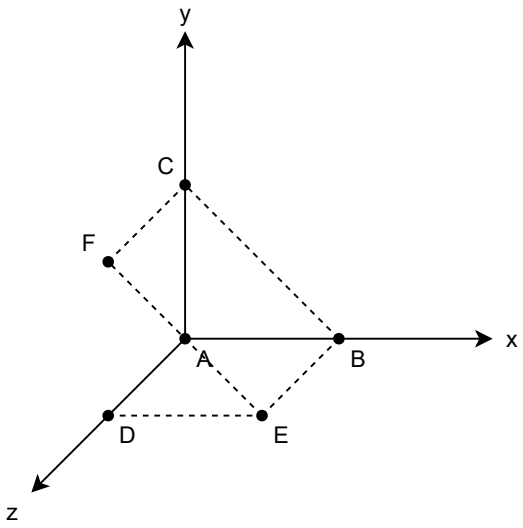

Figure 3: Skizze zu *meinZweitesVRML.wrl* anhand der Punkte

Die Szene wurde anschließend im VRML-Browser *view3dscene* geöffnet, da es leider für Linux keine vernünftige Alternative gibt.

b)

Die Standardblickrichtung von view3dscene ist:

- POS: 0.50 0.50 3.00
- DIR: 0.00 0.00 -1.00
- UP: 0.00 1.00 0.00

Man sieht aus dieser Richtung nichts, weil nur die Rückseiten der Flächen sichtbar sind. Wenn man **solid false** wieder "reinkommentiert", werden die Flächen nicht mehr als Teile eines Körpers mit Volumen (wo die Innenseiten nicht dargstellt werden müssen) betrachtet und daher werden die Rückseiten auch dargestellt.

"The solid field indicates whether the shape encloses a volume (TRUE), and can be used as a hint to perform backface culling. If nothing is known about the shape, this field value is FALSE (and implies that backface culling cannot be performed and that the polygons are two-sided)."

Quelle (11.11.2021):
`http://graphcomp.com/info/specs/sgi/vrml/spec/part1/concepts.html#GeometryNodes`

c)

Aufgabe in der Datei erledigt. Siehe *meinZweitesVRML.wrl*.

d)

Der Mittelpunkt des Vierecks BCFE ist [0.5 0.5 0.5].

e) und f) (Zusatzaufgaben)

Beide Aufgaben wurden in *meinZweitesVRML.wrl* erledigt.

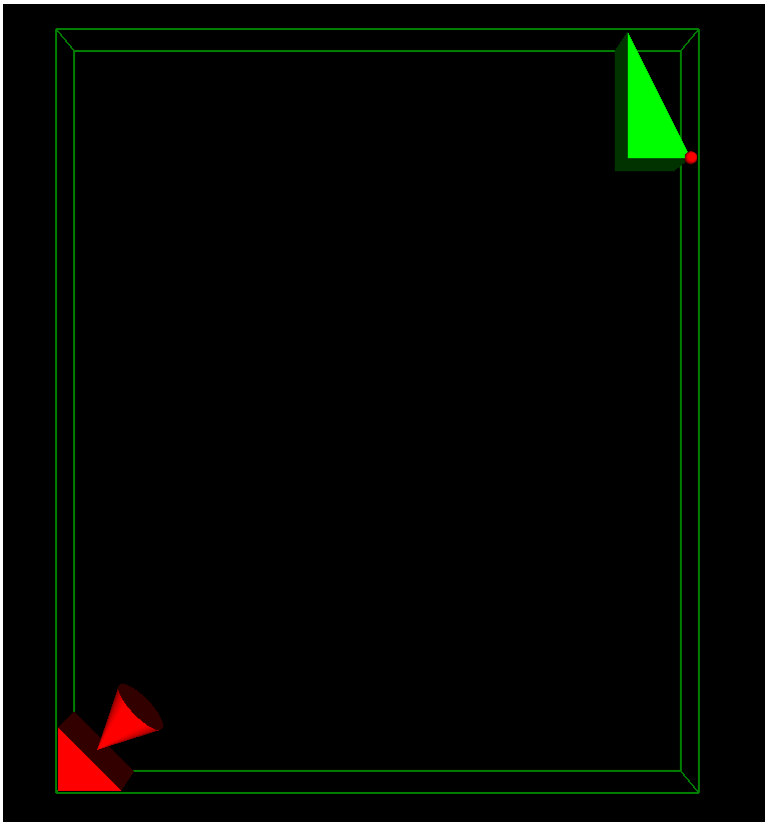

Figure 4: Das gleiche Prisma mit verschiedenen Eigenschaften. Die rote Kugel dient als Hilfspunkt.

Das eigentliche VRML Projekt

Nicht jede Aufgabe ab hier benötigt eine Beschreibung, da die Arbeit aus der Visualisierung der .wrl Datei genügen dürfte. Hier eine Liste der Aufgaben, die erledigt wurden.

- V3:
  - a) Datei anlegen - **DONE**
  - b) Skifahrer - **DONE**
  - c) Berghang - **DONE**
  - d) Skifahrer zu Proto - **DONE**
- V4:
  - a) Vogelperspektive - **DONE**
  - b) 4 weitere Viewpoints - **DONE**
  - c) FPV Skifahrer + Name - **DONE**
- V5:
  - a) Grüne Scheinwerfer - **DONE**
  - b) (Zusatz) Lichtkegel - **DONE**
- V6:
  - a) Skybox - **DONE**
  - b) Scheinwerfer AN/AUS Skript - **DONE**
- V7:
  - a) Skifahrer Animation (Gerade + Kreis) - **DONE**
  - b) Klick auf Skifahrer für Animation -**DONE**
  - c) Schlitten verfolgt Skifahrer
- V8:
  - a) Kastenhaus + Baum Texturen - **DONE**
  - b) Nachtskifahren + Fackel - **DONE**
- V9 (Zusatz): Verschönerungen - **DONE**

Anmerkungen zu den Aufgaben

V3b

Für das Anlegen des Skifahrers wurde zuerst eine Skizze erstellt, um die Verbindung der Punkte im indexedFaceSet zu vereinfachen. Da ich dieses Kunstwerk mit der Welt teilen möchte, füge ich es hier mal ein. Dies entspricht nicht das Endergebnis, welches in *miniproject.wrl* zu sehen ist. Das Objekt besteht außerdem aus mehr als 50 Oberflächen, allerdings wurde dies in Rücksprache mit Herrn Dörner gemacht und soll kein Problem darstellen.

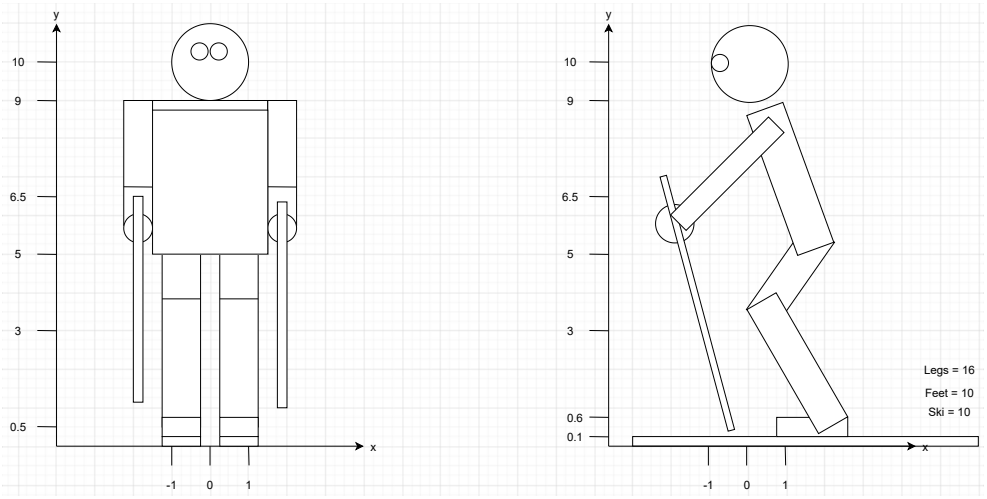

Figure 5: Erste Skizze des Skifahrers

V9

Es wurden zwei Verschönerungen durchgeführt. folgendes gemacht:

Die erste Verschönerung besteht aus einer innerhalb des Hauses integrierte Werbewand. Die soll eher als Easter Egg gesehen werden, als ernsthafter Versuch, die Szene deutlich schöner zu machen. Ich hoffe, ich kann dennoch hiermit zeigen, dass ich in VMRL mit Video-Dateien umgehen kann, wie in der Aufgabe gefordert.

Die zweite Verschönerung war die Verbesserung der Skifahrt auf kurvengenauen Pfaden. Die Punkte wurden mit einem selbstgeschriebenen Pythonskript ausgerechnet und können somit beliebig genau sein. Die Datei ist ebenfalls im Ordner enthalten.

```
import numpy as np
import math as m

def calculatePointsWithinAngle(ammount, centerX, centerY, radius, startTime, endTime, s

    # calculate increments and create array
    angleIncrement = (endAngle - startAngle) / (ammount + 1)
    timeIncrement = (endTime - startTime) / (ammount + 1)
    points = np.zeros((ammount + 2, 4), dtype=float)

    # calculate each point
    for i in range(ammount + 2):
        currentAngle = startAngle + angleIncrement * i
        currentTime = startTime + timeIncrement * i
```

```
        # x value
        points[i][0] = centerX + radius * m.cos(m.radians(currentAngle))

        # y value
        points[i][1] = centerY + radius * m.sin(m.radians(currentAngle))

        # rotation (depending on clockwise/counterclockwise)
        if(angleIncrement > 0):
            points[i][2] = m.radians(currentAngle + 90)
        else:
            points[i][2] = m.radians(currentAngle - 90)

        # time
        points[i][3] = currentTime

    return points

def printVRMLfromPoints(points):

    print("\npositions:")
    print("key [")
    for point in points:
        print("{:.5f}".format(point[3]))
    print("]")
    print("keyValue [")
    for point in points:
        print("{:.2f} {:.2f} {:.2f}".format(point[0], 0.0, point[1]))
    print("]")

    print("\norientations:")
    print("key [")
    for point in points:
        print("{:.5f}".format(point[3]))
    print("]")
    print("keyValue [")
    for point in points:
        print("0 1 0 {:.2f}".format(point[2]))
    print("]")

# Beispieleingabe
center = (-48.5, 282)
radius = 23.5
ammount = 10
startTime = 0.23
endTime = 0.30
startAngle = 0
endAngle = -90

points = calculatePointsWithinAngle(ammount, center[0], center[1], radius, startTime, e
printVRMLfromPoints(points)
```

Quellen und Lizenzen

Texturen

- Brick Texture, von bart:
  - Datum: 29.11.2021
  - `https://opengameart.org/content/brick-texture`
  - Lizenz: CC-BY 3.0 + CC-BY-SA 3.0 + GPL 3.0
- Baumstamm, von bart:
  - Datum: 29.11.2021
  - `https://opengameart.org/content/seamless-tiling-tree-bark-texture`
  - Lizenz: GPL 2.0 + GPL 3.0 + CC-BY-SA 3.0
- Baumblätter, von qubodup:
  - Datum: 29.11.2021
  - `https://opengameart.org/content/wall-grass-rock-stone-wood-and-dirt-480`
  - Lizenz: CC0
- Door, von qubodup:
  - Datum: 29.11.2021
  - `https://opengameart.org/content/weathered-wood-door`
  - Lizenz: CC-BY 3.0
- Never Gonna Give You Up (video clip)
  - Datum: 30.11.2021
  - `https://www.youtube.com/watch?v=iik25wqIuFo`
  - Lizenz: ehm... Fair Use?