Miniprojekt (V3)

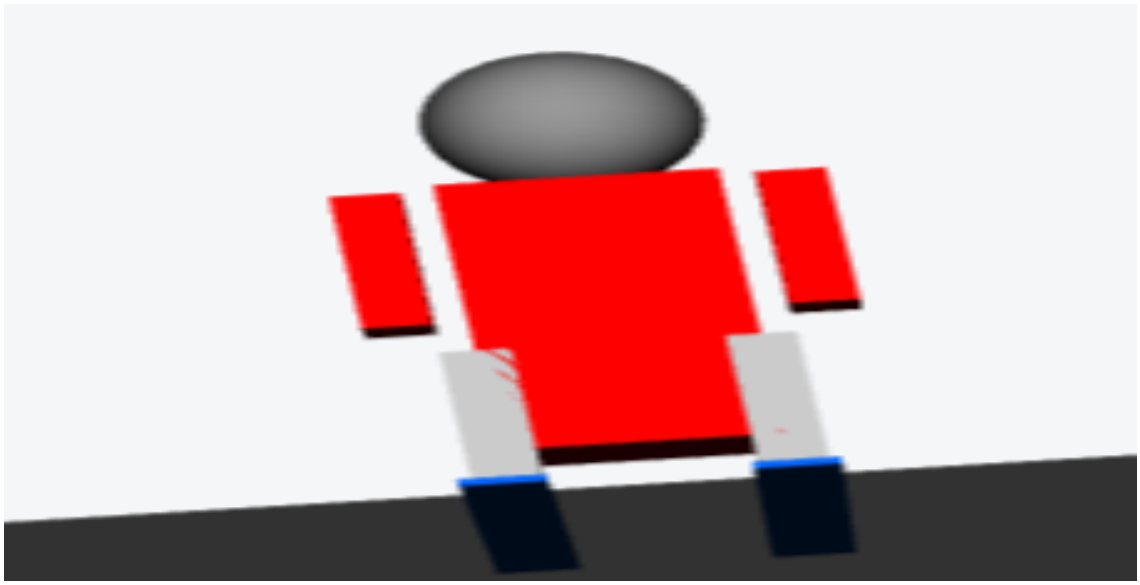
V3:

a.

Die Projektdateien wurden gemäß den Bewertungsrichtlinien organisiert. Die Hauptdatei heißt SkifahrerHaupt.wrl.
Die Struktur der Dateien ist wie folgt: SkifahrerHaupt.wrl: Hauptszene mit Berghang, Bäumen, Hütte und mehreren Skifahrern.

b.

Der Skifahrer wurde mithilfe von IndexedFaceSet-Objekten modelliert, wobei grundlegende Primitive wie Sphere für den Kopf und Box für die Skier verwendet wurden. Die Skier wurden mit Transform-Nodes positioniert, um sie korrekt unter den Füßen des Skifahrers zu platzieren.

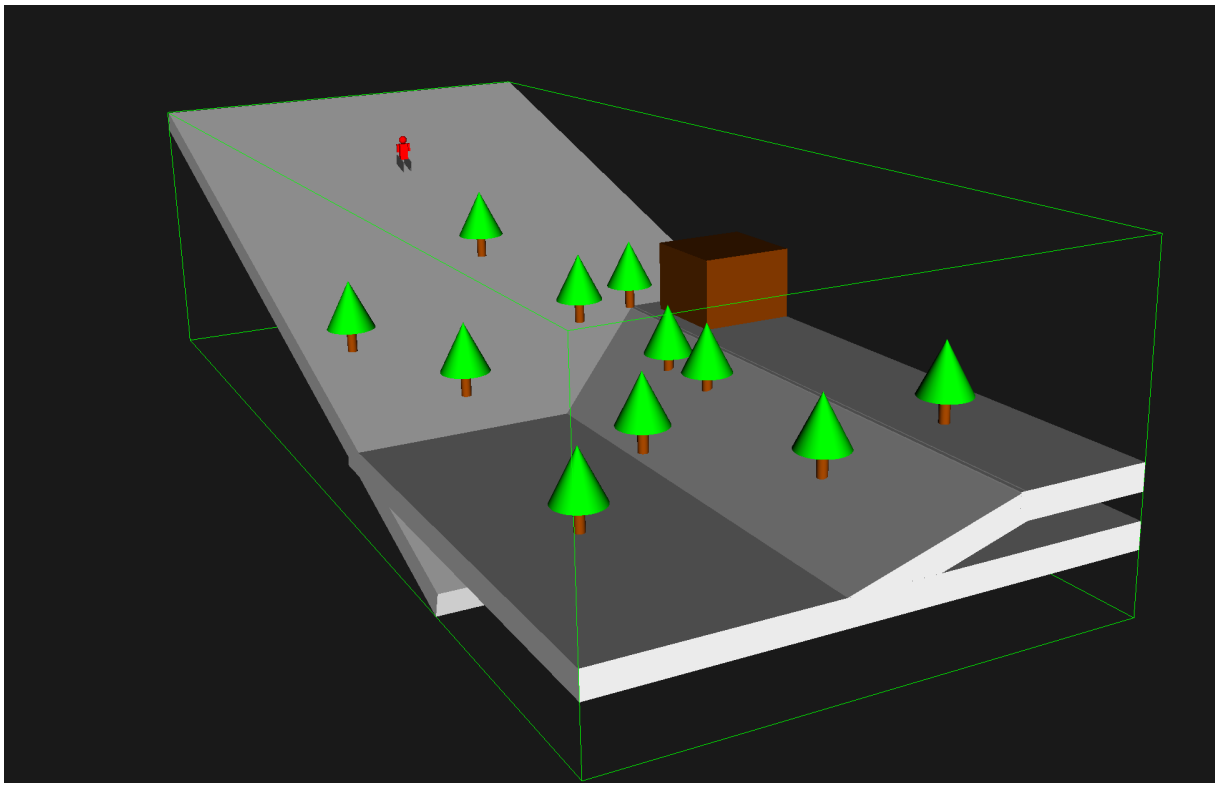


C.

Der Berghang wurde aus einer schrägen weißen Fläche (einer geneigten Box) und einer geraden Auslauffläche erstellt. Zehn Tannen wurden aus Cone- und Cylinder-Primitiven modelliert, und eine Berghütte wurde aus Box-Objekten zusammengesetzt.

Hauptmerkmale:

- Der Hang ist weiß, um Schnee darzustellen.
- Die Bäume bestehen aus einem braunen Zylinder (Stamm) und einem grünen Kegel (Baumkrone).
- Die Berghütte steht stabil auf der geraden Fläche.



d. Ich habe eine andere Datei namens **SkifahrerProto.wrl** erstellt, meinen Skier darin kopiert und anschließend die Datei **SkifahrerProto.wrl** in der Hauptdatei mit **EXTERNPROTO Skifahrer [ ] "SkifahrerProto.wrl"** aufgerufen.
Drei zusätzliche Skifahrer mit unterschiedlichen Farben wurden oben am Hang platziert.

```
#
#Berghütte mit Textur:
#
# Definiere die Berghütte (Box)
DEF Hut Transform {…
}

#
#Drei Skifahrer mit unterschiedlichen Farben
#
# Roter Skifahrer mit TouchSensor und Animation

DEF rot  Skifahrer {…


 DEF grun  Skifahrer {
           translation 2 0 0
           color 0 1 0   # Grün
}

DEF blau Skifahrer {
           translation -2 0 0
           color 0 0 1  # Blau
}

DEF blauu Skifahrer {
           translation -4 0 0
           color 0 2 1  # Blau
}



#
#View
#
#4.a Viewpoint für die Vogelperspektive
Viewpoint {…
}
#4.b Seitliche Ansicht vom Hang Links
Viewpoint {…
}
# Seitliche Ansicht vom Hang Rechts
```

```
#VRML V2.0 utf8
PROTO Skifahrer [
    field SFColor color 1 0 0  # Standardfarbe ist rot
    field SFVec3f translation 0 0 0
] {
    Transform {
        translation IS translation
        children [
            # Kopf des Skifahrers
            Transform {
                translation 0 4.6 -10
                children [
                    Shape {
                        appearance Appearance {
                            material Material {
                                diffuseColor 0.6 0.6 0.6
                            }
                        }
                        geometry Sphere {
                            radius 0.2
                        }
                    }

                ]
            }

            # Körper des Skifahrers
            Transform {
                translation 0 4 -10
                children [
                    Shape {
                        appearance Appearance {
                            material Material {
                                diffuseColor IS color
                            }
                        }
                        geometry Box {
                            size 0.4 0.8 0.2
                        }
                    }
                ]
            }

            # Linker Arm
            Transform {
```

V4:

a. *Ein Viewpoint wurde definiert, um den Berghang von oben zu betrachten. Dieser Viewpoint positioniert die Kamera hoch über dem Hang und richtet sie nach unten aus, sodass der gesamte Berghang sichtbar ist.*

```
# Viewpoint für die Vogelperspektive
Viewpoint {
    position 0 50 0   # Hoch über dem Zentrum des Hangs
    orientation 1 0 0 -1.57 # Die Kamera neigt nach unten (90 Grad in Bogenmaß ist ~-1.57)
    description "Vogelperspektive"
    fieldOfView 0.785  # Ein etwas weiteres Sichtfeld
}
```

b. Weitere sinnvolle Viewpoints:

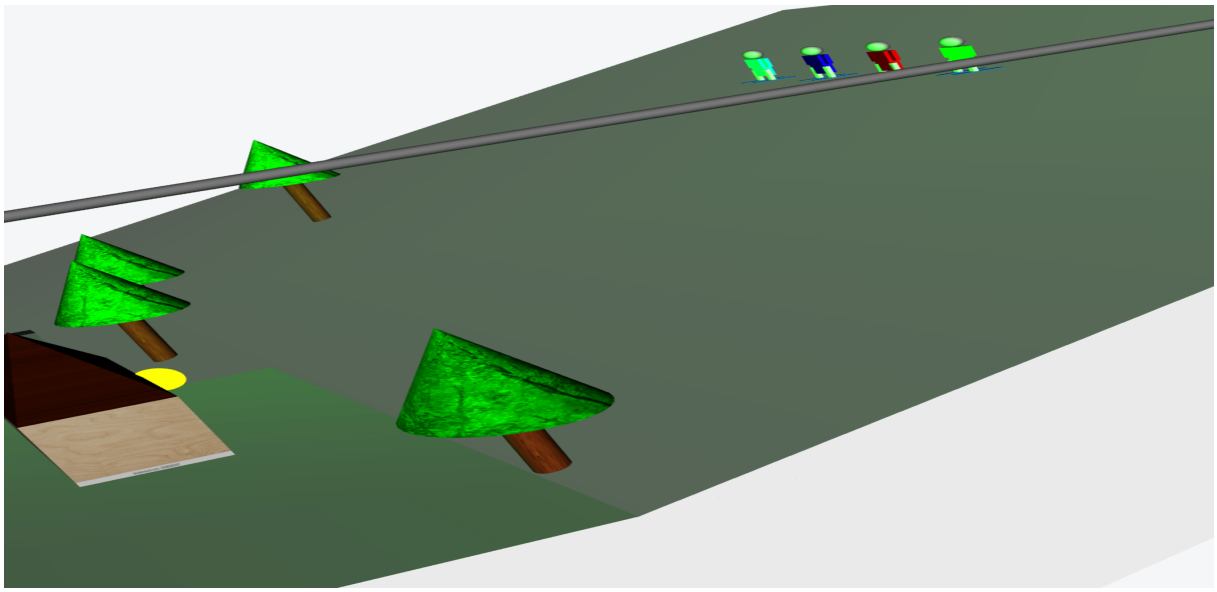
```
# Seitliche Ansicht vom Hang Links
Viewpoint {
    position -30 0 0  # Auf der linken Seite des Hangs
    orientation 0 1 0 -1.57  # 90 Grad Drehung zur rechten Seite, auf den Hang blickend
    description "Seitliche Ansicht vom Hang Links"
    fieldOfView 0.785
}


# Seitliche Ansicht vom Hang Rechts
Viewpoint {
    position 30 0 0  # Auf der rechten Seite des Hangs
    orientation 0 1 0 1.57  # 90 Grad Drehung, um zum Hang zu blicken
    description "Seitliche Ansicht vom Hang Rechts"
    fieldOfView 0.785
}

#  Ansicht von Vorne
Viewpoint {
    position 0 2 45  # In der Nähe von Bäumen
    orientation 0 1 0 0  # Blick auf den Wald
    description "Dichte Waldansicht"
    fieldOfView 0.785
}


# Ansicht von hinten
Viewpoint {
    position 0 12 -30  # Direkt hinter dem Kopf des Skifahrers
    orientation 0 1 0 3.14159  # Blick in die Z-Richtung (also ins Negativ, da im Minus)
    description "Ansicht von hinter dem Skifahrer"
    fieldOfView 0.785  # Weites Sichtfeld für eine vollständigere Perspektive
```

V5:

a.

In dieser Aufgabe wurde ein Scheinwerfer in der Nähe der Berghütte positioniert, um den Berghang mit grünem Licht zu bestrahlen. Zusätzlich wurde eine sichtbare gelbe Sphere hinzugefügt, um die Position der Lichtquelle zu kennzeichne

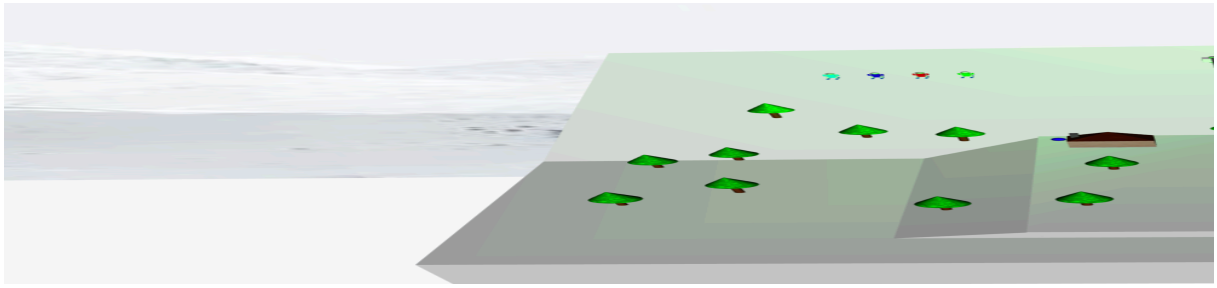
```
#Licht
#-----------------------------------------------
DEF Hut Transform {
    children [
        # Scheinwerfer
        DEF MySpotLight SpotLight {
            location 8 0 8.5  # Position des Scheinwerfers
            direction 0 0 -1  # Licht nach unten
            color 0 1 0  # Grünes Licht
            intensity 2  # Standard-Intensität (Licht an)
            cutOffAngle 1.5  # Strahlwinkel
            beamWidth 0  # Breite des Kernstrahls
        }

        # Sichtbare Kugel zur Darstellung der Lichtquelle
        Transform {
            translation 8 0 7  # Gleiche Position wie der Scheinwerfer
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor 1 1 0  # Gelb
                            emissiveColor 1 1 0  # Leuchtend
                        }
                    }
                    geometry Sphere { radius 0.3 }   # Kugel als visuelle Marki
                }
            ]
        },
```



b. Es wird ein halbtransparenter Lichtkegel erstellt, der die Szene mit einem Leuchteffekt visualisiert. Der Kegel ist in der Nähe der Berghütte positioniert, um eine Lichtquelle darzustellen.

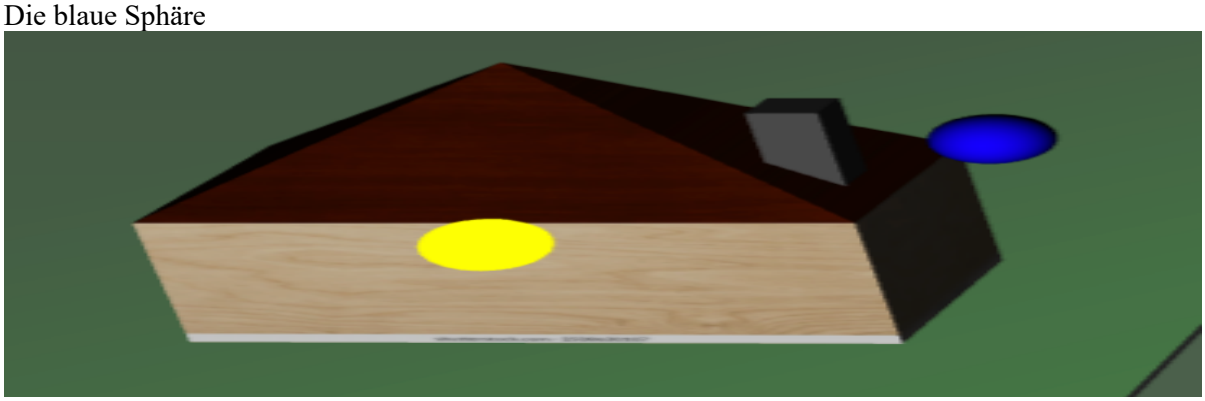
V6:

a. Um eine ansprechendere Darstellung zu erzielen, kann der Hintergrund der Szene angepasst werden. Eine Möglichkeit, dies zu tun, ist das Hinzufügen von Bildern.



```
#-----------------------------------------------------------------
#Background
#-----------------------------------------------------------------
Background {
    backUrl "bild1.jpg"    # Texture for the back side
    frontUrl "bild1.jpg"   # Texture for the front side
    leftUrl "bild1.jpg"        # Texture for the left side
    rightUrl "bild1.jpg"     # Texture for the right side
    topUrl "bild2.jpg"        # Texture for the top (sky)
    bottomUrl  "bild1.jpg"
}
```

b. Ein einfacher Ansatz ist, einen **TouchSensor** zu verwenden, der in VRML integriert ist und mit einem Skript gekoppelt wird, um die Lichtintensität des Scheinwerfers zu steuern.
Die blaue Sphäre



Scheinwerfer (SpotLight): Der Scheinwerfer ist ein SpotLight-Element, das eine Richtung, Intensität und Farbe hat. Es gibt auch einen cutOffAngle und beamWidth, die den Lichtkegel definieren.
Schalter (TouchSensor): Ein TouchSensor reagiert auf Klicks und sendet ein Signal an das Skript.
Skript (Script): Das Skript ändert die Intensität des Scheinwerfers basierend auf dem Schalterzustand.
Routen: Die Routen verbinden den Schalter mit dem Skript und das Skript mit dem Scheinwerfer, um die Intensität zu steuern.

```
        # Schalter-Kugel
        Transform {
            translation 6 0.5 8.5  # Position des Schalters
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor 0 0 1  # Blau
                        }
                    }
                    geometry Sphere { radius 0.3 }  # Schalter-Kugel
                }
                DEF SwitchSensor TouchSensor { }
            ]
        }
    ]

# Skript zum Umschalten des Lichts
DEF LightToggleScript Script {
    field SFBool isOn TRUE  # Initialzustand (Licht an)
    eventIn SFBool toggle  # Eingangssignal
    eventOut SFFloat intensityChanged  # Ausgangssignal für Intensität

    url "javascript:
        function toggle(isActive) {
            if (isActive) {  // Nur auf positive Zustandsänderung reagieren
                isOn = !isOn;  // Zustand umschalten
                intensityChanged = isOn ? 2.0 : 0.0;  // Licht ein/aus
            }
        }
    "

# Routen zur Verbindung
ROUTE SwitchSensor.isActive TO LightToggleScript.toggle
ROUTE LightToggleScript.intensityChanged TO MySpotLight.intensity
```

V7:

a. Animieren der Abfahrt des Skifahrers

- Der Skifahrer bewegt sich entlang von Kreis- und Geradenstücken.
- Gerade Abschnitte folgen der linearen Bewegung, und nach dem Erreichen eines Punktes wird der Skifahrer kurz warten, bevor er die Rückfahrt antritt.

```
# Positionsinterpolator
DEF SkierPosition PositionInterpolator {
    key [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
    keyValue [
        0 0 0,       # Startposition
        3 -1.2 2,
        0 -1.9 5,
        -3 -3.5 10,
        0 -4.5 15,
        3 -6.1 20,
        0 -6.1 25,
        0 -6.1 25,
        0 0 -20        # Zurück zur Startposition
    ]
}

# Orientierungsinterpolator
DEF SkierRotation OrientationInterpolator {
    key [0, 0.1, 0.2, 0.3, 0.4, 0.6, 0.7, 0.8, 1]
    keyValue [
        0 1 0 0,          # Startorientierung (geradeaus)
        0 1 0 0.785,      # Drehung (45° nach rechts)
        0 1 0 0,          # Geradeaus
        0 1 0 -0.785,     # Drehung (45° nach links)
        0 1 0 0,          # Geradeaus
        0 1 0 3.14159,    # Drehung um 180° vor Rückfahrt
        0 1 0 3.14159,    # Beibehaltung
    ]
}
```

b. Starten der Animation durch Klick

- Ein **TouchSensor** wird verwendet, um die Animation beim Klicken des Skifahrers zu starten.
- Der Sensor aktiviert den **PositionInterpolator**, der die Bewegung des Skifahrers steuert.

```
# TimeSensor für die Animation
DEF SkierTimer TimeSensor {
    cycleInterval 30   # 30 Sekunden pro Durchlauf
    loop FALSE         # Animation wird nicht automatisch wiederholt
}
```

c. Ausrichtung des Skifahrers

Der Skifahrer wird mithilfe eines **OrientationInterpolator** korrekt entlang der Fahrtrichtung ausgerichtet, sowohl auf dem Hang als auch auf der flachen Strecke.

```
# Verbindungen für die Animation und Interaktionen
ROUTE SkierTouchSensor.touchTime TO SkierTimer.set_startTime
ROUTE SkierTimer.fraction_changed TO SkierPosition.set_fraction
ROUTE SkierTimer.fraction_changed TO SkierRotation.set_fraction
ROUTE SkierPosition.value_changed TO rotSkifahrer.set_translation  # Versch
ROUTE SkierRotation.value_changed TO rotSkifahrer.set_rotation
```

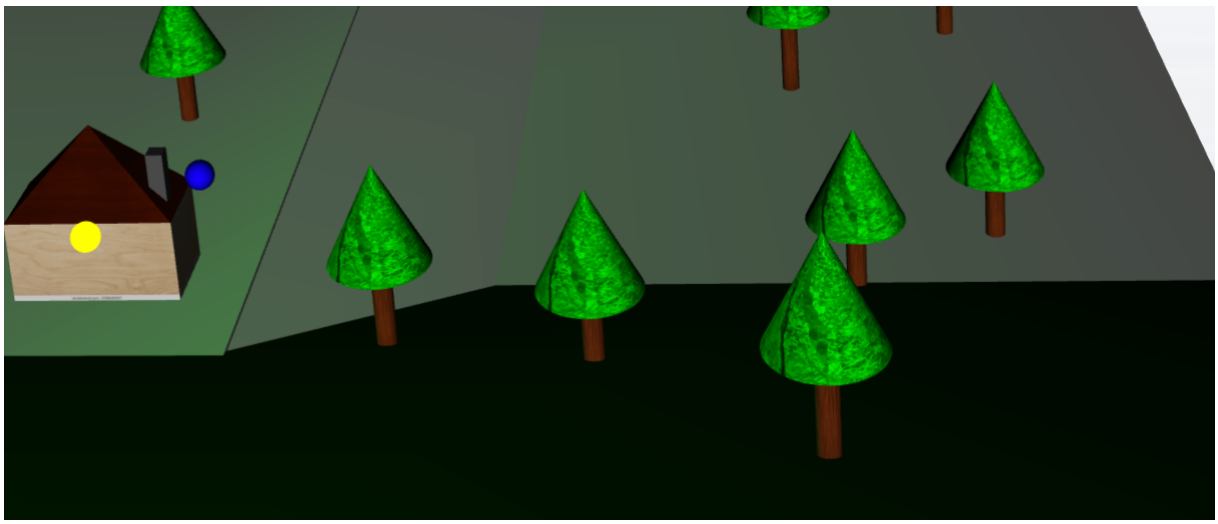
V7:

a. Texturierung der Berghütte und der Bäume

Texturierung ist eine Technik in der 3D-Grafik, um Oberflächen realistischer zu gestalten, ohne die Anzahl der Polygone zu erhöhen. In dieser Aufgabe sollen die Berghütte und Bäume texturiert werden.

```
DEF Hut Transform {
    translation 8 -1.5 8
    scale 1.7 1.7 1.7
    children [
        # Basis der Hütte mit Holztextur
        Transform {
            translation 0 0.5 0
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor 0.8 0.8 0.8  # Helle Grundfarbe, fall
                        }
                        texture ImageTexture {
                            url "haus.webp"  # Holztextur-Datei
                        }
                    }
                    geometry Box {
                        size 2 1 1  # Breite, Höhe, Tiefe
                    }
                }
            ]
        }

        # Dach der Hütte (optional mit Textur)
        Transform {
            translation 0 1.5 0  # Position über der Basis
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            diffuseColor 0.8 0.3 0  # Rote Grundfarbe für das
                        }
                        texture ImageTexture {
                            url "dach.webp"  # Optional: Dachtextur
                        }
                    }
                    geometry IndexedFaceSet {
                        coord Coordinate {
                            point [
                                -1 -0.5 -0.5,  1 -0.5 -0.5,  1 -0.5 0.5,  -1 -
                                0 0.5 0  # Dachspitze
                            ]
                        }
```

```
# Definiere den Baumstamm (Zylinder) mit Textur
DEF Tree_Stamm Transform {
    translation -6 1 0  # Position des Baumstamms
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 0.6 0.3 0.1  # Braune Farbe für den Stamm
                }
                texture ImageTexture {
                    url "holz.jpeg"  # Ersetze mit der tatsächlichen Textur-Dat
                }
            }
            geometry Cylinder {
                height 2.5  # Höhe des Baumstamms
                radius 0.2  # Radius des Baumstamms
            }
        }
    ]
}

# Definiere die Baumkrone (Kegel) mit Textur
DEF Tree_Krone Transform {
    translation -6 3 0  # Position der Baumkrone
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 0 1 0  # Grüne Farbe für die Blätter
                }
                texture ImageTexture {
                    url "korne.jpeg"  # Ersetze mit der tatsächlichen Textur-Da
                }
            }
            geometry Cone {
                height 2  # Höhe der Baumkrone
                bottomRadius 1  # Radius der Basis der Baumkrone
            }
        }
    ]
}
```



V.9

Sound:

In VRML kann Sound durch das Sound-Objekt hinzugefügt werden. Hier wird ein kontinuierlicher Sound für die Luftseilbahn implementiert, der durch eine AudioClip-Quelle wiedergegeben wird.

- **Parameter:**
  - location: Position des Sounds im Raum (z.B., -12 5 5).
  - maxBack und maxFront: Definieren die Hörreichweite hinter und vor der Quelle (z.B., 20).
  - minBack und minFront: Legen die minimalen Abstände zum Sound fest, innerhalb dessen er gehört wird (z.B., 5).
  - source AudioClip: Gibt die Quelle des Sounds an. Hier wird eine .mp3-Datei (sound.mp3) verwendet und der Sound ist auf "loop" eingestellt, sodass er sich kontinuierlich wiederholt.

```
# 3D-Sound hinzufügen
#-----------------------------------------------------------------
Sound {
    location -12 5 5  # Position des Sounds
     maxBack 20       # Hörreichweite nach hinten
     maxFront 20      # Hörreichweite nach vorne
     minBack 5        # Minimaler Abstand
     minFront 5       # Minimaler Abstand
     source AudioClip {
         url "sound.mp3"  # Sounddatei für den Skilift
         loop TRUE             # Loop für kontinuierlichen Sound
     }
}
```

Luftseilbahn:

Die Luftseilbahn wird durch mehrere VRML-Transformations- und Geometrieobjekte dargestellt. Sie besteht aus mehreren Komponenten:

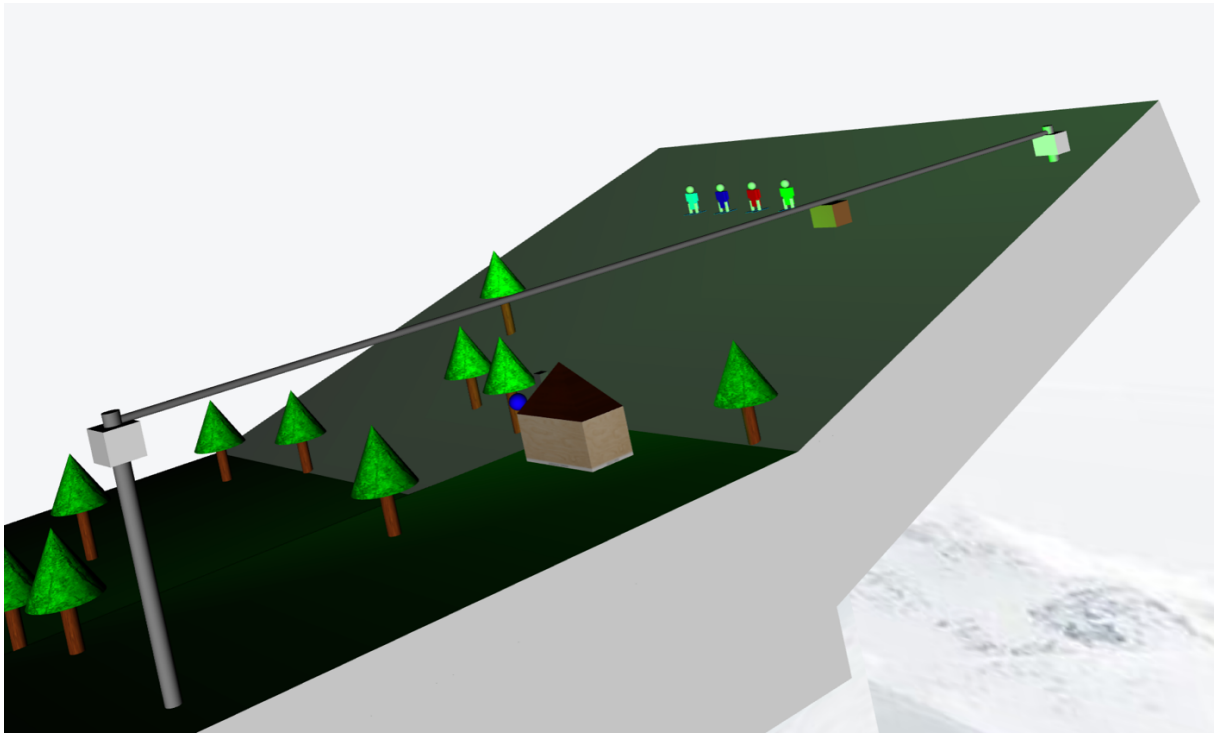
_**Basis und Zimmer**: Die Basis und die Zimmer (Stationen) sind durch Transform-Objekte dargestellt. Diese enthalten Zylinder (für die Stützen) und Boxen (für die Zimmer).

_**Kabel**: Ein Zylinder wird verwendet, um das Kabel der Seilbahn darzustellen.

_**Lift-Gondel**: Die Gondel selbst wird mit einer Box-Geometrie realisiert und ist auf einer B-Spline-Kurve (definiert durch PositionInterpolator) bewegt.

_**B-Spline-Kurve**: Die Gondel folgt einer B-Spline-Kurve, die die Position der Gondel während ihrer Fahrt entlang der Strecke definiert.

_**Zeitschaltuhr**: Ein TimeSensor wird verwendet, um die Zeit für die Bewegung der Gondel zu steuern, mit einer Schleife von 20 Sekunden pro Durchlauf.



Quellen:

- . Vorlesung **Computergrafik**
- "VRML 2.0: A Guide for the Perplexed" von Stephen G. Johnson
- **ChatGPT**
- **YouTube**
- Stack Overflow