3D Dungeon Crawler

Anfängerpraktikum Computergrafik

Stefan Mladenov & Mangkonthong Virasith

25.11.24

Inhaltsverzeichnis

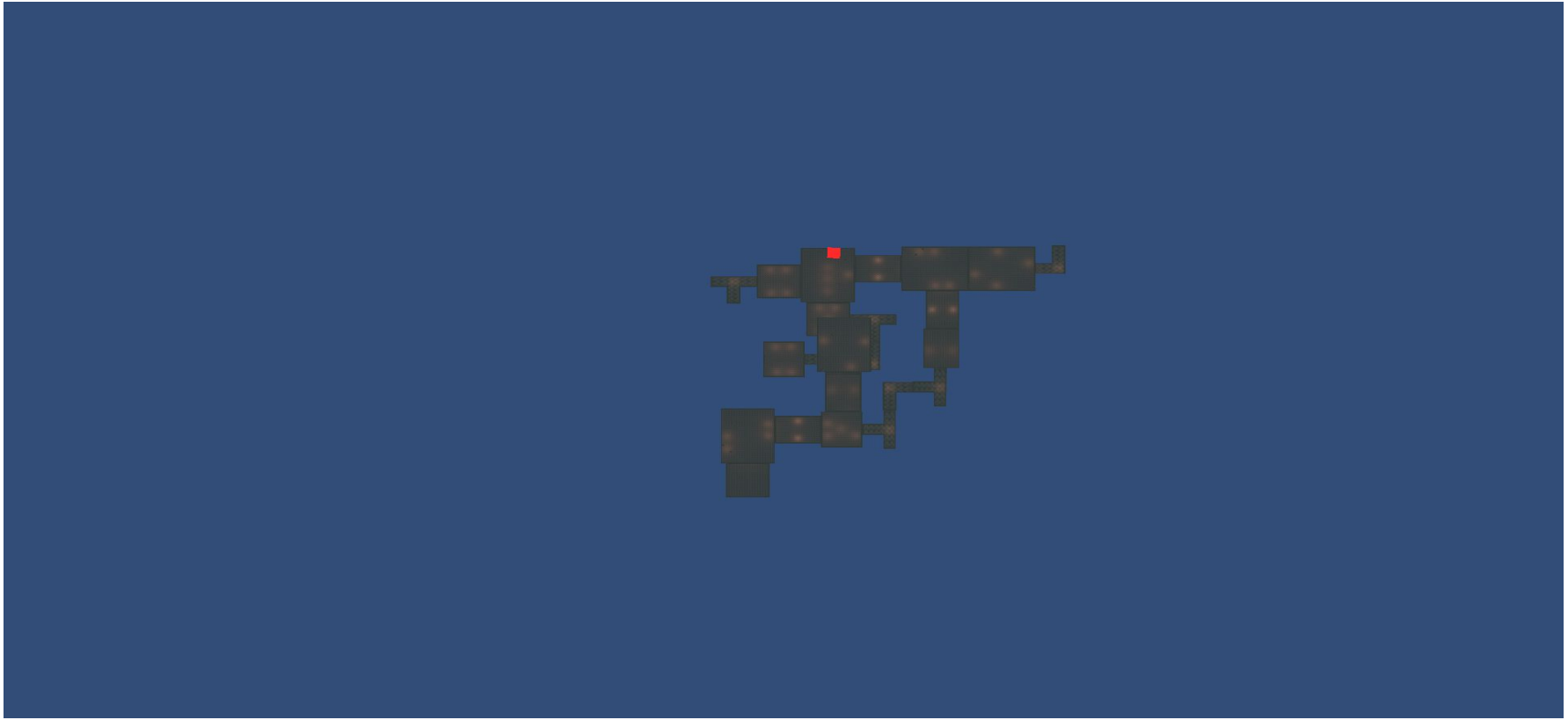
Dungeon Crawler

- Fantasy Rollenspiel
- Labyrinth Umgebung
- Kerker
- Besiegen von Gegnern
- Endlose Level

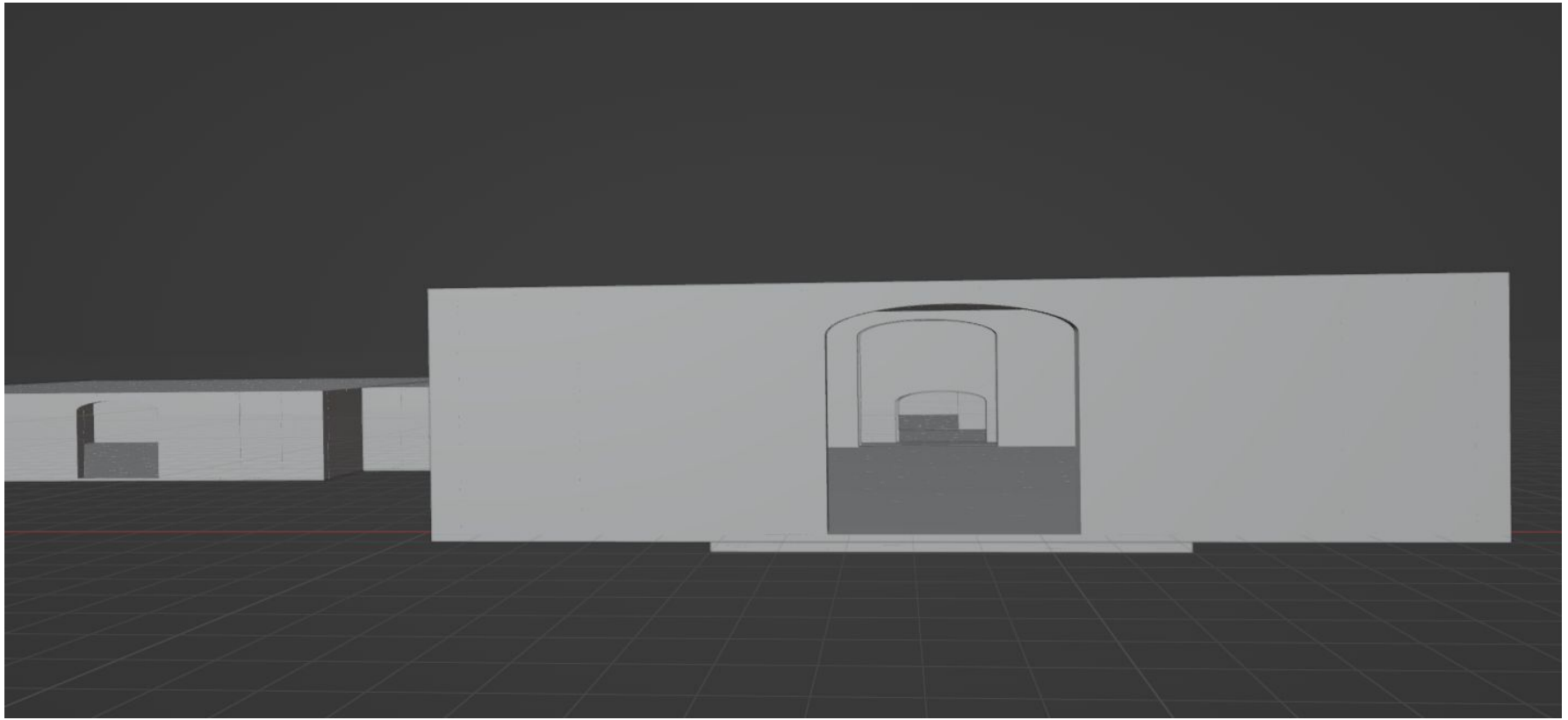
Räume

- In einem Stil gehalten
- Abwechslung
- Start und Endraum

Map



Räume



Thronsaal

Schmiede

Startraum


Move - WASD
Jump - Space
Sprint - Shift
Duck - CTRL
Inventory - E
Interact - F
Attack - Left Mouse Button
Map - M
Pause- Escape
1
3
2
3
4
6
5
6
7
6
8
9
0
5
1346 exp / 1605 exp

Bossraum

Bossraum

Gegenstände

- Rüstung
- Waffen
- Tränke
- Dekoration für Räume
- Inventar als Sammlung
- Verbessern Werte des Spielers

Inventar


Twilight Dagger
Daggers
80 Base Damage
Lifesteal: Give 10% of the damage dealt as healthpoints to the wielder.
Critical strike: 40% chance to multiply damage by 3,5 times.
190
144
47
39
SPD
10
DEF
17
10
2784 exp / 2855 exp

Gegner

- Skalieren mit Level
- Greifen an sobald Spieler in Sicht
- Verschiedene Arten
- Alle 5 Level Boss

Gegner

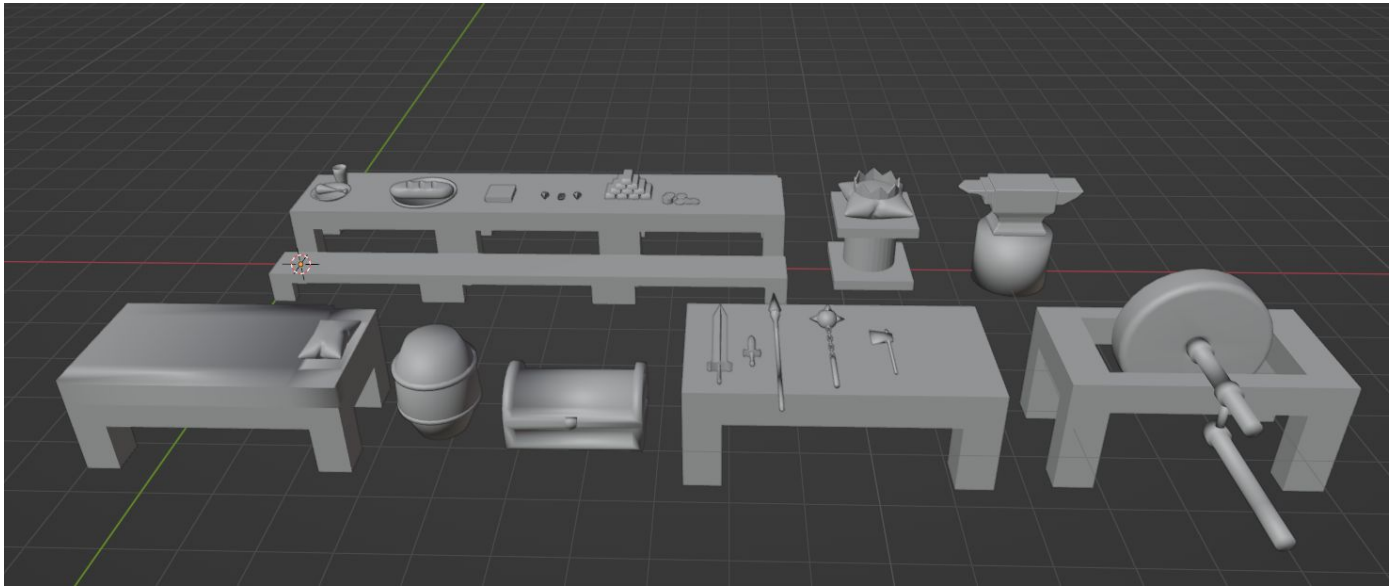
Gegner

Blender

- Modellierung
- Texturen
- Flat und Smooth Shading
- Backface Culling
- UV – Editing
- Export

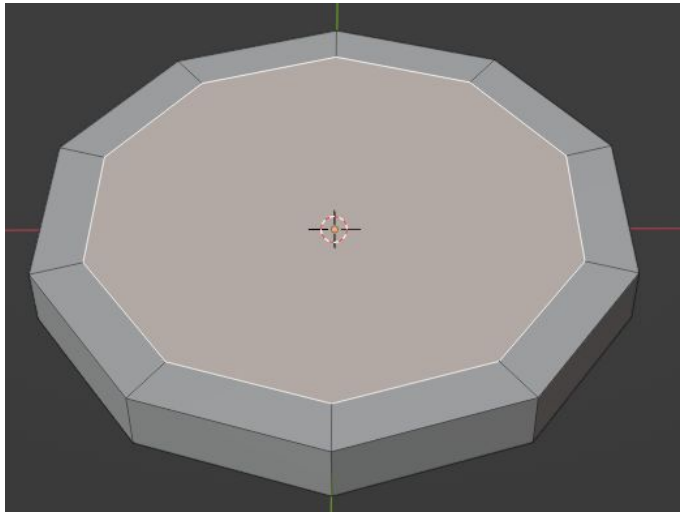
Modellierung

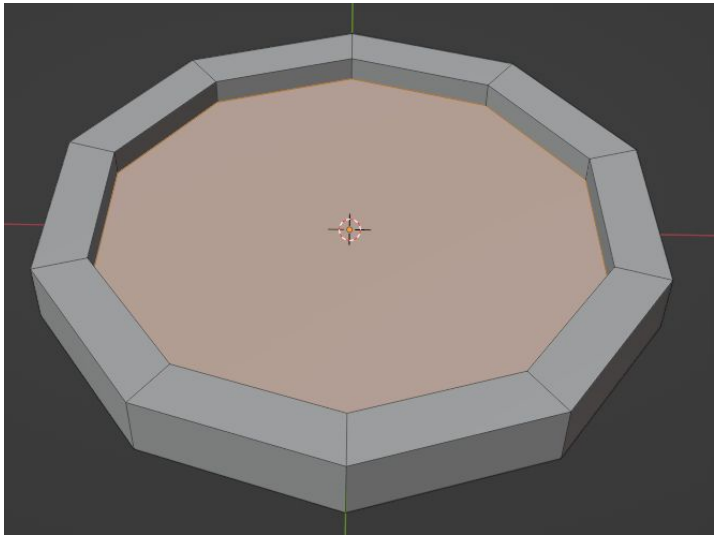
- Referenzen
- Zusammenfügen einzelner Modelle
- Modifikatoren



Modellierung

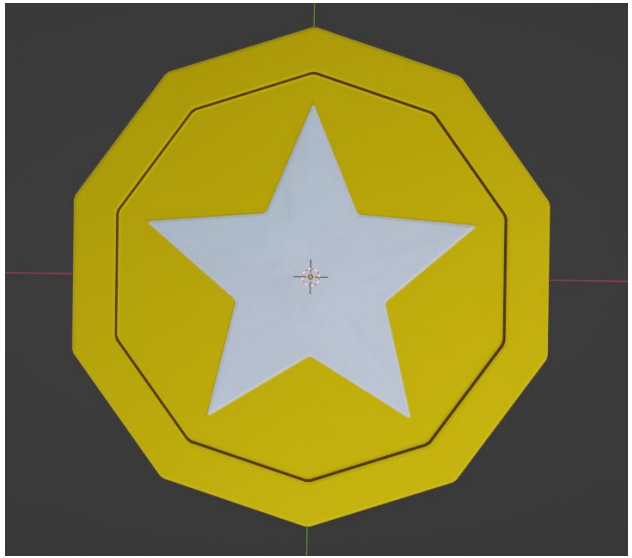
- Fläche gleichmäßig verkleinern mit inset
- Fläche eindrücken/herausziehen mit extrude along normals

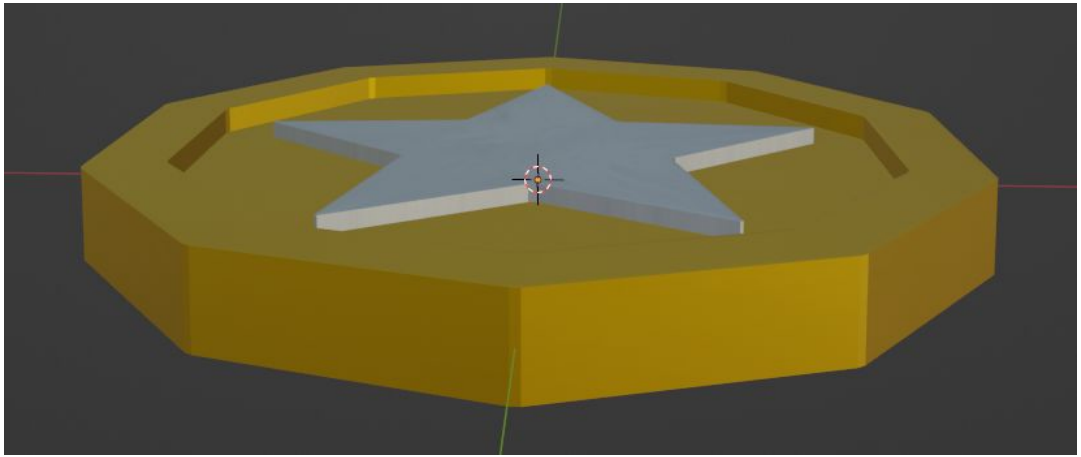




Modellierung

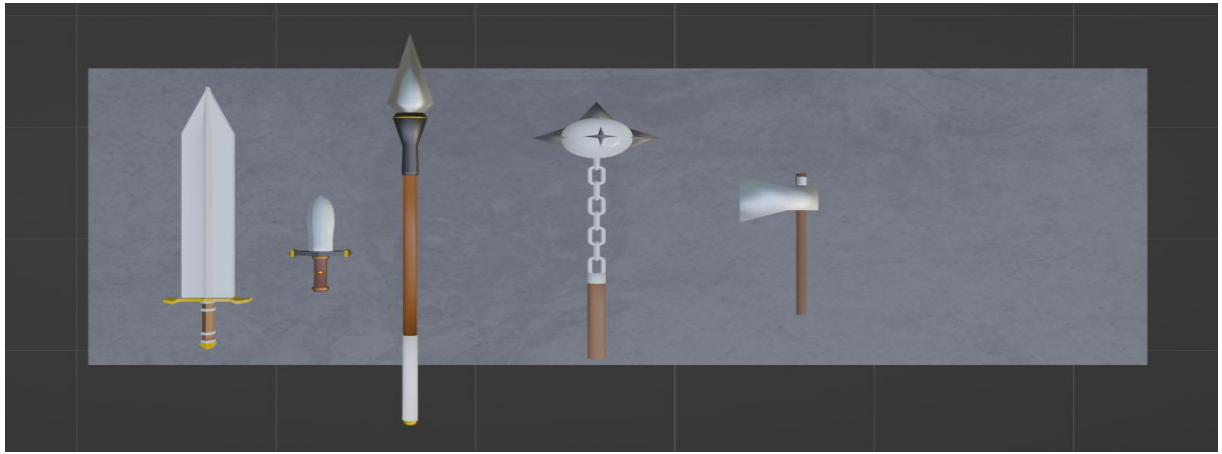
- Fläche gleichmäßig verkleinern mit inset
- Fläche eindrücken/herausziehen mit extrude along normals





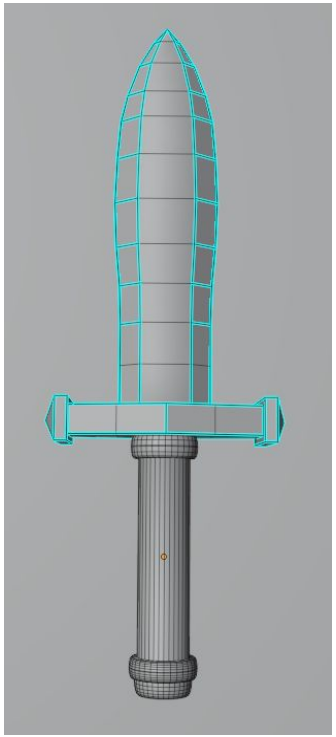
Texturen

- Bilder
- Projektion auf 3D Oberflächen
- Eigenschaften des Materials
- Color Map
- Metall Map
- Normal Map

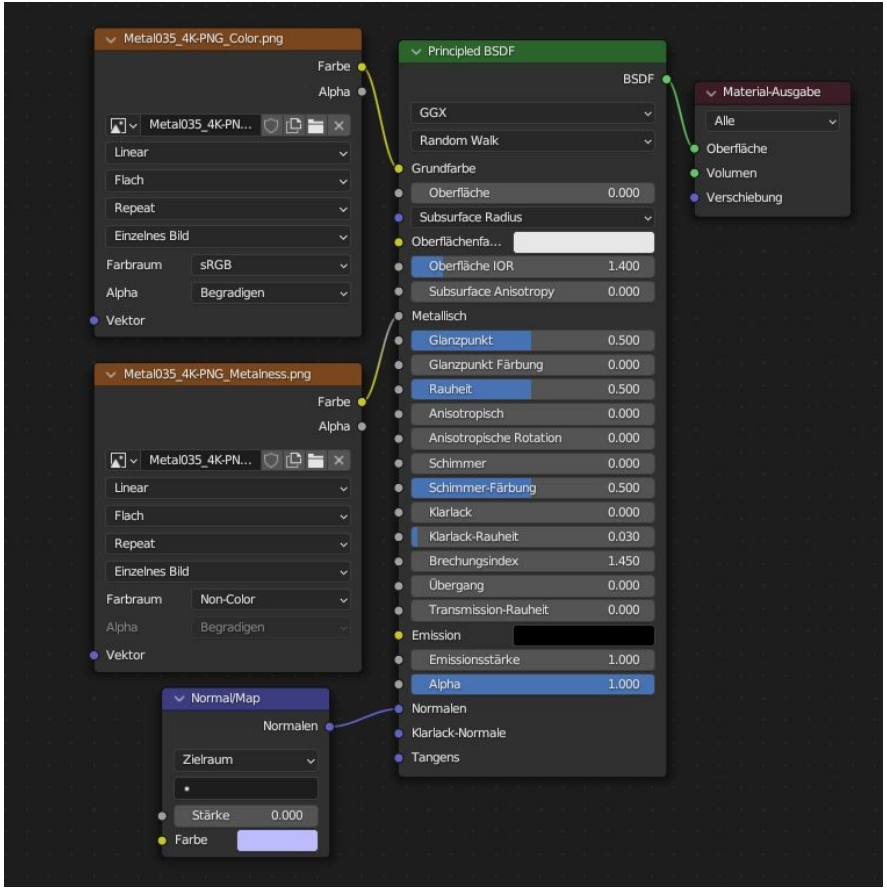


Texturen

- Jede Fläche kann ein Material haben



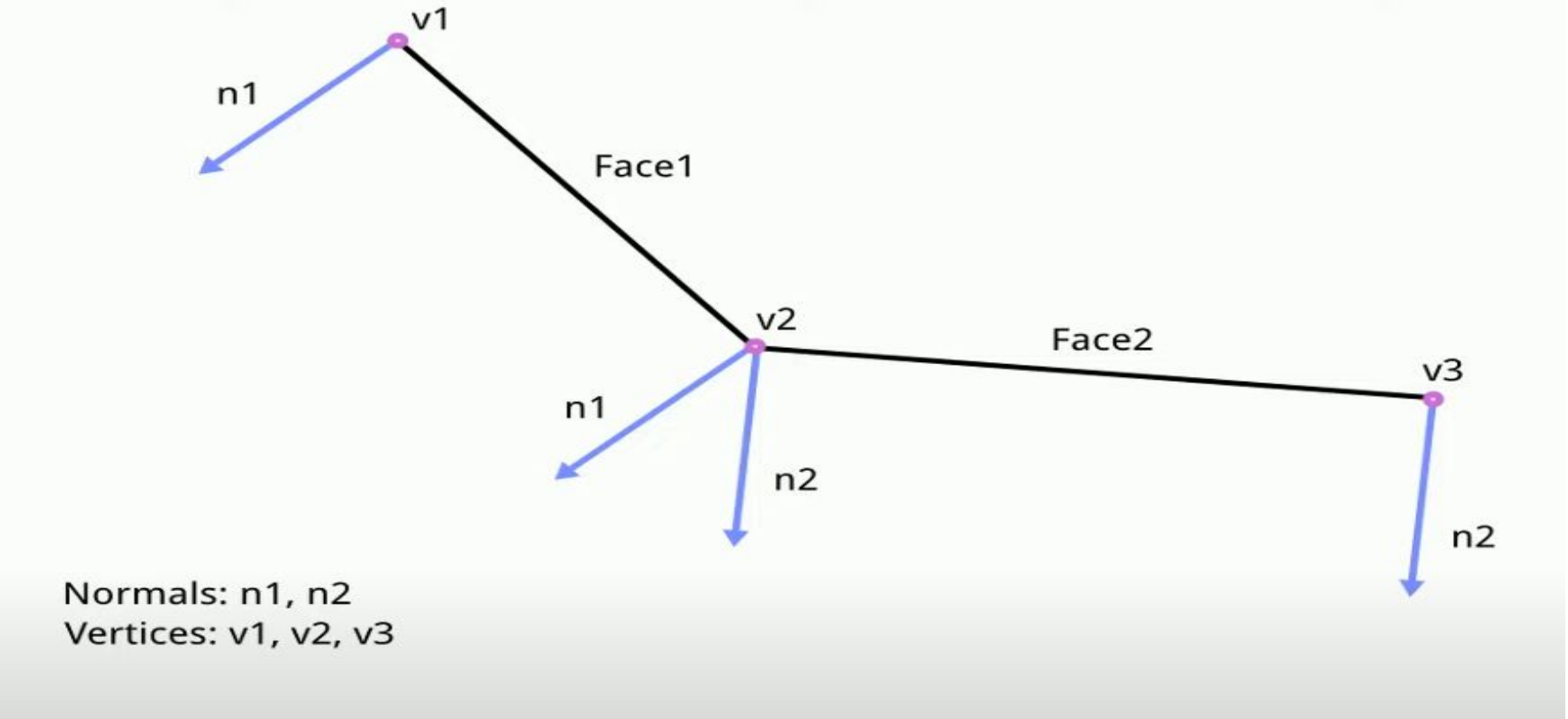
Texturen


Metal035_4K-PNG_Color.png
Farbe
Alpha
Metal035_4K-PN...
Linear
Flach
Repeat
Einzelnes Bild
Farbraum
sRGB
Alpha
Begradigen
Vektor
Metal035_4K-PNG_Metalness.png
Farbe
Alpha
Metal035_4K-PN...
Linear
Flach
Repeat
Einzelnes Bild
Farbraum
Non-Color
Alpha
Begradigen
Vektor
Normal/Map
Normalen
Zielraum
Stärke
0.000
Farbe
Principled BSDF
BSDF
GGX
Random Walk
Grundfarbe
Oberfläche
0.000
Subsurface Radius
Oberflächenfa...
Oberfläche IOR
1.400
Subsurface Anisotropy
0.000
Metallisch
Glanzpunkt
0.500
Glanzpunkt Färbung
0.000
Rauheit
0.500
Anisotropisch
0.000
Anisotropische Rotation
0.000
Schimmer
0.000
Schimmer-Färbung
0.500
Klarlack
0.000
Klarlack-Rauheit
0.030
Brechungsindex
1.450
Übergang
0.000
Transmission-Rauheit
0.000
Emission
Emissionsstärke
1.000
Alpha
1.000
Normalen
Klarlack-Normale
Tangens
Material-Ausgabe
Alle
Oberfläche
Volumen
Verschiebung

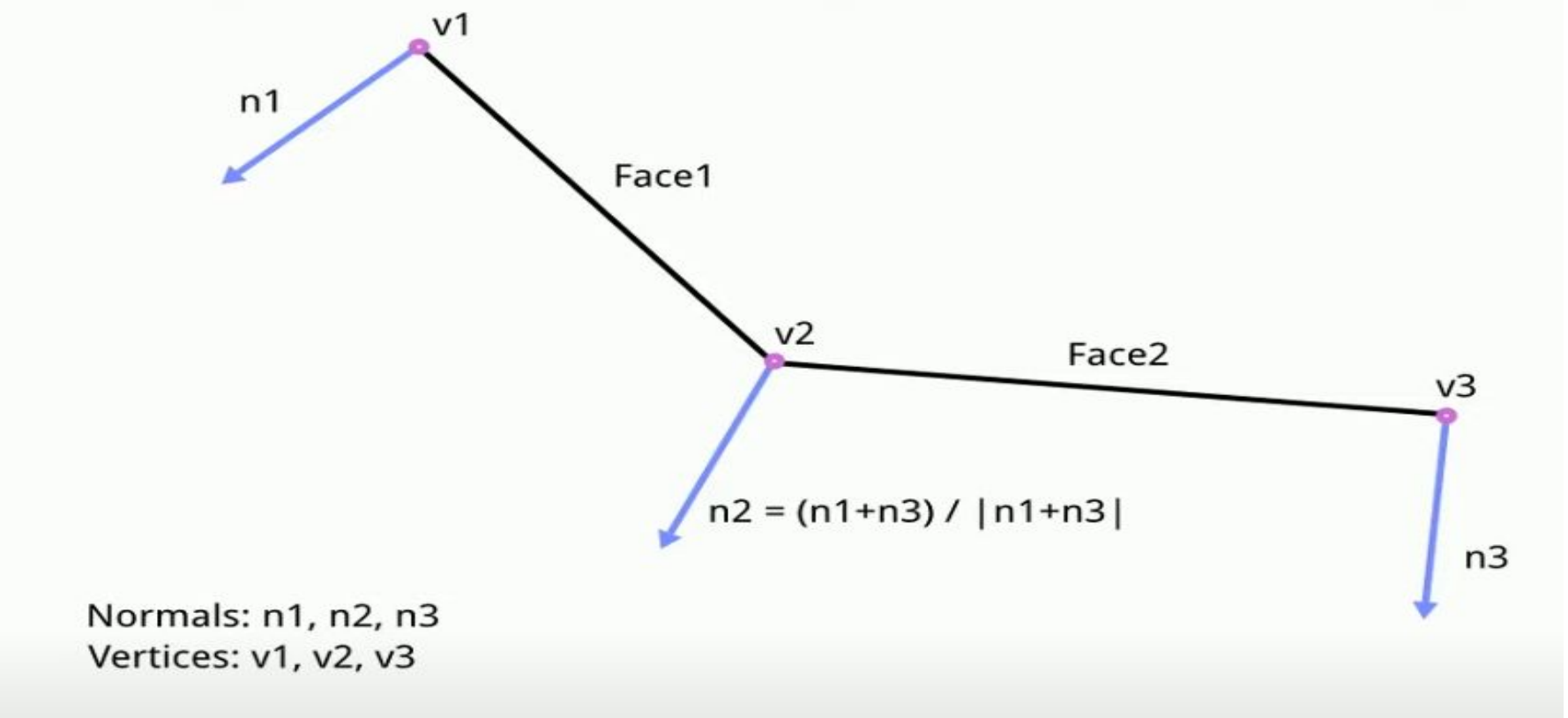
Flat und Smooth Shading

- Flat:
  ⇒ Jede Fläche eigenen Normalenvektor
- Smooth:
  ⇒ Mittelwert angrenzender Flächen als Normalenvektor für Eckpunkt

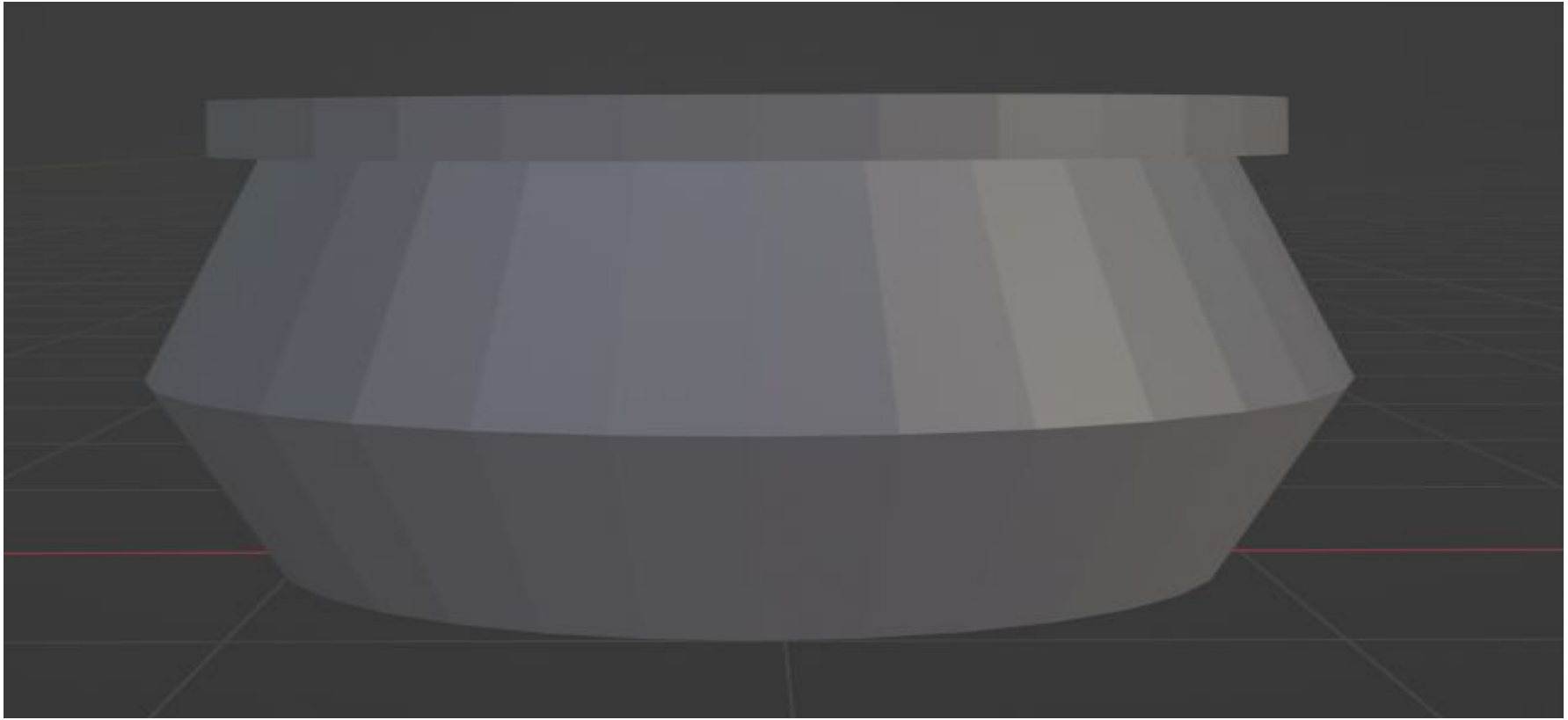
Flat Shading



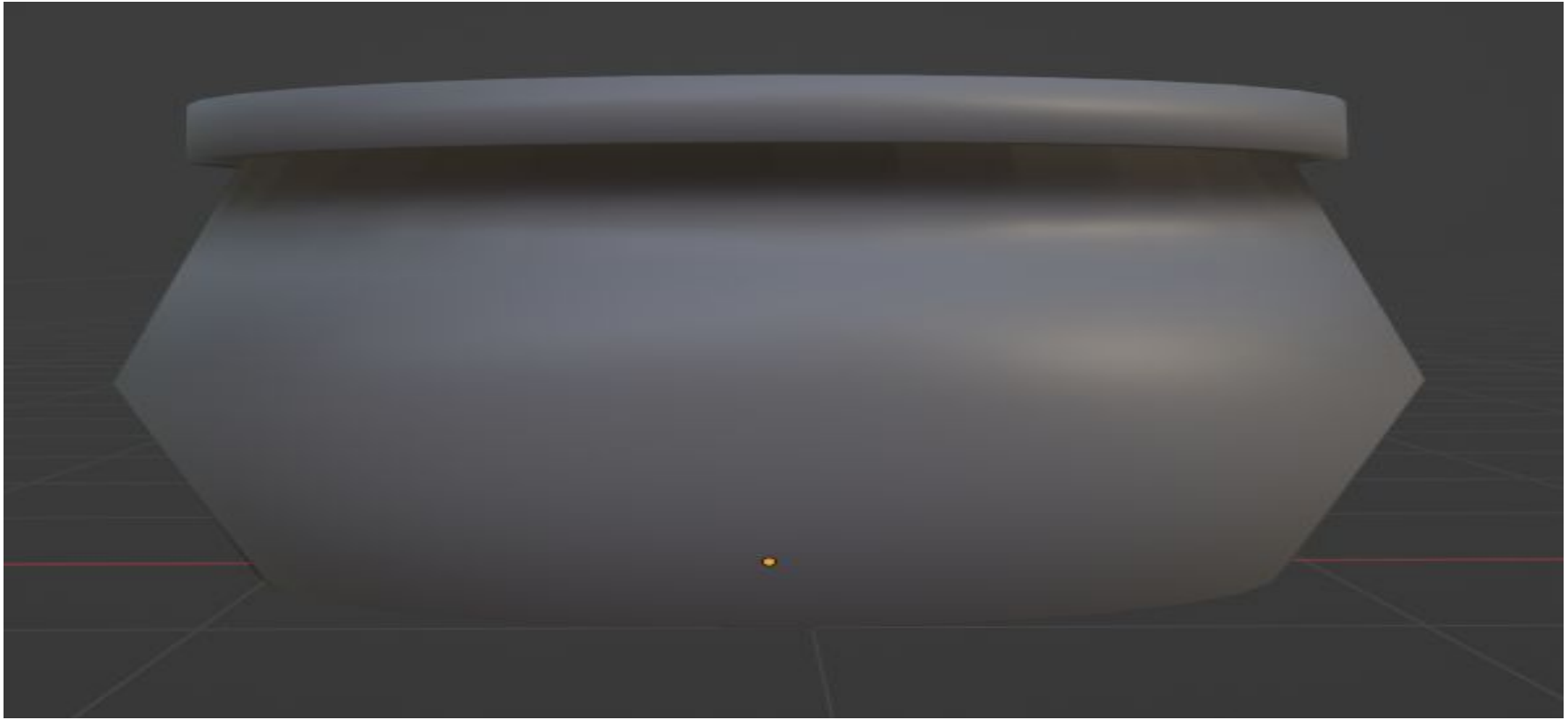
Smooth Shading



Flat Shading

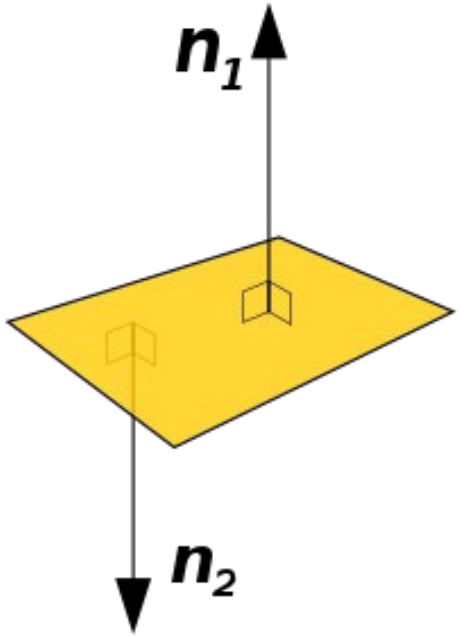


Smooth Shading

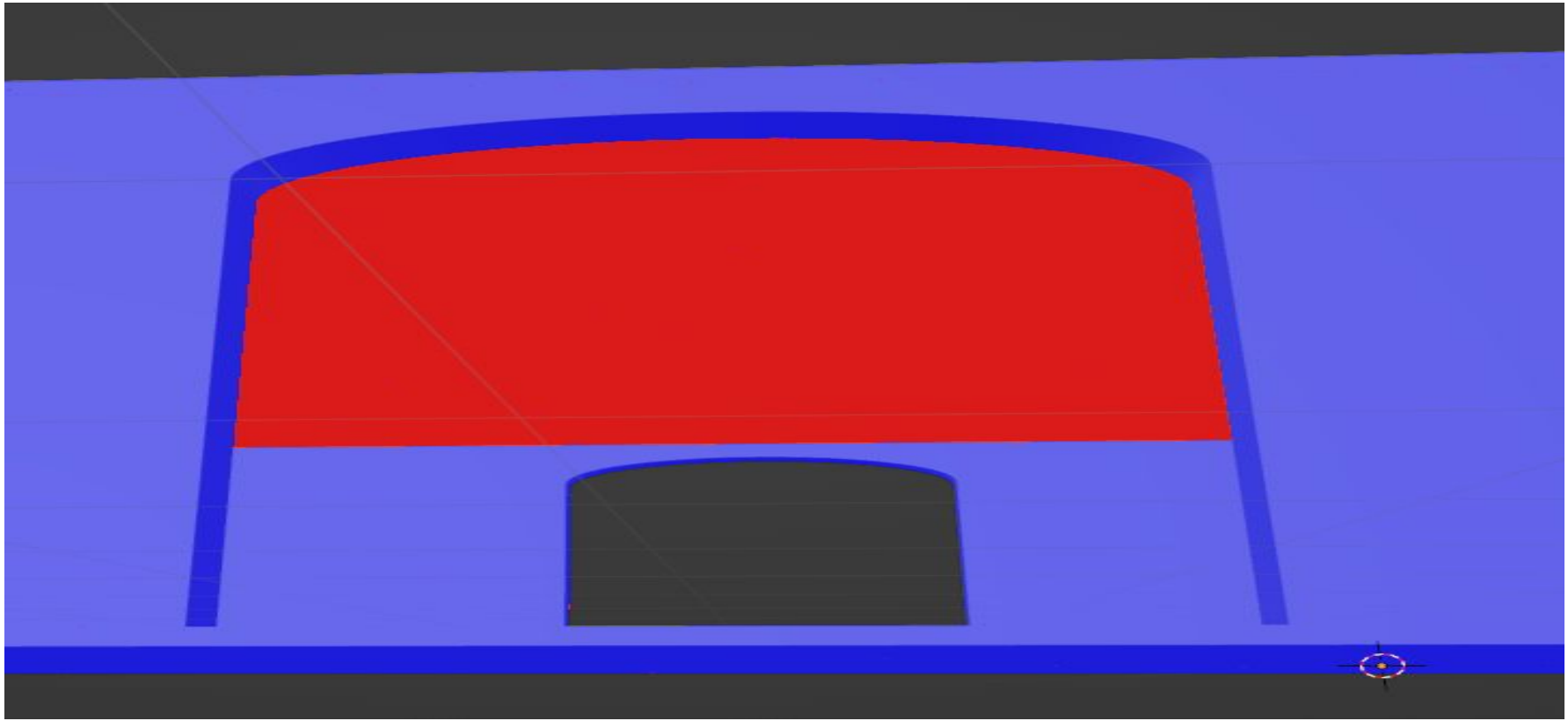


Backface Culling

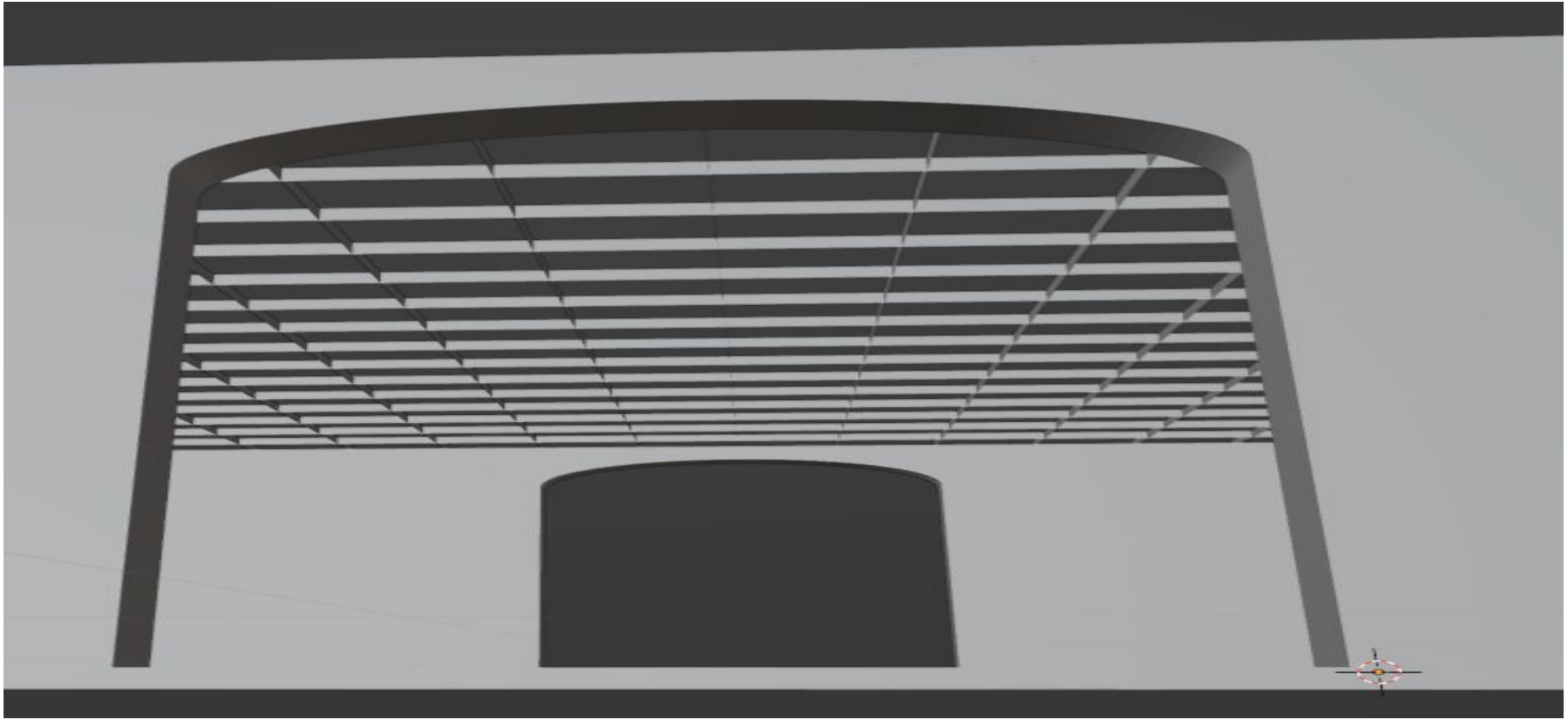
- Entfernt nicht sichtbare Polygone
- Normalenvektor
- Flächenausrichtung



Flächenausrichtung

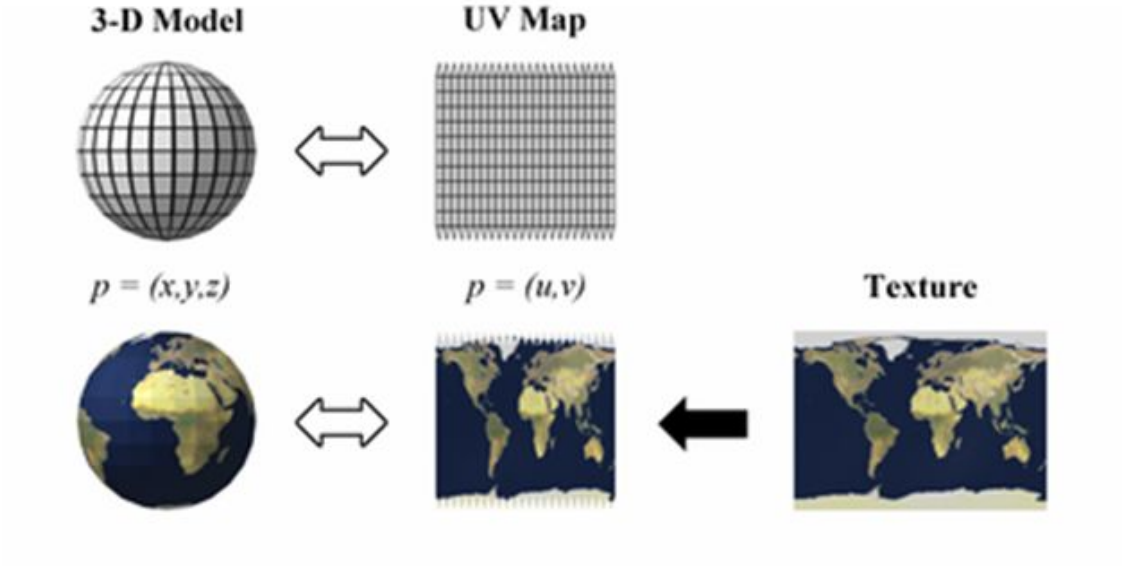


Backface Culling

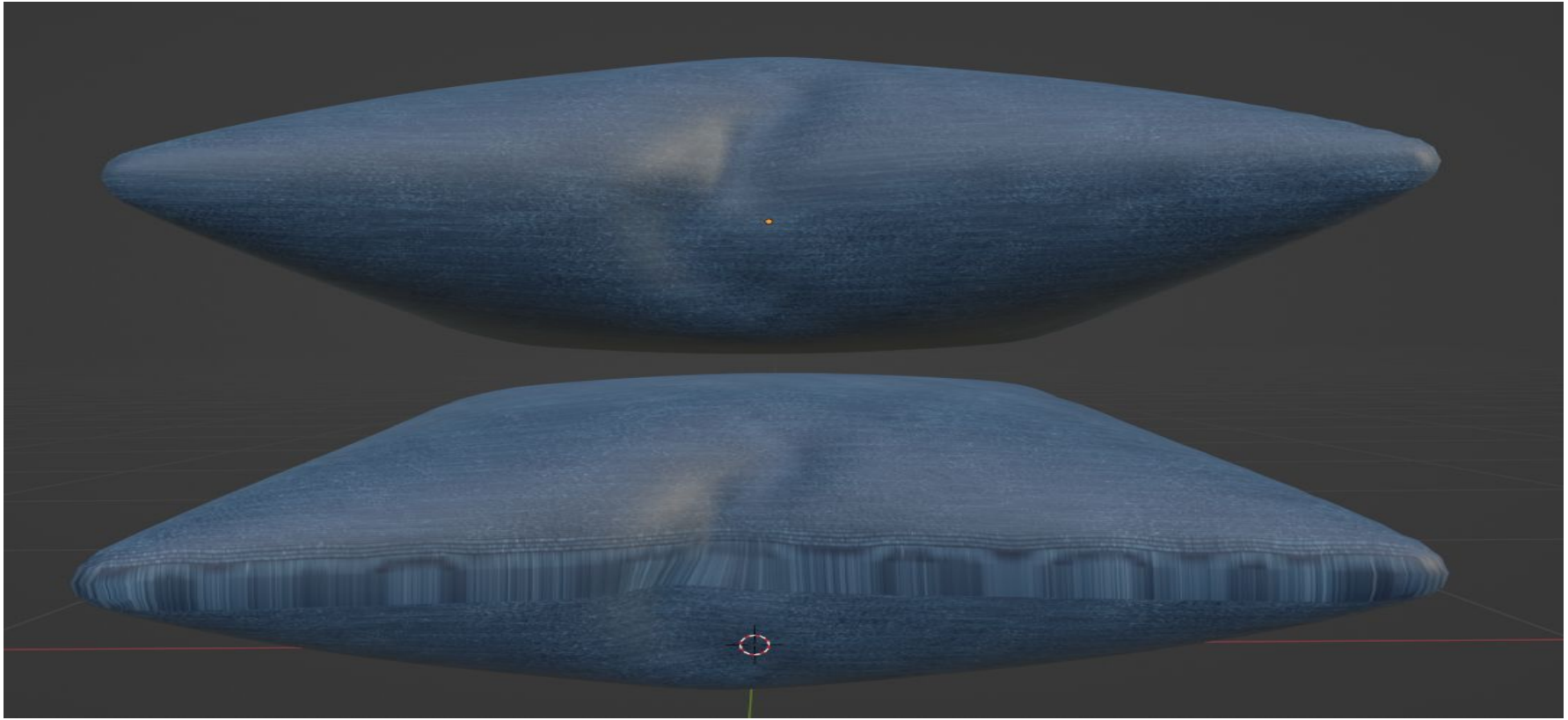


UV Editing

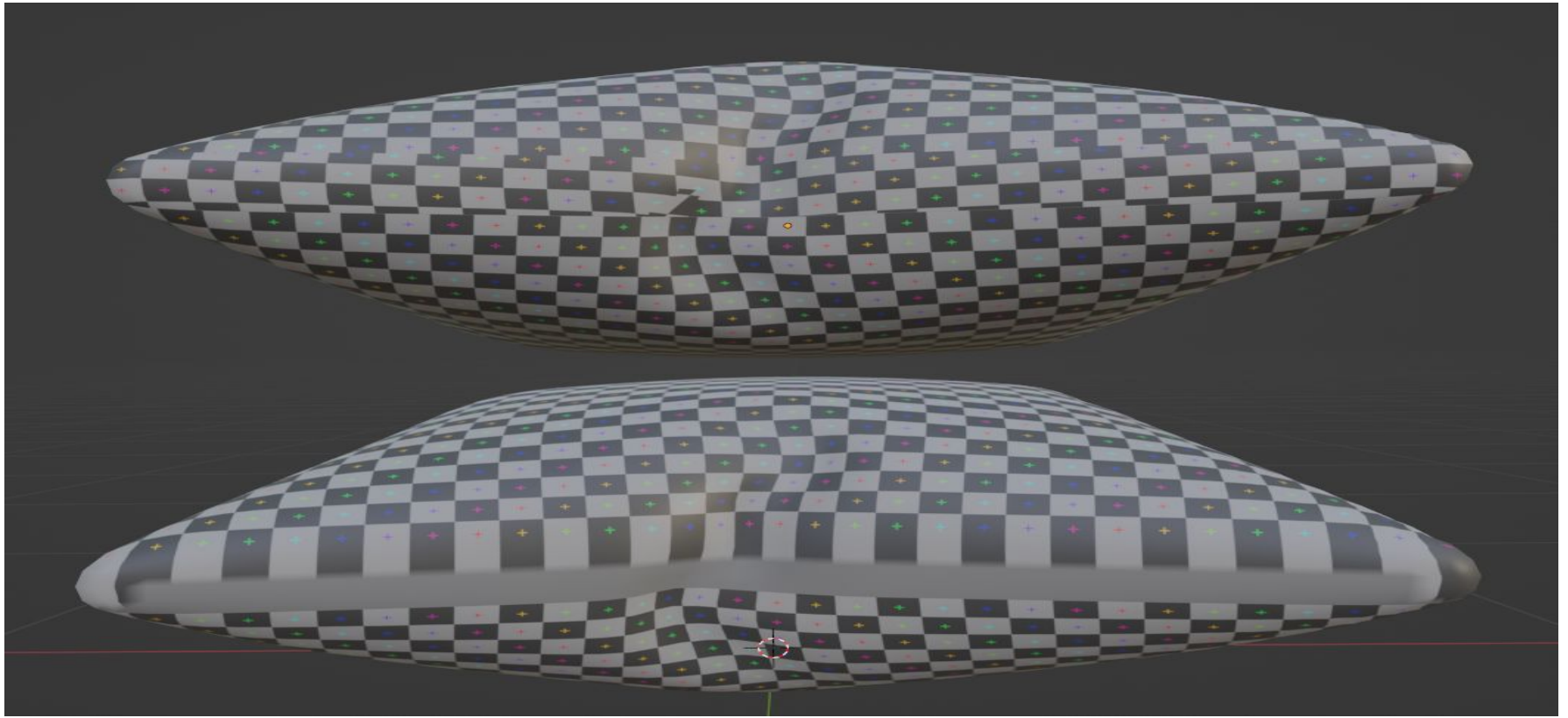
- Polygone von 3D Modell mit Bild texturieren
- Texturkoordinaten
- Jede Fläche eindeutige Texturposition



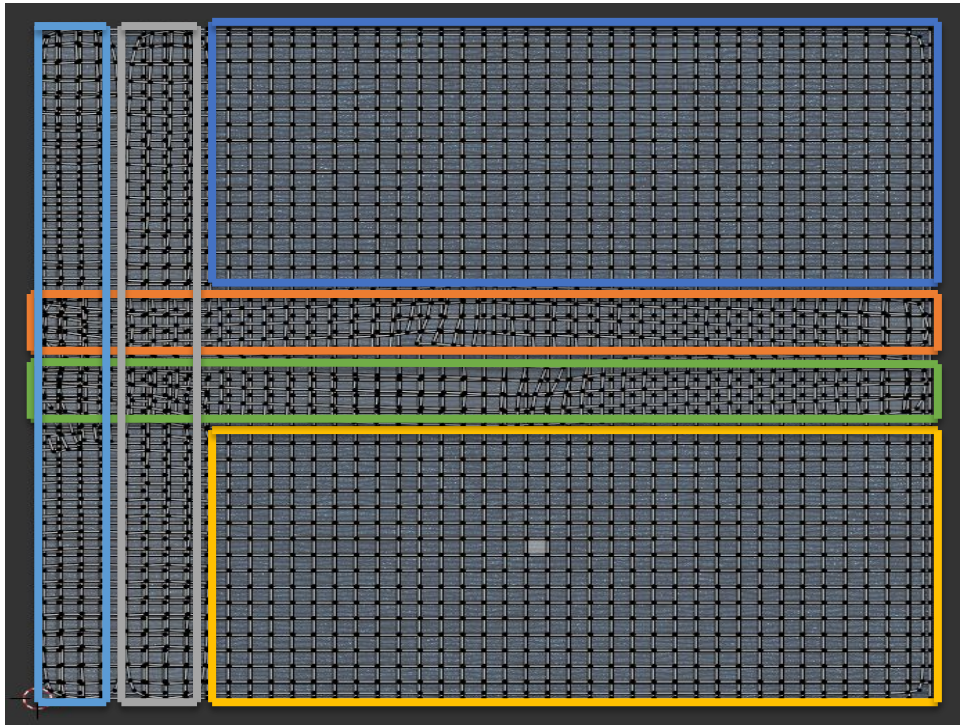
UV Editing

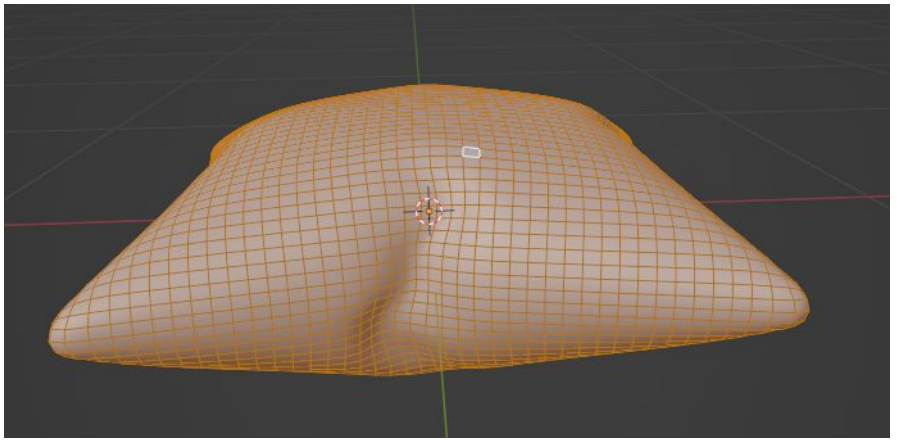


UV Editing



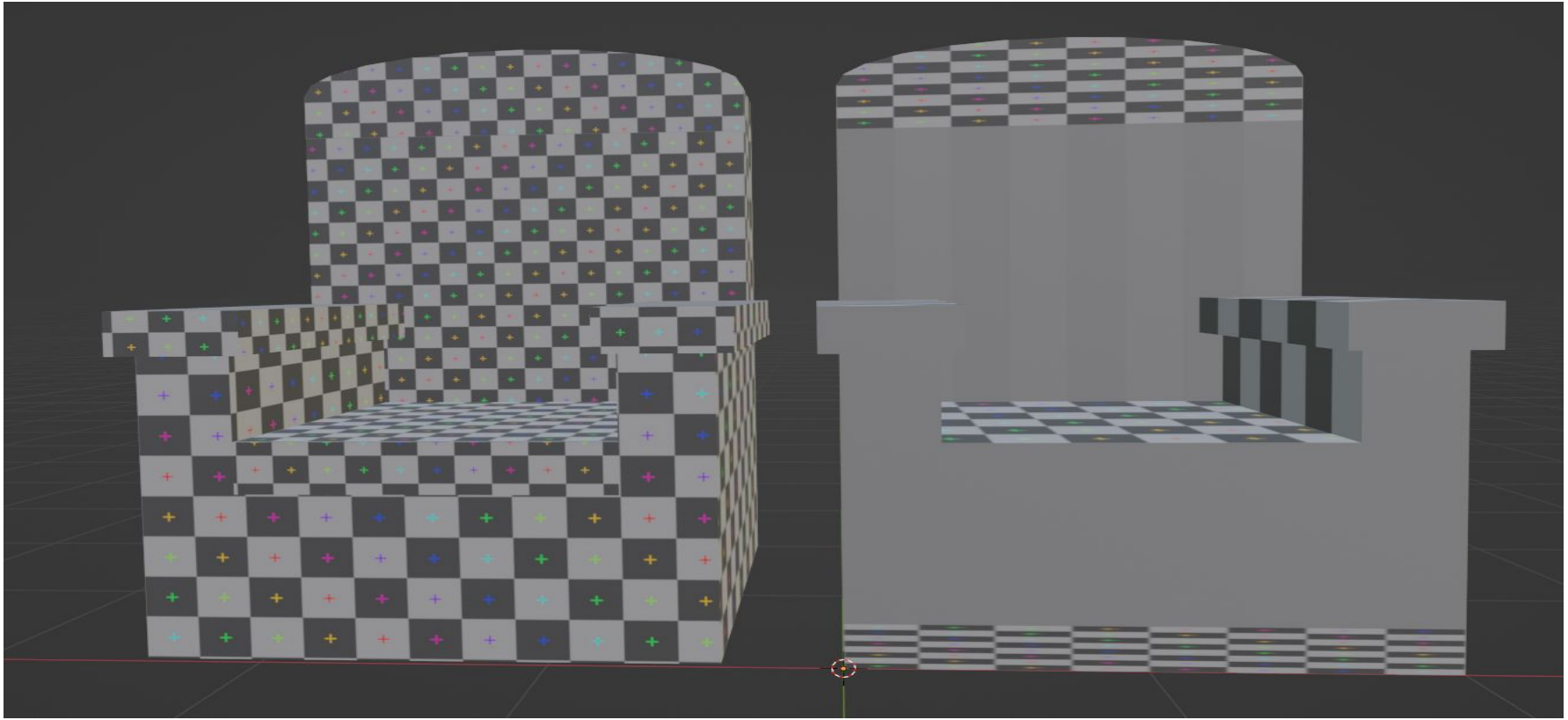
UV Editing





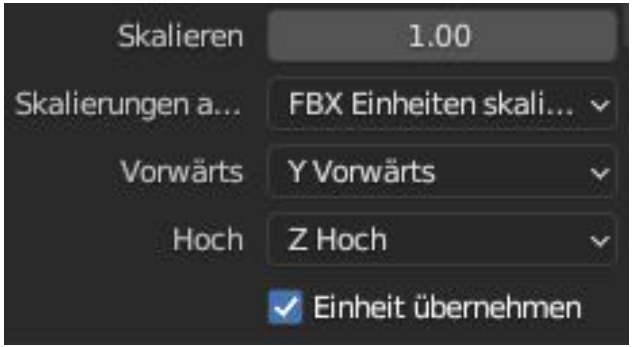
UV Editing

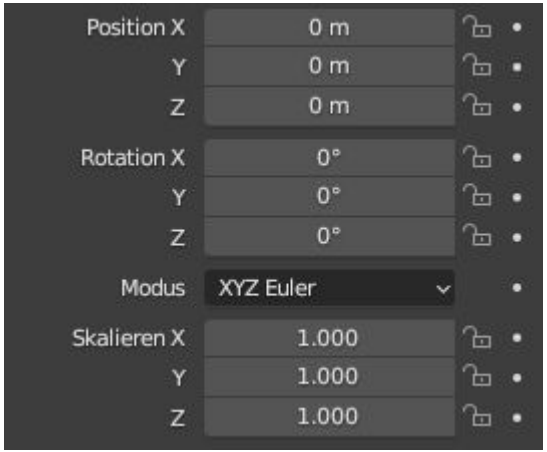
UV Editing



Export

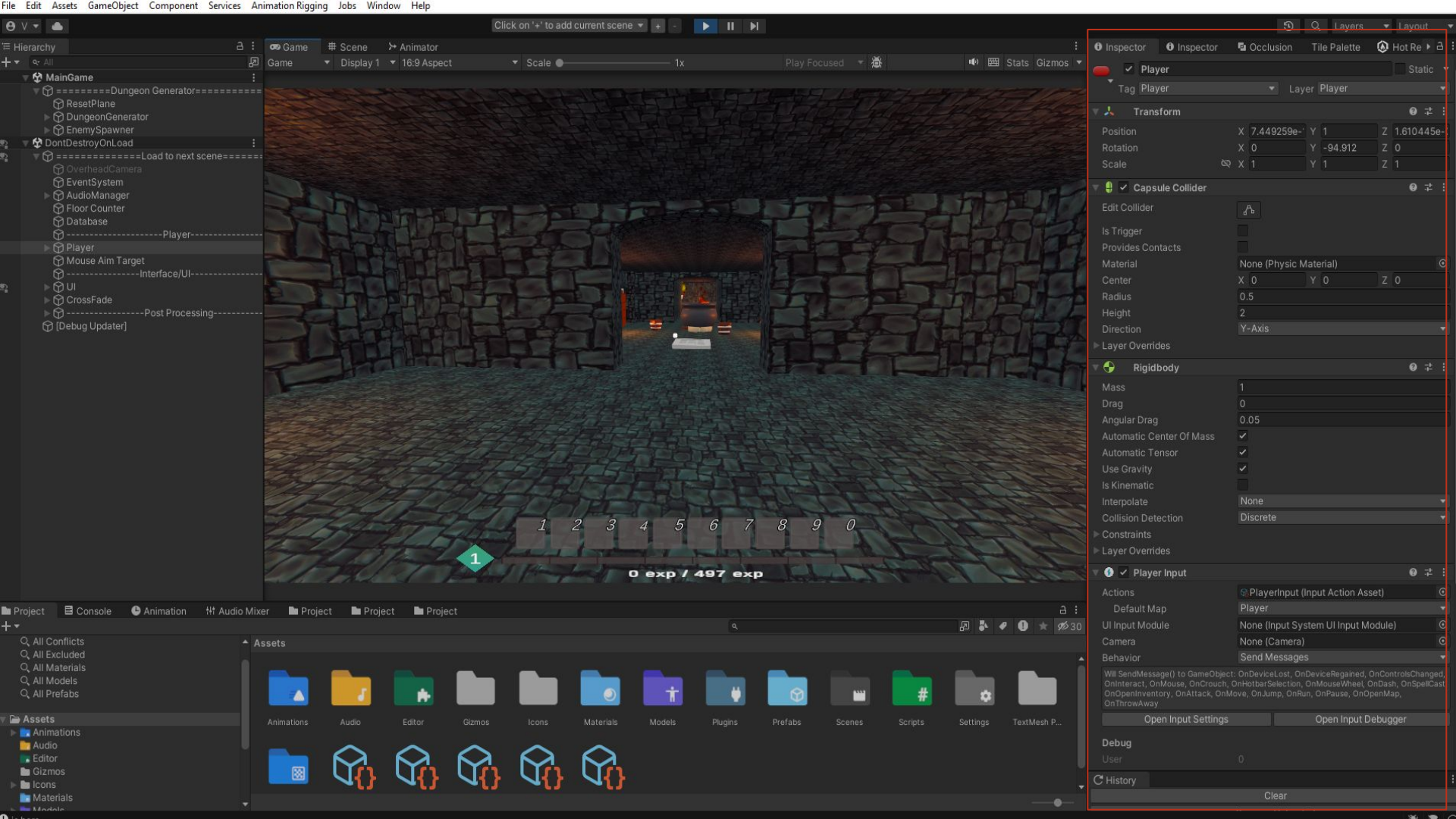
- Blender: x seite, y vorne, z oben
- Unity: x vorne, y oben, z seite
- Ursprung
- Skalierung
- FBX Datei





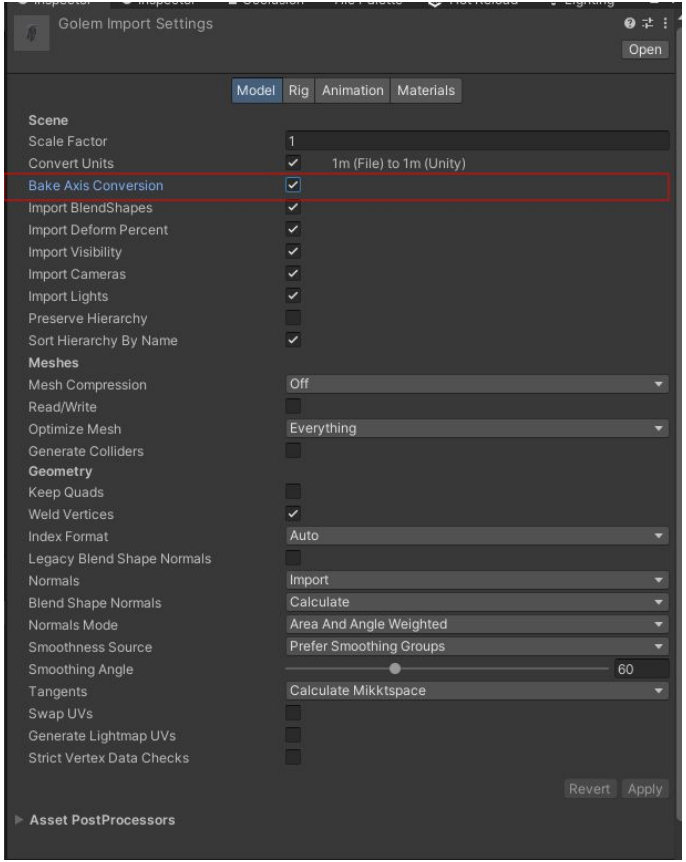
Unity

- Cross-Platform Spiele-Engine
- Integrierte Physik Engine
- Skripting mit C#
- Animationen, AI, Shader, Partikeleffekte


File Edit Assets GameObject Component Services Animation Rigging Jobs Window Help
Hierarchy
MainGame
========Dungeon Generator==========
ResetPlane
DungeonGenerator
EnemySpawner
DontDestroyOnLoad
==============Load to next scene======
OverheadCamera
EventSystem
AudioManager
Floor Counter
Database
----------Player----------
Player
Mouse Aim Target
----------Interface/UI----------
UI
CrossFade
----------Post Processing----------
[Debug Updater]
Game
Scene
Animator
Display 1
16:9 Aspect
Scale
1x
Play Focused
Stats
Gizmos
1 2 3 4 5 6 7 8 9 0
1
0 exp / 497 exp
Inspector
Occlusion
Tile Palette
Player
Static
Tag Player
Layer Player
Transform
Position
Rotation
Scale
Capsule Collider
Edit Collider
Is Trigger
Provides Contacts
Material None (Physic Material)
Center
Radius 0.5
Height 2
Direction Y-Axis
Layer Overrides
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Automatic Center Of Mass
Automatic Tensor
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Layer Overrides
Player Input
Actions PlayerInput (Input Action Asset)
Default Map Player
UI Input Module None (Input System UI Input Module)
Camera None (Camera)
Behavior Send Messages
Will SendMessage() to GameObject: OnDeviceLost, OnDeviceRegained, OnControlsChanged, OnInteract, OnMouse, OnCrouch, OnHotbarSelection, OnMouseWheel, OnDash, OnSpellCast, OnOpenInventory, OnAttack, OnMove, OnJump, OnRun, OnPause, OnOpenMap, OnThrowAway
Open Input Settings
Open Input Debugger
Debug
User 0
History
Clear
Project
Console
Animation
Audio Mixer
All Conflicts
All Excluded
All Materials
All Models
All Prefabs
Assets
Animations
Audio
Editor
Gizmos
Icons
Materials
Models
Plugins
Prefabs
Scenes
Scripts
Settings
TextMesh P...

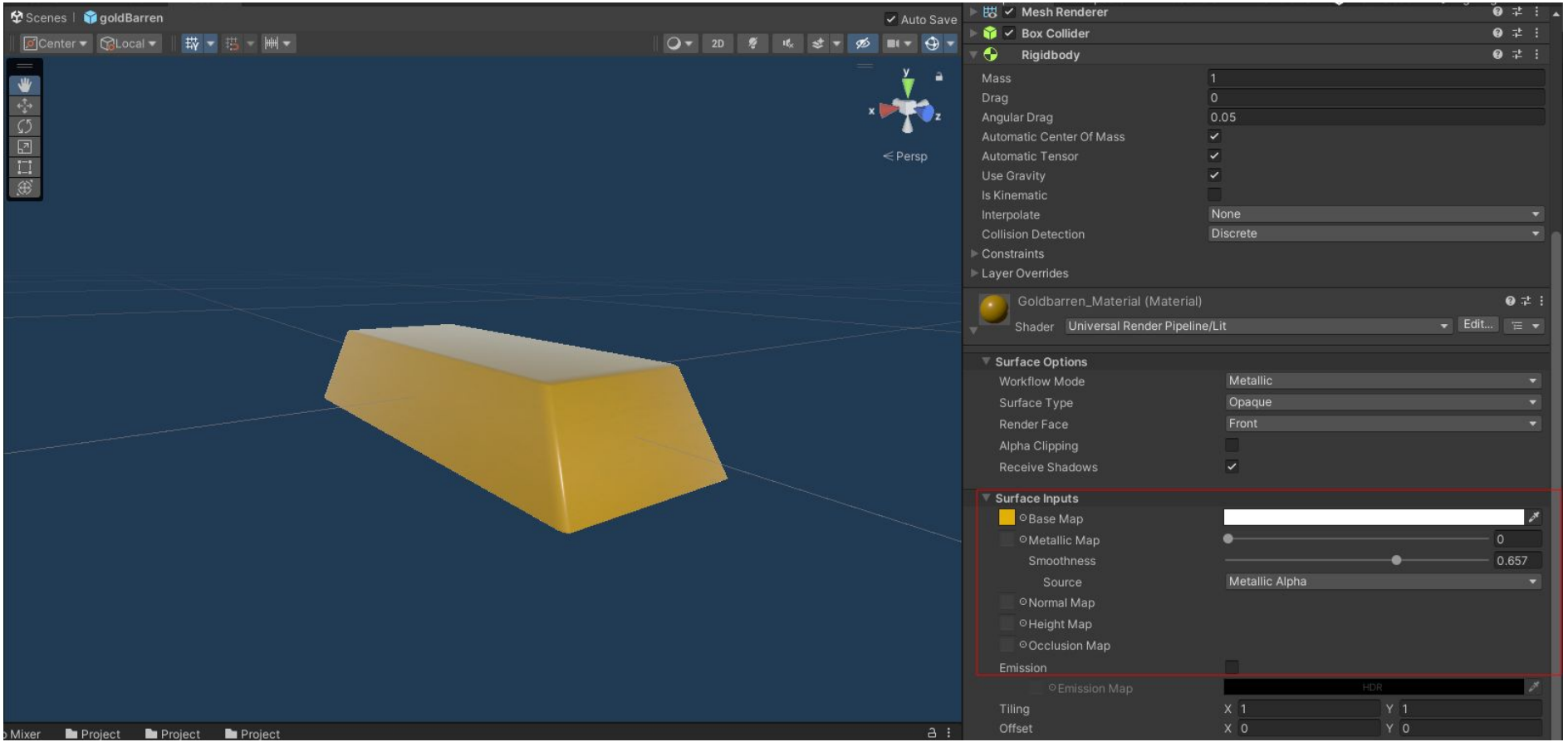
Wie Blender mit Unity benutzen

- Achsenkonvertierung
- Unity und Blender benutzen verschiedene Render Systeme und Shadersprache (OSL vs Shaderlab)



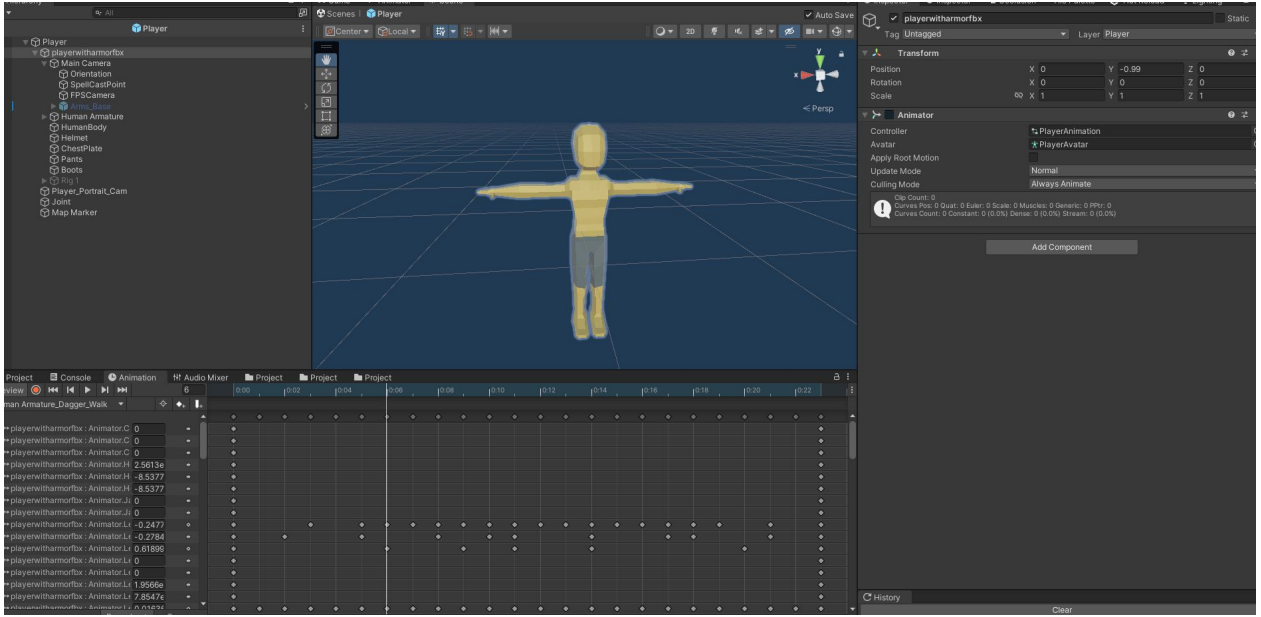
Wie Blender mit Unity benutzen

- Texturen, Normal Map, Height Map etc. in Unity importieren
- Modelle sehen meistens nicht exakt aus wie in Blender
- Shader sollen in Unity gemacht werden


Scenes
goldBarren
Auto Save
Center
Local
2D
Persp
Mesh Renderer
Box Collider
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Automatic Center Of Mass
Automatic Tensor
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Layer Overrides
Goldbarren_Material (Material)
Shader Universal Render Pipeline/Lit
Edit...
Surface Options
Workflow Mode Metallic
Surface Type Opaque
Render Face Front
Alpha Clipping
Receive Shadows
Surface Inputs
Base Map
Metallic Map 0
Smoothness 0.657
Source Metallic Alpha
Normal Map
Height Map
Occlusion Map
Emission
Emission Map
HDR
Tiling X 1 Y 1
Offset X 0 Y 0
Mixer
Project
Project
Project

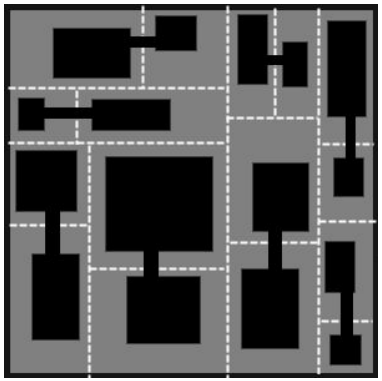
Wie Blender mit Unity benutzen

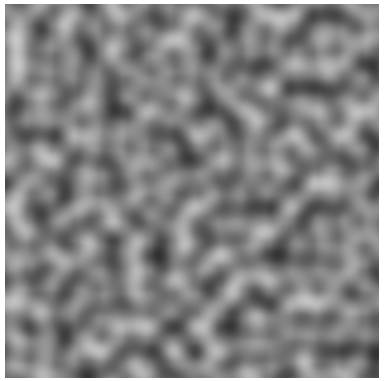
- Animationen können problemlos als FBX Datei importiert werden
- Änderungen innerhalb Unity sind möglich



Dungeon Generator

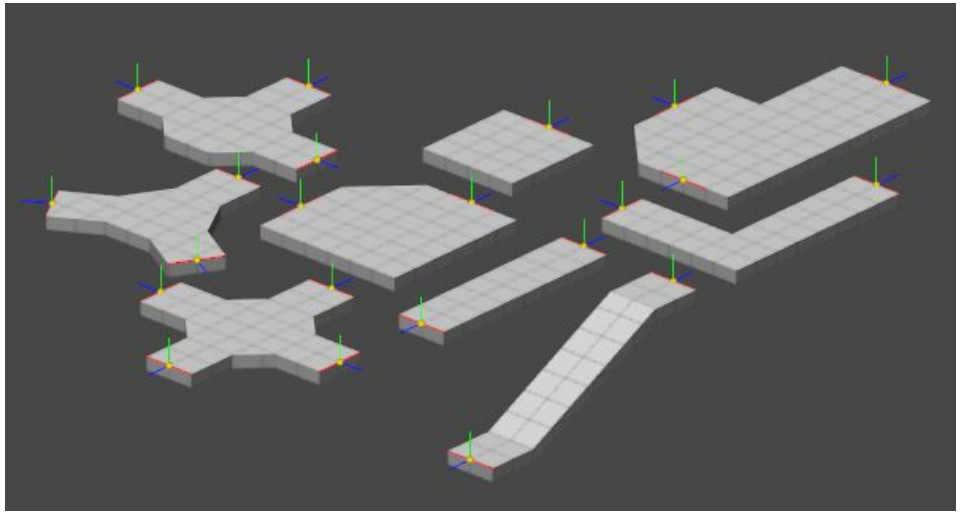
- Prozedurale Generierung des Dungeons
  - Jedes Level ist einzigartig und unvorsehbar
  - Wir müssen nicht jeden Raum manuell gestalten
- Es gibt viele Algorithmen zur Prozedurale Generierung
  - BSP, Cellular Automata, Perlin Noise etc.





→ Generierung mit modularen Bauteilen

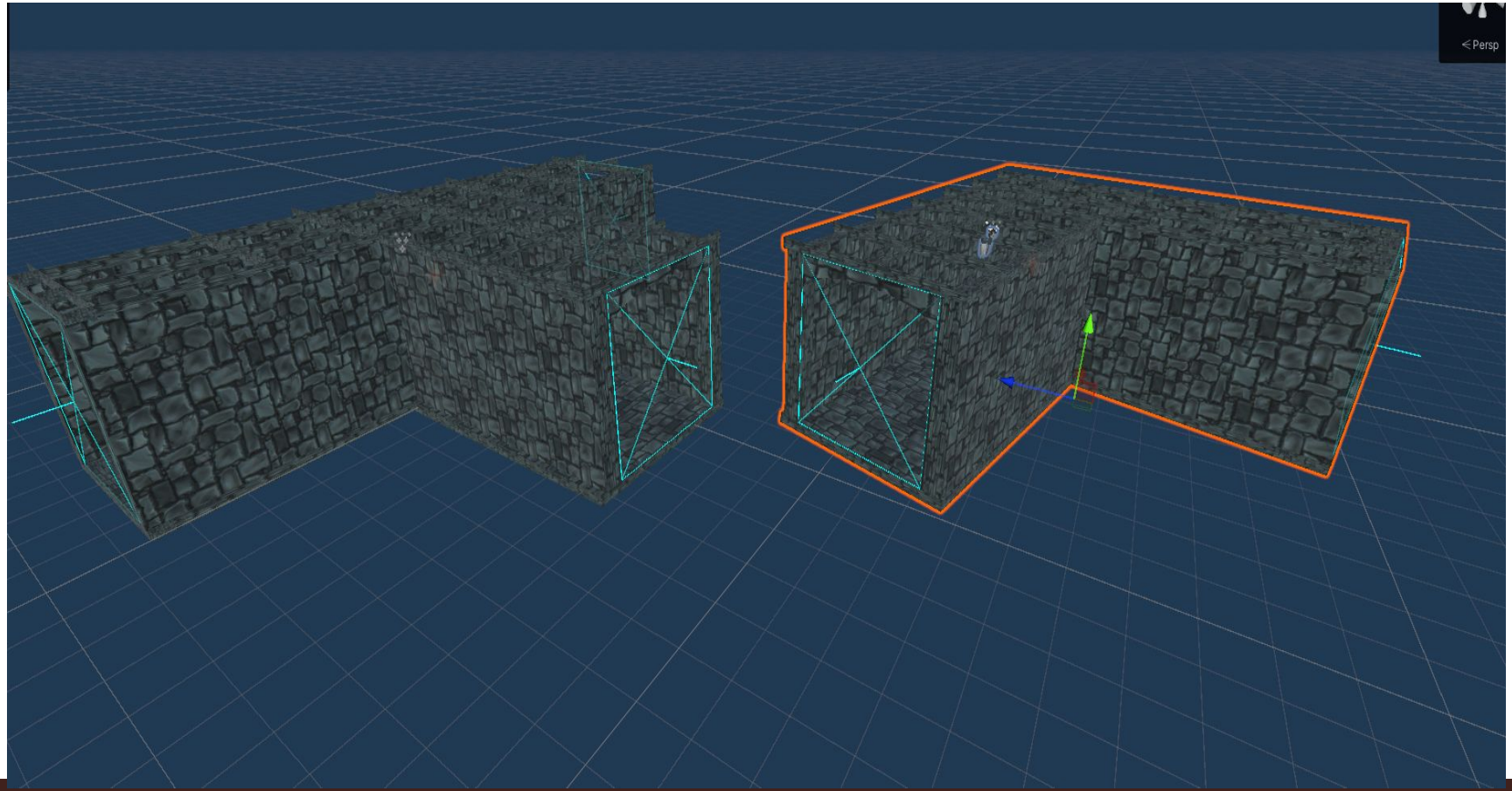
- Leichter in 3D umzusetzen
- Kreative Freiheit bei Gestaltung der einzelnen Räume



Dungeon Generator

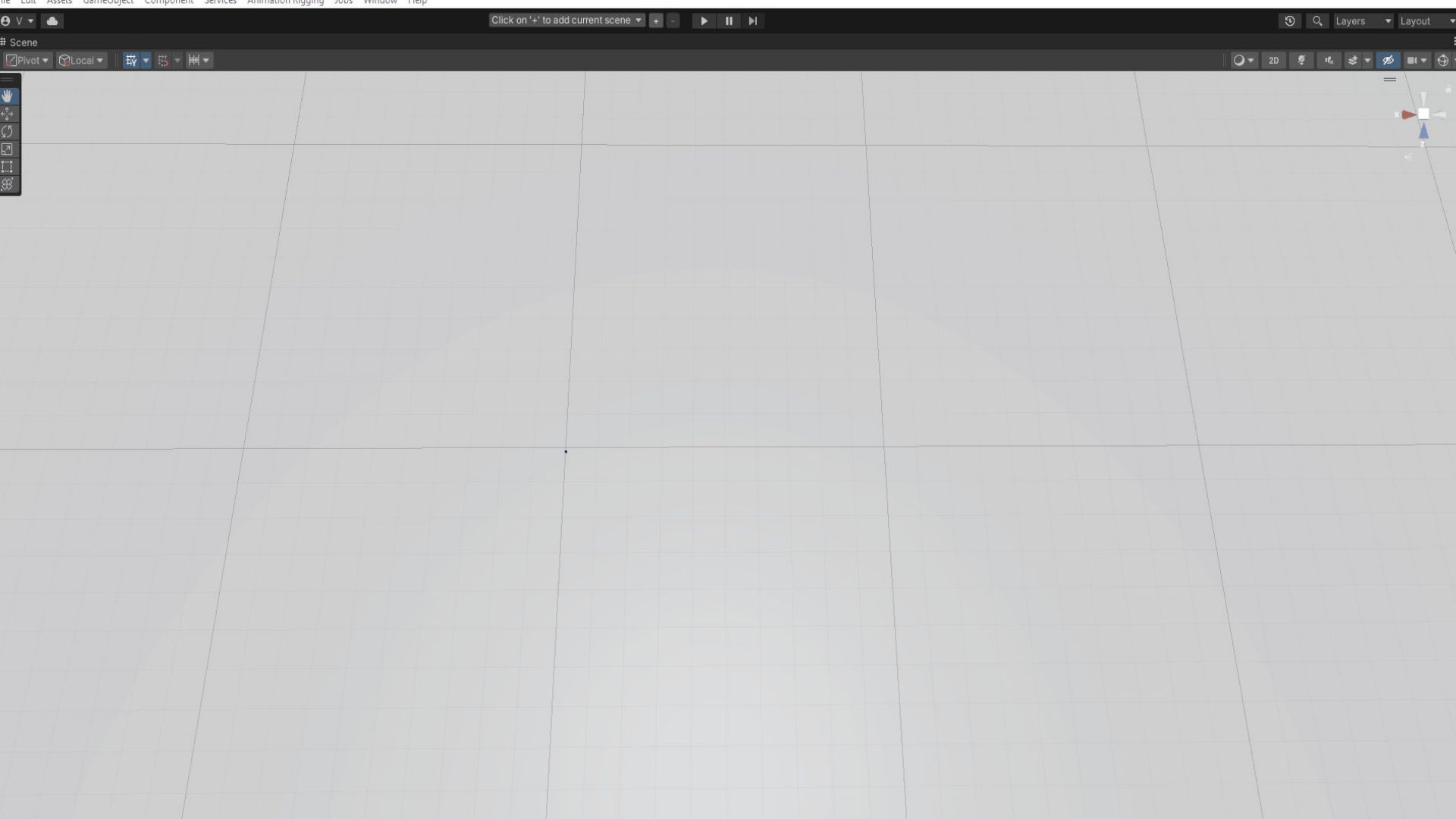
- Der Dungeon besteht aus einer Menge aus Modulen (Räume) welche miteinander verbunden werden
- Räume können einen bis vier Ausgänge haben
- Räume können mit Räumen oder Korridore verbunden werden
- Ein Dungeon muss einen Start- und Endraum haben

Module

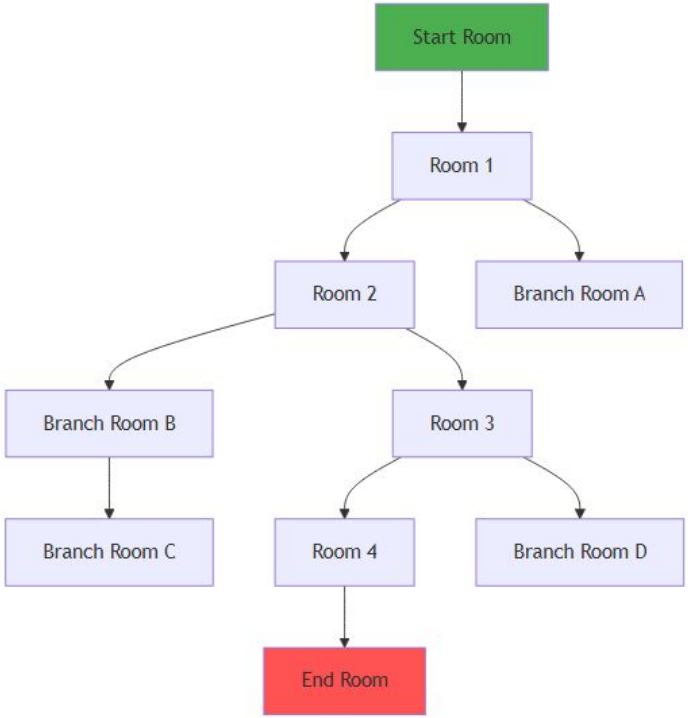


Dungeon Generator Algorithmus

1. Platziere Startraum
2. Wähle zufällig ein Raum aus und verbinde dies mit einem der Ausgänge
3. Falls eine Kollision/Überschneidung mit einem anderen Raum gibt, wähle ein anderen Ausgang aus
4. Wiederhole Schritt 2 bis alle Räume platziert worden sind
5. Platziere den Endraum
6. Platziere Wände an Ausgängen, die nicht mit einem Raum verbunden sind
7. Platziere Monster und Dekorationen

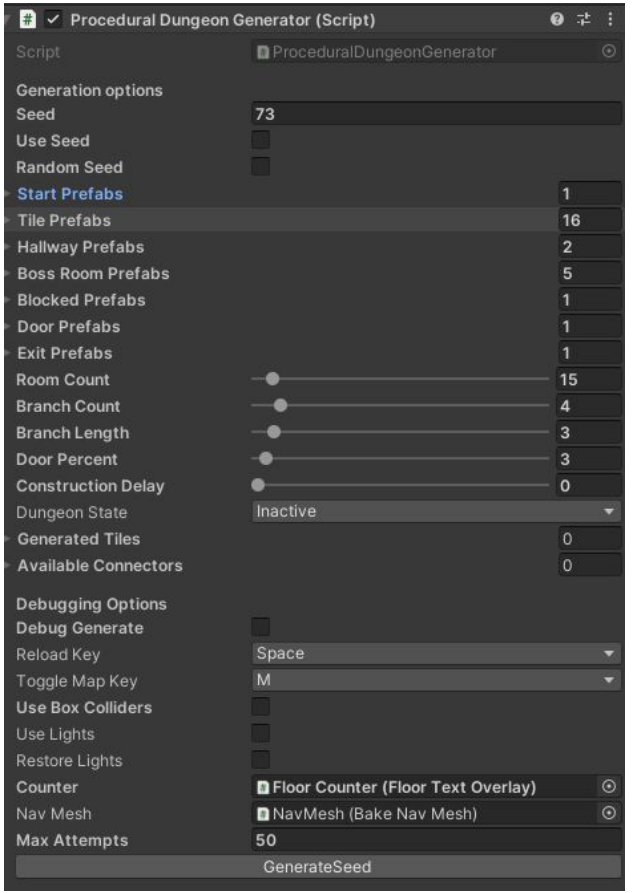

Click on '+' to add current scene
Layers
Layout
Scene
Pivot
Local
2D

Struktur des Dungeons als Baum



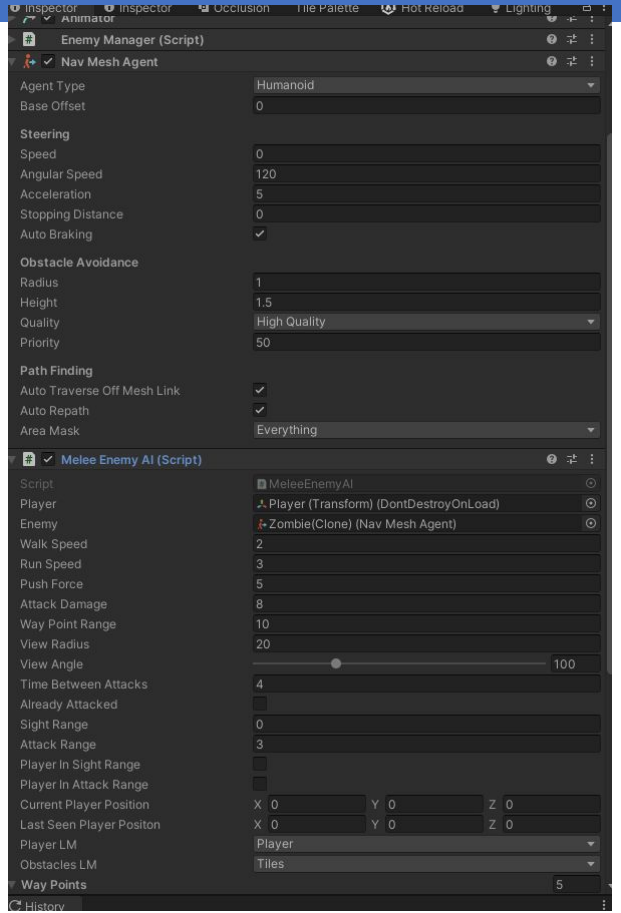
Dungeon Generator

- Dungeon kann beliebig erweitert werden
- Keine Zyklen
- Bugs waren schwer reproduzierbar
- Testing durch festgelegte Seeds

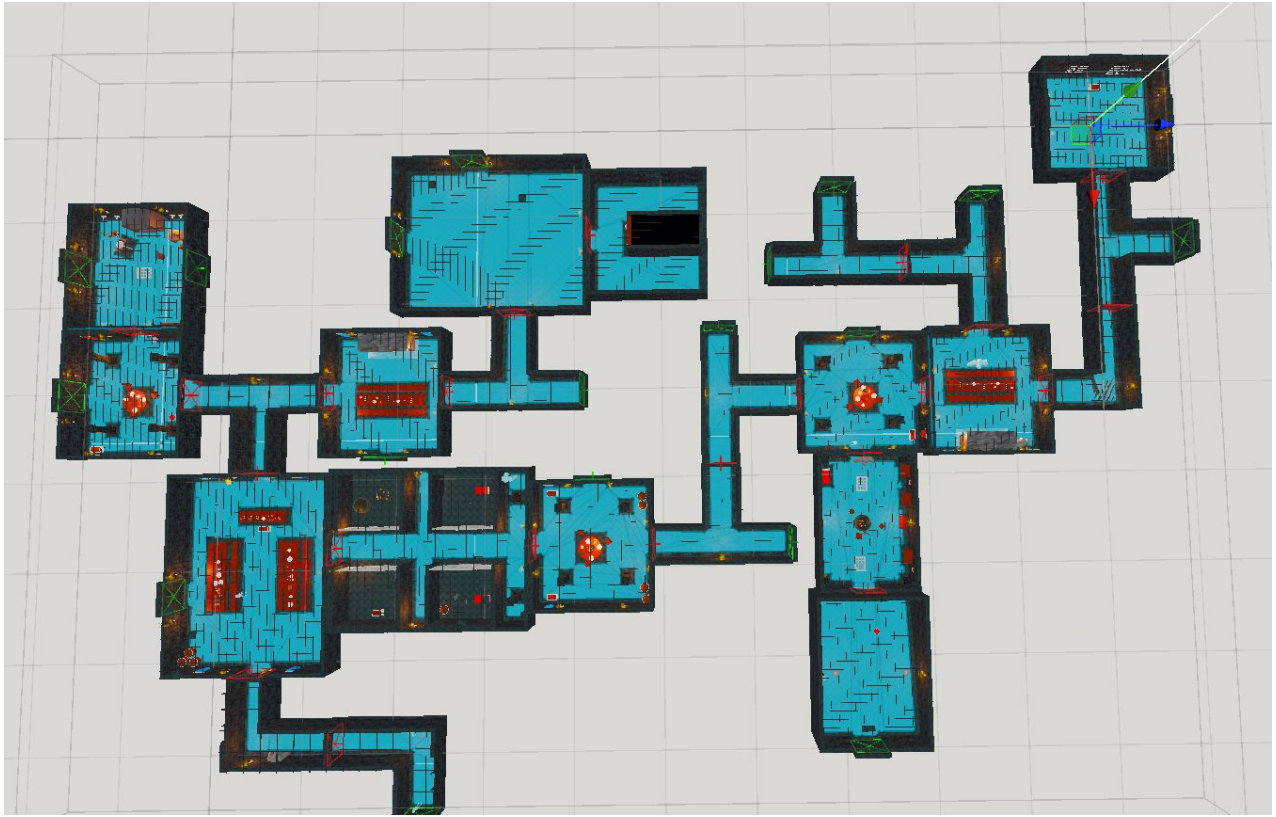


Gegner KI

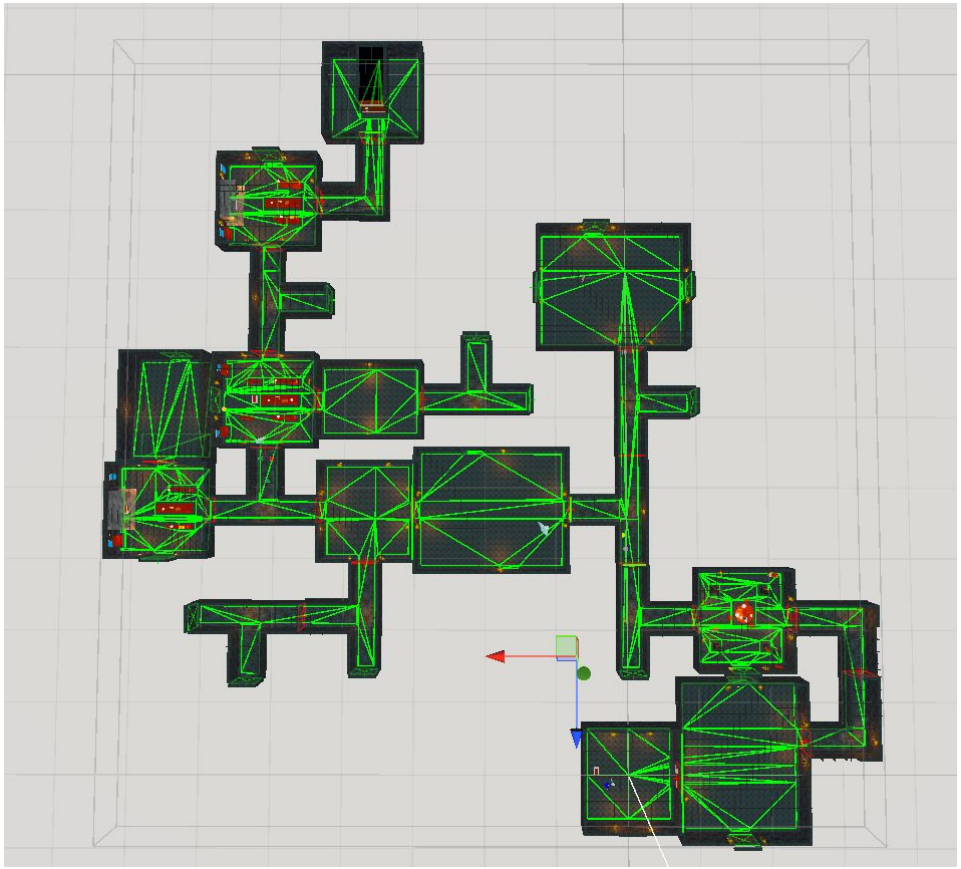
- Unity’s Nav Mesh Agent
  - AI System
- es wird ein NavMesh aus dem Dungeon generiert
- Gegner können sich frei auf diesen NavMesh bewegen über Waypoints



NavMesh



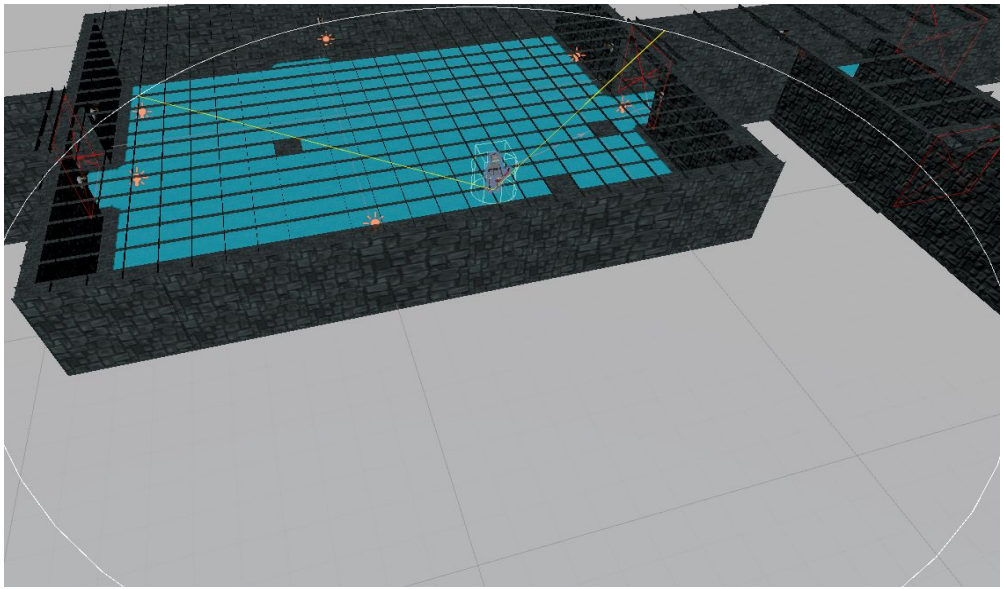
Wegpunkte



```
public void CreateNewWayPoint()
{

    NavMeshTriangulation triangulation = Na

    NavMeshHit Hit;
    while(wayPoints.Count < numberOfWayPoin
    {
        int VertexIndex = UnityEngine.Rando
        if (NavMesh.SamplePosition(triangul
        {
            wayPoints.Add(Hit.position);

        }
    }
}
```

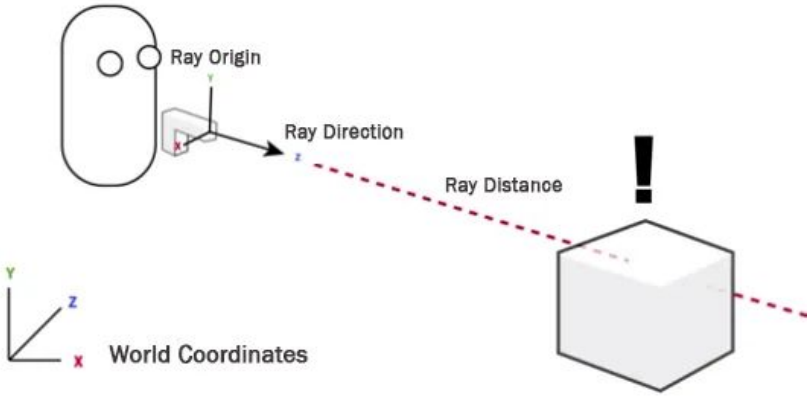
Gegner KI

- Gegner greifen den Spieler an, sobald er sich im Sichtfeld befinden
- Die Abfrage erfolgt mittels Raycasting



Raycasting

- Schießt ein Strahl vom Ursprung mit festgelegter Distanz
- In Unity kann man Abfragen, ob ein und welches Objekt getroffen wurde

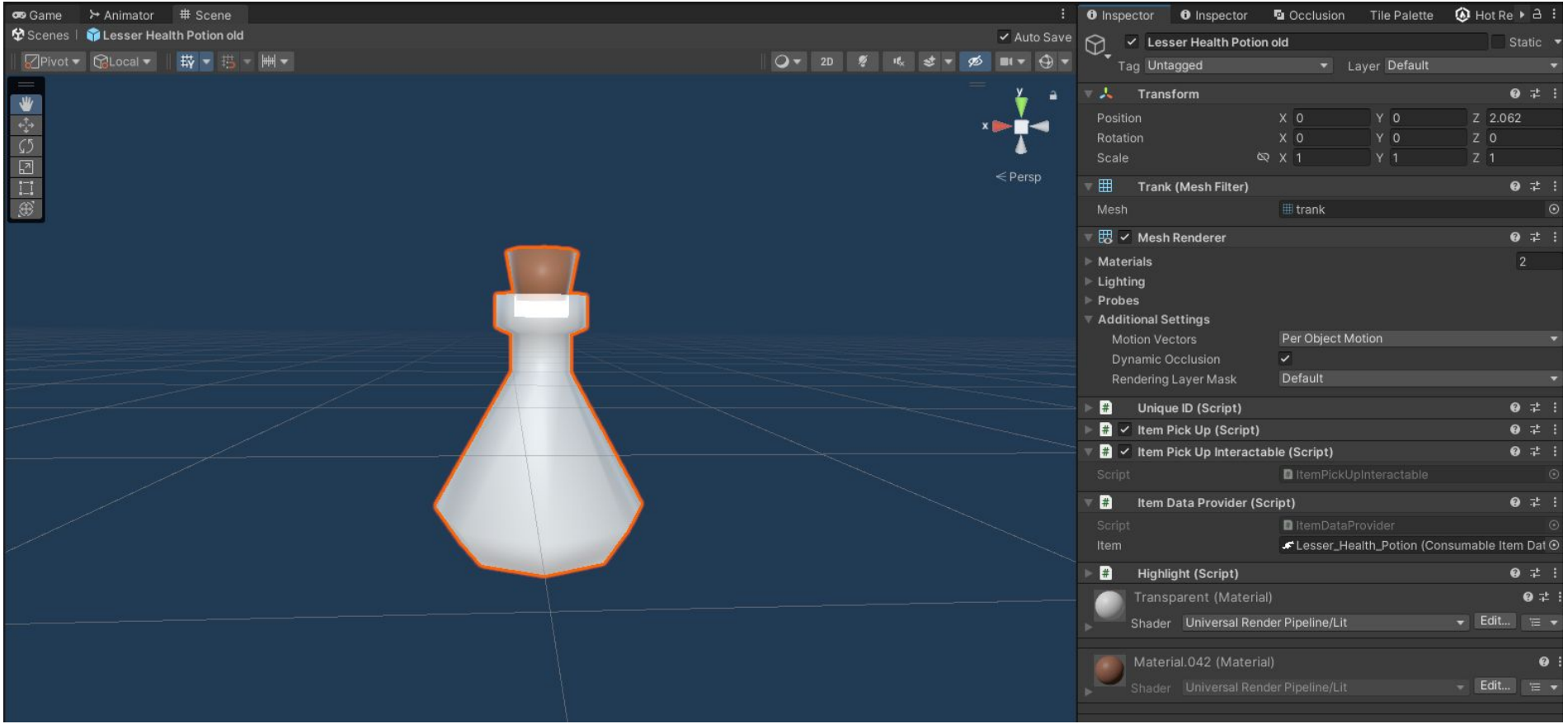


Quelle: https://medium.com/@miguel.araujo/raycast-what-the-hell-is-that-6d36b3c8dd8b

Raycasting

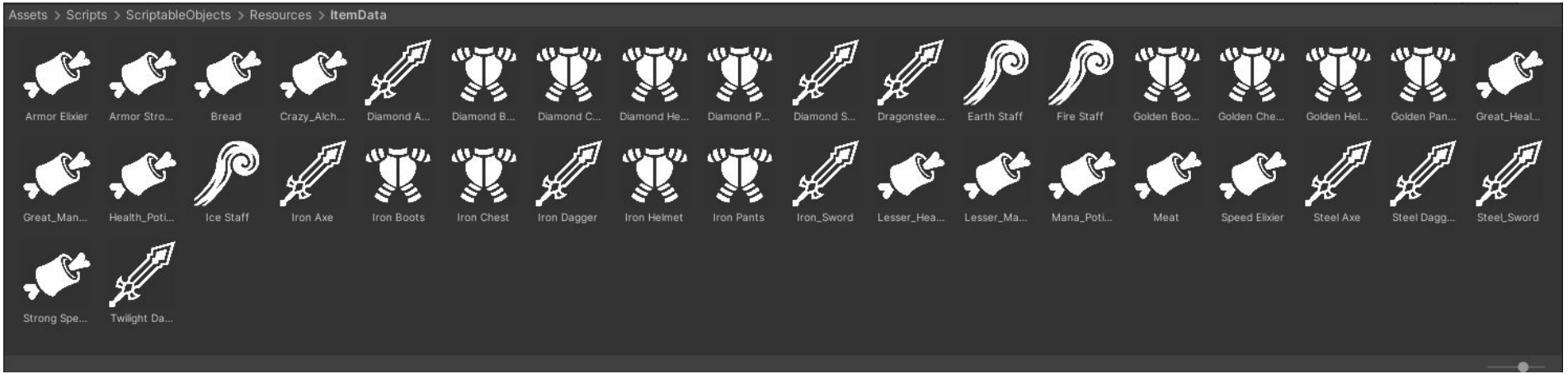
- wird für viele Funktionen angewendet:
  - UI Gegenstandsabfrage
  - Zauberstab (Schusswaffen)
  - Interaktion mit der Umwelt
    - Türe, Kisten, Gegenstände

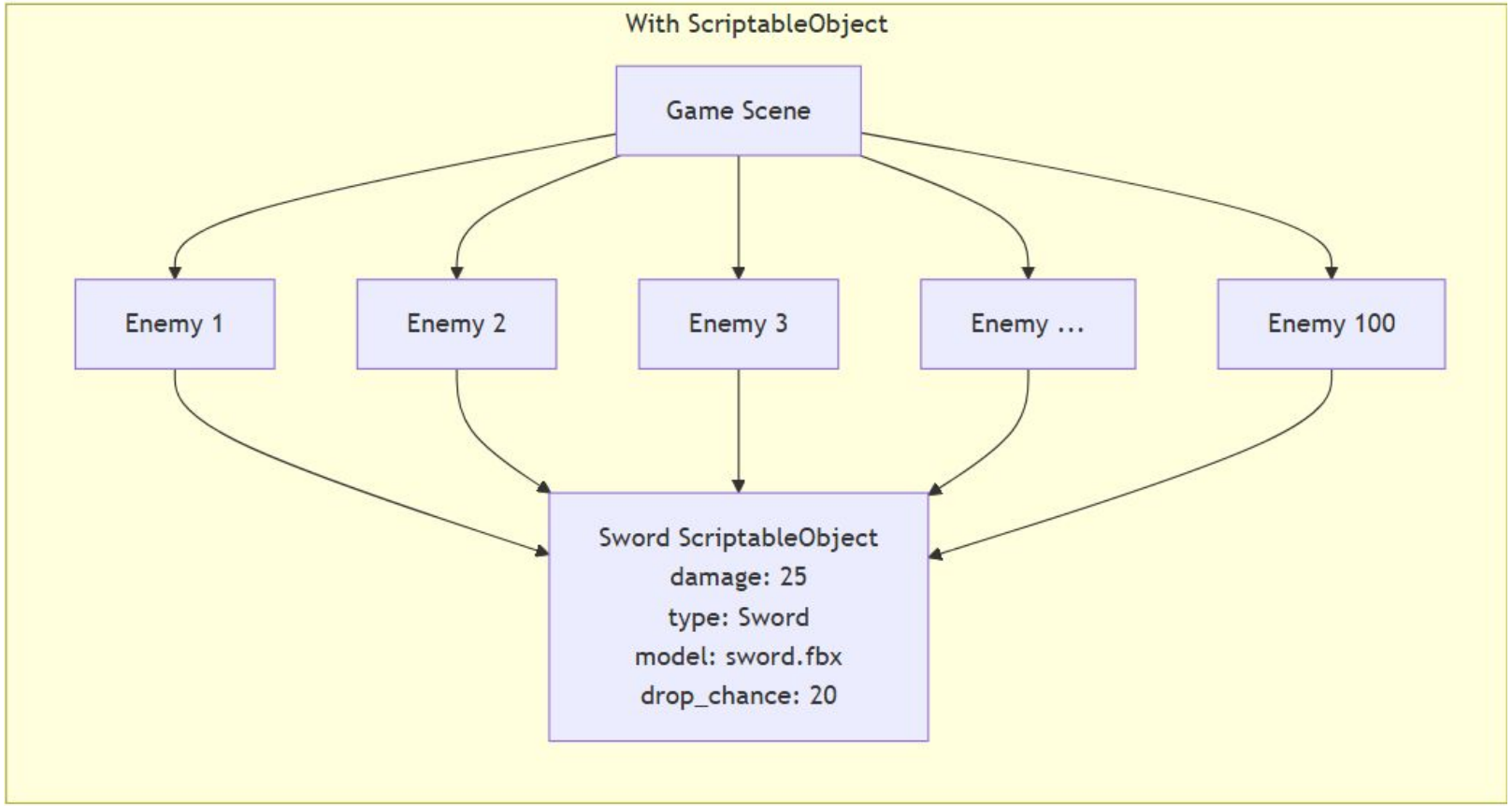
Gegenstände

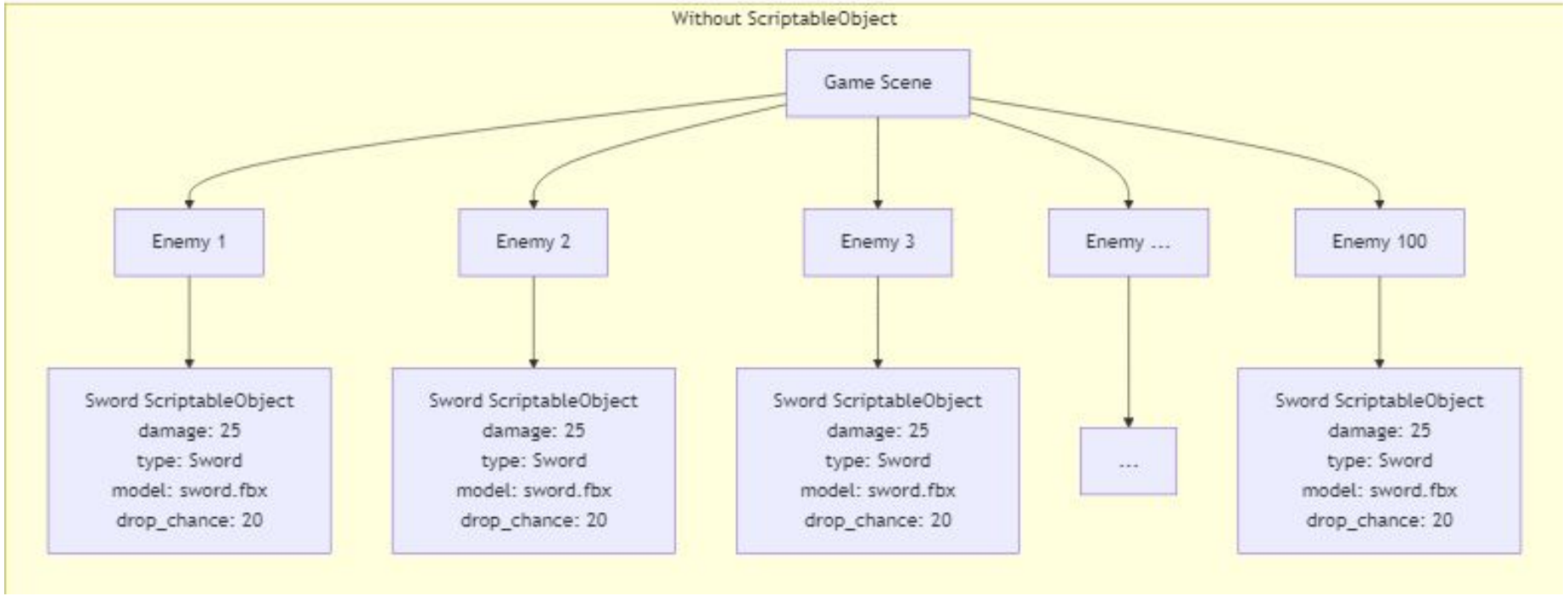


Scriptable Objects

- Datencontainer um große Datenmengen unabhängig von Klasseninstanzen zu speichern
  - Vereinfacht Änderung, Erstellung des Objekts in Runtime




With ScriptableObject
Game Scene
Enemy 1
Enemy 2
Enemy 3
Enemy ...
Enemy 100
Sword ScriptableObject
damage: 25
type: Sword
model: sword.fbx
drop_chance: 20


Without ScriptableObject
Game Scene
Enemy 1
Enemy 2
Enemy 3
Enemy ...
Enemy 100
Sword ScriptableObject
damage: 25
type: Sword
model: sword.fbx
drop_chance: 20
Sword ScriptableObject
damage: 25
type: Sword
model: sword.fbx
drop_chance: 20
Sword ScriptableObject
damage: 25
type: Sword
model: sword.fbx
drop_chance: 20
...
Sword ScriptableObject
damage: 25
type: Sword
model: sword.fbx
drop_chance: 20

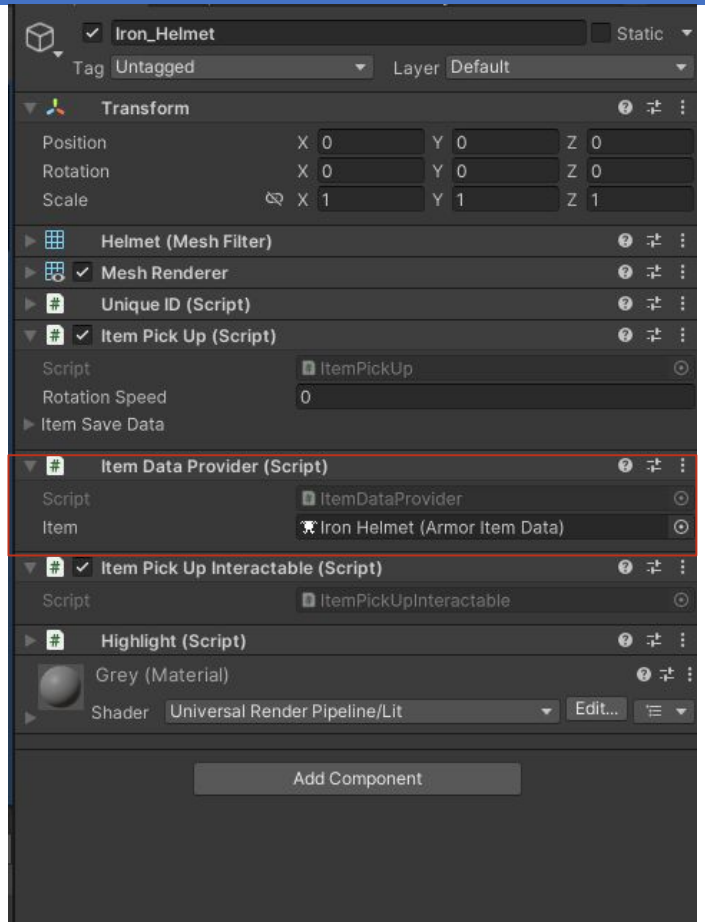
Gegenstände

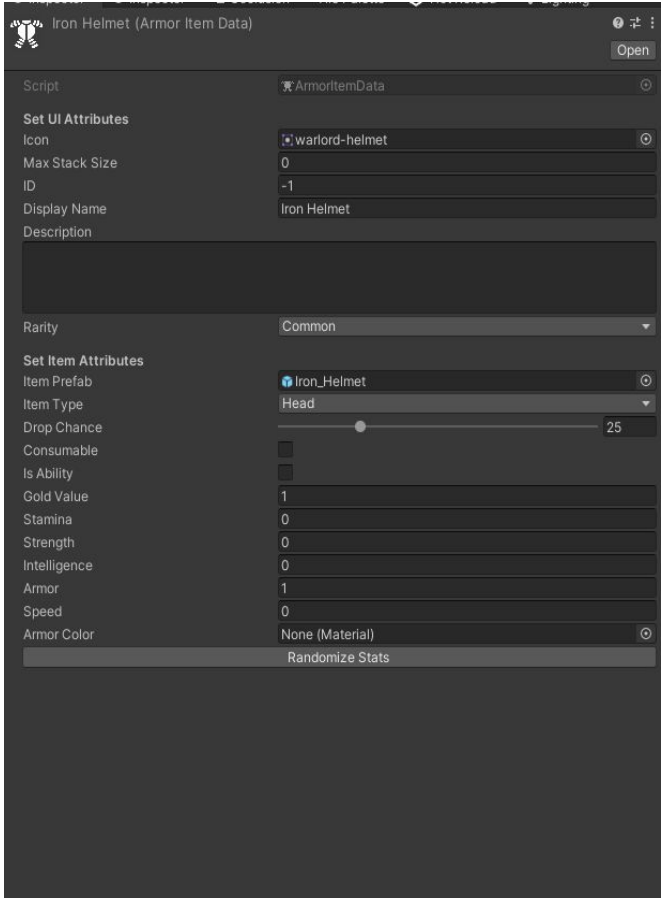
```
[RequireComponent(typeof(ItemPickUp))]
[RequireComponent(typeof(ItemPickUpInteractable))]
[RequireComponent(typeof(Highlight))]
Unity Script (58 asset references) | 8 references
public class ItemDataProvider : MonoBehaviour
{

    [SerializeField] public InventoryItemData item;
    4 references
    public InventoryItemData Item => item;
}
```

```
[CreateAssetMenu(menuName = "Create Items/Item")]
Unity Script | 56 references
public class InventoryItemData : ScriptableObject
{

    Item Data

    3 references
    public virtual void UseItem()
    {
        Debug.Log($"Using {DisplayName}");

    }

}
```

Strategy Pattern

Gegenstände Beispiele



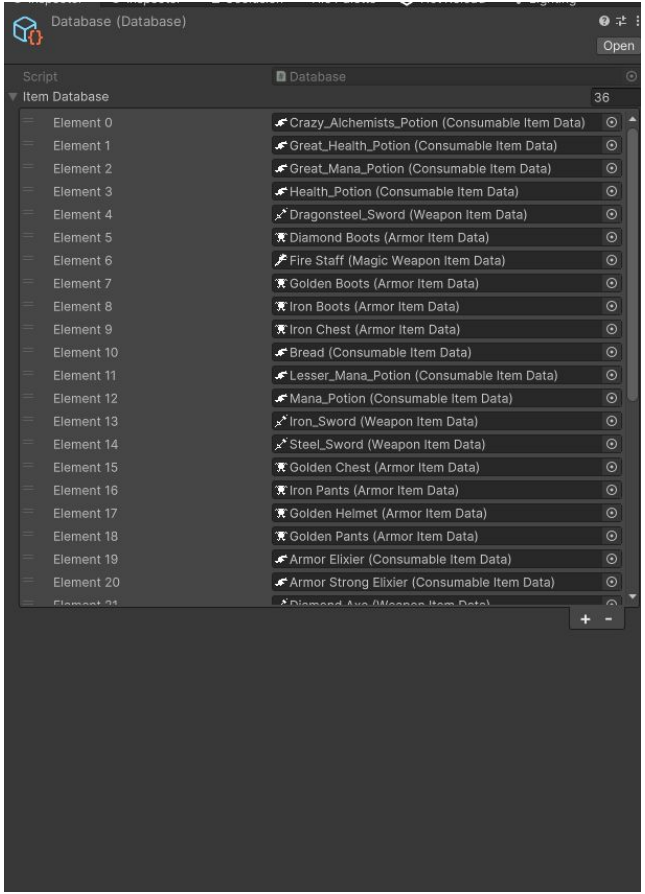


Gegenstandsliste

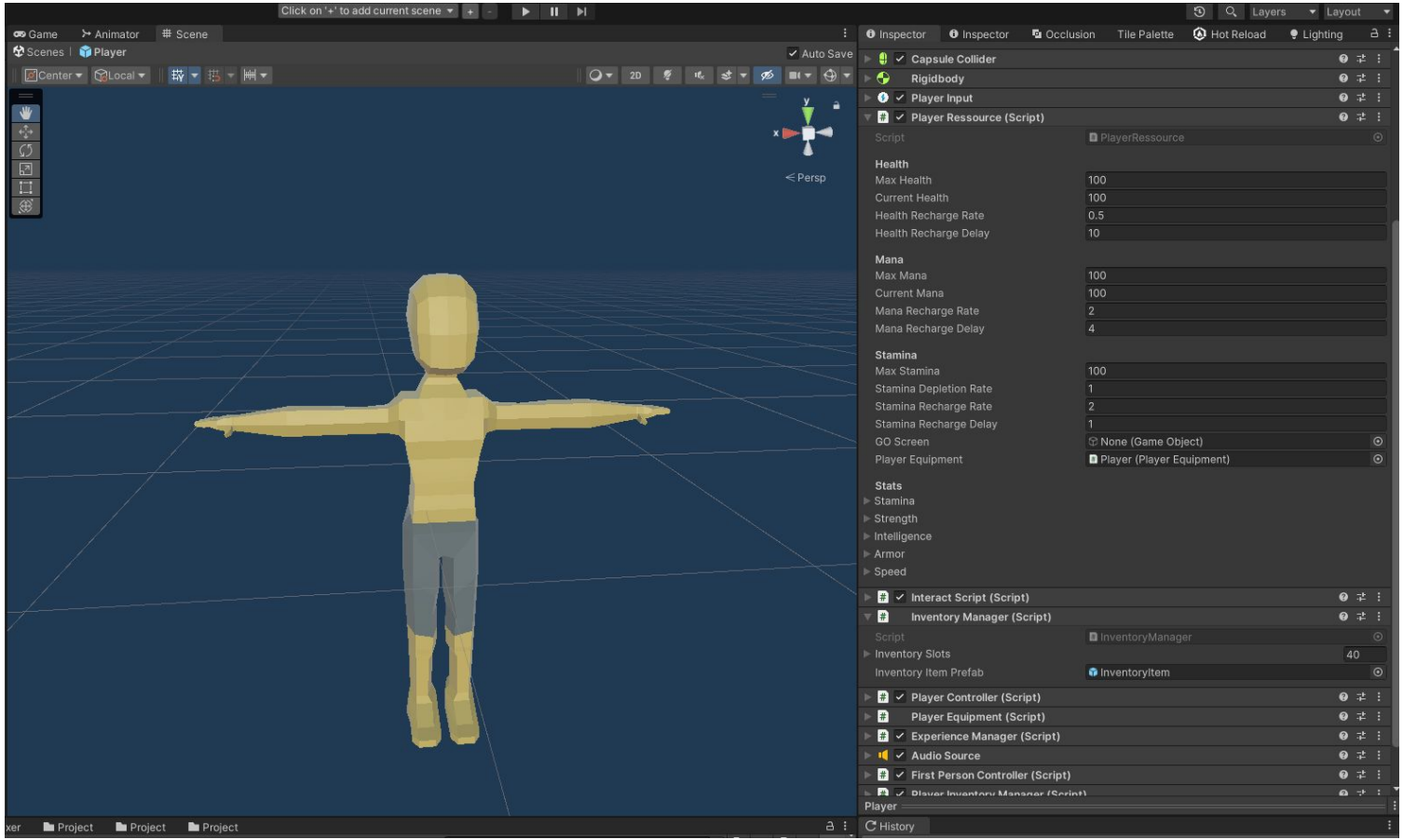
- Man kann SO nutzen, um

  Verweise auf andere SO

  zu speichern
- Loottables
- Gegenstände in Kisten platzieren
- Gegner lassen Gegenstände fallen

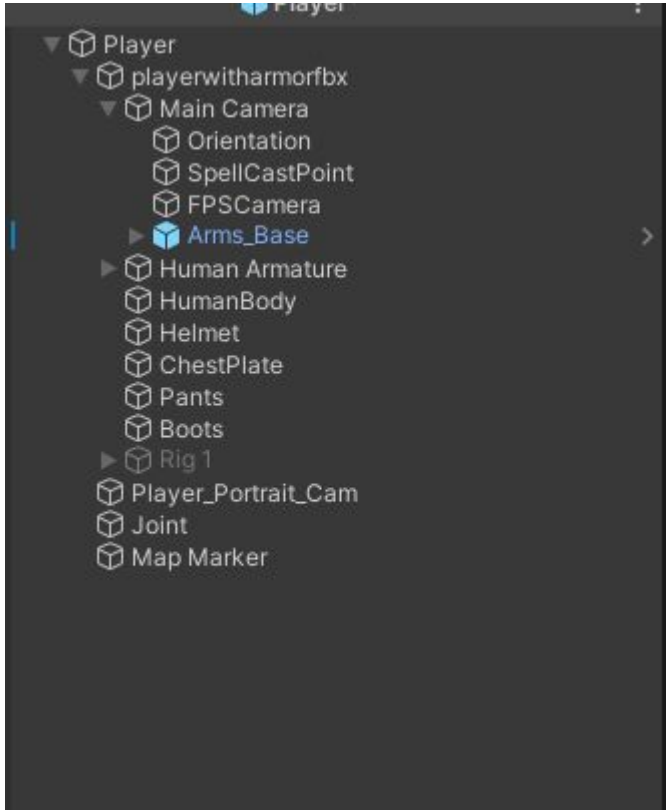


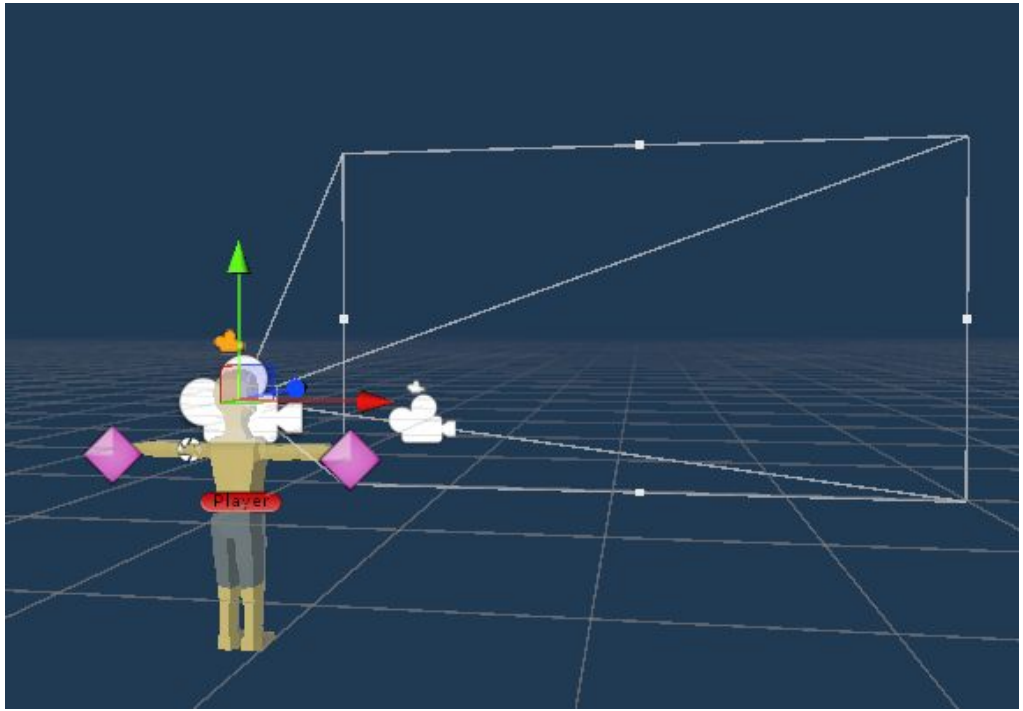
Spieler


Capsule Collider
Rigidbody
Player Input
Player Ressource (Script)
Health
Max Health 100
Current Health 100
Health Recharge Rate 0.5
Health Recharge Delay 10
Mana
Max Mana 100
Current Mana 100
Mana Recharge Rate 2
Mana Recharge Delay 4
Stamina
Max Stamina 100
Stamina Depletion Rate 1
Stamina Recharge Rate 2
Stamina Recharge Delay 1
Stats
Interact Script (Script)
Inventory Manager (Script)
Inventory Slots 40
Player Controller (Script)
Player Equipment (Script)
Experience Manager (Script)
Audio Source
First Person Controller (Script)

Spieler

- Scripts um das Verhalten bzw. Steuerung des Spielers zu definieren
- Man kann problemlos neue Scripts hinzufügen
- Auch gameobjects können hinzugefügt werden

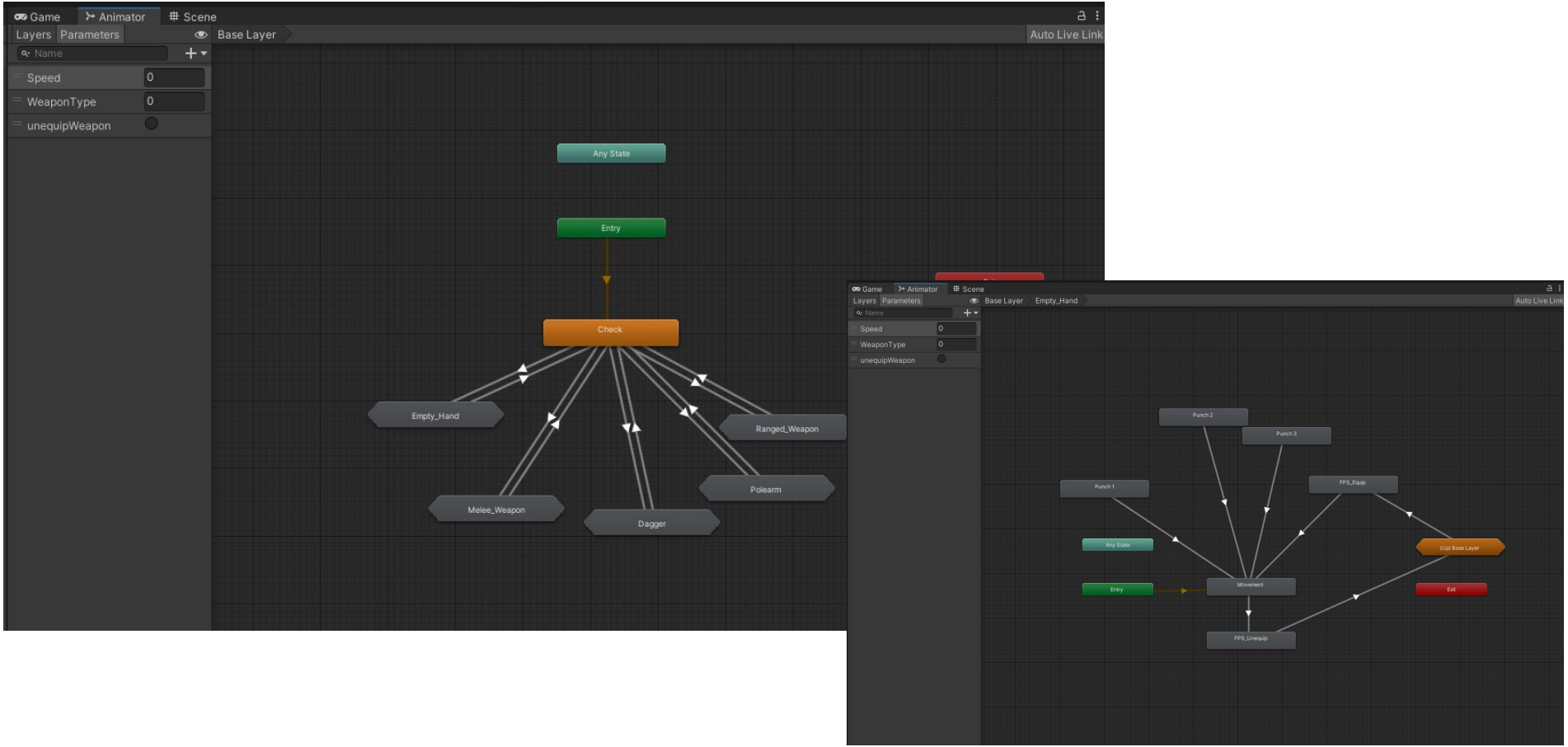

Player
playerwitharmorfbx
Main Camera
Orientation
SpellCastPoint
FPSCamera
Arms_Base
Human Armature
HumanBody
Helmet
ChestPlate
Pants
Boots
Rig 1
Player_Portrait_Cam
Joint
Map Marker


Player

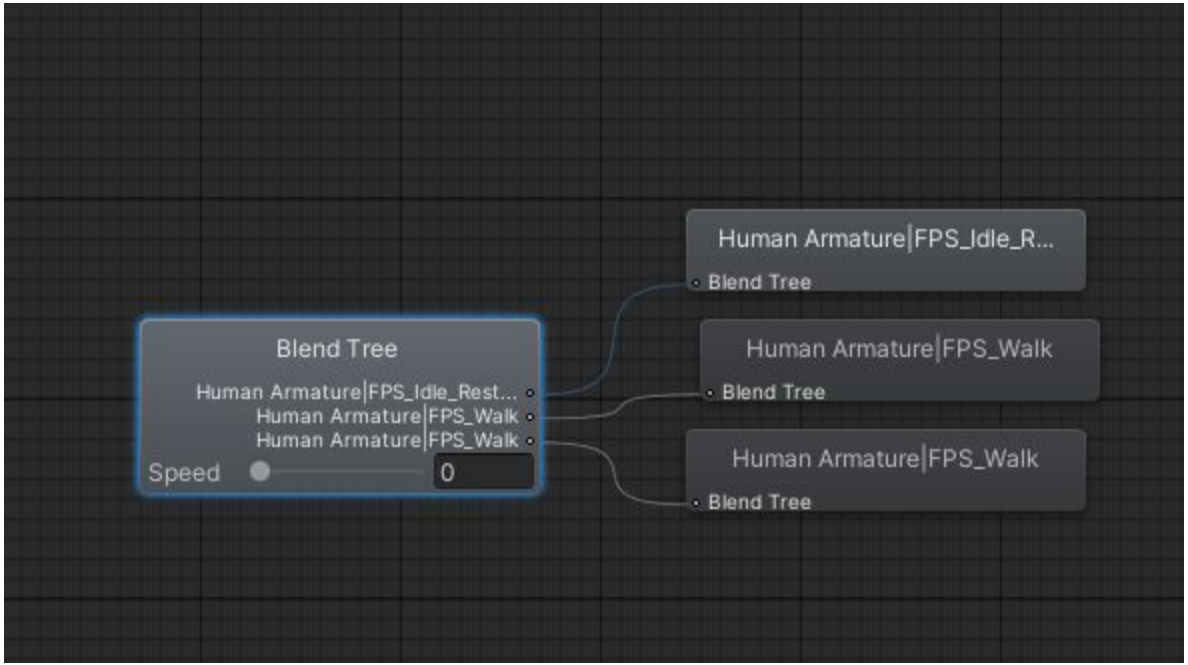
Animationen

- Animator und AnimatorController
  - State Machine
- Zustände definieren und Animationen zuweisen
- Über Skripts kann man auf den Animator zugreifen und die Zustände ändern

Animationen



Blend Tree


Blend Tree
Human Armature|FPS_Idle_Rest...
Human Armature|FPS_Walk
Human Armature|FPS_Walk
Speed
0
Human Armature|FPS_Idle_R...
Blend Tree
Human Armature|FPS_Walk
Blend Tree
Human Armature|FPS_Walk
Blend Tree

Fazit

Was wir gelernt haben:

- Planung und Entwurf eines 3D Computerspiels
- Prozedurale Generierung
- Arbeiten mit Blender(über 90 Modelle) und Unity(100+ commits)
- Blender und Unity Zusammenarbeit

Quellen

1. https://gamedevelopment.tutsplus.com/bake-your-own-3d-dungeons-with-procedural-recipes--gamedev-14360t
2. https://medium.com/@miguel.araujo/raycast-what-the-hell-is-that-6d36b3c8dd8b
3. https://www.youtube.com/watch?v=t6gYn4cV1vw
4. https://de.wikipedia.org/wiki/Normalenvektor
5. https://de.wikipedia.org/wiki/UV-Koordinaten
6. https://www.roguebasin.com/index.php/Basic_BSP_Dungeon_generation
7. Mermaid für Diagramme benutzt