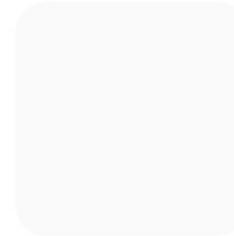
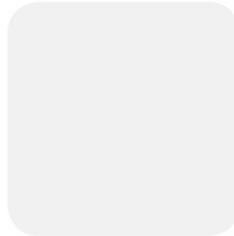
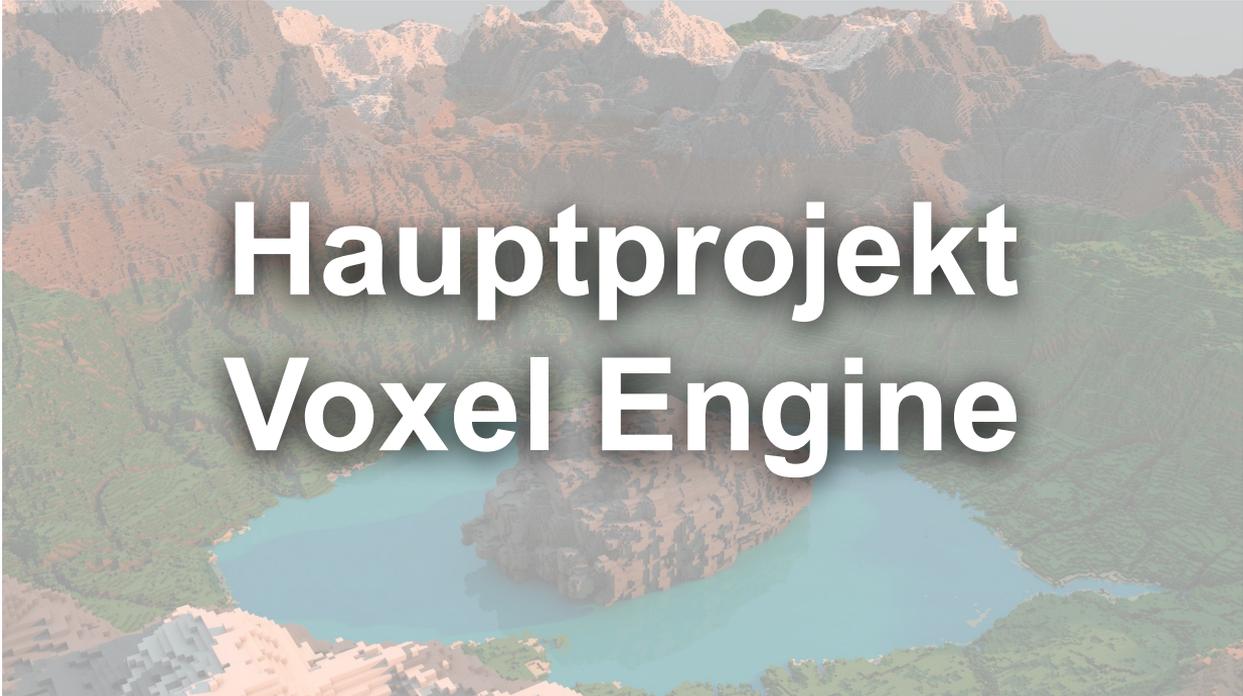


Fachprojekt DET SS 2019 - Hauptprojekt -





Hauptprojekt Voxel Engine

Ziel

- Tiefgehender Einstieg in Elemente der Game Engine Unity
 - z.B. Job System
 - z.B. Entity-Component-System
 - z.B. UI & Event System
 - z.B. Assetbundle
 - z.B. Profiler

Ziel

- Tiefgehender Einstieg in Entertainment Technologies
 - z.B. Procedural Content Generation
 - z.B. AI, Pathfinding Methoden
 - z.B. UI

Wahlthemen

- Procedural Content Generation
- User Interface, Crafting System, Modding System
- Kampfsystem und KI

- Thema wird bis zum 07.05.2019 festgelegt
- Mehrfach- oder Nichtbelegung von Themen ist möglich
- Andere Themen, insbesondere komplexere Themen, nur nach Vereinbarung (z.B. Multiplayer)

Aufgaben

- Entwickeln Sie Spielideen/Features und skizzieren diese (z.B. auf Papier)
- Planen Sie Ihr Projekt mit einem Meilensteinplan und einem Gantt Chart
- Implementieren Sie Ihre Spielidee und dessen Features anhand Ihrer Projektplanung
- Versionieren Sie häufig Ihr Projekt

Voxel Engine Download

- <https://github.com/MarcoMeter/DET-SS2019>

Referenz: <https://www.udemy.com/unityminecraft/>

Einstieg in die Grundlage

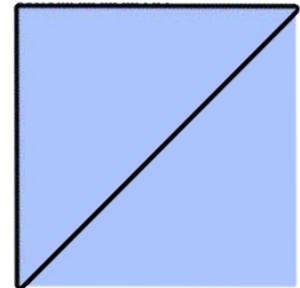
- Was ist ein Voxel?
- Was ist ein Cube?
- Mesh Zusammenfassung
- Datenverarbeitung in einem Voxel Array
- Texture Atlas
- Procedural Content Generation
- Perlin Noise
- Fractional Brownian Noise

Was ist ein Voxel (Volume Pixel)?

- Ein Voxel repräsentiert einen einzelnen Datenpunkt in einem dreidimensionalen Array
- z.B. `chunk[0, 2, 1] = DirtBlock`
- Ein Voxel kennt seine relative Position in einem Volumen (z.B. Position innerhalb eines Chunks)
- Rendering: Interpretation der Daten
- Anwendung: z.B. Terrain

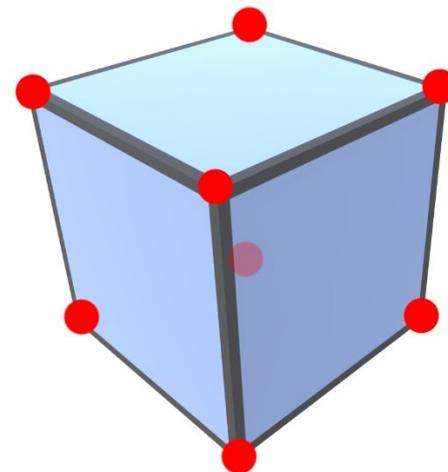
Was ist ein Cube?

- Ein Mesh aus Polygonen
- Die einfachste Form eines Polygons in Unity is ein Triangle
- Die drei Punkte des Dreiecks haben 3D Koordinaten
- Eine Fläche eines Würfels hat zwei Dreiecke
- Ein Dreieck hat eine Normale



Was ist ein Cube?

- Ein Cube in Unity besteht aus 12 Triangles
- GetComponent<MeshFilter>().mesh
 - Vertex Array
 - Normal Array
 - UV Array
 - Triangle Array



Vertex Array

| | | |
|---------------------|----------------------|----------------------|
| $(0.5, -0.5, 0.5)$ | $(-0.5, -0.5, -0.5)$ | $(-0.5, -0.5, 0.5)$ |
| $(-0.5, -0.5, 0.5)$ | $(0.5, 0.5, 0.5)$ | $(-0.5, -0.5, -0.5)$ |
| $(0.5, 0.5, 0.5)$ | $(-0.5, 0.5, 0.5)$ | $(-0.5, -0.5, 0.5)$ |
| $(-0.5, 0.5, 0.5)$ | $(0.5, 0.5, -0.5)$ | $(-0.5, 0.5, 0.5)$ |
| $(0.5, 0.5, -0.5)$ | $(-0.5, 0.5, -0.5)$ | $(-0.5, 0.5, -0.5)$ |
| $(-0.5, 0.5, -0.5)$ | $(0.5, -0.5, -0.5)$ | $(-0.5, -0.5, -0.5)$ |
| $(0.5, -0.5, -0.5)$ | $(0.5, -0.5, 0.5)$ | $(0.5, -0.5, -0.5)$ |
| $(0.5, 0.5, -0.5)$ | $(0.5, 0.5, 0.5)$ | $(0.5, -0.5, 0.5)$ |

Normal Array

| | | |
|------------------|------------------|------------------|
| (0.0, 0.0, 1.0) | (0.0, 1.0, 0.0) | (-1.0, 0.0, 0.0) |
| (0.0, 0.0, 1.0) | (0.0, 1.0, 0.0) | (-1.0, 0.0, 0.0) |
| (0.0, 0.0, 1.0) | (0.0, 0.0, -1.0) | (-1.0, 0.0, 0.0) |
| (0.0, 0.0, 1.0) | (0.0, 0.0, -1.0) | (-1.0, 0.0, 0.0) |
| (0.0, 1.0, 0.0) | (0.0, -1.0, 0.0) | (1.0, 0.0, 0.0) |
| (0.0, 1.0, 0.0) | (0.0, -1.0, 0.0) | (1.0, 0.0, 0.0) |
| (0.0, 0.0, -1.0) | (0.0, -1.0, 0.0) | (1.0, 0.0, 0.0) |
| (0.0, 0.0, -1.0) | (0.0, -1.0, 0.0) | (1.0, 0.0, 0.0) |

UV Array

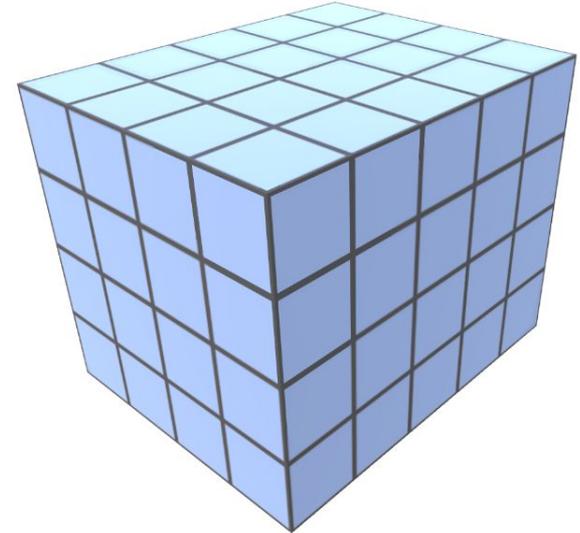
| | | |
|------------|------------|------------|
| (0.0, 0.0) | (0.0, 0.0) | (0.0, 0.0) |
| (1.0, 0.0) | (1.0, 0.0) | (0.0, 1.0) |
| (0.0, 1.0) | (0.0, 0.0) | (1.0, 1.0) |
| (1.0, 1.0) | (1.0, 0.0) | (1.0, 0.0) |
| (0.0, 1.0) | (0.0, 0.0) | (0.0, 0.0) |
| (1.0, 1.0) | (0.0, 1.0) | (0.0, 1.0) |
| (0.0, 1.0) | (1.0, 1.0) | (1.0, 1.0) |
| (1.0, 1.0) | (1.0, 0.0) | (1.0, 0.0) |

Triangle Array

| | |
|------------|-------------|
| 0, 2, 3, | 12, 13, 14, |
| 0, 3, 1, | 12, 14, 15, |
| 8, 4, 5, | 16, 17, 18, |
| 8, 5, 9, | 16, 18, 19, |
| 10, 6, 7, | 20, 21, 22, |
| 10, 7, 11, | 20, 22, 23 |

Mesh Zusammenfassung

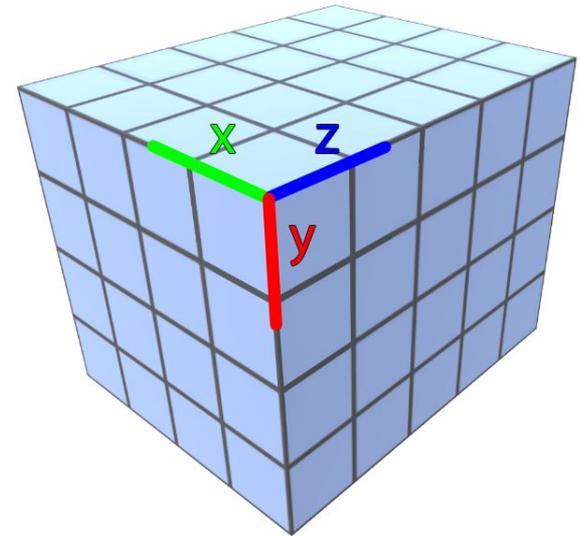
1. Erstelle Quads
 - a. Erstelle Liste mit Vertices
 - b. Erstelle UV Koordinaten
 - c. Berechne Normalen
 - d. Verbinde Vertices zu Triangles
2. Kombiniere Quad Meshes



Datenverarbeitung in einem Voxel Array

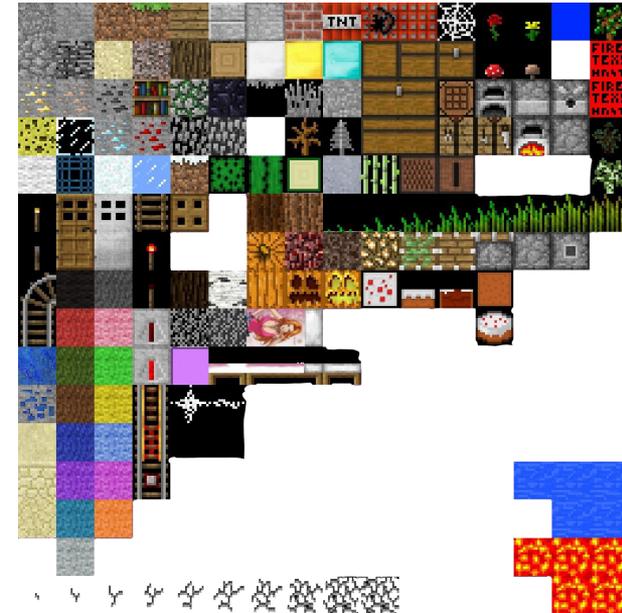
Verschachtelte Schleife:

```
for(int z = 0; z < chunkSize; z++)  
  for(int y = 0; y < chunkSize; y++)  
    for(int x = 0; x < chunkSize; x++)
```



Texture Atlas

- Einsparung von Performance durch ein einziges Material
- UV Koordinaten müssen auf den gewünschten Ausschnitt zugeschnitten werden



Procedural Content Generation

- Autonome Generierung von Inhalten mit limitiertem menschlichen Einfluss
- Terrain
- Level
- Karten
- Texturen
- Musik
- Regeln
- Geschichten
- Charaktere
- ...

Procedural Content Generation

- Schneller und günstigere Produktion
 - 2D und 3D Assets verschlingen 40% der Produktionskosten (Yannanakis, 2017)
- Ziel: Augmentation der Fähigkeiten eines menschlichen Künstlers

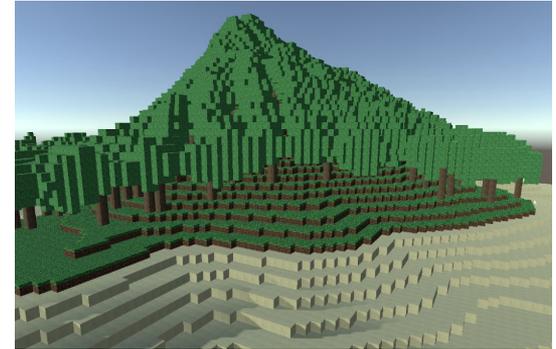
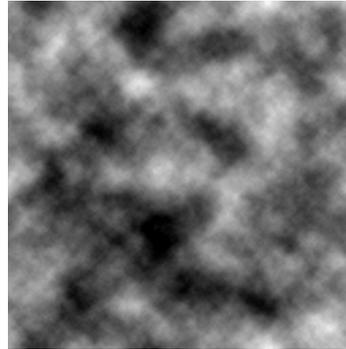
Procedural Content Generation

PCG Schritte:

- Verstehen des Design Prozesses
- Auflistung der Einschränkungen
- Verstehen des Prozesses
- Auswählen einer Methoden zum Generieren
- Iterieren

Perlin Noise

- “Smooth and organic” noise
- The Theory of Noise: An Overview of Perlin Noise
 - <https://www.youtube.com/watch?v=H6FhG9VKhJg>



Fractional Brownian Motion

- Auch genannt “Fractal Noise”
- Mehrere “Oktaven” von Wellen werden überlagert und führen zu einer höheren Detailgenauigkeit
- <https://thebookofshaders.com/13/>

Ressourcen

- Implementieren eines Cubes von Hand
<http://ilkinulas.github.io/development/unity/2016/04/30/cube-mesh-in-unity3d.html>
- Implementieren eines Cubes und Kombination mehrerer Meshes
https://drive.google.com/open?id=1eqwZCBHvnuJtTfQwH1b_hDI8C2jCW4z5

Referenzen

Georgios Yannakakis & Julian Togelius, 2018, Artificial Intelligence and Games, Springer