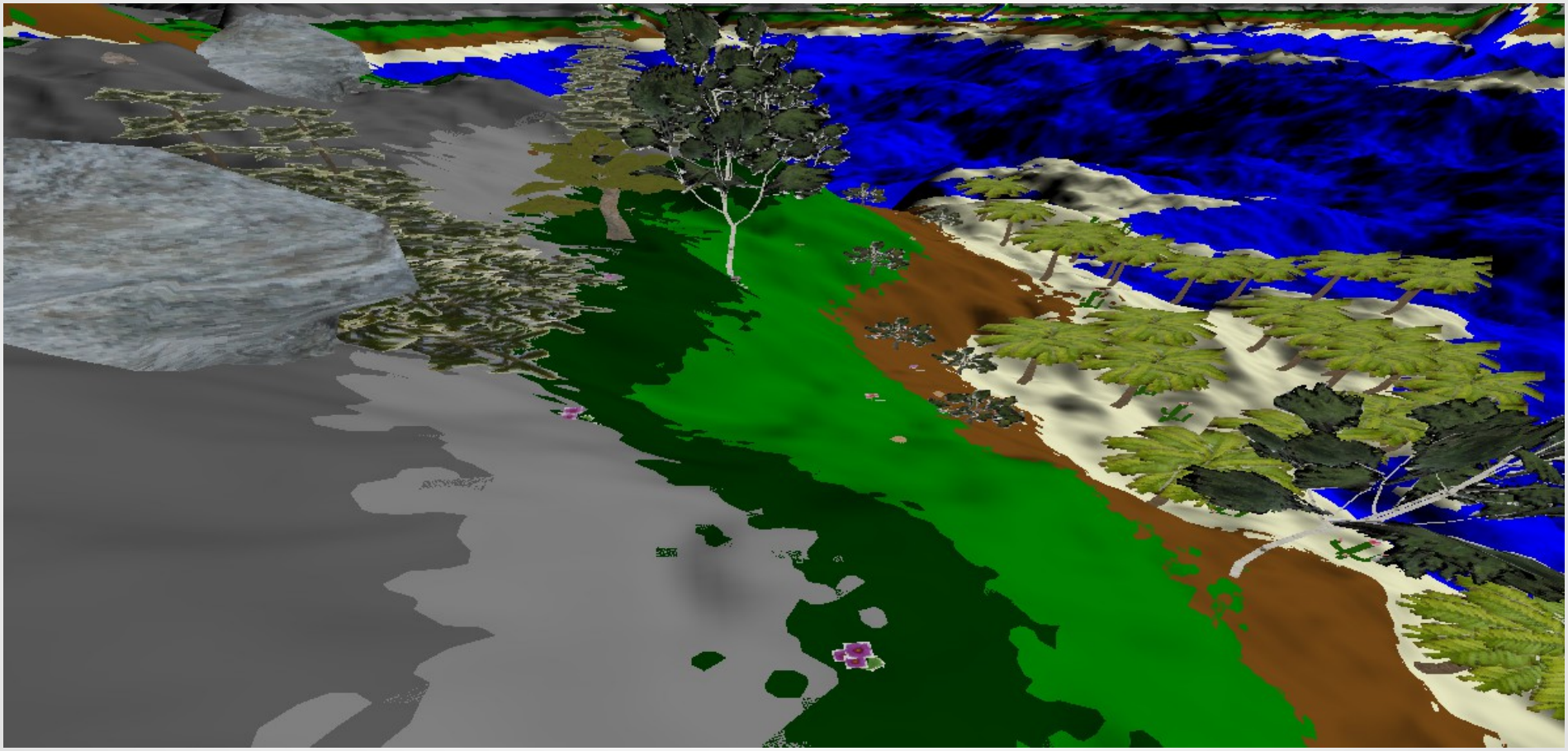
# Blender Models in OpenGL importieren



# Gliederung

- Ziele der Implementation
- Modelle Importieren
- Hauptbestandteile der zu importierenden Dateien
- Anschauungsmaterial und Shader
- Modelle auf Karte platzieren
- Besondere Schwierigkeiten

# Ziele der Implementation

- In Blender erstellte 3D Modelle in unserem Java OpenGL Programm verfügbar machen
- Modelle je nach Beschaffenheit des Untergrunds sinnvoll auf der Karte verteilen

# Modelle Importieren

- Modelinformationen (OBJ Format) anhand von Beispielen und der Spezifikation einlesen
- Materialinformationen (MTL Format) anhand von Beispielen und der Spezifikation  einlesen
- Vertex- und Indexbuffer erstellen
- Texturen einlesen
- Alle genannten Eigenschaften für jedes Teil des Models in einer Klasse zusammenfassen
- Alle Teile des Models in einer Liste speichern

# Hauptbestandteile einer OBJ-Datei

- Aufbau: Textdatei mit zeilenweisen Anweisungen gruppiert in Teilblöcke für Teilobjekte (Baumstamm z.B.)
- Vertex-Block (zeilenweise x, y, z)
- Vertextextur-Block (zeilenweise s,t)
- Vertexnormalen-Block (zeilenweise x,y,z)
- Materialname des Teilobjekt (Pro Teilobjekt ein Material)
- Face-Block (Zusammenhang Vertex, Vertextextur, Vertexnormale)
  - Je Zeile 3 oder 4 Vertices zusammengefügt (xv,xt,xn / yv,yt,yn, zv,zt,zn)
  - Stellt ein Drei- oder Viereck dar

# Hauptbestandteile einer MTL-Datei

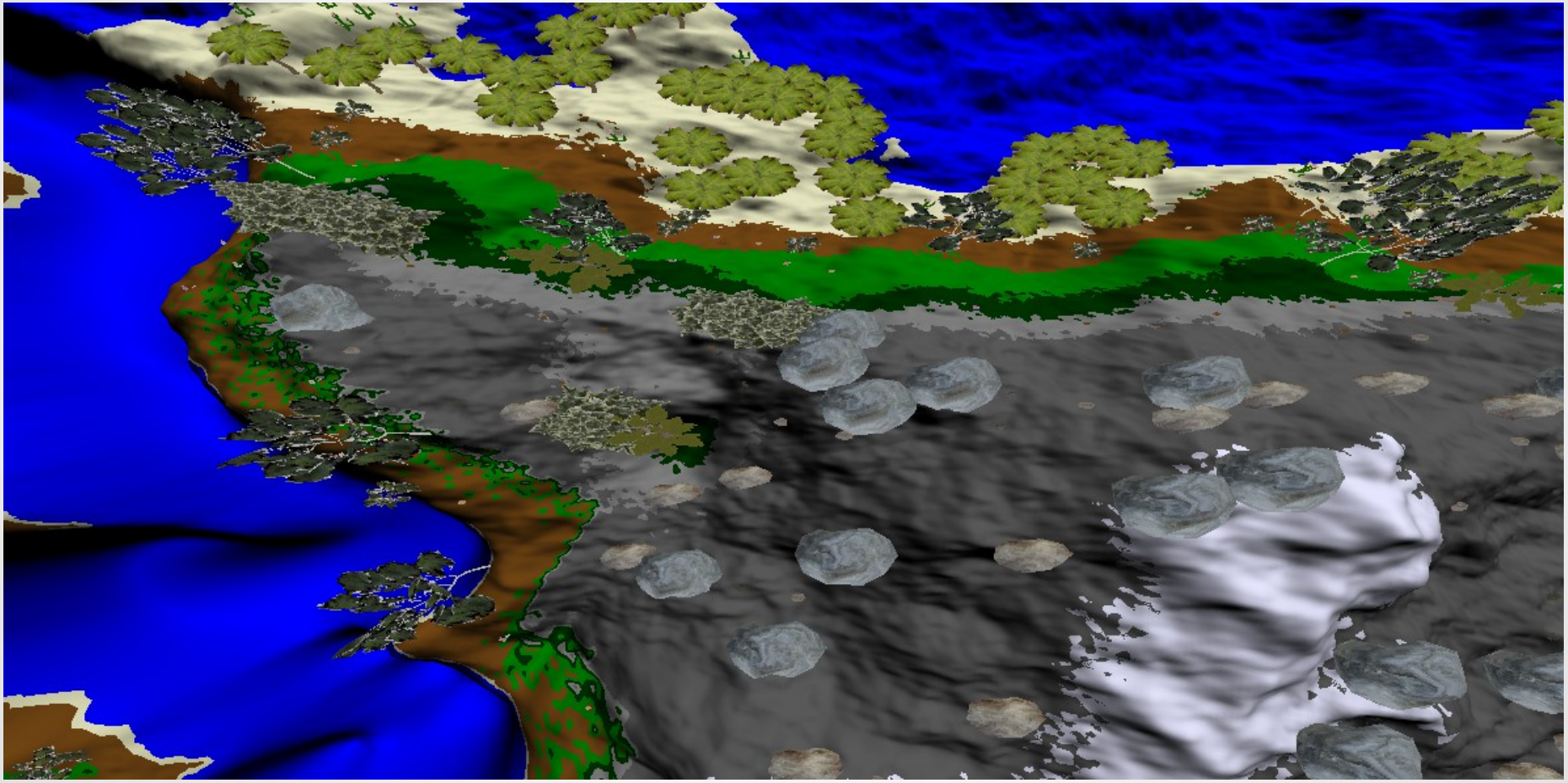
- Aufbau: Textdatei mit zeilenweisen Anweisungen gruppiert in Blöcke für jedes Material
- Materialname
- Ambient reflectance
- Diffuse reflectance
- Specular reflectance
- Dissolve factor
- Diffuse Color Map (Textur)
- Specular Color Map
- Specular exponent

# Modell nach Import

# Modelle auf Karte platzieren

- TerrainGrid durchlaufen (bspw. 2048x2048)
- Für jeden Punkt des Grids die Bodenart auslesen
  - Über gewichtete Zufallswerte bestimmen ob ein Model platziert werden soll und welches
- Passendes Model an Punkt des Grids translatieren
  - Per Skalierung und festen Faktoren, gegeben durch das Terrain und die Implementierung der ClipMap

# Art des Untergrundes und Zufallsverteilung



# Besondere Schwierigkeiten

- Scheinbar verschiedene Interpretationen des OBJ-Formats von verschiedenen Programmen vorhanden
  - Lösung: implementation verschiedener Ausnahmen
- Performanceprobleme
  - Lösungen:
    - Nutzung von Threading beim Erstellen der Buffer
    - Nur Models in direkter Umgebung zeichnen
    - Models nur auf jedem X. Punkt der Karte erlauben
- Probleme beim Merge-Vorgang durch fehlende Dokumentation und Kapselung einzelner Abschnitte
  - Lösung: ausprobieren