Paketbeschreibung - Softwareprojekt 2

HFT Stuttgart - Sommersemester 2014 - Prof. Coors

Paket Model

Die Klassen in dem Paket dienen zur Speicherung der eingelesenen Daten. Sie werden von allen anderen Paketen verwendet.

Klasse:

- City: ist Singleton, zentraler Speicherort für Gebäude (Building, ...)
- MeshInterface: abstrakte Klasse zur Speicherung des Meshs.
- Building: leitet von MeshInterface ab, dient außerdem zur Speicherung von Volumen und anderen Metadaten.
- Triangle: Speichert ein Dreieck, bestehend aus drei Punkten (Vertex).
- Vertex: Speichert ein Punkt, bestehend aus drei float-Koordinaten.
- ShadowTriangle: leitet von Triangle ab, speichert zusätzlich noch Schattenwerte.

Paket OpenCL

Dieses Paket beinhaltet die Methoden um die Volumen- und Schattenberechnung auf der GPU auszuführen.

Klasse:

- VolumeCalculator: berechnet das Volumen der Gebäude die im City-Singleton enthalten sind.
- ShadowCalculator: berechnet die Schatten der Gebäude die im City-Singleton enthalten sind.

Paket Render

Dieses Paket beinhaltet die Klassen und deren Methoden die zur 3D-Berechnung eines Stadtmodell (im CityGML-Format) notwendig sind. Hier wird die 3D-Oberfläche erstellt und die Main gestartet, mit der ein erstes "Willkommen-Frame" ausgeführt wird.

Klasse:

- Main: erste Initialisierung der Anwendung.
- StartFrame: "Willkommen Frame", hier kann man Einstellungen vornehmen wie eine (neue) CityGML-Datei auswählen, und ob man eine Stadt 3D rendern will, oder nur an der Berechnung des Volumens interessiert ist, die Funktion Schattenberechnung wird durch eine Checkbox zusätzlich noch auswählbar sein, und die Farbwahl der Gebäude könnte man hier auch noch einstellen.
- CoordinateSystem: Dies ist das Frame welches das Koordinatensystem anzeigen lässt, man kann hier im Raum per (linken) Mausdruck die Anzeige drehen und mit der Tastatur sich mit den Cursor Tasten in x (rechts und links) und in z Richtung bewegen mit y und a geht man die y Achse auf und ab. Durch ein Menüleiste per hide/show-Button, kann man die Optionen, wie z.B. die Auswahl einer neuen CityGML-Datei noch nachträglich ändern.
- CameraPerspective: Hier soll die Kameraperspektive eingestellt werden, das heißt in verschiedenen Skalierungen der Achsen, bspw. kann man bei der reshape-Funktion z mit 0,5 belegen und die Häuser werden um 50% kleiner. Und nach Modus ob man eine zentrale Projektion verlangt oder eine parallel Projektion.
  Es kann hier auch der Ausgangspunkt/Startpunkt der Sicht verändert werden und verschiedene Sichten mit dem Modus angeben werden, wie bspw. die Vogelperspektive oder die Fußgänger-Ansicht.
- CityMap3D: Hier werden auf dem Koordinatensystem die Gebäude Stück für Stück gerendert, einzelne Gebäude können durch die im Parent (CoordinateSystem) vorhandenen MouseListener selektiert werden und eine Information zum Gebäude wird ausgegeben.
- GraphicalBuilding: Das Gebäude wird aus den Dreiecken der Polygonen des Building-Objektes erstellt und in die OpenGL-Logik überführt.

Externe Abhängikeiten:

Paket Parser:
Daten müssen im Parser transformiert (skaliert) und trianguliert (aufteilen in Dreiecke) sein, so dass man problemlos mit OpenGL die Polygone eines Gebäudes durch die Eckpunkte in Dreiecke zusammensetzen kann.

Paket Parser:

Wir benötigen zusätzlich zu den Klassen im Parser-Klassendiagramm bisher Objekte aus dem model-Package:
- Building
- Vertex
- Triangle

Unsere public Methoden:
parse():
- liest die Polygone ein
- transformiert die Koordinaten von den bisherigen Werten zum Ursprung
- konvertiert die Polygone zu Vertices
- erstellt ein neues Building-Objekt (mit Building-id, dem polygon und den vertices)

export():
- kann "id,volume" exportieren, wenn sie eine Liste von Buildings nach den Berechnungen erhält

triangulate():
- nutzt die jogl-Bibliothek um aus den Polygonen Dreiecke zu erzeugen und sie in eine List<Vertex> zu speichern

transform():
- transformiert die Koordinaten (näher) an den Ursprung des Koordinatensystems zur einfacheren Weiterverarbeitung