Entwurf und Programmierung einer Rendering Engine

186.166 – WS 2.0

Robert F. Tobler

VRVis Research Center
Vienna, Austria

Oranisatorisches

Vorlesung

- jeden Montag, **15:15 (s.t) - 16:45**
- Seminarraum Institut für Computer Graphik und Algorithmen

Übung

- eigene Module für eine Rendering Engine
- einfache Rendering Engine wird bereitgestellt in die das Modul eingebunden werdn kann (nicht zwingend):
  Microsoft Visual Studio / C# / Direct-X

Prüfung

Abgagbe des Programms + einer schriftlichen Ausarbeitung

- bis 2 Tage vor dem Prüfungstermin
- Analyse des implementierten Moduls in 2-4 Seiten:
  - was wurde implementiert, sowie Analyse der Performance

Mündlicher Prüfungstermin

- Ende Jänner bis Ende März nach Terminvereinbarung (email mit mind. 2 möglichen Terminen an rft@vrvis.at) am VRVis
- Vorführung des implementierten Programms
- zwei Fragen zum Stoffgebiet
  - Details sind weniger wichtig, das Verständnis ist wichtig

Bereitgestellte Entwicklungsumgebung

Programmierumgebung

- visual studio community: kostenlos von Microsoft
- möglicherweise DirectX (Details am 9. November in der Übungsvorbesprechung): https://www.microsoft.com/en-us/download/details.aspx?id=6812
- AARDVARK-Student:Rendering Kernel in den das Module hineingehängt werden kann.
  Kann verwendet werden, muss aber nicht!

  wird ab 9. November verfügbar (= Übungsvorbesprechung)
- jeder verwendet seinen eigenen Rechner

Kontakt

Robert F. Tobler

- VRVis Forschungszentrum
  Donau-City Str. 1/3
- rft@vrvis.at
- **Bitte eine email mit Name und Matrikelnummer an mich zur informellen Anmeldung!**

VO-Homepage

- Institut für Computergraphik und Algorithmen
- http://www.cg.tuwien.ac.at/courses/RendEng/

Motivation

Echtzeit 3D-Darstellung

- große Datenmengen
- statische und dynamische Geometrie
- Verwendung der Graphikhardware
- Berücksichtigung von Special-Effects

Anwendung

- Terrain Visualisierung, Architektur, Lichtsimulation (Global Illumination), Spiele, …

Fragestellungen

Wie strukturiert man eine Rendering Engine?

- Welche Interfaces und Module braucht man?
- Welche Daten werden transferiert?
- Wie wird Speicher gehandhabt?
- Wie sollen die Daten im Speicher strukturiert werden?
- Wie soll man optimieren, parallelisieren?

Wie wird die Graphikhardware berücksichtigt?

- Was sollte man in Shadern abwickeln?
- Was sollte man auf der CPU abwickeln?

Fragen die in dieser VU *nicht* behandelt werden

Grundlagen der Graphikprogrammierung

- Vektor- und Matrizenrechnung, Standard-Renderingpipeline, Schattierungsmodelle

Shader Programmierung

- unterscheidet sich von Graphikkarte zu Graphikkarte und Generation zu Generation

Details von DirectX und OpenGL Programmierung

- nur generelle Aspekte der hardwarenahen Graphikprogrammierung werden berücksichtigt

Themen (1)

Organisation von großen Datenmengen

- moderne objektorienterte Datenstrukturen

Handhabung von 2D und 3D Datenarrays

- Generische Datenstrukturen und funktionale Programmierung

Using/Building Code-Generators

- Erzeugen von Gruppen von Klassen mit konsistenten APIs

Notwendige Graphik-Datenstrukturen

- meshes, lines, point-sets, rendering properties

Themen (2)

Pre-Processing Algorithmen für Graphik-Daten

- merging, splitting, sorting by properties, mesh topology

Strukturierung von Rendering Engines in Module

- interfaces, data-flow, control structures

Notwendige 3D-Suchstrukturen für Rendering Engines

- k-d-Trees, BSP-trees für Rendering von Transparenzen

Inkrementelle Berechnung für Dynamische Daten

- Funktionale Programmierung zur Handhabung dynamischer Daten

Themen (3)

High-Level Szenenorganisation

- Szenengraphen, Instancing, Traversierung, LODs

Subdivision & Procdedural Generation of Detail

- CPU based vs. GPU based, smooth surfaces, subdivsion surfaces

Rendering von Großen Datenmengen

- terrain-rendering, rendering precision, caches

Parallelization of Rendering Engines

- parallel rendering, splitting of views, parallel algorithms

Danke für Ihre Aufmerksamkeit!

Robert F. Tobler
rft@vrvis.at

vrvis