VU Entwurf und Programmierung einer Rendering-Engine

Organization

186.166 - WS 2.0

Harald Steinlechner, Georg Haaser, Christian Luksch, Stefan Maierhofer
Organization

● **Vorlesung**
  ○ Monday, 16:15 (s.t.) - 17:45
  ○ Seminarraum 186, Institut für Computer Graphik und Algorithmen
  ○ ECTS efforts: approx half/half

● **Übung**
  ○ As a project, implement a module for a rendering engine
  ○ Topics can be chosen by students
  ○ There will be an extra “Vorbesprechung” for the exercise: 05.11.2018
    ▪ Including ideas and a discussion of possible topics
Student project

- Will have extra lecture which describes tasks more precisely (05.11.2018)
- Similar to previous years:
  - Rendering and optimization a scene. This includes:
    - Geometry processing (e.g. Terrain generation, Meshes,...) or model loading
    - Acceleration data structure or optimization algorithm
    - Rendering of the scene
Exam

- Hand in (per email) the project + a written report
  - Till 2 days before the exam date
  - Written report (2-4 pages)
    - Description of the project
    - Description of the used techniques
    - Analysis of the performance

- Oral exam
  - End of January till end of March
    - Email with 2 possible dates to hs@vrvis.at
  - Demo of the project
  - Two questions of the lecture content
    - Details not that important, but understanding of the topics.
Contact

Harald Steinlechner

- VRVis Research Center, Donau-City-Straße 11
- hs@vrvis.at
- Register in TISS
- When projects/team is fixed: write email with task description to hs@vrvis.at

VO Homepage

- https://www.cg.tuwien.ac.at/courses/RendEng/
The mission of a rendering engine….

- Provide easy to use software components...
- which can be used to solve rendering engine tasks (like a toolbox)

In order to accomplish this, we need:

- Algorithms and Datastructures
- Graphics API & Hardware Insights
- API design
- Domain specific languages (e.g. scene description)
Content of this LV

- Requirements for the design of rendering engines
- Hardware and Graphics APIs (OpenGL, Direct3D, Vulkan,..)
- Scene Representation (Scene graphs, display lists, command buffers,...)
- Static and Dynamic Data (Incremental Update Techniques)
- Optimizations (Caching, Culling, Level of Detail, Bounding Volume Hierarchies, Just-In-Time Optimization)
- Resource Management
- Domain Specific Languages (HLSL, Spark, FShade, Semantic Scene Graph,..)
- Reusable Components/Design for Rendering Engines
About the LV team & Aardvark

- LV Team is basically the aardvark core development team.
- Active development of the aardvark rendering engine since 2006 with Robert F. Tobler.
- Roberts mission: easy to use but high-performance rendering engine.
- Aardvark - An Advanced Rapid Development Visualization and Rendering Kernel
  - Heavily used in research + industry projects
<table>
<thead>
<tr>
<th>Timeline</th>
<th>Language/Graph Type</th>
</tr>
</thead>
</table>
| Till approx 2002 | Aart (Obj C)  
Approx. 2002 Ave (C++)  
Traditional Scene Graph | Managed language for rendering engine?  
Approx. 2005 Aardvark (C#)  
Semantic Scene Graph [Tobler 2011] |
| Approx. 2005 Aardvark 2008 (C#) | Clean Semantics for Rendering |
| Approx. 2008 | Aardvark 2010  
Lazy Incremental Computation  
For efficient Scene Graph Rendering [Wörister et al. 2013] | Performance!  
Aardvark 2015  
Composable Shaders  
[Haaser et al. 2014]  
Towards Incremental Computation,  
Attribute Grammars for Incremental Scene Graph Rendering |
| Approx 2016 | Fast and flexible!  
aardvark.rendering  
aardvark.base | Usability, Remote Rendering,  
Aardvark in the browser  
2017  
Vulkan, ELM architecture,  
Aardvark goes web |
Challenges

● Size of data-sets
  ○ Often requires out of core approaches

● Dynamic and static geometry

● Efficient graphics hardware utilization

● Support for special effects
  ○ APIs for accessing special hardware features
  ○ Provide mechanisms to specify for example shaders and post processing

● Many different application areas: Focus on real-time applications
  ○ Terrain, laserscan, reconstructed data, game levels
  ○ Architecture and planning
  ○ Light simulation (Global Illumination)
  ○ Games
  ○ Interactive Editing applications
Design Space

How to structure a rendering engine

- What interfaces and modules useful
- How to transfer data
- How to manage memory (we have GPU and main memory)
- How to store data in memory (e.g. for efficiency reasons)
- How to optimize, how to make use of multiple CPU cores

Graphics hardware specific questions

- What to compute in shaders
- What to compute on CPU (in what precision?)
What to expect

● Tools/Algorithms/Concepts to implement rendering engines
● Hardware/Graphics API insights
● How to structure rendering engine into modules
  ○ (Low cost) abstraction techniques
  ○ Compiler techniques
● Important data-structures in practise
  ○ k-d-Trees, Octrees
● Performance considerations
  ○ Optimizations (how to pack buffers etc)
  ○ Costs of programming language abstractions (e.g. can we afford virtual function calls?)
● How to manage large scenes (performance + memory)
● Approaches for implementing lighting/material systems
What to expect

- Dependencies and incremental computation for rendering engines
  - Efficient ways to handle dynamic data
- Scene representation
- Rendering of big scenes
  - Terrain-rendering, rendering precision, caches
- Parallelization for rendering engines

- Not content of this LV:
  - Graphics programming tutorial
  - How to use existing engines
  - How to implement concrete tooling (e.g. level editor, material editor)
Timeline

- 08.10.2018 - Organization, Introduction
- 15.10.2018 - Scene representation
- 22.10.2018 - Optimization techniques for rendering engines
- 29.10.2018 - Applied Data Structures
- 05.11.2018 - Representing fully dynamic scenes, Aardvark Tutorial, Vorbesprechung UE
- 12.11.2018 - Optimization techniques for fully dynamic scenes
- 19.11.2018 - Domain Specific Languages for Rendering Engines, Composable Shaders
- 26.11.2018 - Materials and Lights for Rendering Engines
- 03.12.2018 - Shading System and Global Illumination
  - Including lightmap packing, instant radiosity, deferred rendering techniques,.. 
- 14.01.2018 - Questions regarding the lecture/project
Questions