Algorithm Engineering Lab: Ray Tracing

Jenette Sellin Markus Pawellek

8. Februar 2018

Gliederung

Goal of the Project

Goal of the Project

Application / Library:

- Ray Tracing
- Smoothed Particle Hydrodynamics (SPH)

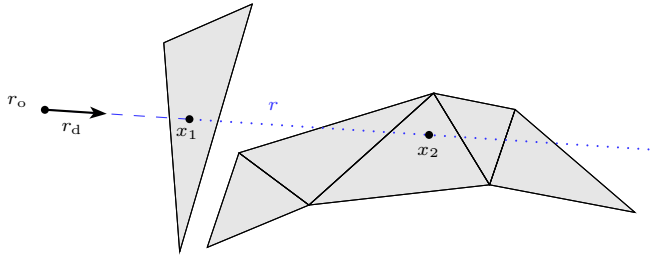
Example:

- ▸ NVIDIA Kepler real-time raytracing demo at GTC 2012 - The Verge

Current state: Starting the application...

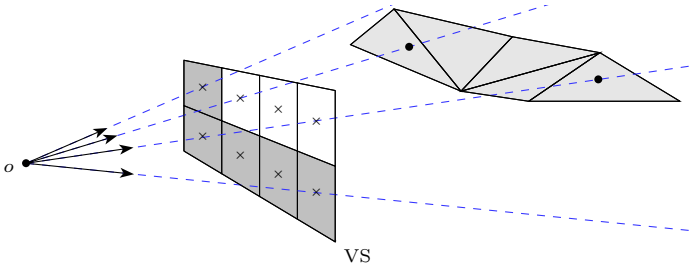
Ray Tracing Background

Ray Tracing Background



- determine the visibility of surfaces in a scene from an origin
- use a ray to compute nearest intersections

Ray Tracing Background



- trace a bunch of rays for every pixel
- apply shading and show the result

Starting Point

Starting Point

Group of 3 People:

- different background and knowledge base
- different hardware
  - for example: ordinary laptops, non-ordinary laptops, desktop computer
- different operating systems
  - for example: Windows 7, macOS, Ubuntu, Arch Linux

Setting up the Environment

Setting up the Environment

- tools known from the lecture:
  - Intel C++ compiler, git, CMake, clang-format, etc...
- choice of code editor:
  - Sublime Text 3 with given configurations for C++ and clang-format
- C++ coding style
  - Google C++ style guide with some minor changes

Setting up the Environment

- git branching model
- git commit message style
- learning modern CMake
- installing GLUT-library in Windows 7

Setting up the Environment

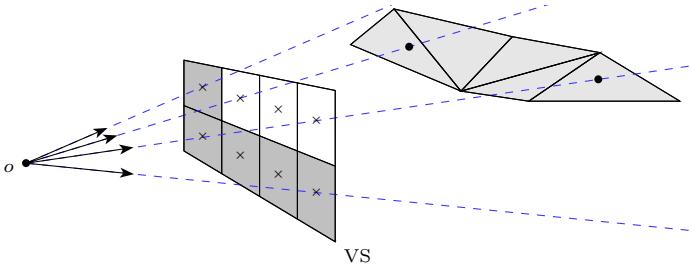
- communication via E-Mail and GitLab-Issue system
- additional meeting every week
- one feature per group member per week

Implementation

Implementation

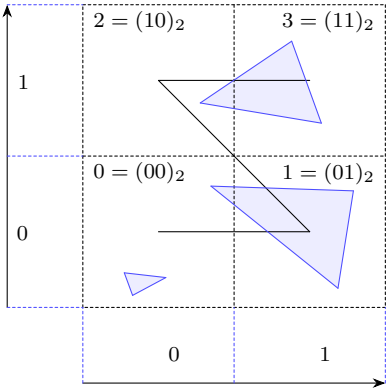
- OpenGL scene viewer with user interaction
- scene loader: STL-files, OBJ-files
- camera
- FPS measurement

Naive Ray Tracing



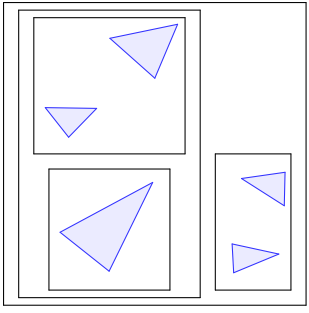
- for each ray check existance of intersection with each primitive

Morton Code



- enclose scene in a bounding box
- order primitives of scene with respect to their morton code

Bounding Volume Hierarchy (BVH)



- use morton code to build a tree structure
- enclose primitives of every node in a bounding box

Parallelization

naive ray tracing:

- static parallelization over pixels

BVH ray tracing:

- FPS depends on camera position and screen pixel
- use blocking principle with block size of 8
- dynamic scheduling in for loop

Vectorization

Now:

- done by compiler through Eigen-library

Future:

- vectorization of intersection computation
- vectorization of BVH traversal
- vectorized random number generation

Serialization / Deserialization

Serialization / Deserialization

Goal:

- Track user camera movement over time while running spray
- Serialize the movement by storing camera parameters and time data externally
- Deserialize by reading the parameters back into the camera object
- Render the recorded camera movement over time

Serialization / Deserialization

Limitation:

- Frame resolution is bounded by the rendering speed during recording; playback is always as choppy as the rendering was on the machine that recorded it.

Solution:

- Interpolate between recorded camera parameters, display extra frames for a smoothing effect.

More Future Work

More Future Work

- template-based library
- more objective measurements
- implementing physicaly based shading of primitives
- adding smoothed particle hydrodynamics
- and even more...

Thank you very much for your attention!