Real time intersections on Space Scale Cube based data

Thesis project of Mattijs Driel

STW | Amsterdam | 2014-11-19

Outline

- Problem definition
- Approach
  - Implicit intersections
  - Data structure
  - Unsolved problems
- Demonstration

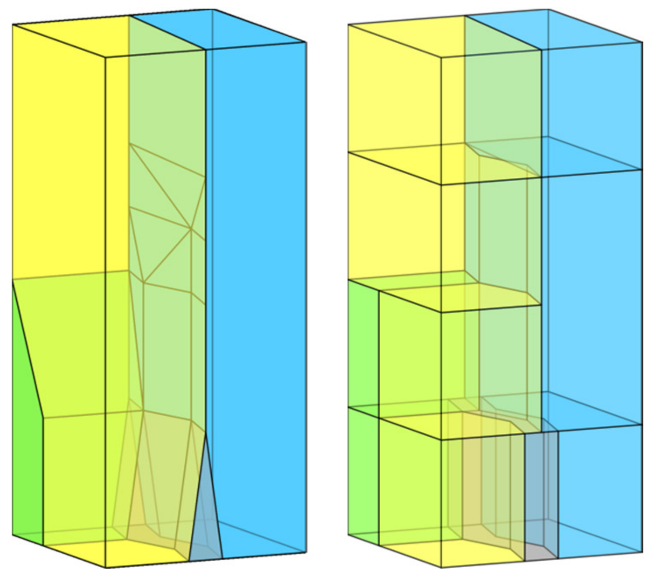
Problem Definition

Given: Space Scale Cube smooth data

- Horizontal intersections are non-trivial

Explicit intersection

- Or geometric / algebraic intersection
- 3d Plane – Mesh intersections
- Gives the exact solution as a polygon set, which can be displayed using vector data rendering software.
- Calculation is costly on dense data sets (even with good acceleration structures), so not real time



Problem Definition

Move intersecting to the GPU

- Increasingly common in consumer hardware

Downside

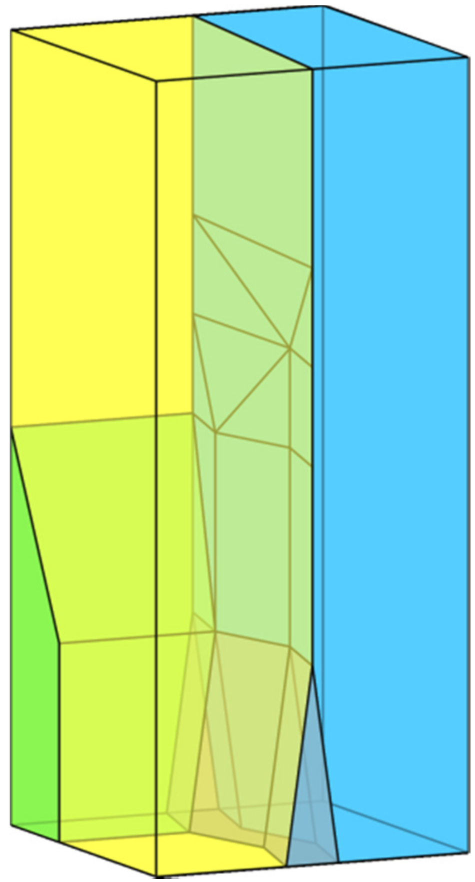
- Explicit intersection algorithms are not very parallelizable (the main strength of the GPU)
- Needs high memory bandwidth
- Needs GPGPU, which is not widely supported yet

Still, GPU based approach can be useful

Approach

Instead use implicit intersecting. Why?

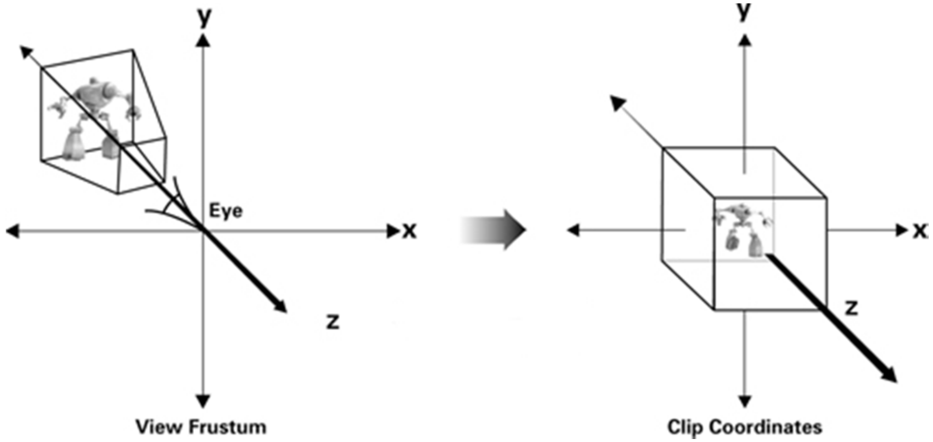
- Exact results aren't necessary, pixel precision is enough for the end user
- Graphics programming has always included clipping techniques
- More direct benefit from GPU strengths



The approach will require a few assumptions, specifically an input data structure of closed, tightly fitting polyhedra (as illustrated).

Implicit Intersections

In games, clipping issues occur when the virtual camera is placed too close to the geometry: Part of the geometry won't be drawn.



We can use this if we place the camera properly.

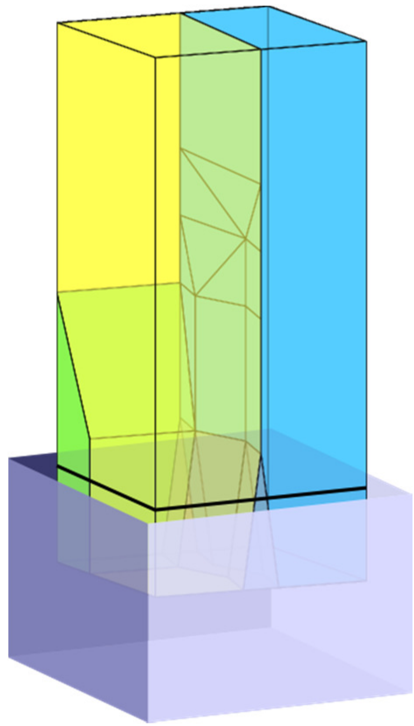
Implicit Intersections

Place the camera such that the desired intersection plane matches the clipping region.

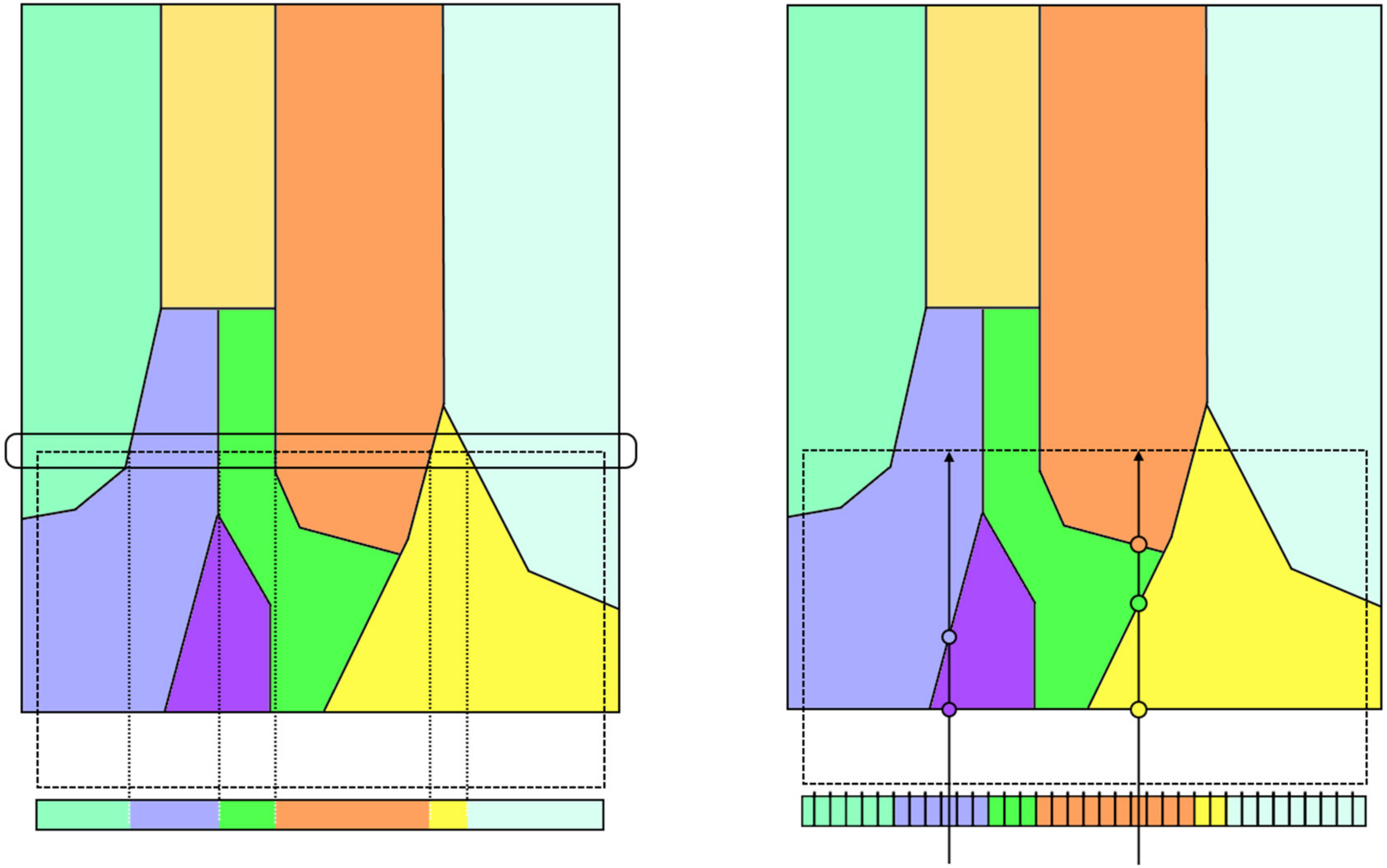
- All geometry outside the clipping region is automatically discarded.

New problem: How do we render the remaining geometry to show the desired cross section?

- We want to avoid calculating the exact perimeter.



Implicit Intersections



Implicit Intersections

Solution is to use the following render settings:

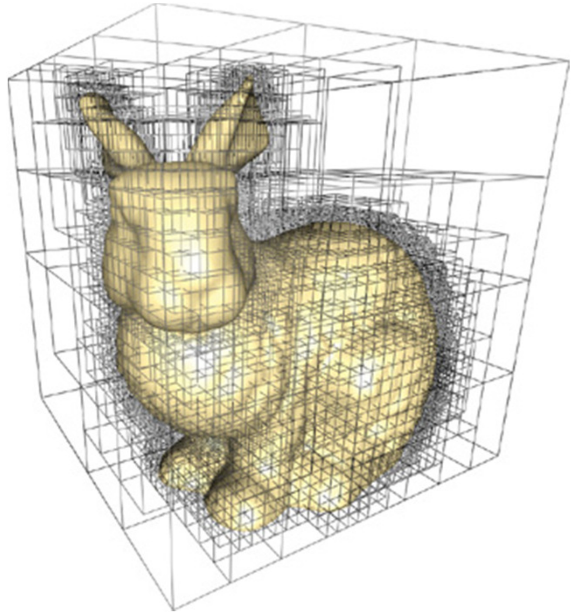
- Discard (cull) outside facing triangles for each separate polyhedron
- Use depth testing to discard each pixel except for the closest ones

This is enough to fill pixels with the correct corresponding terrain feature. Some complexities remain though:

1. There has to be a 'bottom'
2. Segmentation of the data (large data, streaming)
3. Needs filtering of what geometry is guaranteed to be outside of the clipping region, and thus not visible

These can be remedied using a good data structure

Data Structure



Proposed structure is an axis aligned octree

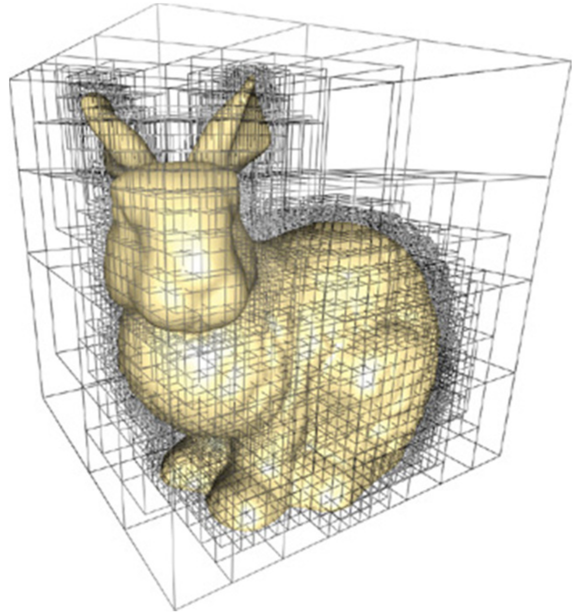
- Geometry is stored in the box shaped octree leaves.
- Require that every leaf is individually closed.

This guarantees that

1. Parts of the geometry are always bounded on the bottom, as the boxes have a predetermined size, and have no gaps.
2. Tree structure provides density-sensitive segmentation.
3. Visibility testing is trivial in planar intersections[*], if the octree internal structure is known. Only those octree leaves intersecting the plane are rendered.

Data Structure

[*] Octree structures allow for some clever visibility testing that applies to non-planar intersection shapes.

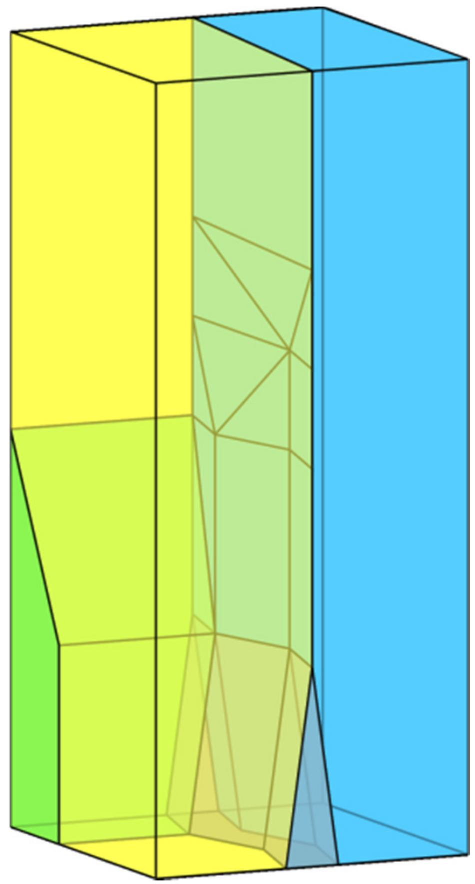


Using GPU shaders, both visibility testing and clipping is possible for curved and polygonal intersection shapes.

Using non-octree data structures is likely possible, but the above advantages might become impossible.

Unsolved problems

- This approach solves only volumetric objects from SSC. If objects are defined without volume (roads at high scale levels), they can't render properly.



Progress

Implementation details

- Octree data generated from wavefront obj (requires closed polyhedra) in a preprocessor (c++)
- Rendering system requires opengl es 3.0 for integer texture support and other minor features (java & libgdx)

Thesis work is ongoing

- Implementation is functional
- Some features incomplete

Conclusion

- Real time intersection visualization of SSC based data
- Utilizes the GPU pipeline
- Octree data structure that ensures reliable operation

Demonstration (feel free to ask questions.)