Occlusion Culling for Sub-Surface Models in Geo-Scientific Applications

John Plate

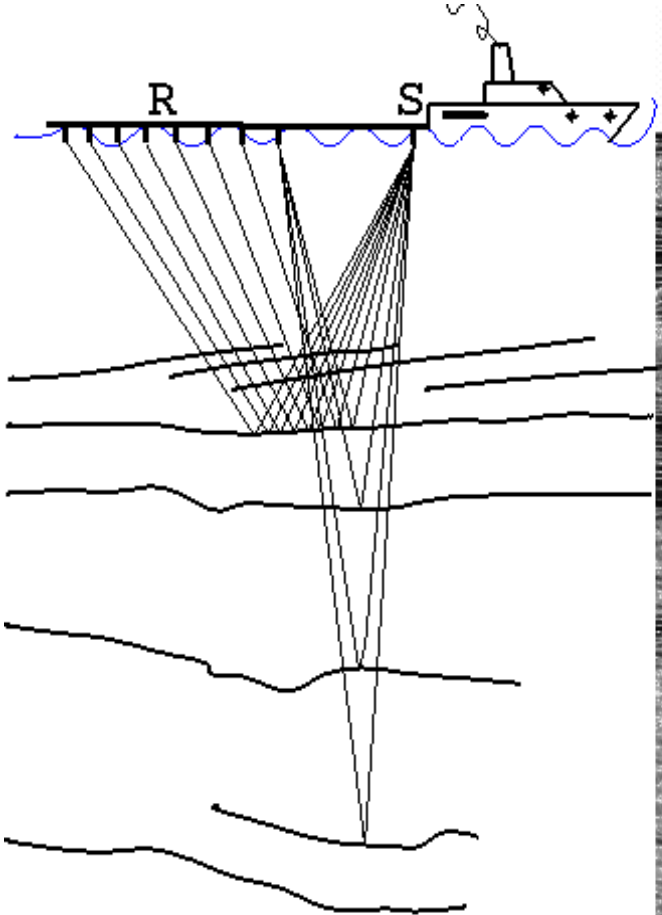
Anselm Grundhöfer, Benjamin Schmidt, Bernd Fröhlich

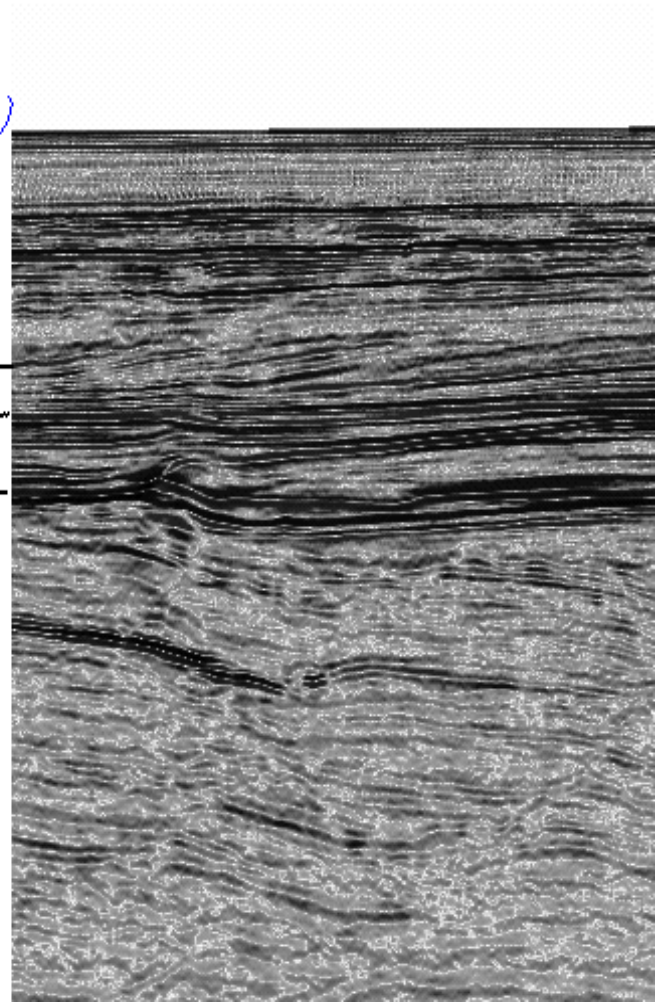
john.plate@imk.fraunhofer.de

Virtual Environments Research
Fraunhofer IMK
Sankt Augustin, Germany

Bauhaus-University Weimar
Weimar, Germany

Reflection Seismic

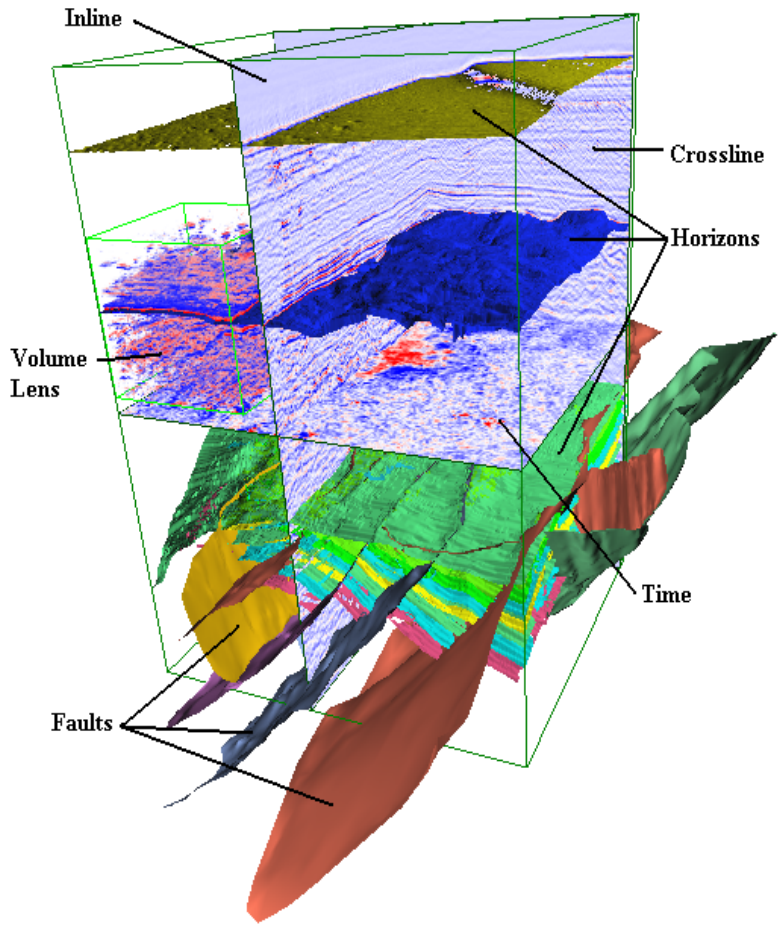




Occlusion Culling

Used Data Types

- **Horizons**
  - High resolution
  - Layered
  - Often height fields
- **Faults**
  - Often lower resolution than horizons
- **Volumes**
  - Slices
    - Opaque
    - Few polygons
  - Volume rendering
    - Semi transparent
    - Does not significantly occlude – may be occluded



Occlusion Culling

Occlusion Culling

- **Basic idea**
  - Don’t render invisible parts of the scene
  - But how to know which parts are invisible?
- **Naive approach**
  - Sort scene front to back
  - For each object in front to back order
    - Submit occlusion query using the bounding box
    - If object’s bounding box is visible
      - Render object
- **Helps for scenes with high depth complexity**
- **Requires sorting**
- **Bounding box is not very precise**
- **Requires waiting for results of occlusion queries – stalls graphics pipe**

Occlusion Queries in Hardware

- **nVidia and ATI support occlusion queries in hardware**
  - Setup occlusion query
  - Render object
  - Result: the number of the visible pixels of the object

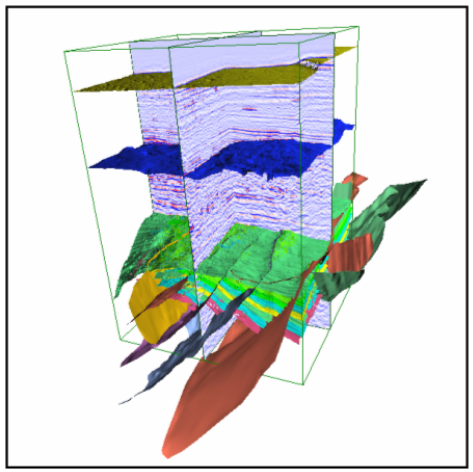
- **Problem: if you ask for the result right after you render, the graphics pipeline needs to finish computing the partial image before the result can be returned**

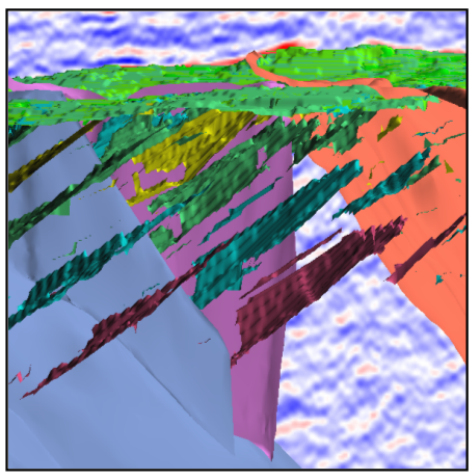
- **Better: render a collection of objects and ask for the occlusion results afterwards**

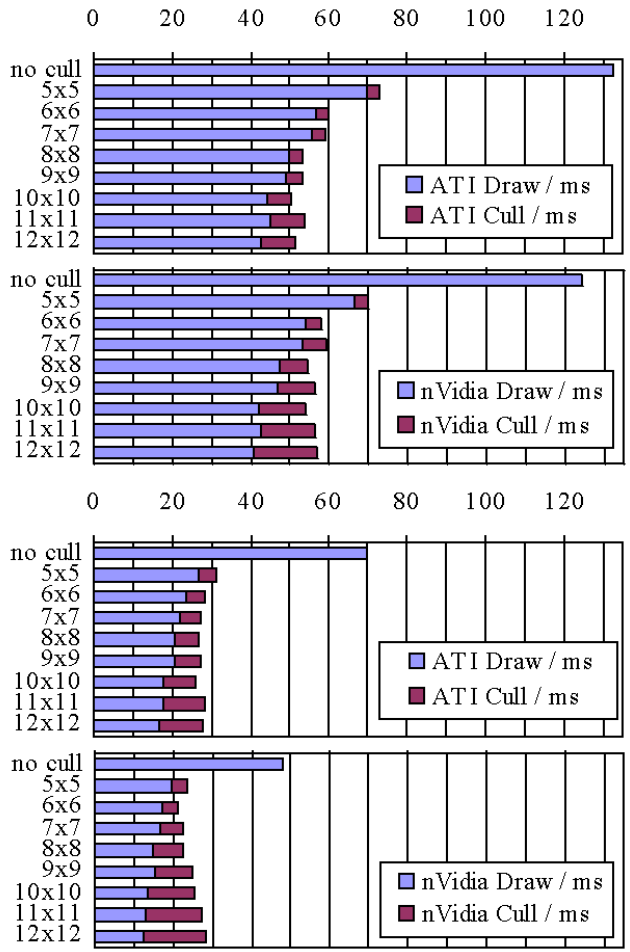
Basic Algorithm

- **Preprocess**
  - Generate low resolution objects
  - Divide objects into tiles
- **First pass – create depth image**
  - Disable lighting, shading, texturing and frame buffer writes
  - Clear depth buffer
  - Render low resolution objects and additional occluders
- **Second pass – query visibility**
  - Disable depth buffer writes
  - Render low resolution objects with occlusion queries
  - Read results of occlusion queries. Object is “visible”, if the number of visible pixels is above of a defined threshold
- **Third pass – render visible objects**
  - Enable lighting, shading and texturing
  - Enable depth and frame buffer writes and clear both buffers
  - Render all “visible” high resolution objects

Benchmarks







Occlusion Culling

Conclusions and Future Work

- **Simple and easy to implement**
- **Efficient**
  - No sorting
  - More precise than bounding boxes
  - Stalls graphics pipe only once per frame
- **Works for dynamic scenes**
- **Small overhead**
  - Rendering of low resolution objects (1:100 reduction works)
  - No lighting, texturing or pixel shaders
  - Worst case: requires three times the fill rate
  - Reduce fill requirements: use smaller image for pre-rendering
- **Should be combined with level-of-detail rendering**
- **Develop occlusion relationship preserving mesh simplification algorithms and measures that predict potential occlusion errors**