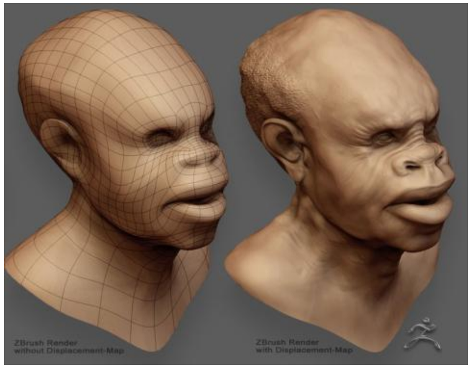
Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering

*Art Tevs**, Ivo Ihrke** and Hans-Peter Seidel*

**MPI Informatik, Saarbruecken, Germany*
***University of British Columbia*
February 17, 2008

Overview

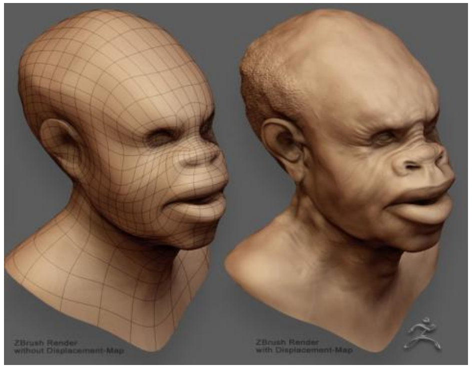


- Height Field Rendering
  - Per-Pixel Displacement mapping
  - Effect of a highly tesselated mesh
  - Used in games and scientific visualizations
- Ray - Height Field Intersection
  - Depth-Map intersection (refraction, reflection)
  - Collision detection

Related work

- Robert L. Cook, Shade Trees, Siggraph'84
  - one of the first mentions of displacement mapping
- Fabio Policarpo et al., I3D'05 and I3D'06
  - real-time relief mapping
  - relief mapping of non-height-field surface details
- F. Policarpo and M. Oliviera, GPU Gems 3’07
  - relaxed cone step mapping (CSM)
- Wyman et al., Szirmay-Kalos et al., Hu and Qin
  - ray depth-map intersection for refraction and reflection
- Kyoungsu Oh et al. (VRST'06)
  - pyramidal displacement mapping

Overview



- Problem Statement
  - Fast, accurate and scalable algorithms are desired
  - Fast pre-computation time of acceleration structure
  - GPU optimized

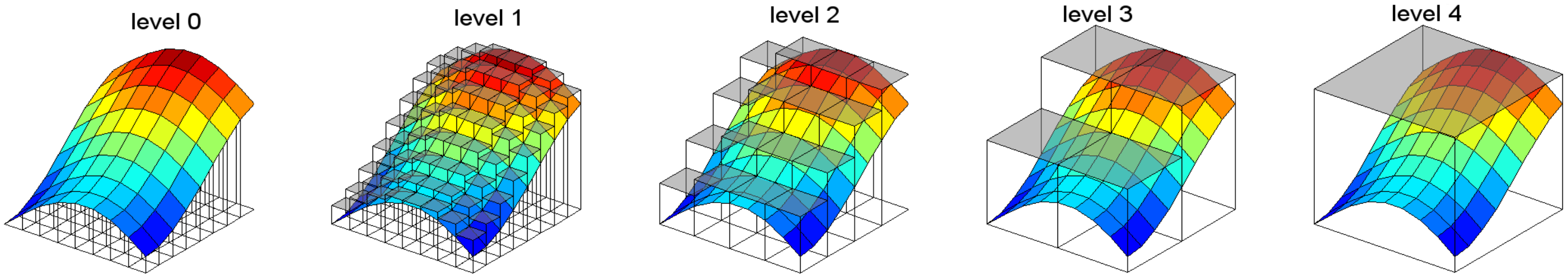
Outline

- Maximum mipmaps data structure (MMM)
- Intersection Algorithm
- Performance
- Optimization
- Discussion
- Your questions

Maximum Mipmaps

- Equivalent to fully sub-divided quad-tree [Samet 1990]
- Already used in CG publications
  - soft shadow rendering [Guennebaud 2006]
  - geometry image intersection [Carr et al. 2006]
- MMM data structure is dynamic
  - precomputation time in order of ms
- 1/3 additional memory required

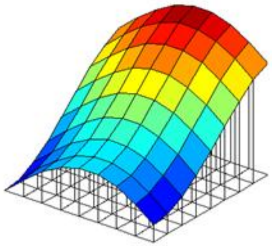
Maximum Mipmaps

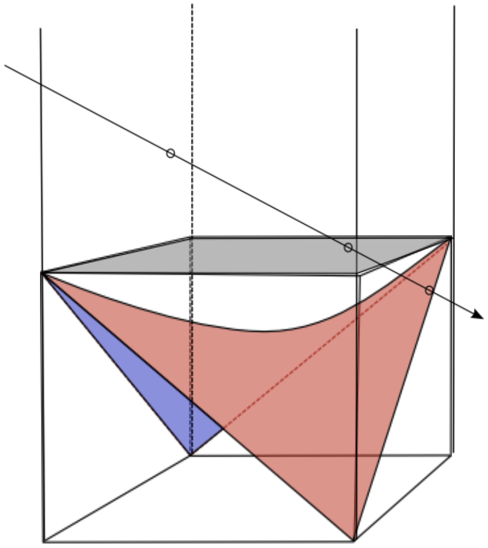


- Collection of bilinear patches placed on a regular grid
- Level 0 – vec4 (RGBA) value storing height of the bilinear patch data points
- Level 1 to n – maximum height of underlying patches
- due to optimized hardware the construction time is very fast

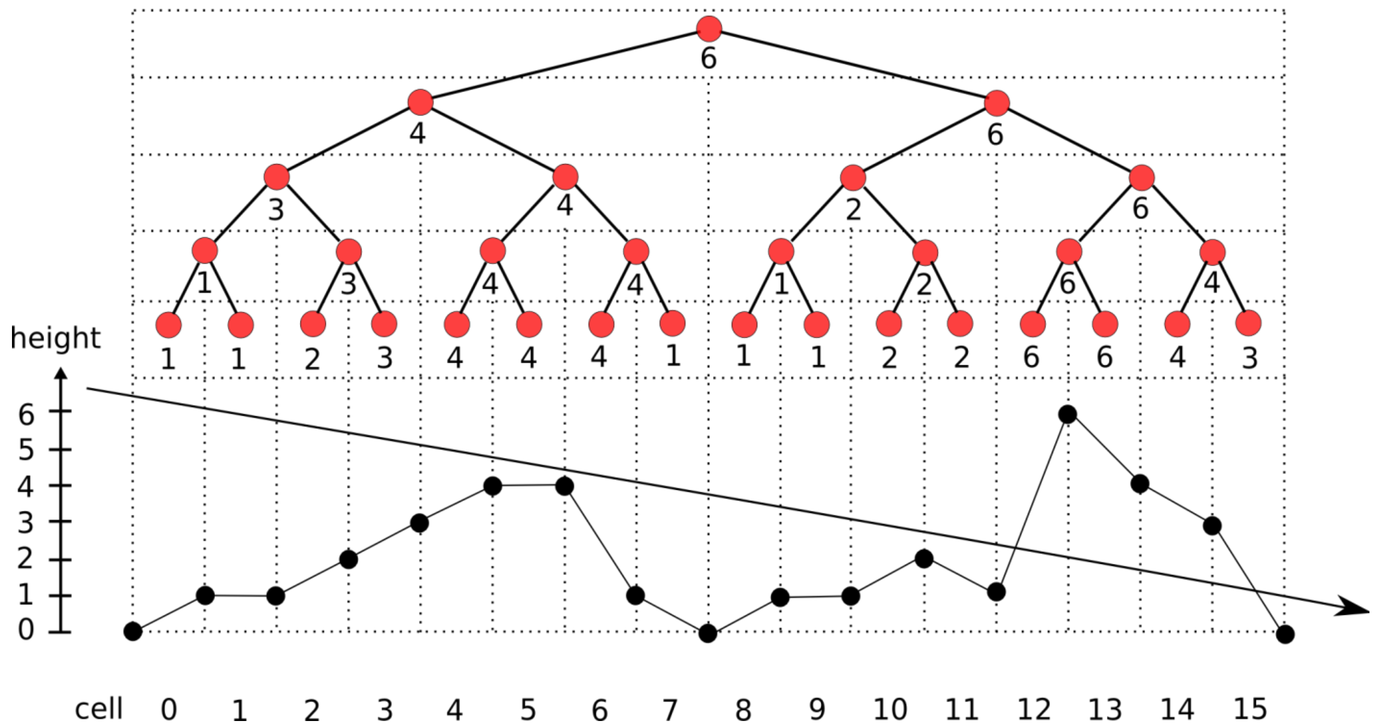
Intersection Algorithm

- Utilize the MMM data structure as quad-tree to perform *empty space skipping*
  - Adaptive ray step length
  - Hierarchical traversal, starting at the highest level
  - Bilinear patch intersection or binary search in level 0



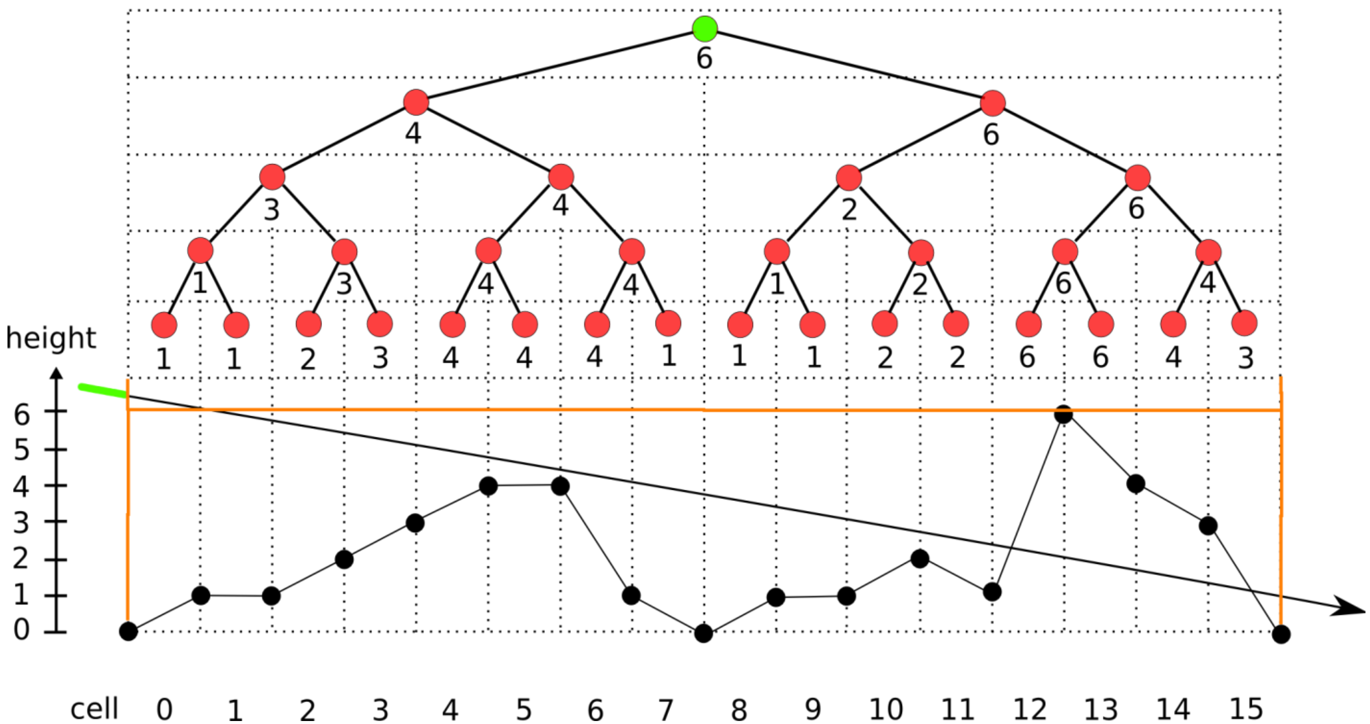


Intersection Algorithm



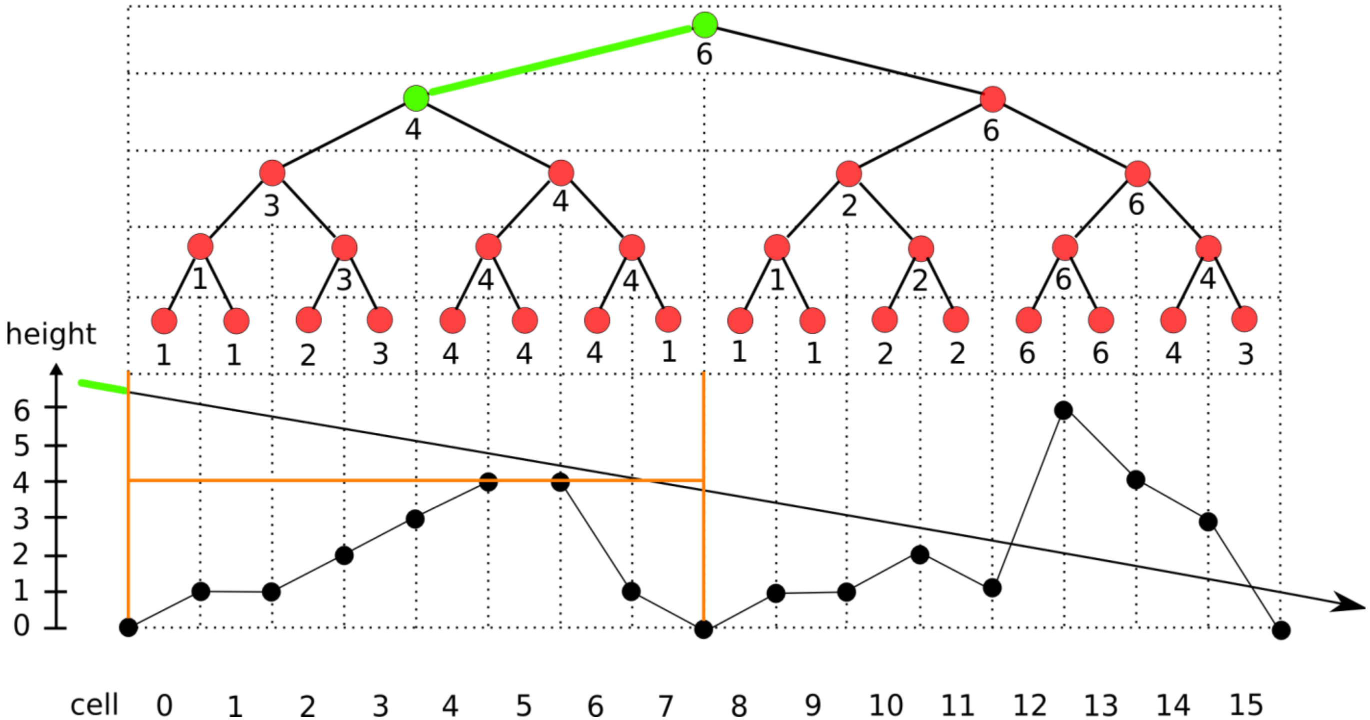
- Example of a Ray – Height Field Intersection
- 1D height field and the corresponding MMM data structure
  - linear elements in the finest levels

Intersection Algorithm



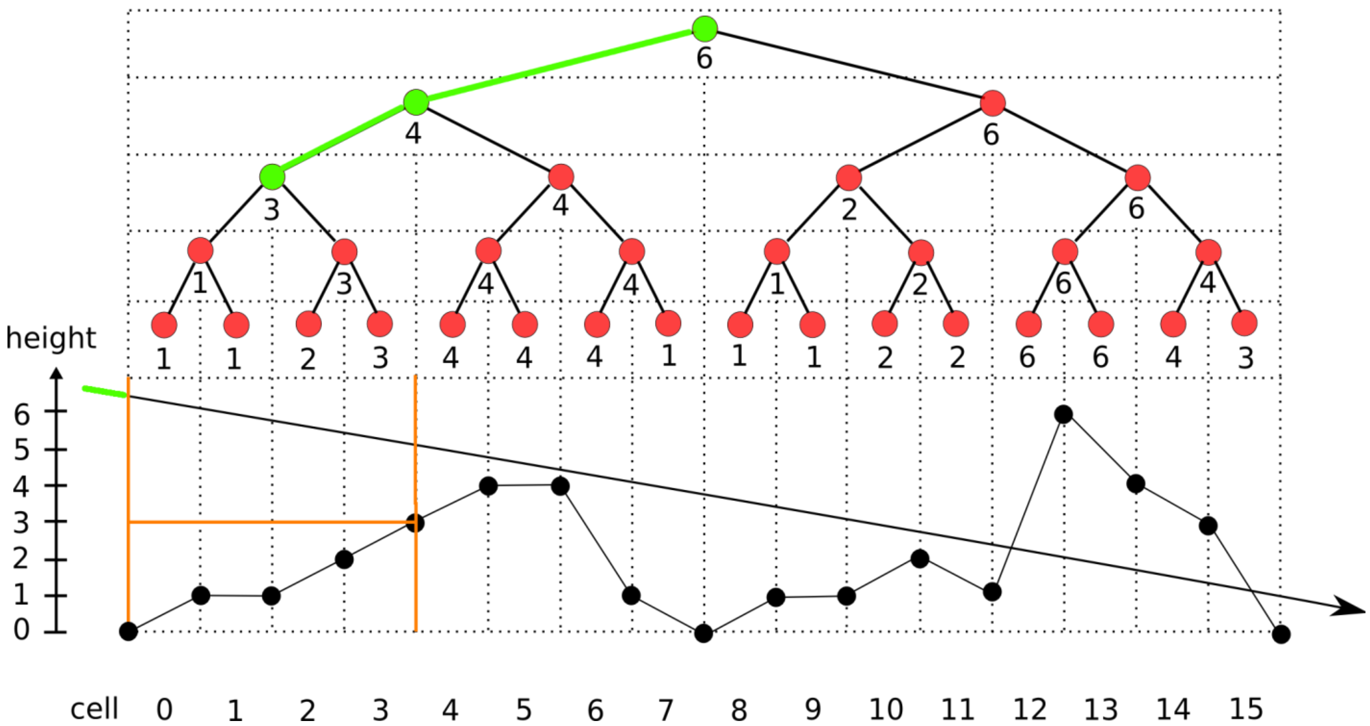
- Ray hits the bounding box of the Height Field
- Exit points below the maximum value -> refine

Intersection Algorithm



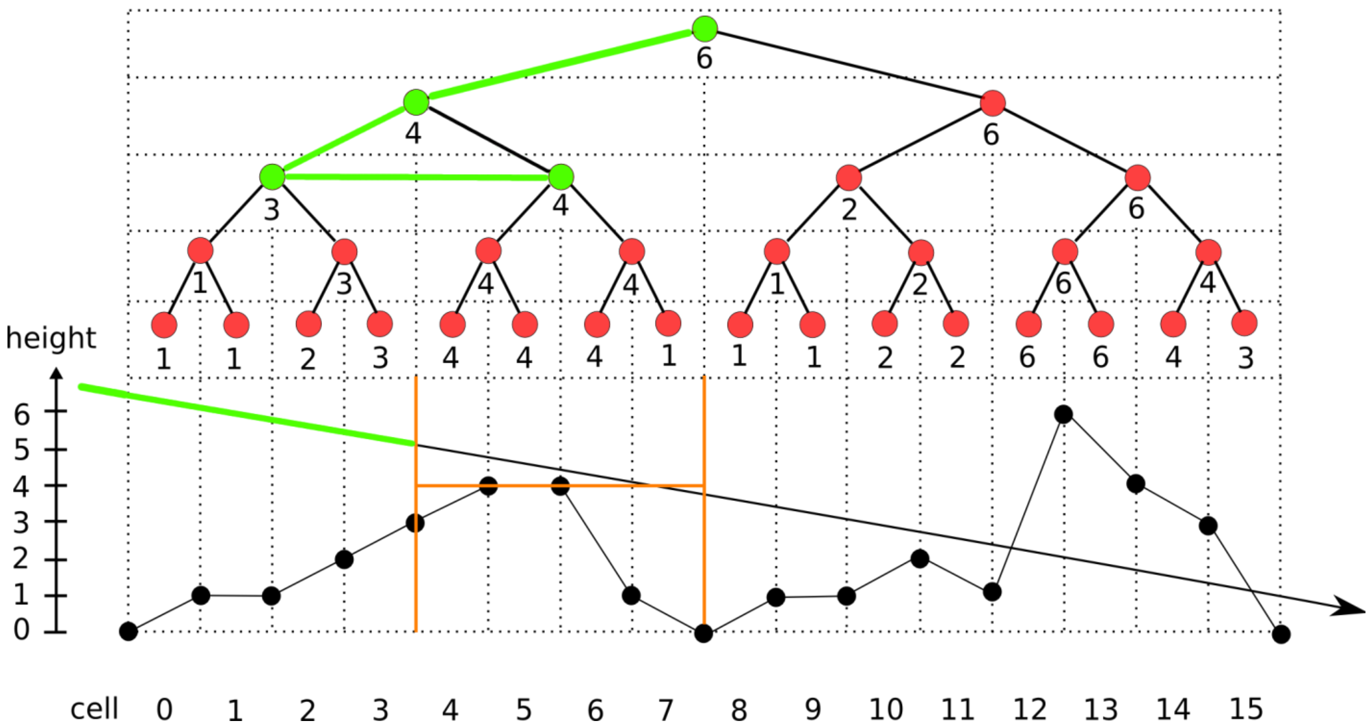
- Ray intersects the maximum value plane -> refine

Intersection Algorithm



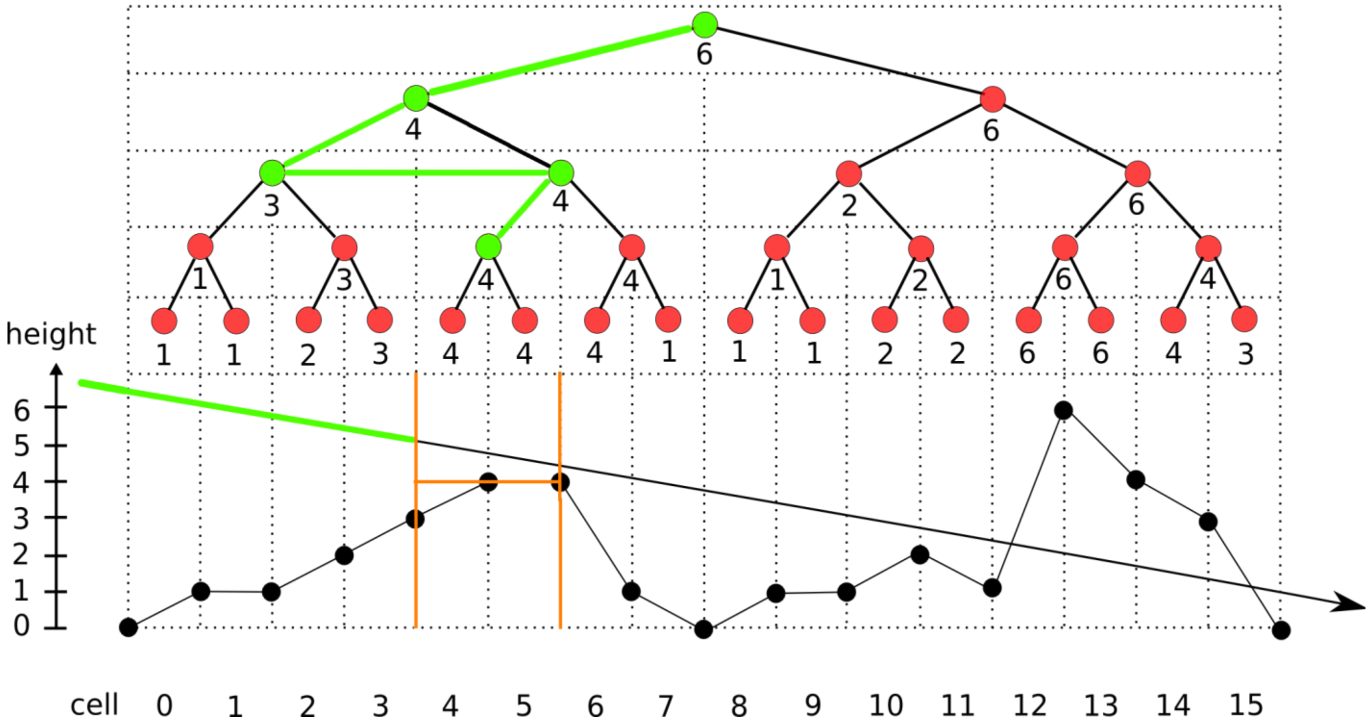
- Ray does not hit the maximum plane -> move

Intersection Algorithm



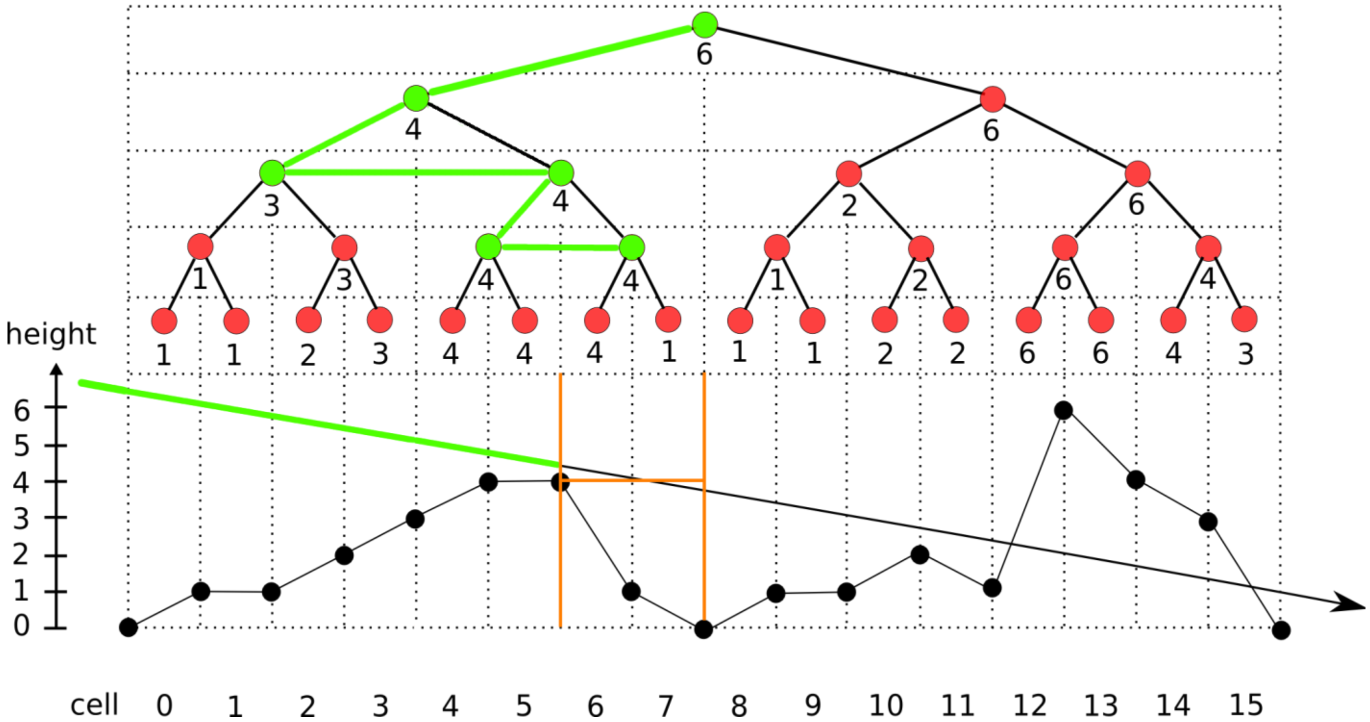
- Ray exits below the maximum value -> refine

Intersection Algorithm



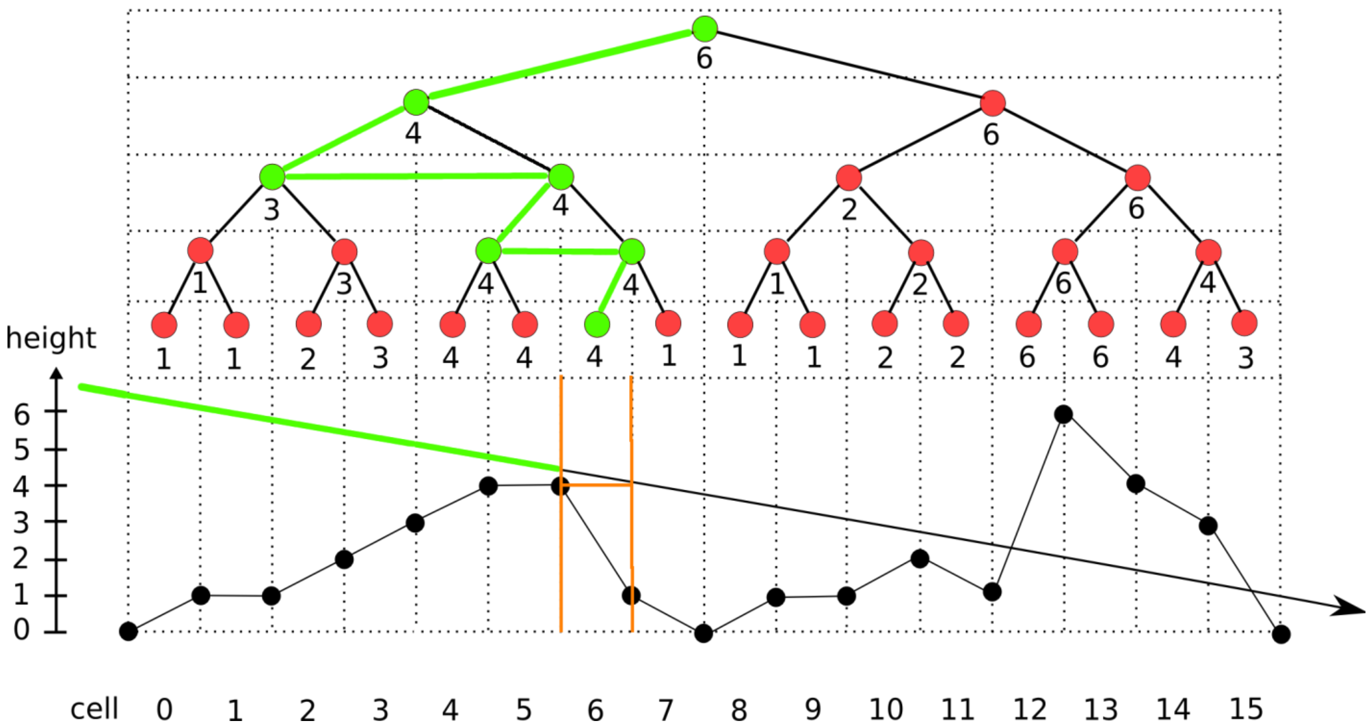
- no maximum height plane intersection -> move

Intersection Algorithm



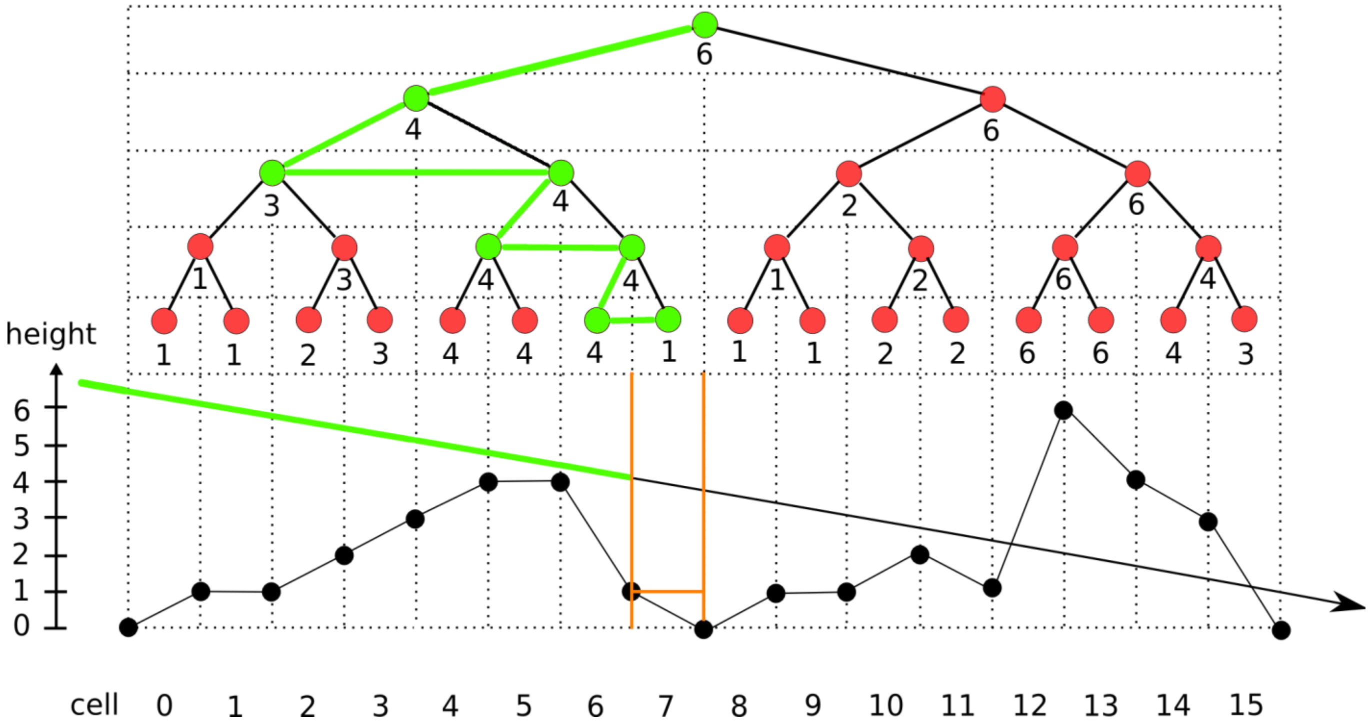
- again here refinement is required

Intersection Algorithm



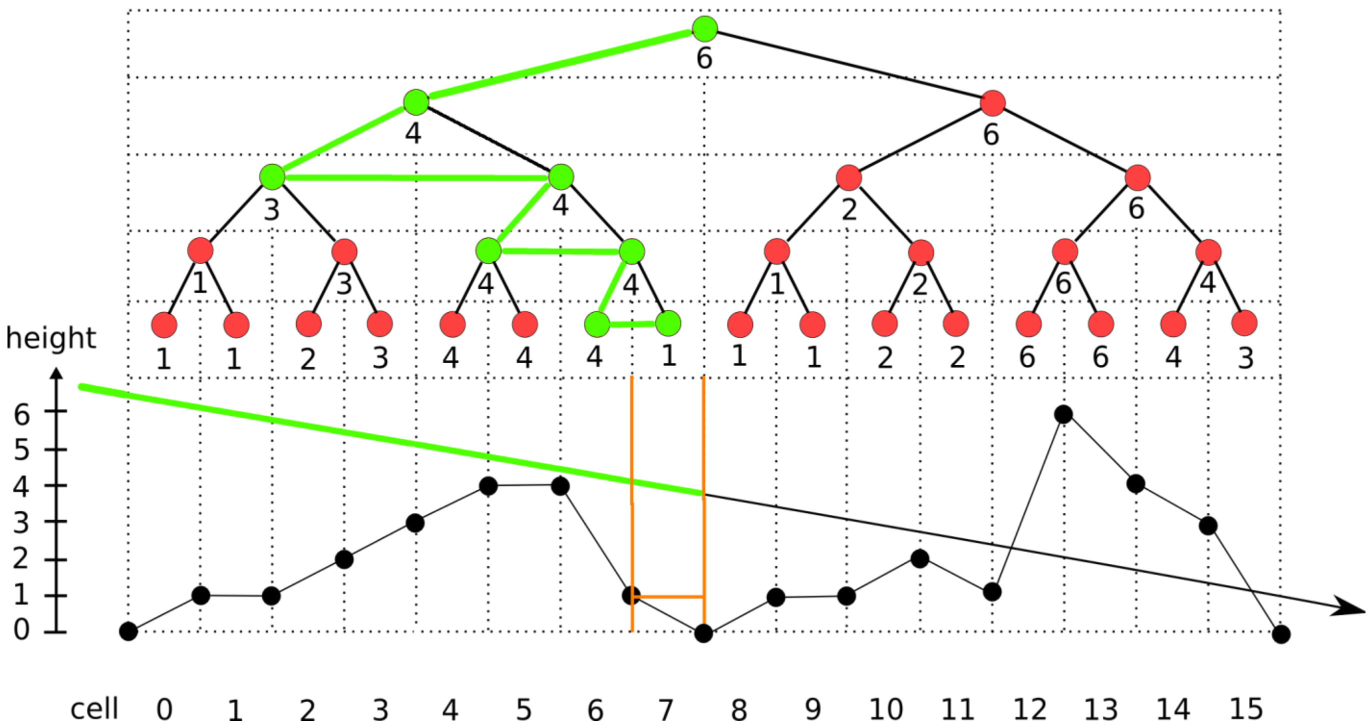
- traverse down

Intersection Algorithm



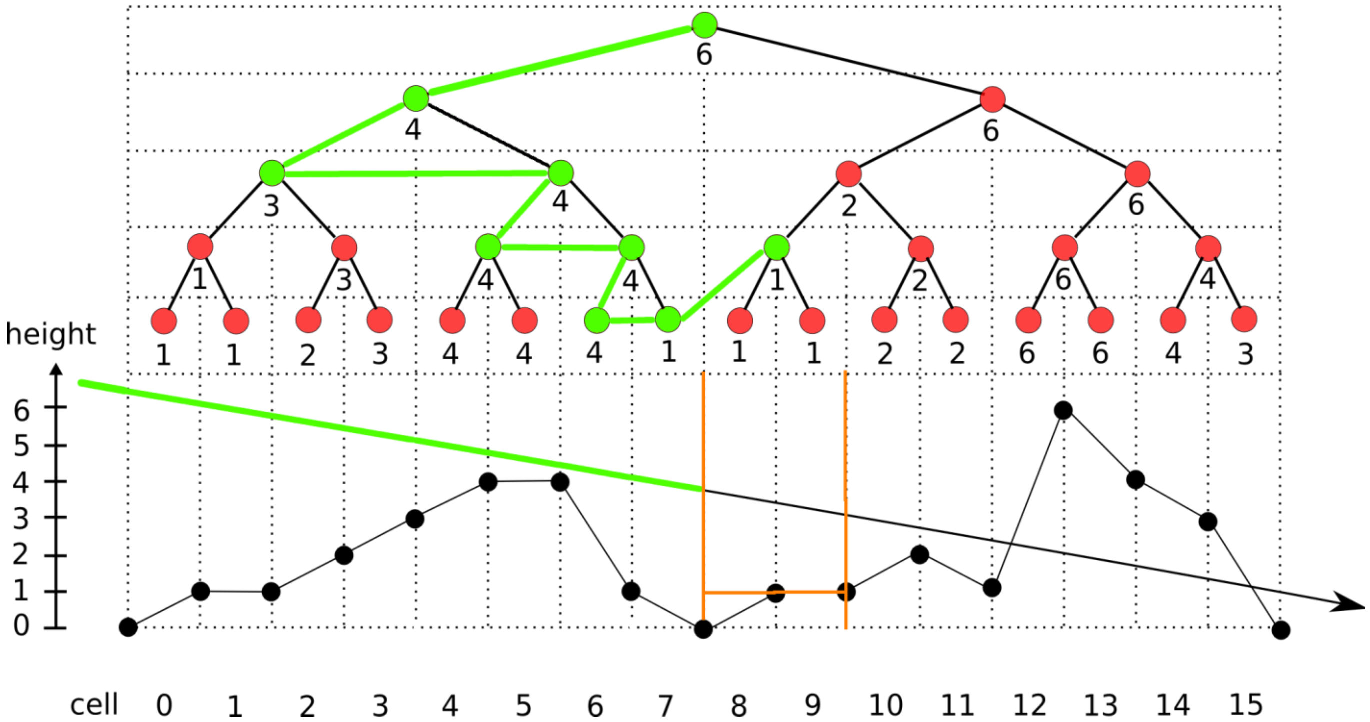
- move ray to the boundary

Intersection Algorithm



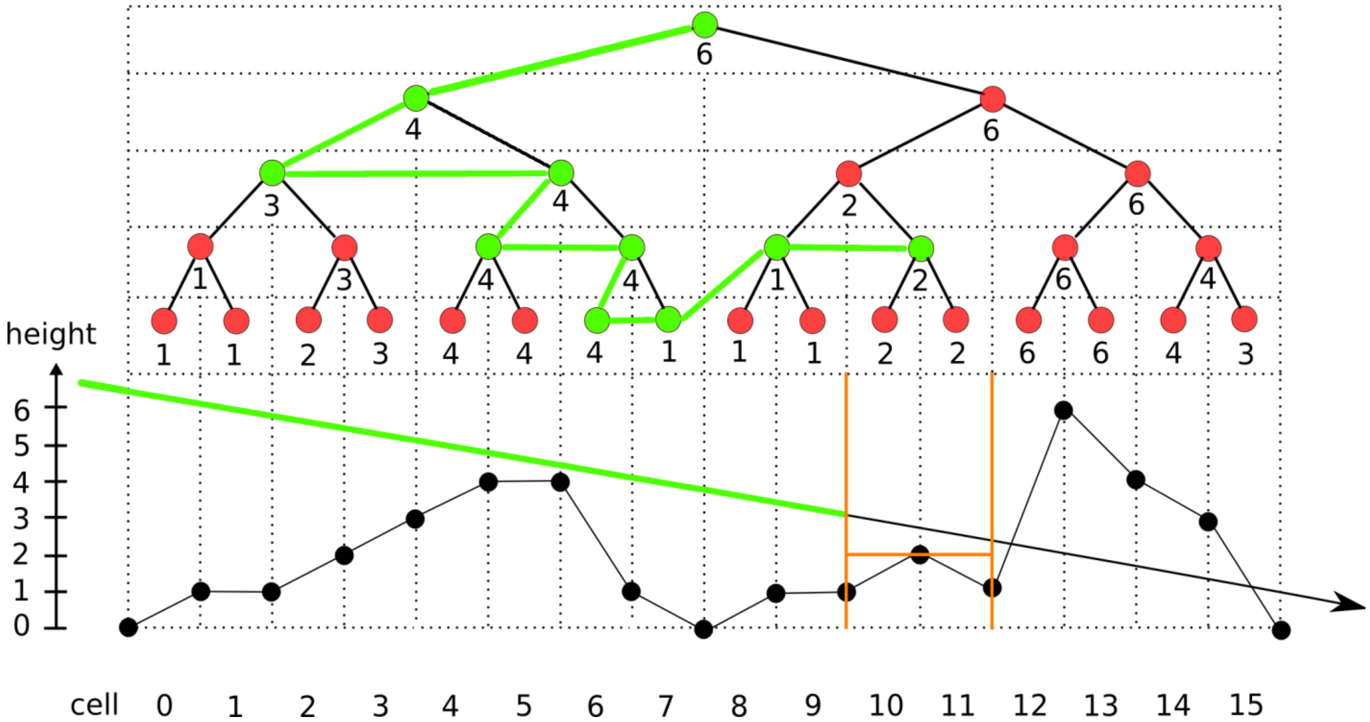
- ray at cell boundary with index divisible by two

Intersection Algorithm



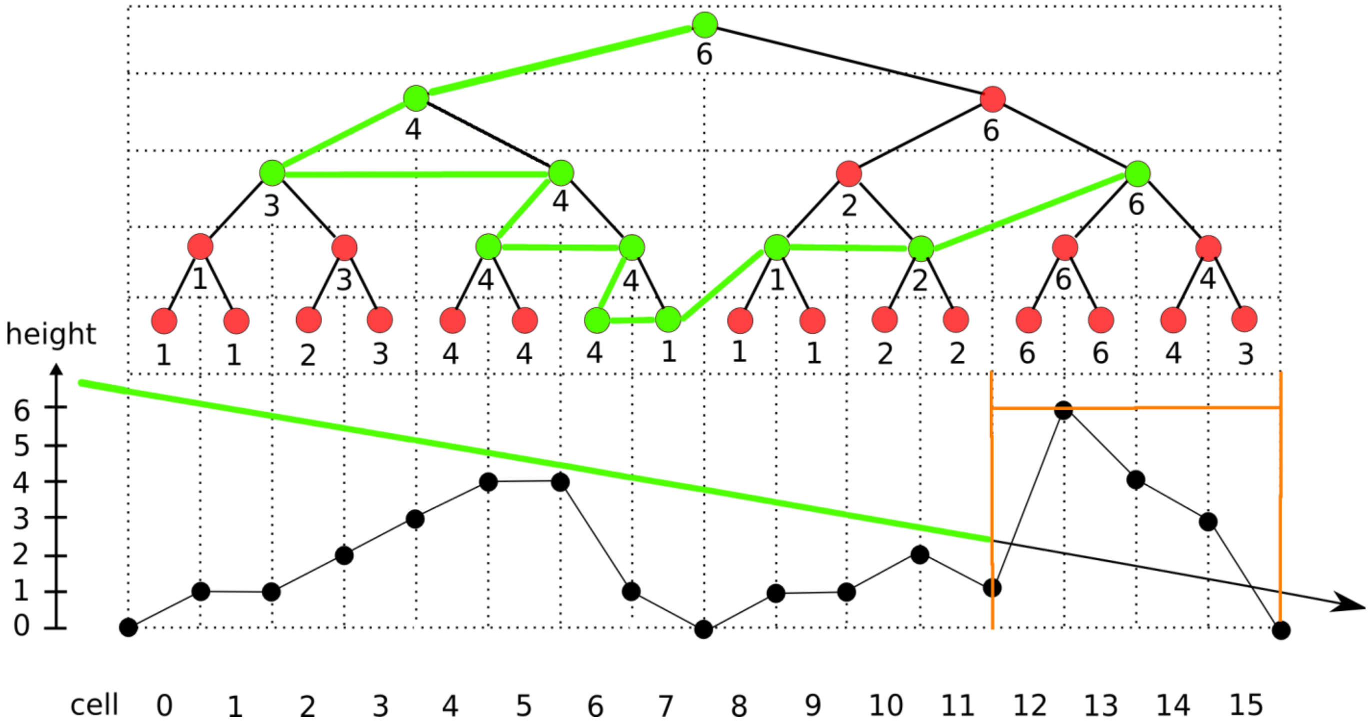
- increase the mipmap level (traverse up in the tree)

Intersection Algorithm



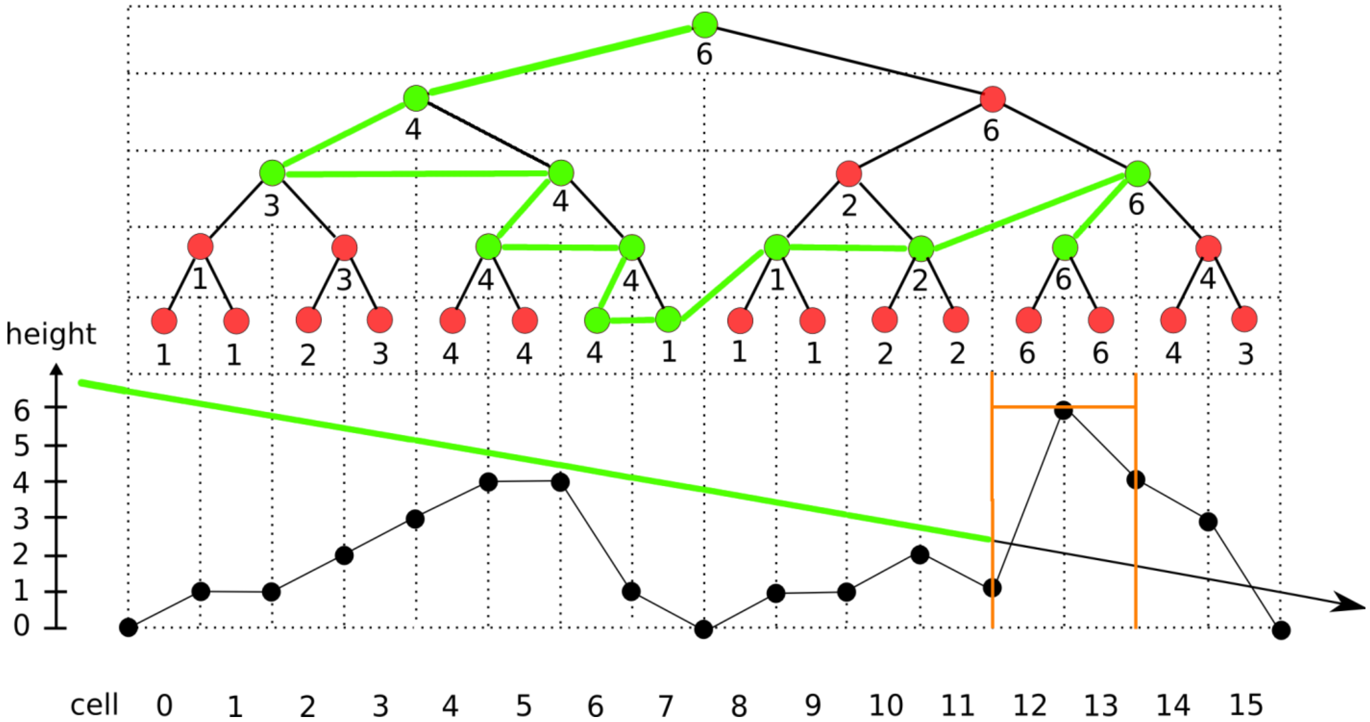
- move ray to the boundary

Intersection Algorithm



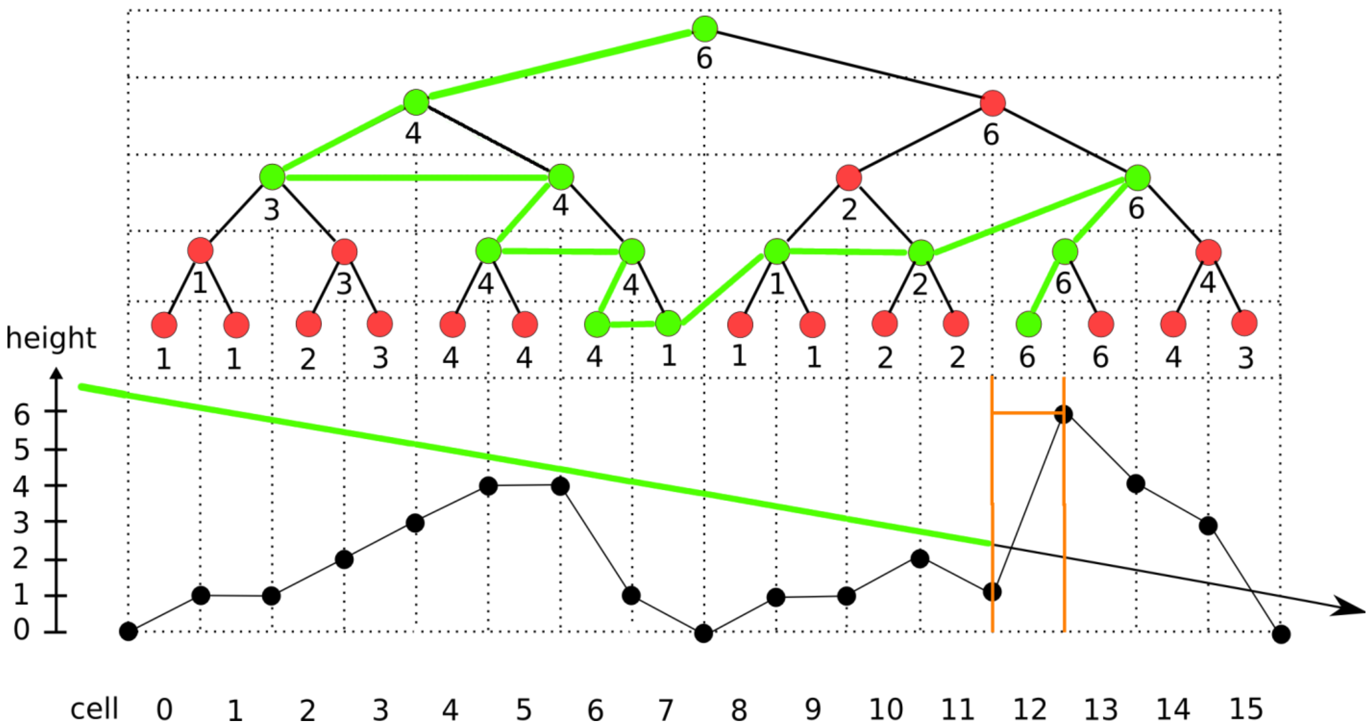
- ray at cell with index divisible by two -> increase the level

Intersection Algorithm



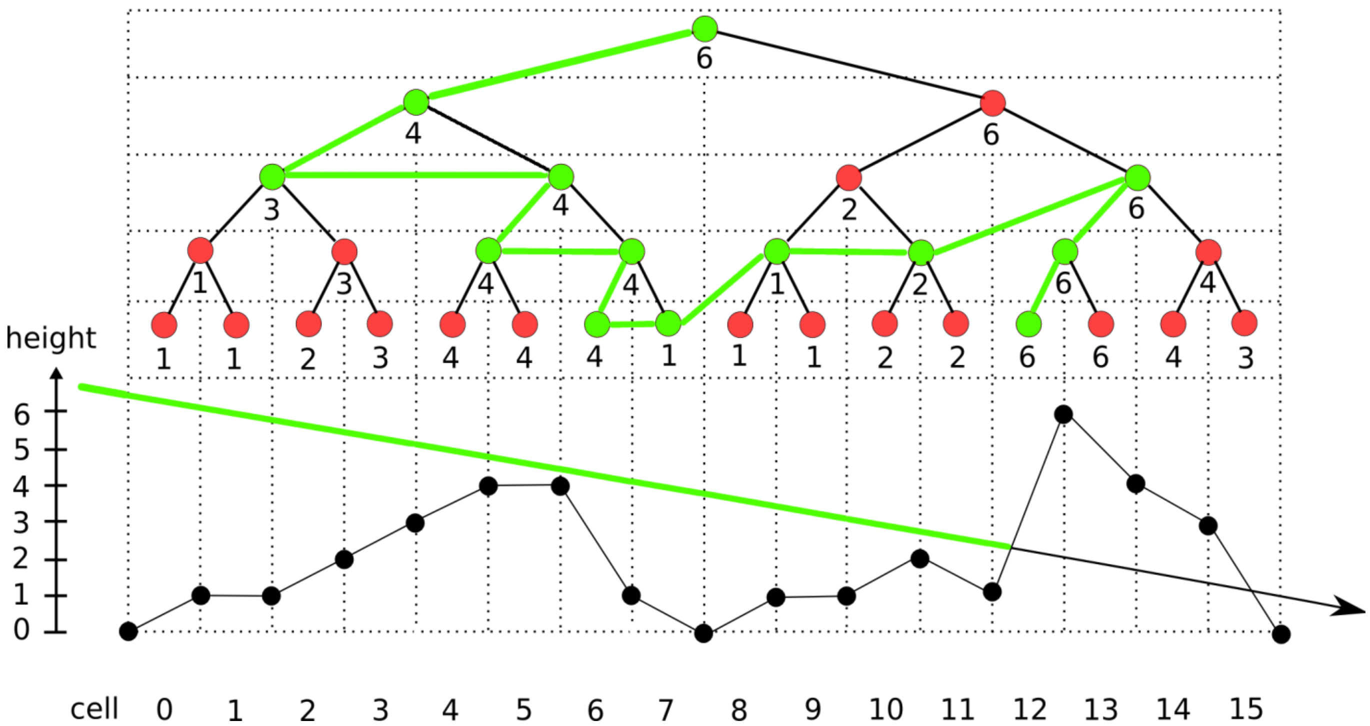
- ray below the maximum height -> refine

Intersection Algorithm



- ray below the maximum height and level = 0 -> perform ray-line intersection test

Intersection Algorithm



- Ray – Height Field Intersection point is found

Intersection Algorithm

- Hierarchy level updates:
  - no level update
    - linear stepping
  - start from root
    - not real effective because of more iteration steps
  - optimal parent
    - require extra look-up or dynamic branching computation to find the optimum
    - not very GPU friendly
  - one level up
    - optimal solution for todays GPUs

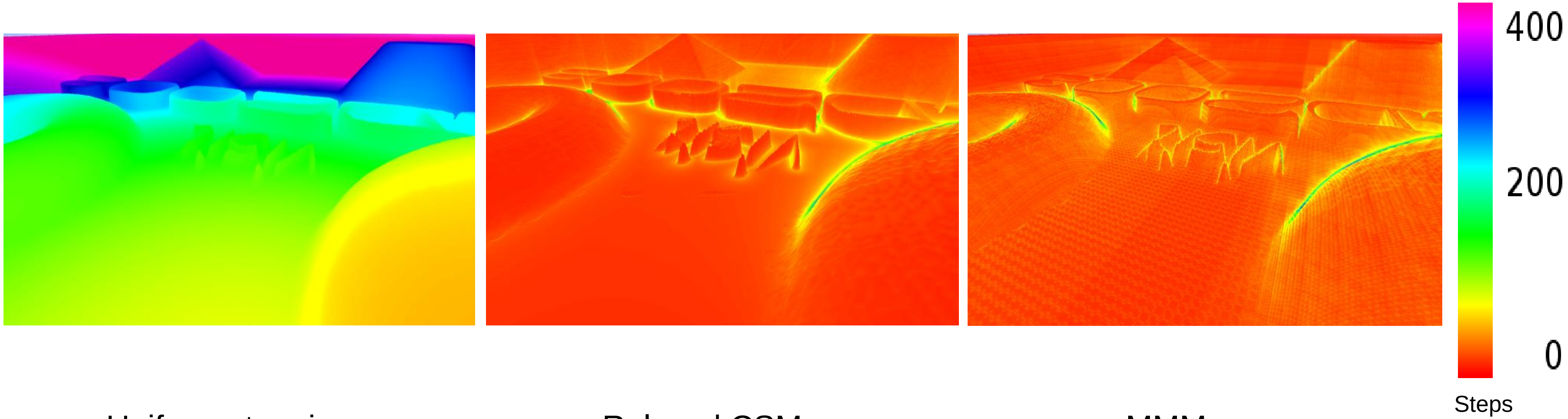
Optimized Implementation

- Level of Detail
  - easily derived from the mipmap structure
  - prune tree nodes below treshold computed by the distance of the ray to the viewer
- Cache optimized mipmap data structure
  - mipmap structure as 3D texture (consecutive levels in Z-slices)
  - Interleaved mipmap data
- Perform binary + linear search in level 0
  - very hardware friendly, however may cause artifacts

	maximum mipmap	+ bilinear patch intersection	+ linear binary search
256^2	95 FPS	43 FPS	70 FPS
512^2	87	38	58
1024^2	75	33	50
2048^2	70	27	40
4096^2	49	22	35

Performance

- Iteration step comparison of common algorithms and MMM



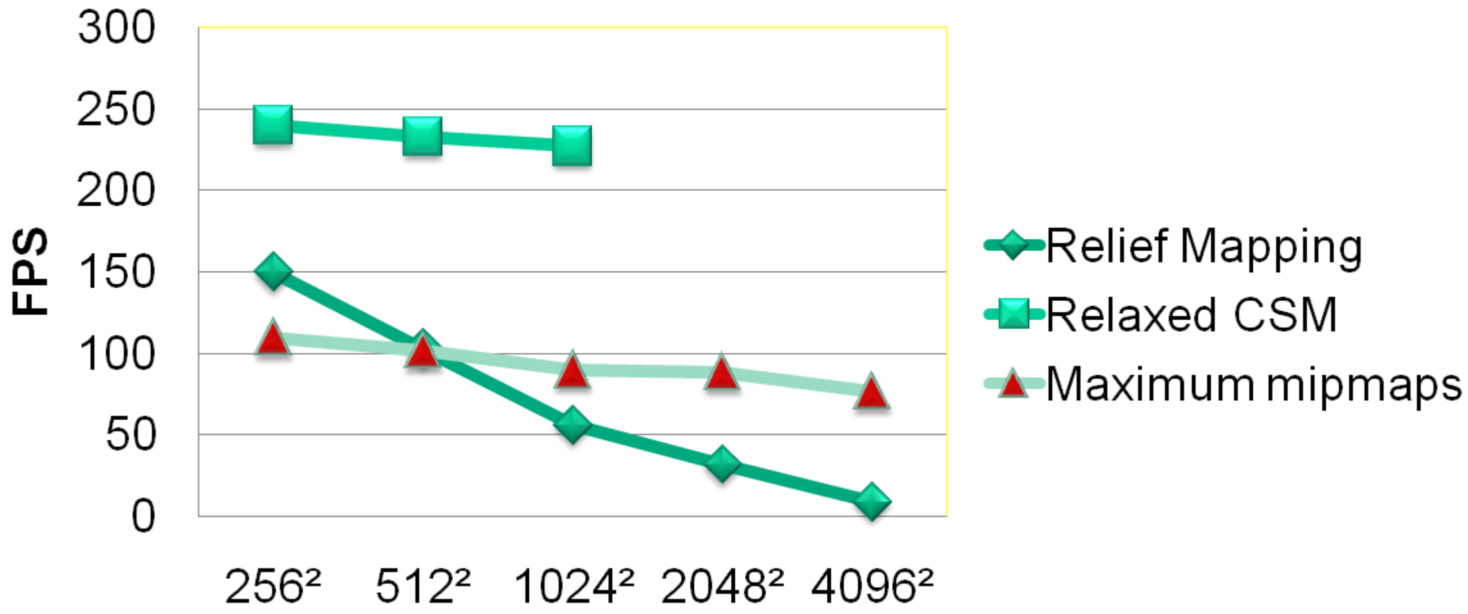
Uniform stepping | Relaxed CSM | MMM

Comparison

- Algorithm complexity
  - Relief mapping $\approx \sqrt{n}$
  - CSM = hard to analyse but in practice $\leq \sqrt{n}$
  - **MMM = in practice logarithmic time**

- Precomputation time
  - Relief mapping = 0
  - CSM = minutes to hours
  - **MMM = mipmap build time = < 10 ms**

Comparison



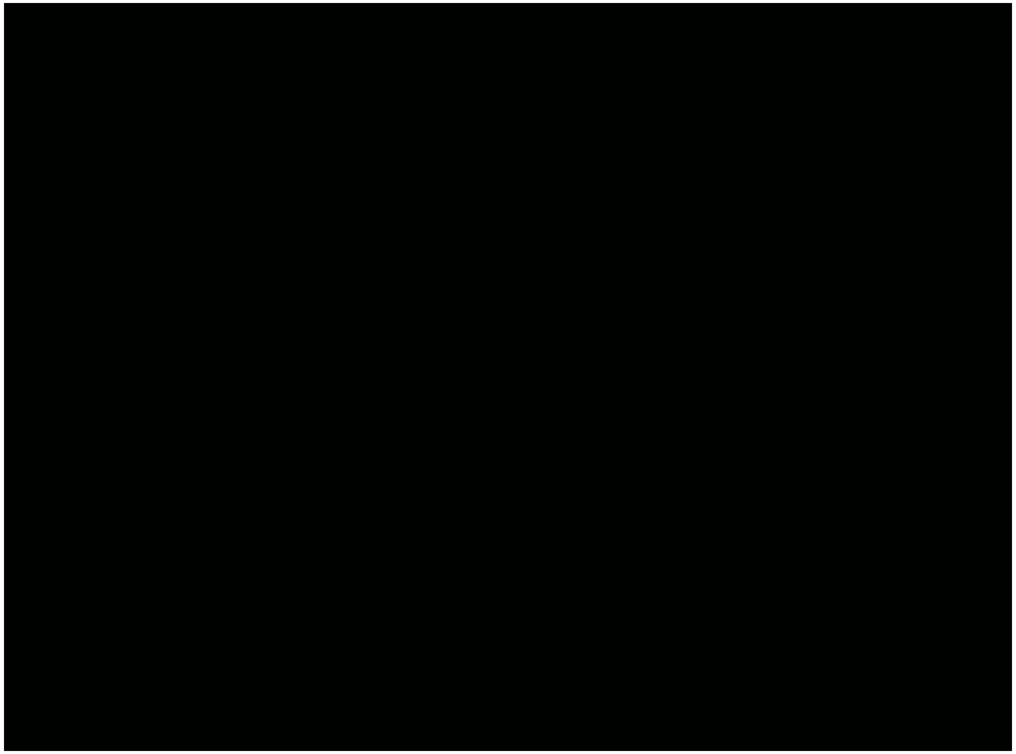
	relief mapping	relaxed CSM	maximum mipmap
256^2	150	240 (~2min)	110 (0.17ms)
512^2	103	233 (~15min)	102 (0.27ms)
1024^2	56	227 (>8h)	90 (1.2ms)
2048^2	32	n.a.	89 (2.13ms)
4096^2	9	n.a.	77 (7.52ms)

Contribution

- Accurate
  - ray – height field intersections with no artifacts

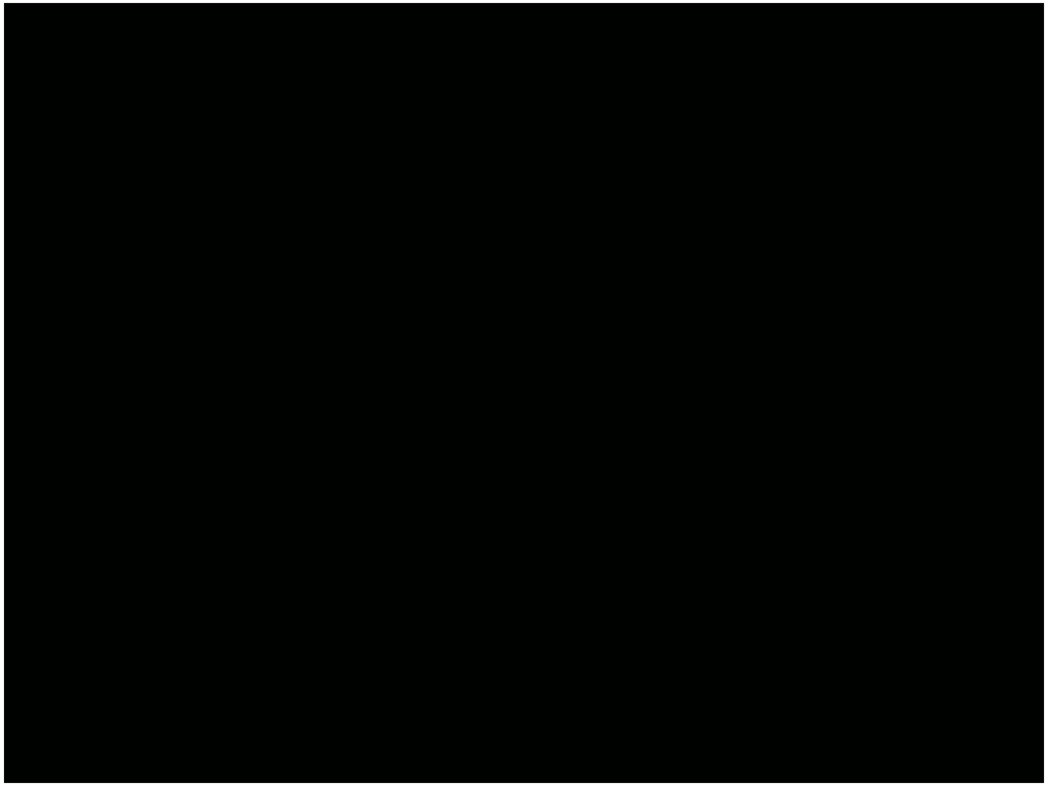
Contribution

- Dynamic
  - negligible update time for dynamic height fields



Contribution

- Scalable
  - high resolution height fields (4096^2) at real-time frame rates

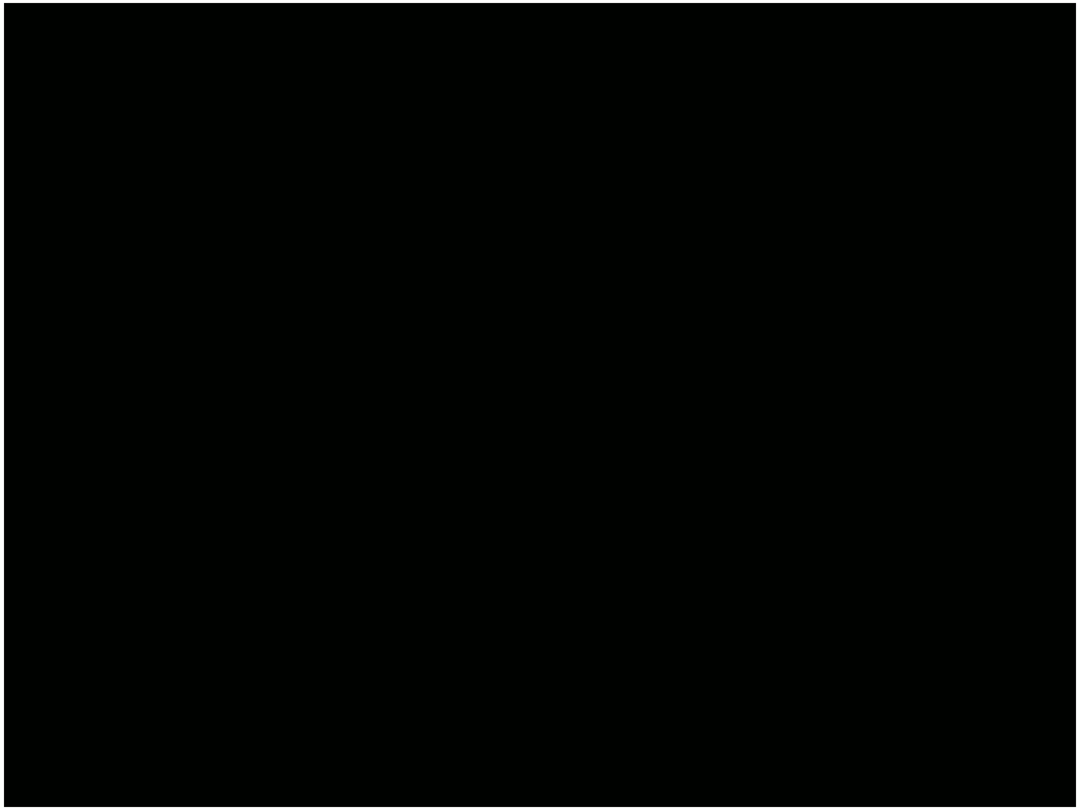


Discussion

- Drawbacks
  - depends on random mipmap level access, which seems to be non optimized on current GPUs
  - for small heightmap resolution the brute force search on current GPUs might even be faster

Questions

Thank you!



http://www.tevs.eu/projects_i3d08.html