Scientific Visualization

Dick, C., Krüger, J. H., & Westermann, R. (2009, March). GPU Ray-Casting for Scalable Terrain Rendering. In *Eurographics (Areas Papers)* (pp. 43-50).

Content

- Terrain Visualization
- Data Structures
- Terrain Rendering
- Implementation
- Batched Methods
- Terrain Shading
- References

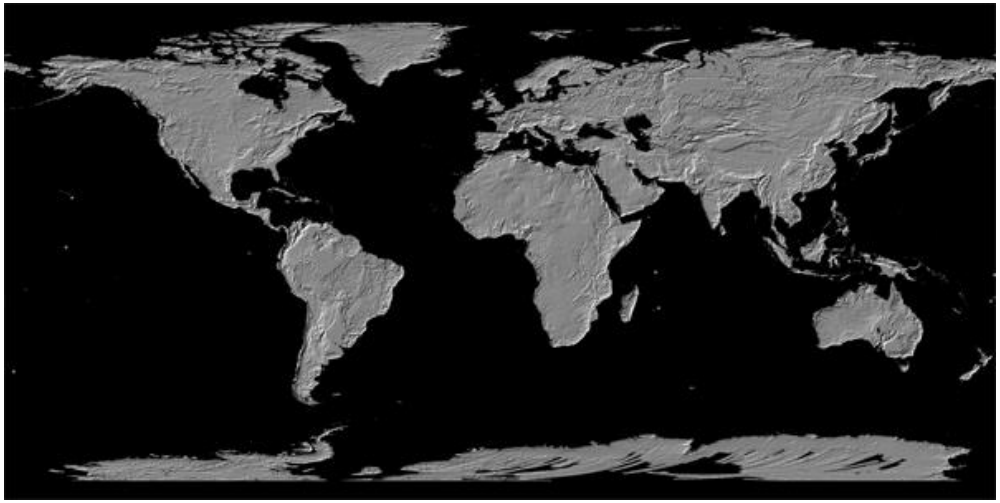
Terrain

VISUALIZATION

Data Sources

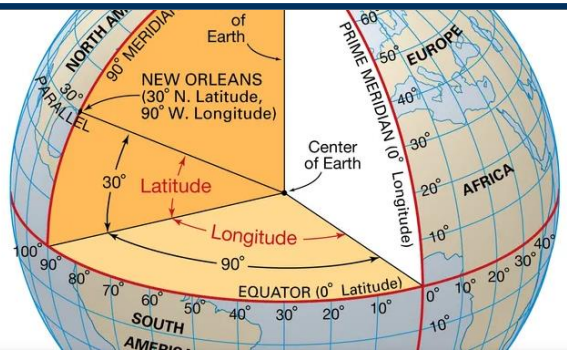
- DEM … Digital elevation Model
- DEM data is usually stored in 16-bit GeoTiff images and projected with Mercator projection
- Acquisition is done with laser scanners in airplanes or high-resolution line cameras in satellites
- processing is costly and error-prone in each case

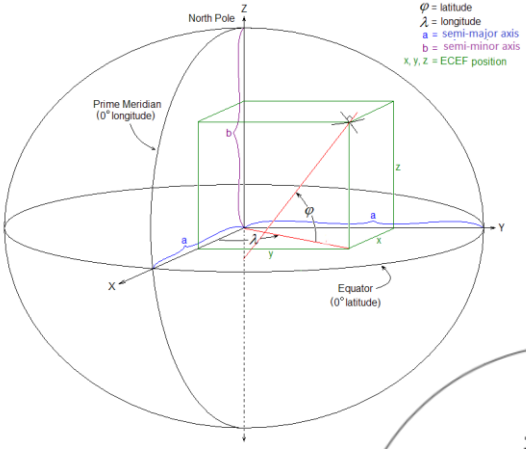
- Example for ASTER GDEM satellite data:
  - ASTER GDEM version 2 (G = global, i.e. the whole earth surface is covered)
  - Resolution: hor/ver 15/20m
  - 22600 tiles of 4000^2 pixels each (approx. 750 GB)

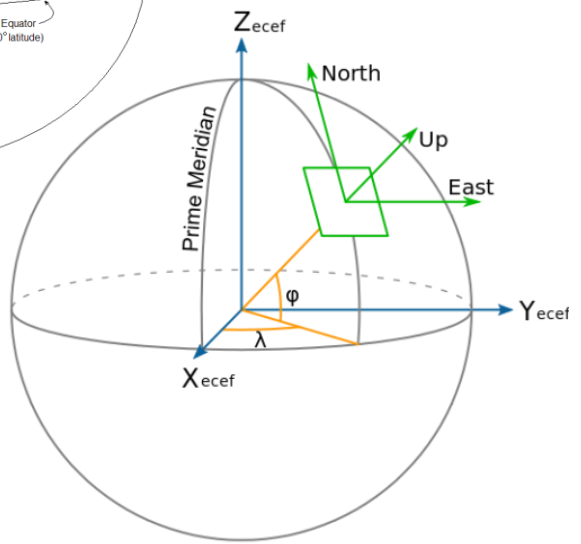


Geographic Coordinate Systems

- geodetic datum / system is composed of horizontal datum (typically Latitude & Longitude) and a vertical datum (typically Height in meters above mean sea level – MSL)
- World Geodetic System (WGS) in current Version 84 is used in for satellite based navigation. It uses reference ellipsoid
- Earth-centered, Earth-fixed coordinate system (ECEF) is 3D coordinate system with earth center in origin
- East-North coordinates are 3D coordinates relative to geo location aligned with geographic directions

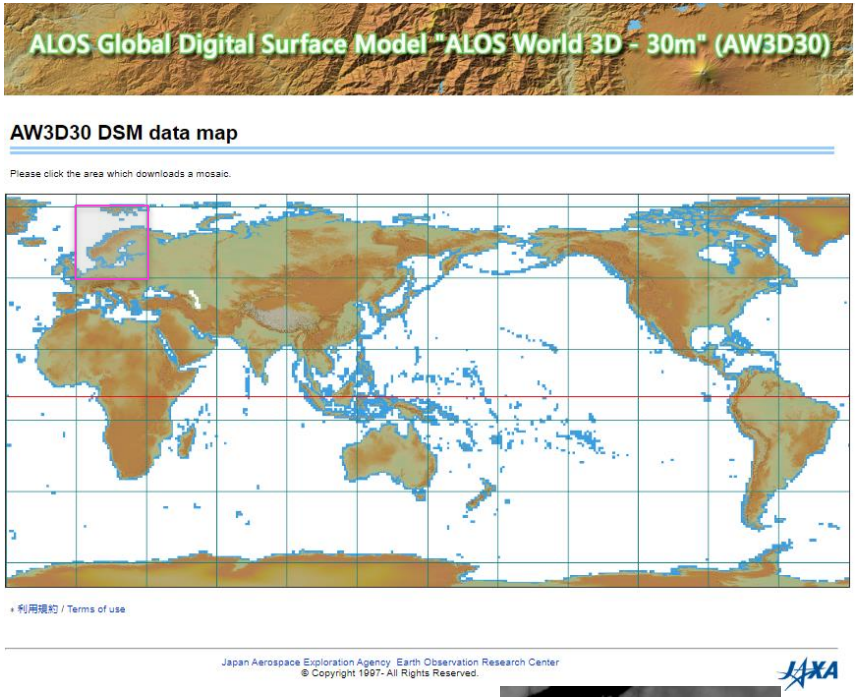


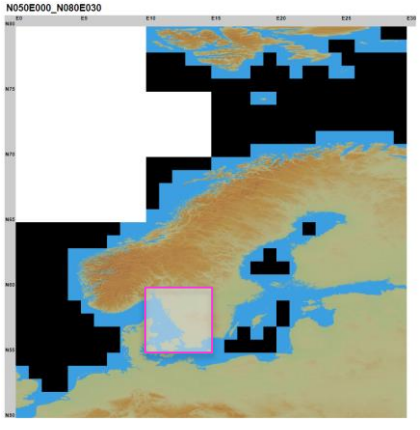


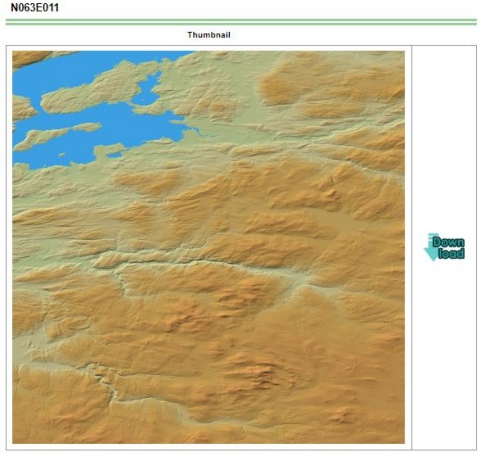


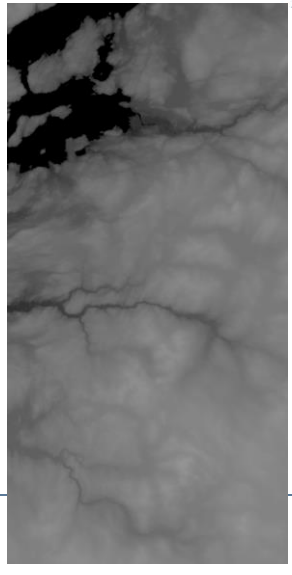
JAXA's global ALOS-2 3D World

- Global dem is provided in tiles that correspond to geodetic datum ranges
- Preview in the JAXA browser is based on a color palette and non uniform image extends with respect to pixels









More Data Sources

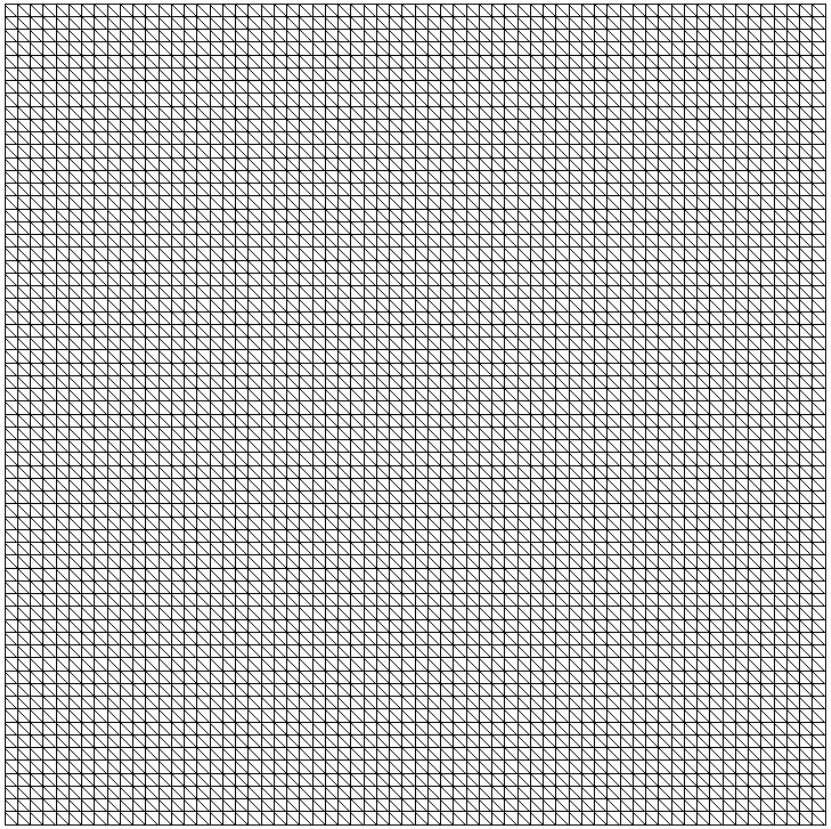
Free data sources (gisgeography.com/free-global-dem-data-sources)

- **Space Shuttle Radar Topography Mission (SRTM)**
  (30m spacing, 16m height accuracy)
- **ASTER global Digital elevation Model**
  (90m spacing, 30m height accuracy)
- **JAXA's global ALOS-2 3D World**
  (30m spacing, 5-17m height accuracy)
- **Mars Orbiter Laser altimeter (MolA)**
  (100m spacing, 3m height accuracy)

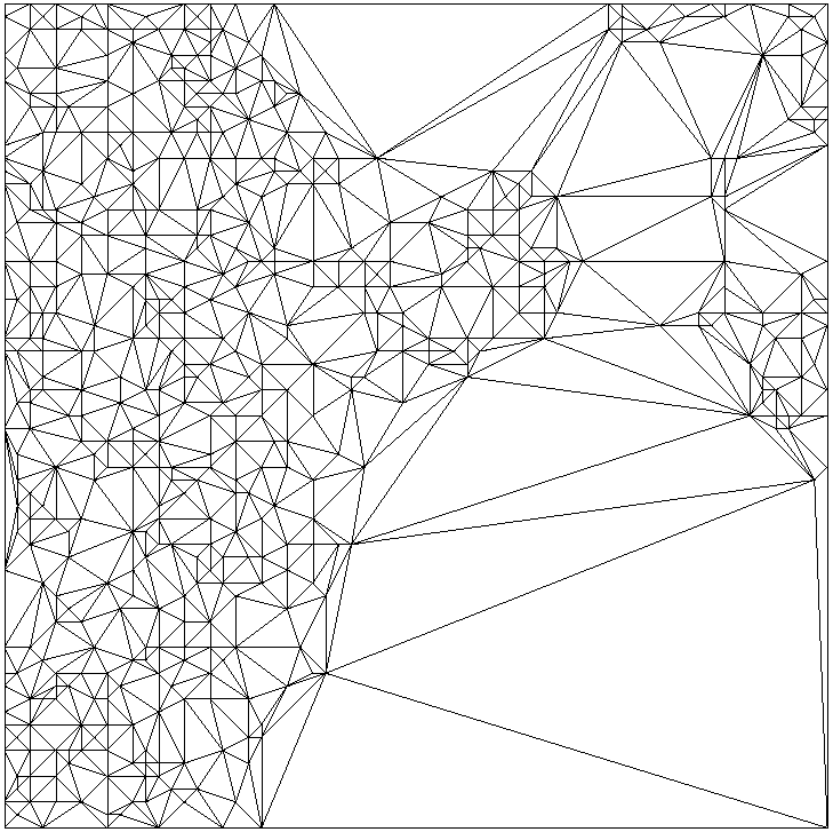
Commercial Sources

- **WorldDEM™**
  (12m spacing, 2m accuracy)

Terrain Representation



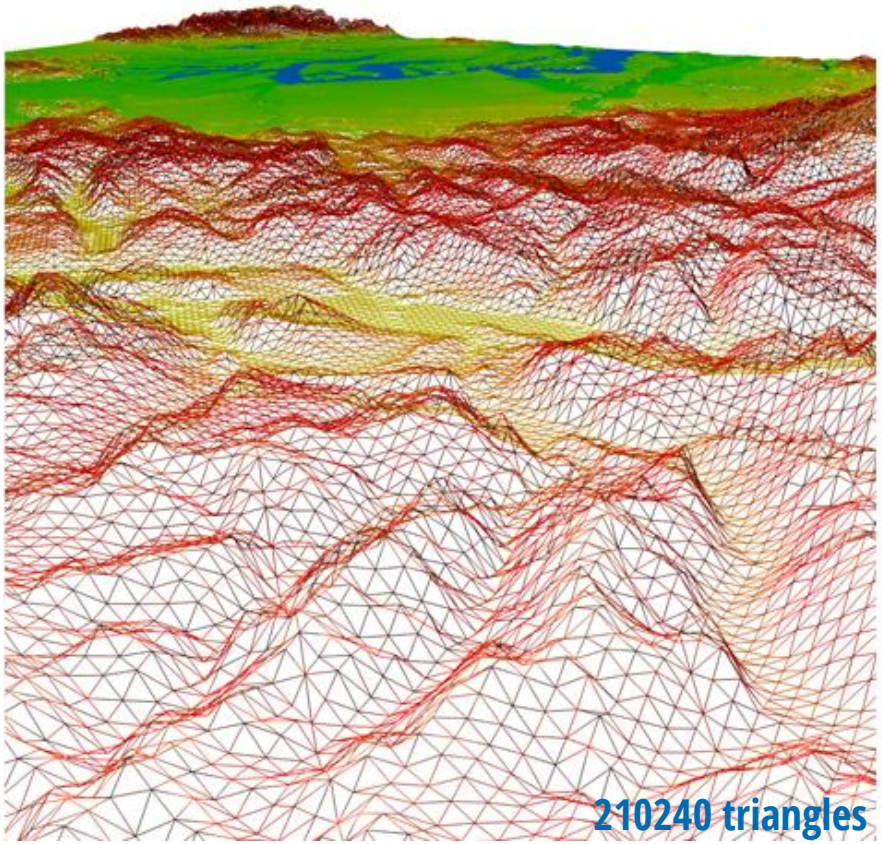
- render as 2.5D height map
- tesselation of regular grid yields far too many triangles

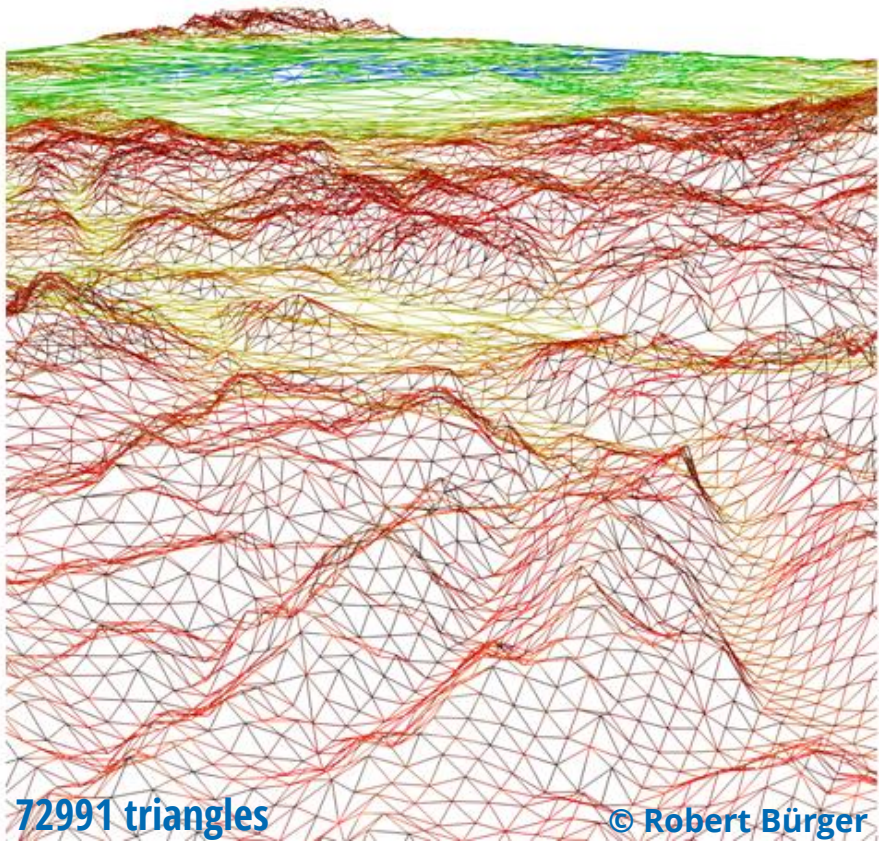


- TIN … triangular irregular networks adapt to data
- but demand for additional storage of x and y coordinates

View-Dependent Tessellation

- Significant performance boost by adaptive refinement with respect to view-dependent approximation error





left: no adaptation, right: adaptation with respect to 1-pixel threshold of projected error

Comparison to Standard LOD

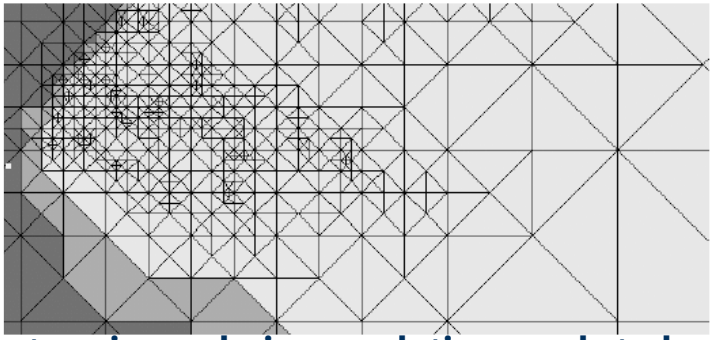
terrain data is simple

- 2 ½ Data
- more specialized simple algorithms

terrain rendering is difficult

- very large contiguous models
- very close and very far away at the same time
- LOD dependent on point of view necessary
- out-of-core

for compact 3D models one simply selects to be rendered mesh from collection of approximations with decreasing complexity



in terrain rendering resolution needs to be varied over model in a view-dependent manner

Importance of Geospatial Data

S. Hahmann and D. Burghardt. How much information is geospatially referenced? networks and cognition. International Journal of Geographical Information Science, 27(6):1171–1189, 2013.

- Study on how often geospatial references are used
- Examined wikipedia articles and asked people
- They found that 57% of the information in German wikipedia articles are geospatially referenced

2D or 3D Viewing?

- **Tory, M., Kirkpatrick, A. E., Atkins, M. S., & Moller, T. (2006). Visualization task performance with 2D, 3D, and combination displays. *IEEE transactions on visualization and computer graphics*, *12* (1), 2-13.**
- Tory et al. conducted experiments of 2D, 3D, and combined visualizations for estimation of relative positioning and orientation as well as region selection.
- their results show that 3D can be effective for approximate navigation and relative positioning, but 2D is more suitable for precise measurement
- 3D views should support
  - lighting cues like shadows
  - dedicated measurement tools
  - view point optimization
- Combined 2D/3D views are suggested

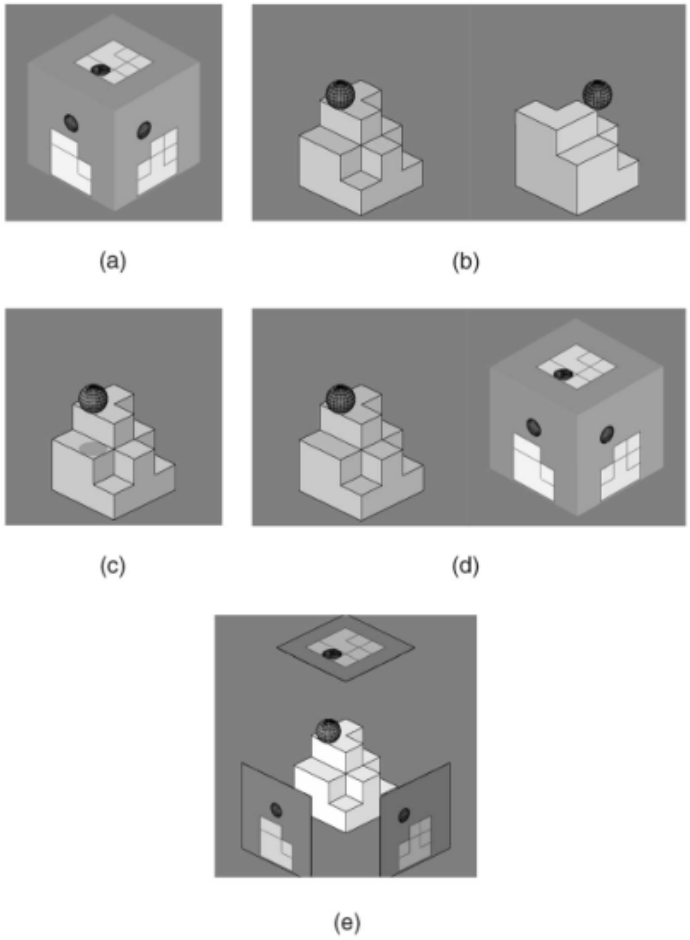
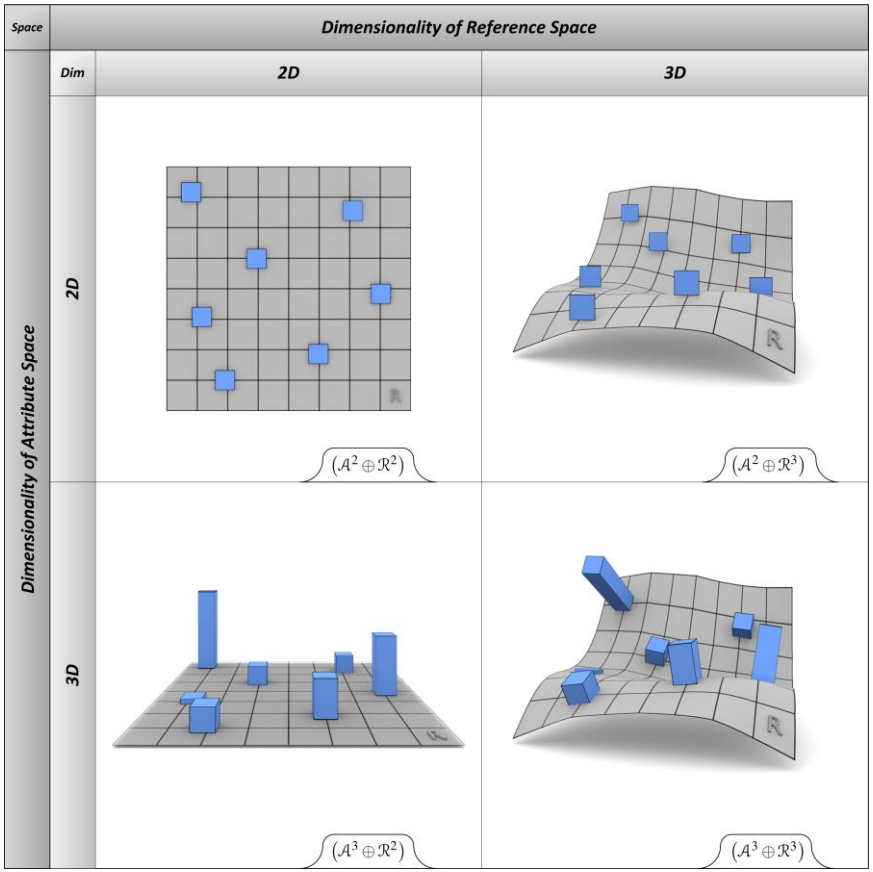
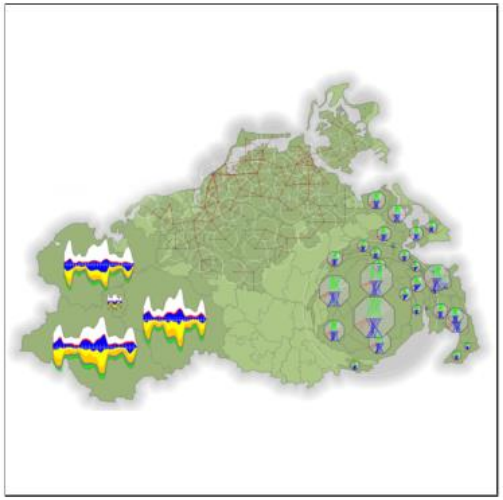


Fig. 1. (a) 2D, (b) 3D rotated, (c) 3D shadow, (d) orientation icon, and (e) ExoVis displays used in Experiment 1 (position estimation). Participants estimated the height of the ball relative to the block shape. In this example, the ball is at height 1.5 diameters above the block shape.

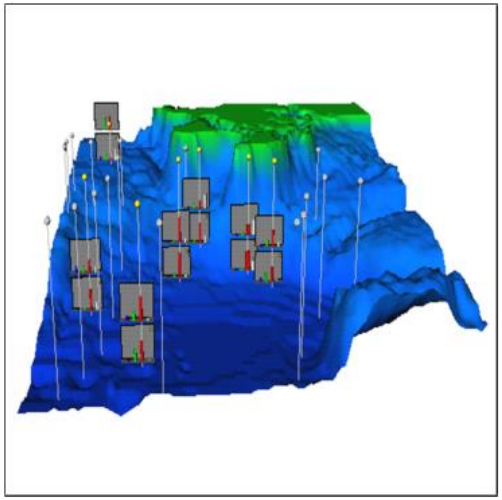
classification of GeoVis Techniques



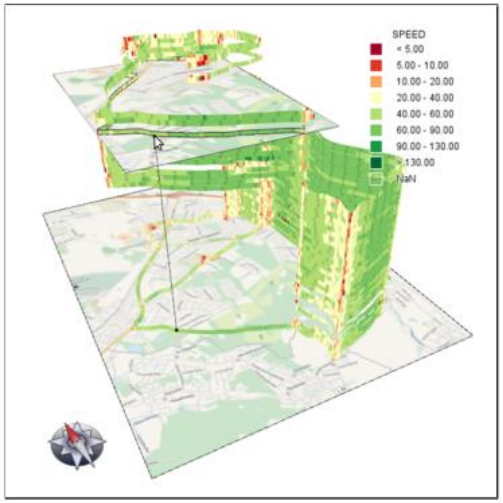
- **Dübel, S., Röhlig, M., Schumann, H., & Trapp, M. (2014, November). 2D and 3D presentation of spatial data: A systematic review. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)* (pp. 11-18). IEEE.**
- $\mathcal{A}^i$: selected attributes are visualized using i-dimensional graphical elements,
- $\mathcal{R}^j$: the reference space is visualized using j-dimensional graphical elements.



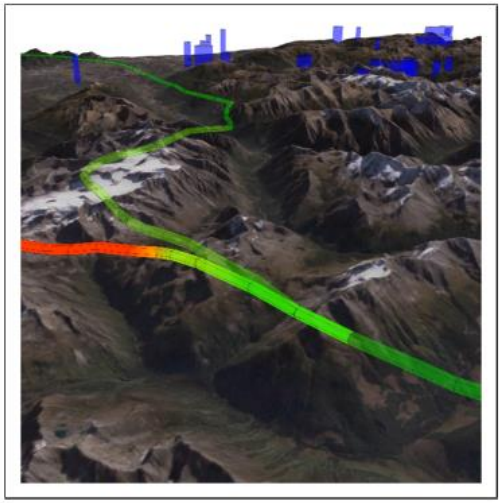
(a) $(\mathcal{A}^2 \oplus \mathcal{R}^2)$



(b) $(\mathcal{A}^2 \oplus \mathcal{R}^3)$



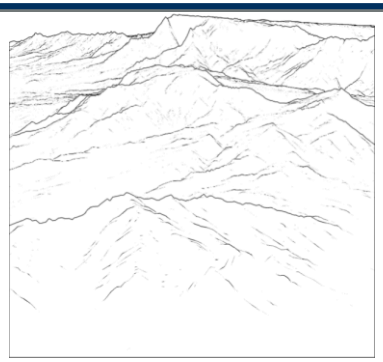
(c) $(\mathcal{A}^3 \oplus \mathcal{R}^2)$



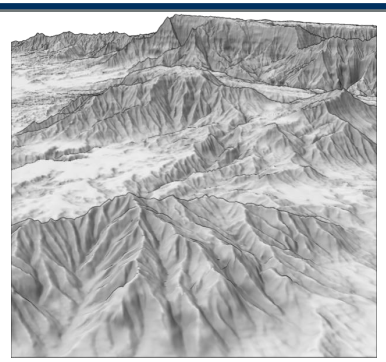
(d) $(\mathcal{A}^3 \oplus \mathcal{R}^3)$

Focus and Context in Terrain Vis

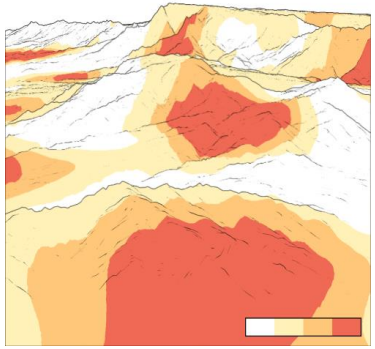
- **Dübel, S., Röhlig, M., Tominski, C., & Schumann, H. (2017, March). Visualizing 3D terrain, geo-spatial data, and uncertainty. In *Informatics* (Vol. 4, No. 1, p. 6).**
- Joint visualization of Terrain, Data, and Uncertainty
- Propose different viewing scenarios like take-off or overflight
- Refined focus & context techniques by prioritizing (+) or deemphasizing (-) terrain (T^- vs T^+), data (D^- vs D^+) and uncertainty (U^- vs U^+) individually



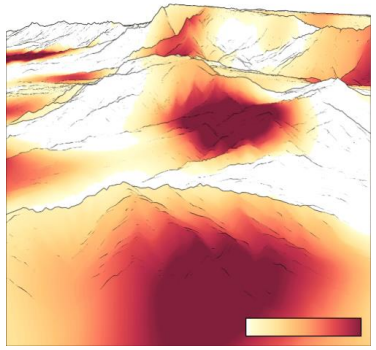
T^-: line drawing



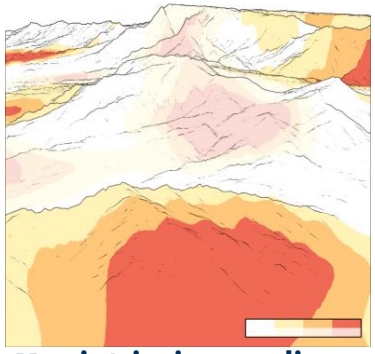
T^+: shading



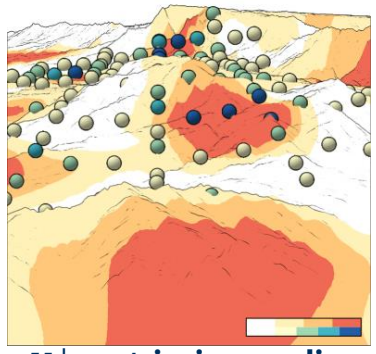
D^-: aggregration



D^+: individual values

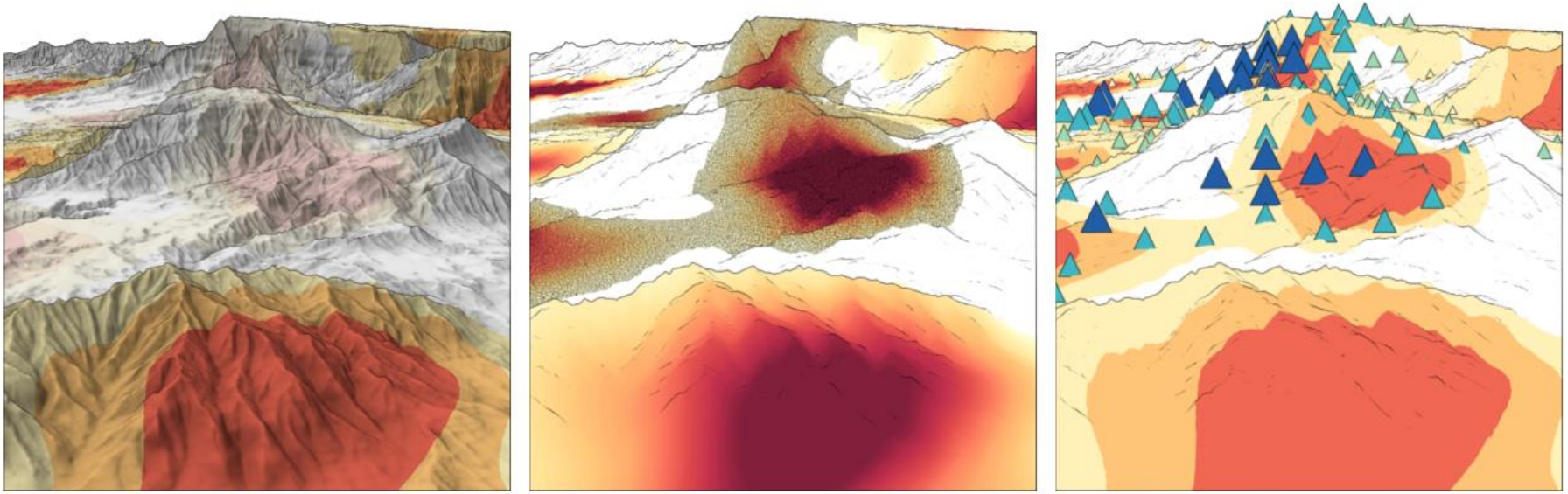


U^-: intrinsic encoding



U^+: extrinsic encoding

Focus and Context in Terrain Vis



(a) Prioritizing terrain (b) Prioritizing data (c) Prioritizing uncertainty

Figure 5. Differently prioritized visual representations of the same terrain, data, and uncertainty. (**a**) The terrain is prioritized and rendered as a detailed shaded surface. The data are visualized in an aggregated fashion by a segmented color scale, and uncertainty is depicted intrinsically by using transparency; (**b**) The data are prioritized and individual values are visualized with a continuous color scale. The terrain is rendered at less detail by drawing only lines, and uncertainty is depicted intrinsically by using noise; (**c**) Uncertainty is prioritized and depicted extrinsically via colored triangular glyphs. The terrain is rendered at less detail as in (**b**), and the data are visualized in an aggregated fashion as in (**a**).

Terrain Rendering

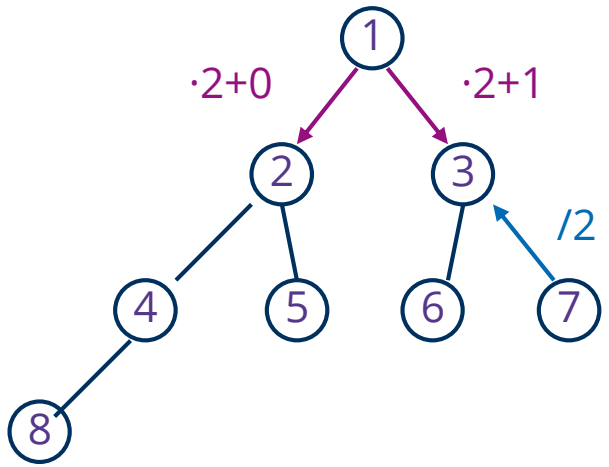
DATA STRUCTURES

Data Structures

- DEM … digital elevation models (images stored in tiles)
- TIN … Triangular Irregular Networks (typically stored as triangle mesh in indexed format
- bintree … used to store hierarchy over triangles
- quadtree … used to store color texture information
- HRT … Hierarchy of Right[-angled] Triangles

Implicit Binary and Quad Trees

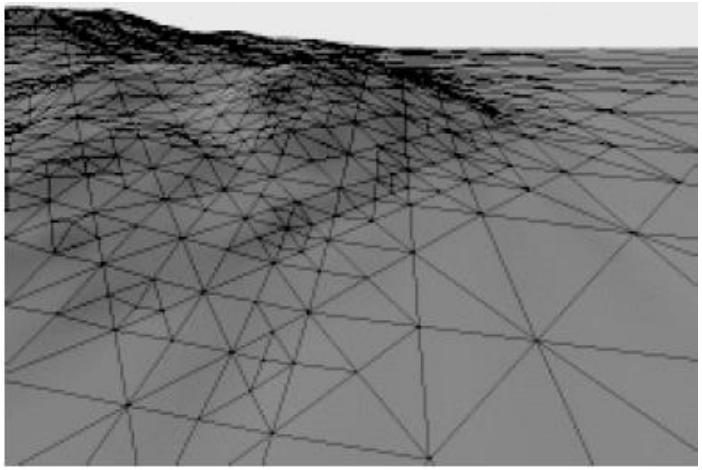
- Complete binary trees can be stored implicitly by indexing the nodes in a breadth first traversal
- access to parent:
$$p(c) = c/2$$
- access to children:
$$c(p,k) = 2p + k$$
- one can interleave two binary trees with indices 2 and 3 as roots, by adding tree specific offset m:
$$p(c) = (c - m)/2$$
$$c(p,k) = 2p + k + m$$

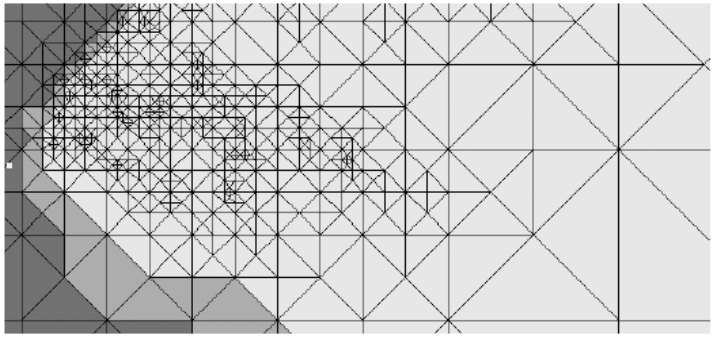


- This also works for quad trees:
$$p(c) = (c - m)/4$$
$$c(p,k) = 4p + k + m$$

Hierarchy of Right Triangles

- ROAM is one of the first and most important publications in terrain rendering
- defines adaptive tessellation
- optimal refinement algorithm
- preprocessing independent of point of view
- view-dependent refinement criterion



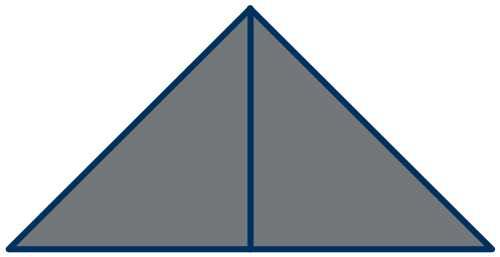


Mark Duchaineau et al., *ROAMing terrain: real-time optimally adapting meshes,*
IEEE Visualization, pp 81-88, 1997
http://www.cognigraph.com/ROAM_homepage

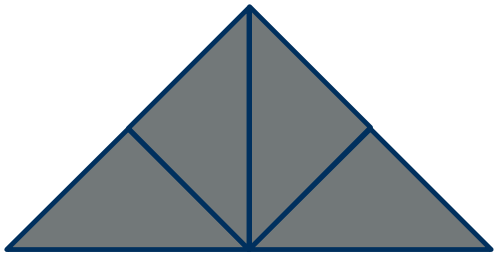
Hierarchy of Right Triangles

- HRT is created by iterated subdivision of a right-angled triangle along the longest side
- combination of two HRTs results in the tessellation of a square
- **Idea:** vary subdivision depth to achieve adaptive triangulation

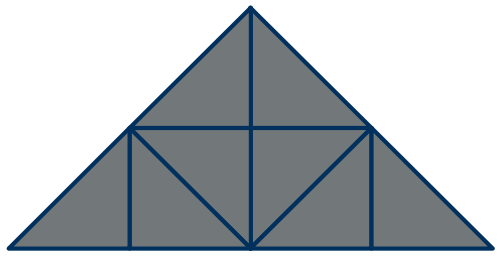
$l = 0$



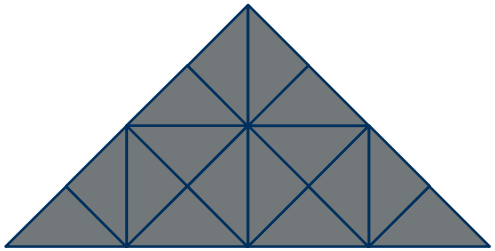
$l = 1$



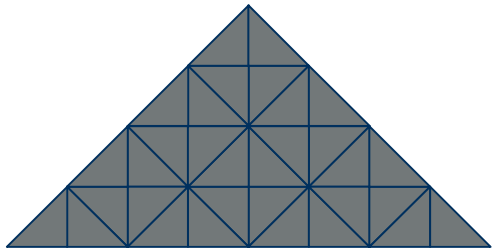
$l = 2$



$l = 3$



$l = 5$

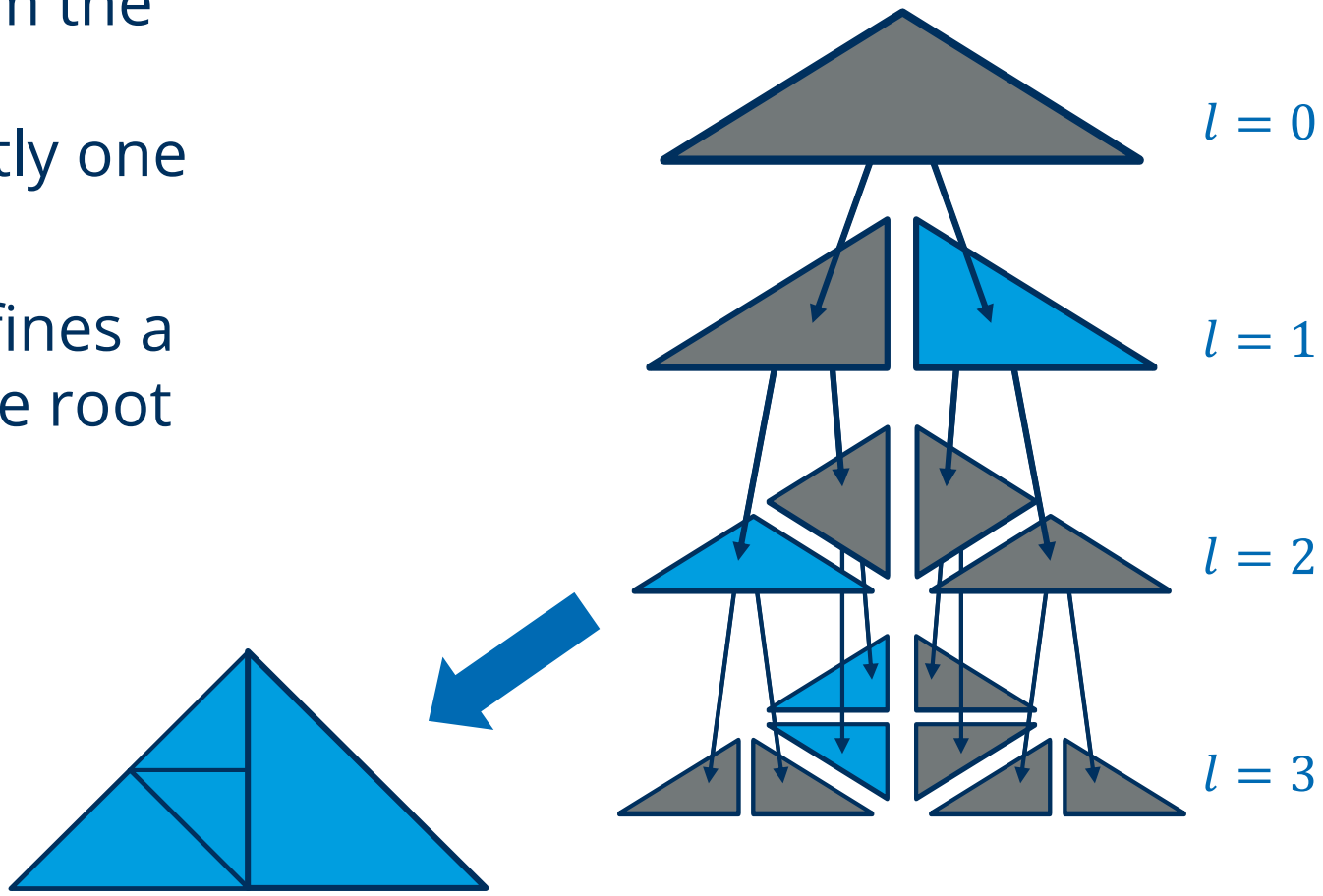


$l = 6$

Hierarchy of Right Triangles

- A front is a subset of nodes, such that each path from the root to a leaf contains exactly one node.
- Each front defines a partition of the root triangle,

for example:

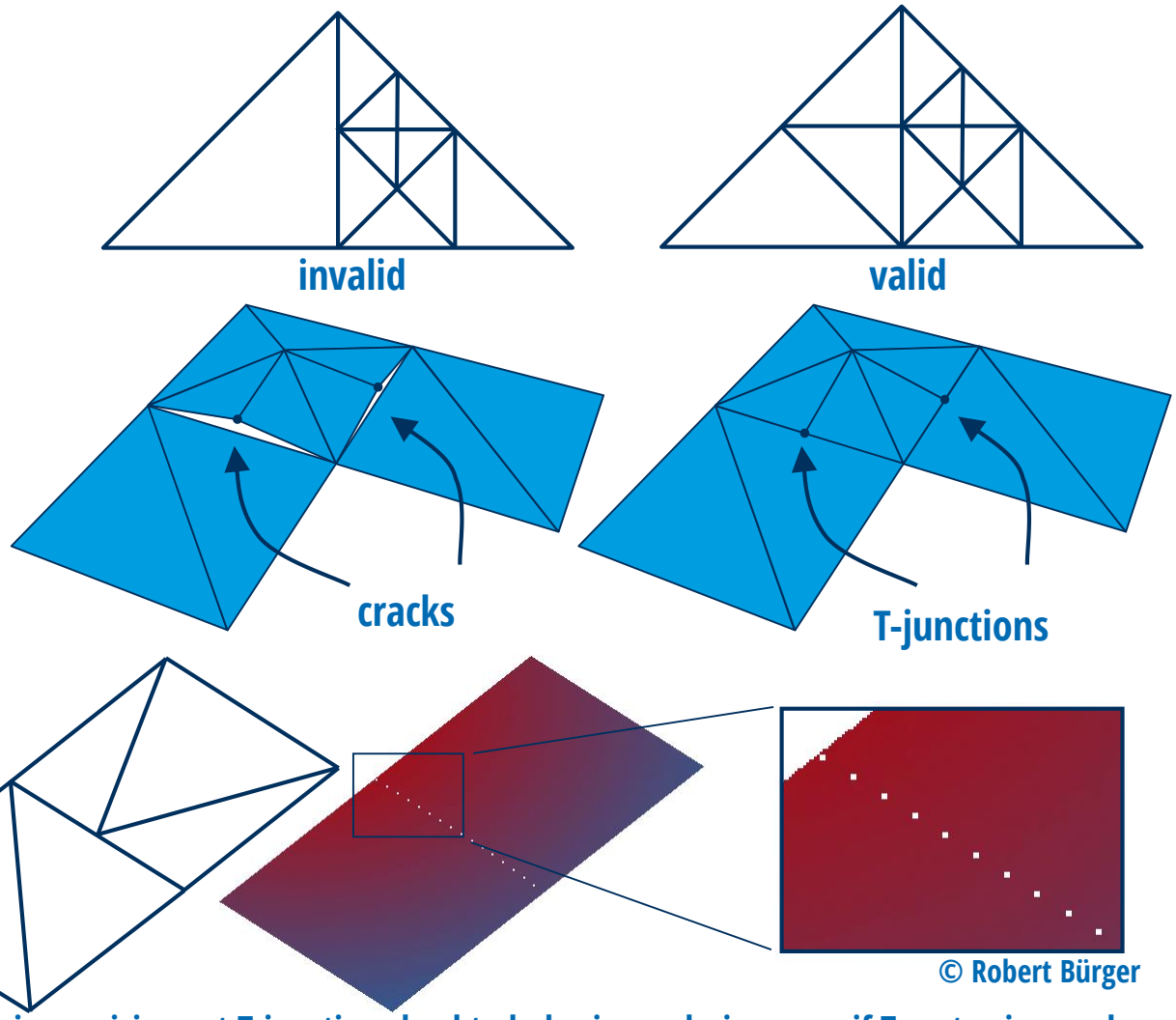


Hierarchy of Right Triangles

- A triangulation is valid if the intersection of two triangles is
  - empty,
  - common corner or
  - common edge

➔ no cracks

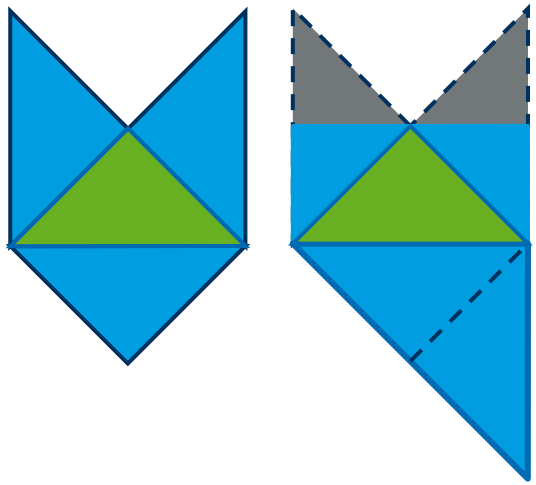
➔ no T-junctions



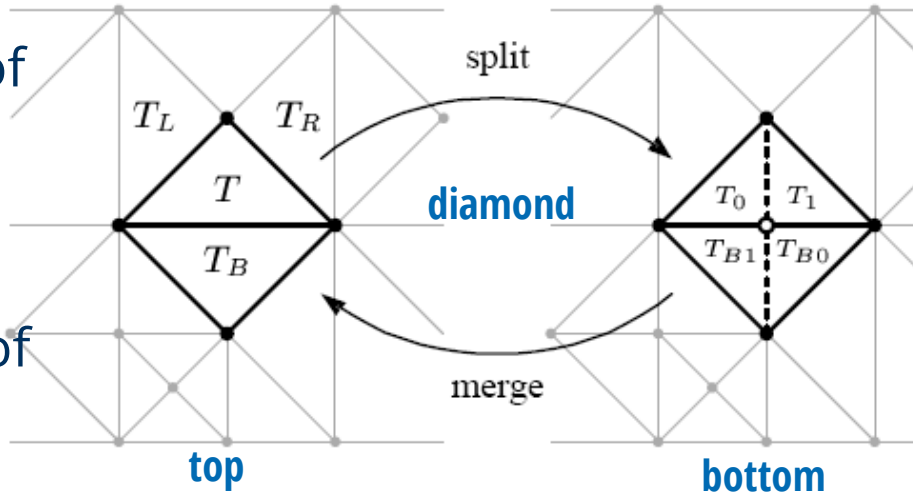
numeric imprecisions at T-junctions lead to holes in rendering even if T-vertex is on edge

Hierarchy of Right Triangles

Conditions for valid triangulation



- For each triangle of a valid triangulation it must hold
  - on the short side there can be triangles of the same or a finer level
  - the long side can have a triangle of the same or a coarser level
- two triangles T, T_B adjacent over long edge form top of a diamond
- bottom of diamond is composed of 4 triangles T_0, T_1, T_{B0}, T_{B1}
- **Theorem:** Any valid triangulation can be transformed to any other valid triangulation by a sequence of diamond split and merge operations



Terrain

RENDERING

Adaptive Refinement – Control

Refinement Criterion

- binary predicate that maps each triangle / diamond to binary value {accurate,inaccurate}
- It is based on
  - user defined threshold τ on geometric screen space error measured in pixels
  - user defined threshold γ on color difference caused by texture approximation or difference in lighting calculations

Priority

- per triangle / diamond measure of importance for further refinement
- Used for adaptive refinement with limited time budget
- It can be based on
  - screen space error
  - view-frustum visibility
  - silhouette edges
  - occlusion culling
  - backface culling
  - objects placed in terrain

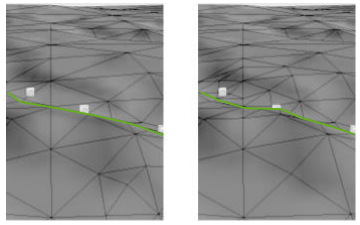


Figure 9: Left side is before LOS correction, right is after.

Adaptive Refinement – Strategy

Top Down

- The starting point is the coarsest diamond, which covers the entire terrain
- refinement
  - Split operations are performed recursively until all triangles meet an adaptation criterion, or
  - unprocessed triangles / diamonds are processed in order of decreasing priority with the help of a priority queue

Incremental

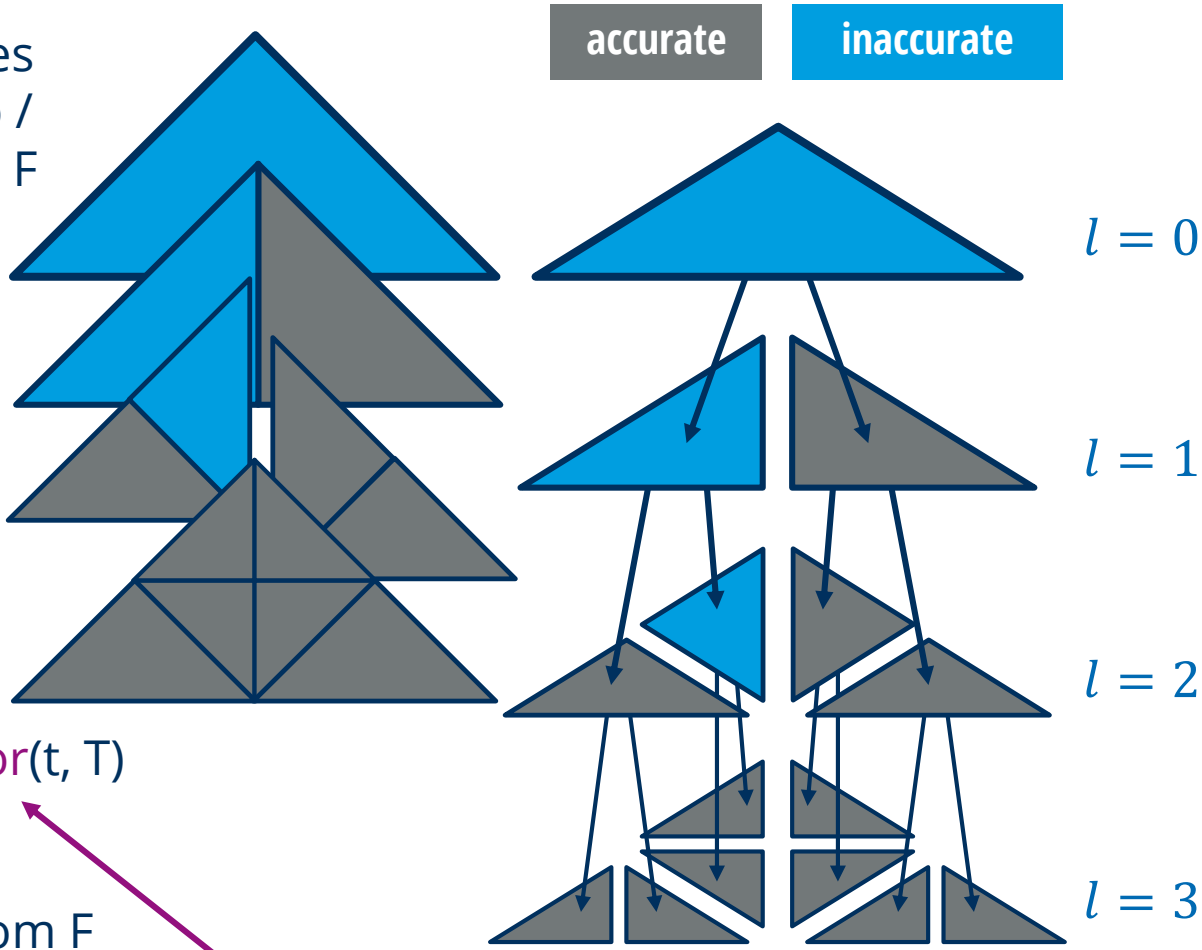
- Starting point is the adaptive tessellation of the previous time step
- two [priority] queues are necessary and all previous triangles / diamonds are inserted into both:
  - Split queue performs refinement as in top down approach
  - Merge queue performs coarsening where less resolution is necessary

Adaptive Refinement – Algorithm

Top Down Refinement on Triangles

```
T := mesh with base triangles
insert triangles of T into fifo /
                    priority queue F
while not F.empty()
    t = F.extract()
    if not accurate(t)
        force_split(t, F, T)

output triangles in T
```



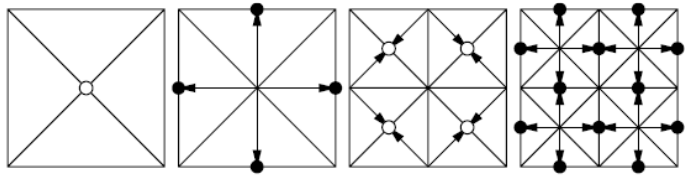

```
force_split(t, F, T)
    t_n := long_side_neighbor(t, T)
    if level(t_n,T) < level(t,T)
        force_split(t_n, F, T)
    remove diamond top from F
    split diamond of t in T
    insert diamond bottom into F
```

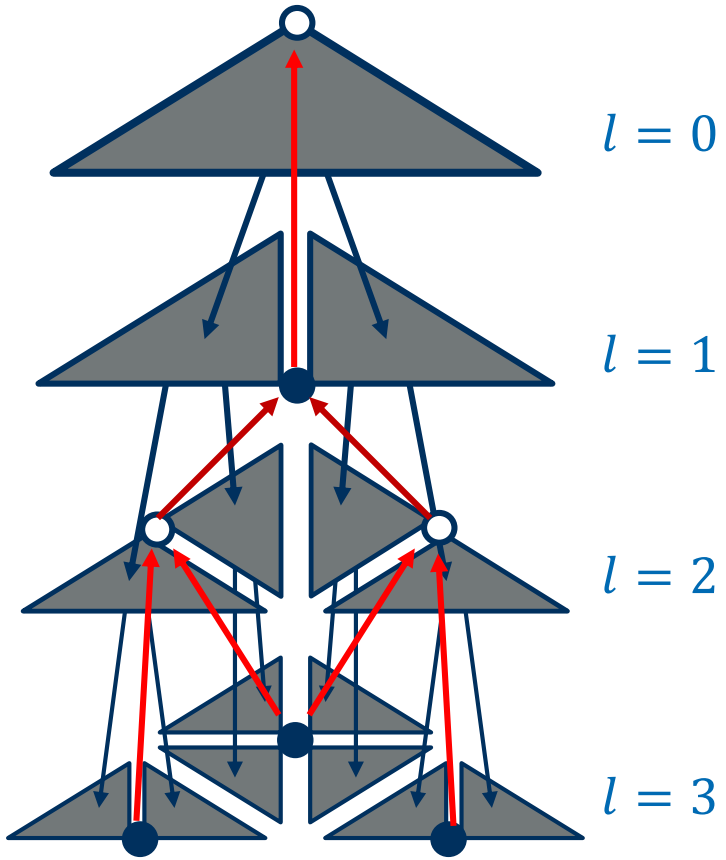
to implement this either long side neighbor or triangle selection status need to be stored and updated during adaptive refinement

this is called state based

Diamond DAG – Definition

- Diamonds can be identified with the vertex index i inserted in their bottom tessellation
- In top figure on the right the even level vertices are filled white and the odd leveled vertices black
- Red arrows show dependencies among diamonds: in a valid triangulation a diamond vertex can only be inserted if the dependent vertices on coarser levels are also present
- Dependencies form a directed acyclic graph (DAG)

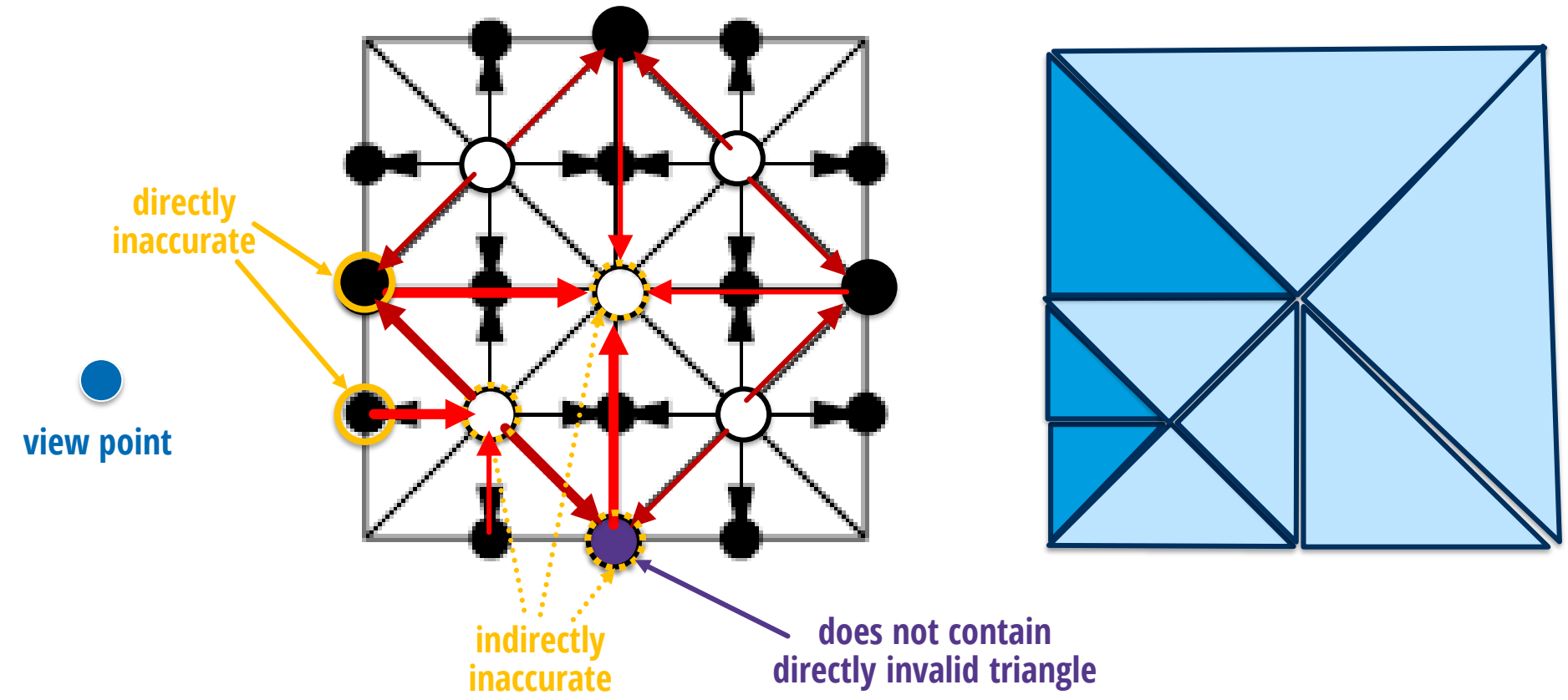




dependency arrows point in oposite direction as child arrows

Diamond DAG – Example

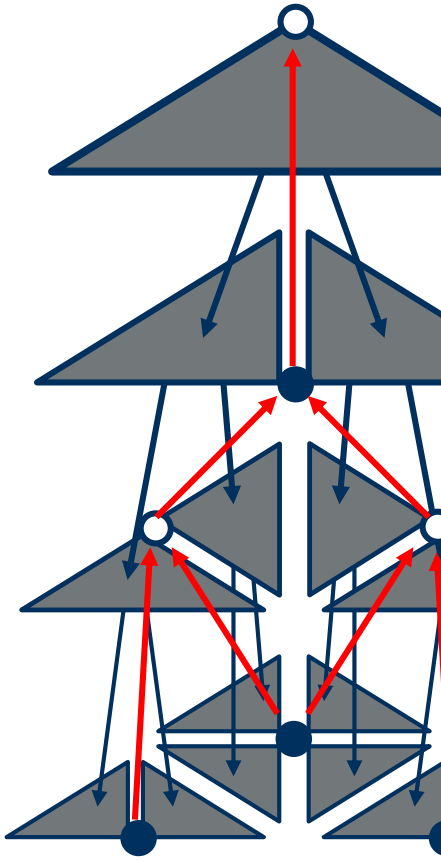
- DAG dependencies enforce presence of diamonds not containing any inaccurate triangle



Diamond DAG – Implement with Monotonicity

Diamond Monotonicity

- A refinement criterion $\rho(i)$ is diamond monotonous if $\rho(i) = inaccurate$ implies $\rho(j) = inaccurate$ for all parents j of which i is a child ($i \in C_j$).
- A priority $\pi(i)$ is diamond monotonous if $\pi(i) < \min \pi(j)$ for all parents j of which i is a child: $i \in C_j$.
- One can extent diamond monotonous refinement criteria and priorities to right triangles by mapping both top triangles to the value of the diamond
- Diamond monotonicity yields top down and incremental refinement algorithms without the need to check or store DAG dependencies. Implementations can be done without storing state of diamonds.

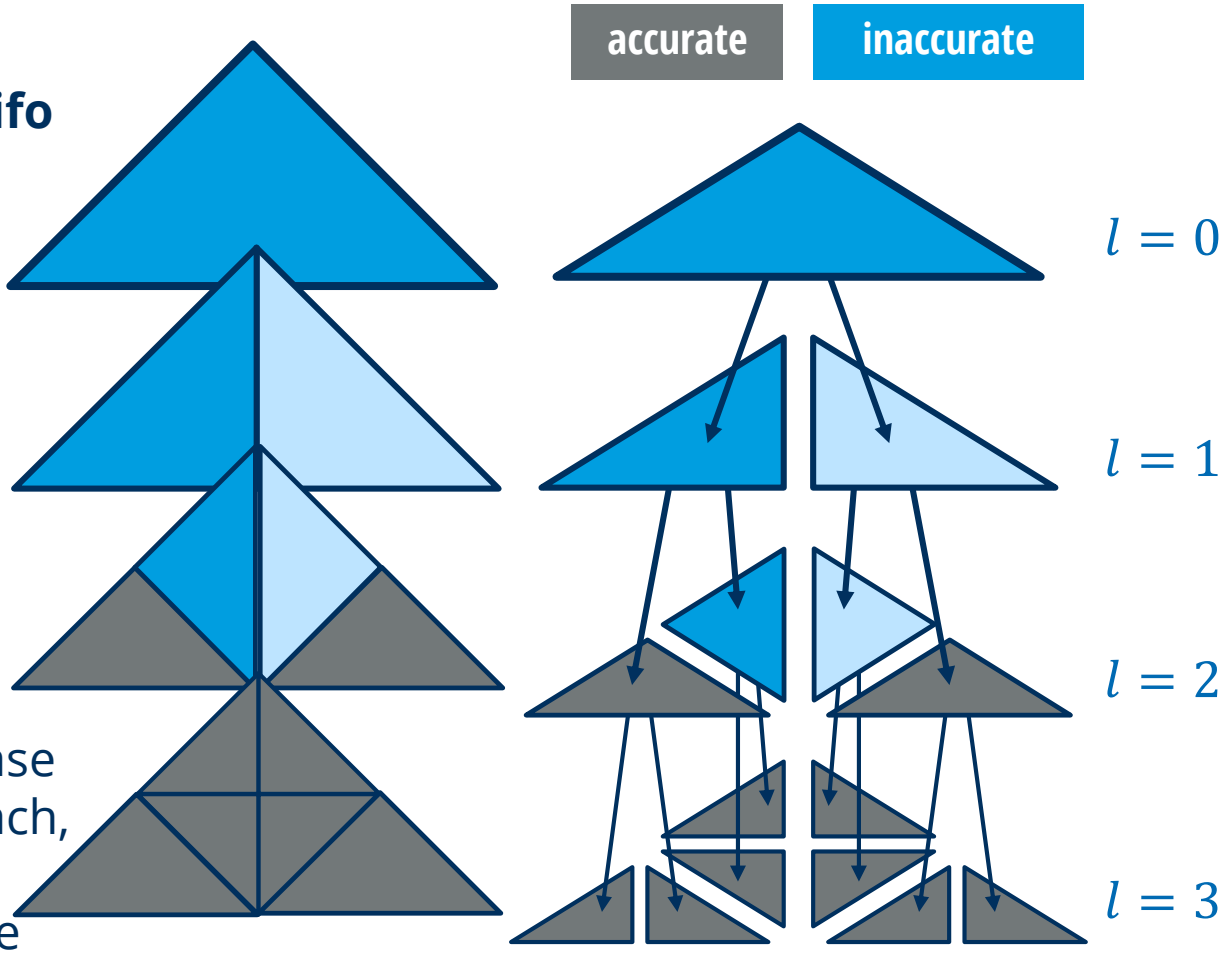


dependency arrows point in oposite direction as child arrows

Diamond DAG – Refinement Algorithm

State-Less Refinement with Diamond Monotonicity

```
T := ∅
insert base triangles into fifo
   or priority queue F
while not F.empty()
        t = F.extract()
        if accurate(t)
                T := T ∪ t
        else
                F.add(children(t))
```



- Before termination in case of priority based approach, process top element in queue if it is long edge neighbor of last processed triangle

Error Controlled Refinement – Overview

Preprocessing

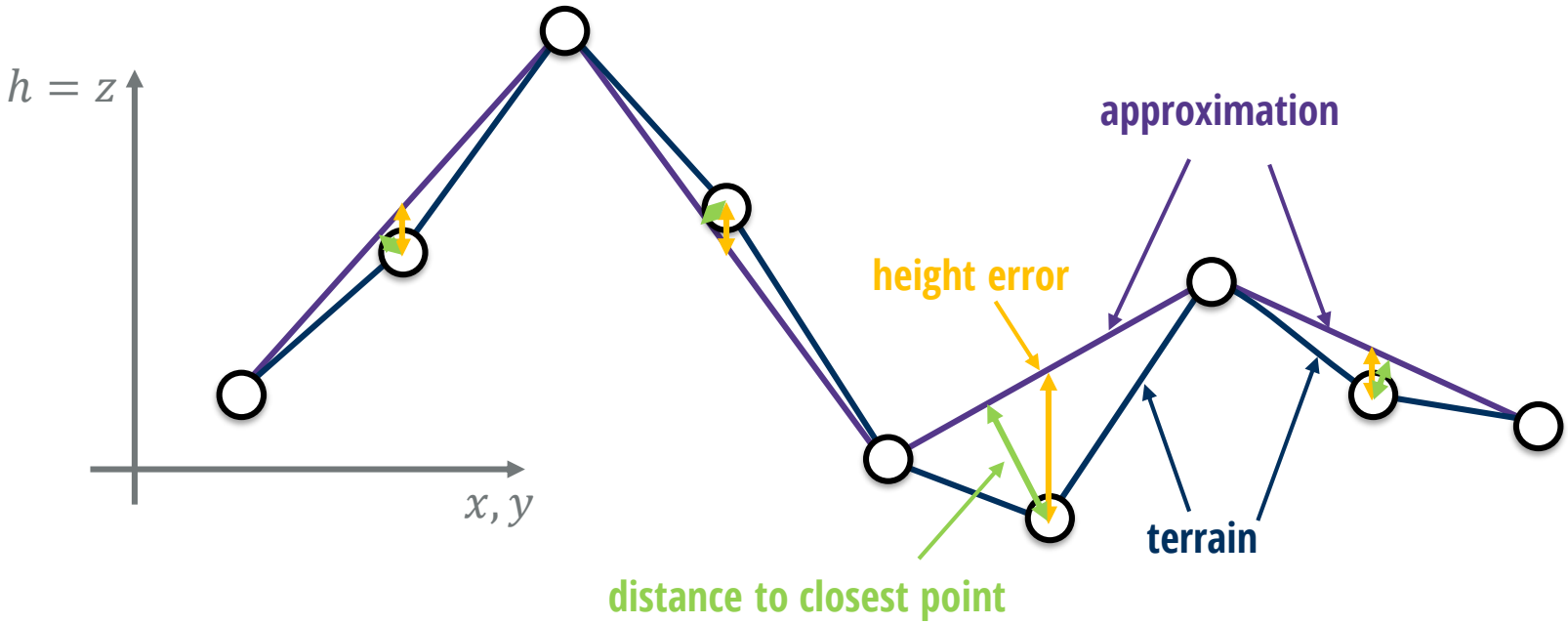
- determine geometric error introduced by removing diamond vertex (typically directed along height field direction)
- define bounding volume (extrusion of right triangle or sphere around diamond center point)
- post process error and bounding volume to ensure monotonicity (to avoid storing the Diamond DAG)

Adaptive refinement

- Estimate screen space error conservatively by maximizing screen space projection of geometric error over bounding volume

Geometric Error

- Geometric Error can be measured by finding for each point on the terrain the closest point on the approximation and maximizing over all terrain points (one-sided Hausdorff-distance), or
- maximizing the error on the height value, which is typically used for terrain adaptation as it also provides a single direction of the error.



Monotonous Geometric Error over HRT

According to ROAM 1997 Paper:

- The geometric errors are computed from bottom to top
- Leaf triangles are assigned 0 error
- for each triangle further up in the hierarchy compute height/z-error of vertex splitting long edge and its projection onto the edge: $|z(v_c) -$

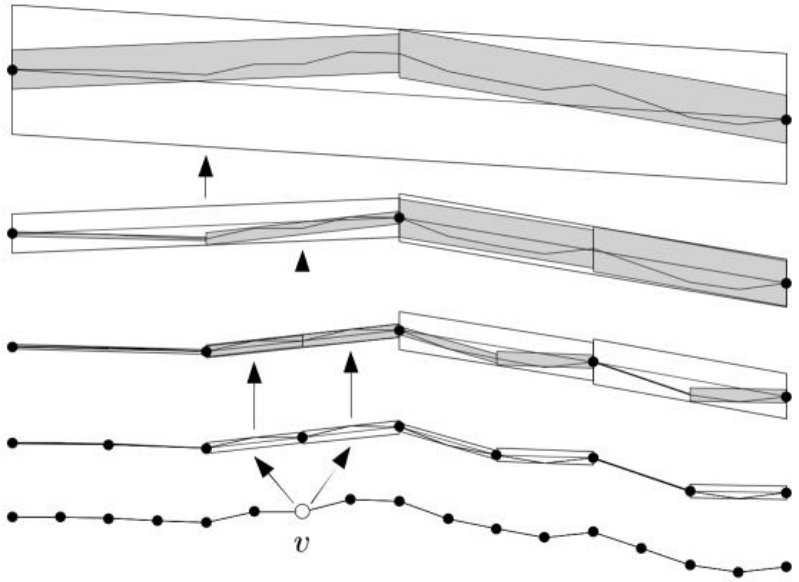
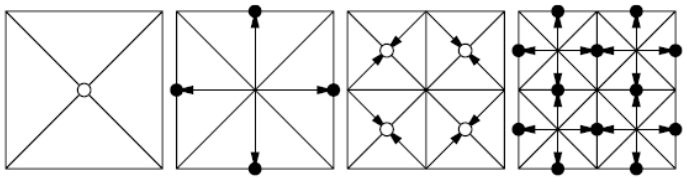
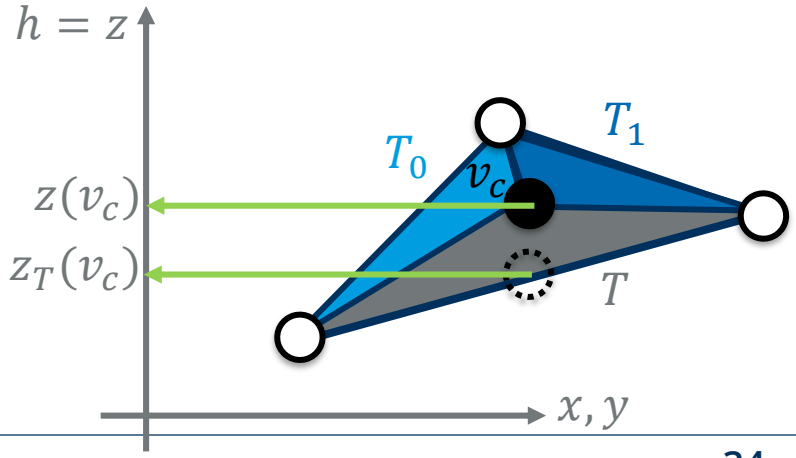


Figure 6: Nested wedgies for 1-D domain with dependents of v.

Duchaineau et al.: ROAM 1997

$$e_T = \max\{e_{T_0}, e_{T_1}\} + |z(v_c) - z_T(v_c)|$$

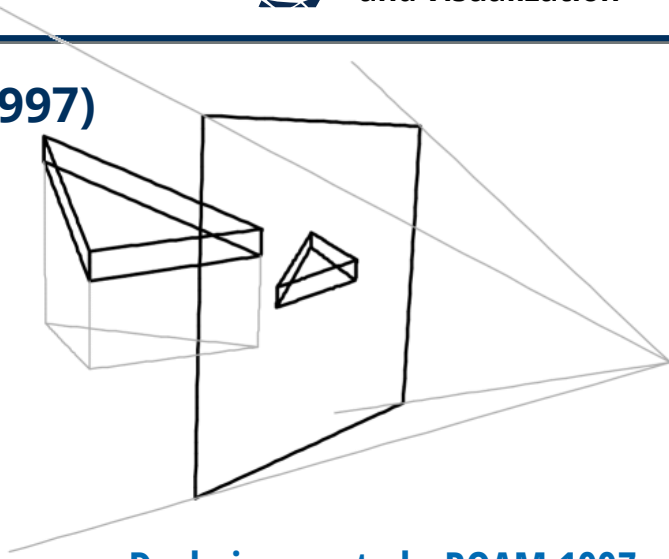




Thresholding Screen Space Error – ROAM

Maximum Project Length Error Vectors (ROAM 1997)

- extruding right triangles along the z-direction by the nested error value yields wedgies that can be used to estimate screen space error:
- For a wedge of a triangle T all error vectors can be spanned by the point pairs $\underline{\boldsymbol{w}}_{\pm}$ with
$$\underline{\boldsymbol{w}}_{\pm} = \underline{\boldsymbol{v}} \pm e_T \hat{\boldsymbol{z}}$$
for triangle point $\underline{\boldsymbol{v}}$ and z-axis $\hat{\boldsymbol{z}}$.
- Next transform to clip space:
$$\boldsymbol{MVP} \cdot \tilde{\boldsymbol{v}} \pm \boldsymbol{MVP} \cdot e_T \tilde{\boldsymbol{z}} \qquad \text{with } \tilde{\boldsymbol{z}} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$
- Error only depends on clip x, y, w components.
- We delay w-clip, apply viewport transform to x and y to get homo. screen coords of $\underline{\boldsymbol{v}}$ and $e_T \hat{\boldsymbol{z}}$:
$$\tilde{\boldsymbol{w}}_{\mathbf{screen},\pm} = (p, q, r) \pm (a, b, c)$$
- Pixel error is length of difference vector after w-clip:
$$\widehat{\mathbf{dist}}(v) = \left\| \frac{p+a}{r+c} - \frac{p-a}{r-c}, \frac{q+b}{r+c} - \frac{q-b}{r-c} \right\|_2 = \frac{2}{r^2 - c^2} \left((ar - cp)^2 + (br - cq)^2 \right)^{1/2}$$



Duchaineau et al.: ROAM 1997

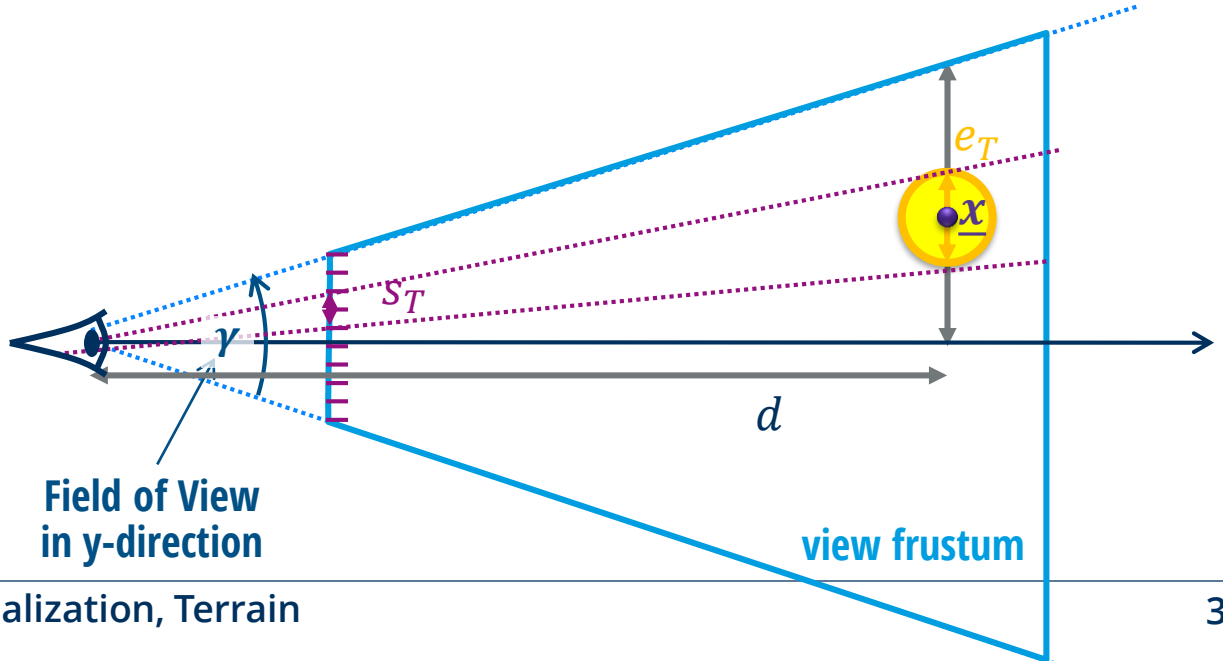
- Duchaineau showed that minimum of $r^2 - c^2$ and maximum of $(ar - cp)^2 + (br - cq)^2$ are achieved at tgl corners
- plugging corner min & max into $\widehat{\mathbf{dist}}$ yields upper bound on screen space error

Thresholding Screen Space Error – Blow

Screen Space Error from Isotropic Object Space Error (Blow 2000)

- In 2000 Blow proposed to use e_T as an isotropic object space error independent of current view direction.
- The screen space error s_T in pixels at point $\underline{\boldsymbol{x}}$ can be estimated from eye location $\underline{\boldsymbol{e}}$, screen height h in pixels, and field of view angle γ in y-direction: $s_T = c \cdot \frac{e_T}{\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}\|}$, with $c = \frac{h}{2 \tan \gamma/2}$
- This can be used as priority or to threshold against pixel error τ

$$\frac{s_T}{\frac{h}{2}} = \frac{e_T}{d \cdot \tan\frac{\gamma}{2}}$$



Thresholding Screen Space Error – Blow

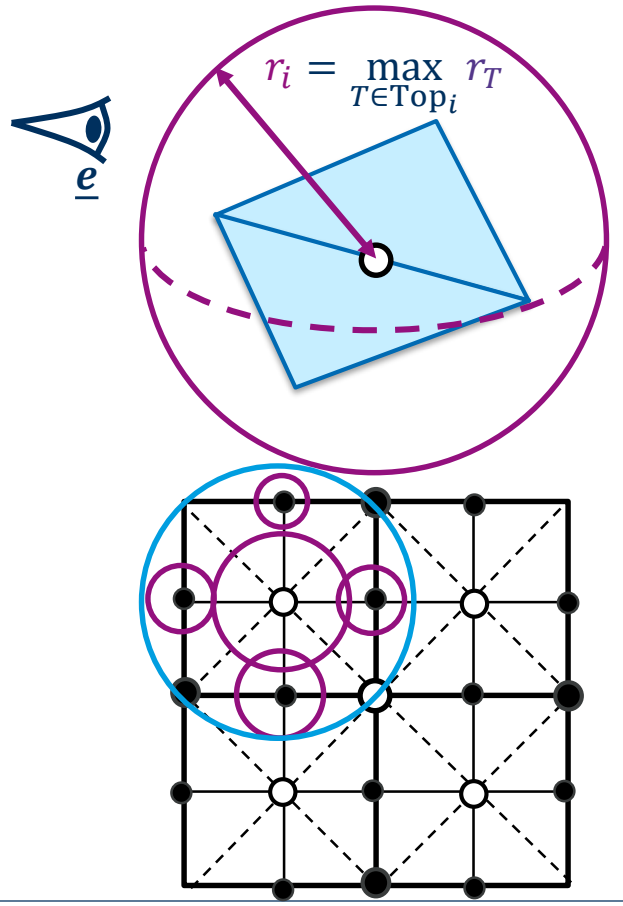
- Screen space threshold τ can be translated per triangle T to sphere radius r_T defining a threshold sphere, when entered by view-point triangle becomes inaccurate
- Per diamond radii r_i are computed as max over diamond top triangle radii.
- For monotonicity parent spheres are increased bottom up to enclose child threshold spheres by computing final radius $\hat{r}_i$:

$$\hat{r}_i := \begin{cases} 0 & \text{if } i \text{ is Leaf} \\ \max\left\{ r_i(\tau), \max_{j \in C_i} \left\{ \left\| \underline{\boldsymbol{p}}_i - \underline{\boldsymbol{p}}_j \right\| + \hat{r}_j \right\} \right\} & \text{otherwise} \end{cases}$$

where the $\underline{\boldsymbol{p}}_i$ are 3D points of diamonds

$$s_T = c \cdot \frac{e_T}{\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}\|} < \tau \qquad c := \frac{h}{2 \tan \frac{\gamma}{2}}$$

$$\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}\| > c \cdot \frac{e_T}{\tau} =: r_T$$



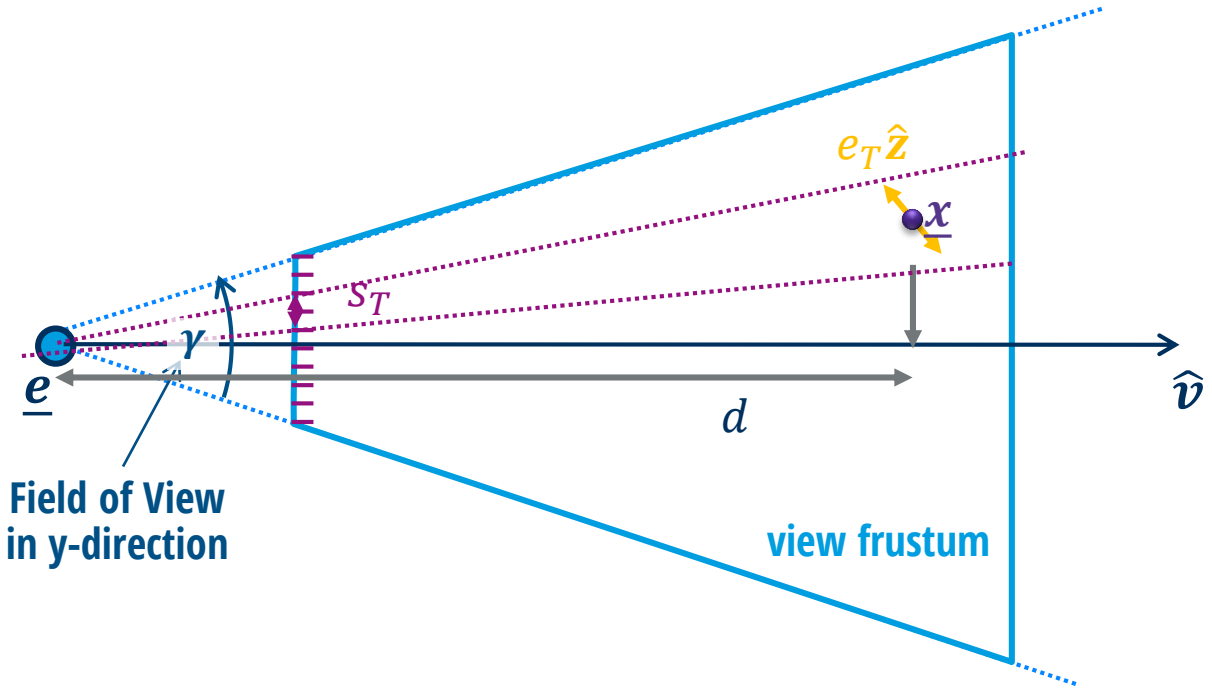
Thresholding Screen Space Error

- In case of an anisotropic error e_i directed along $\hat{\boldsymbol{z}}$ located at point $\underline{\boldsymbol{x}}$ viewed along direction $\hat{\boldsymbol{v}}$, the screen space error s_i in pixels is reduced by the cosine between z and v:

$$s_i = c \cdot (1 - |\langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{v}} \rangle|) \frac{e_i}{\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}\|}, \text{with } c = \frac{h}{2 \tan \gamma/2}$$

- Now threshold sphere depends on view angle

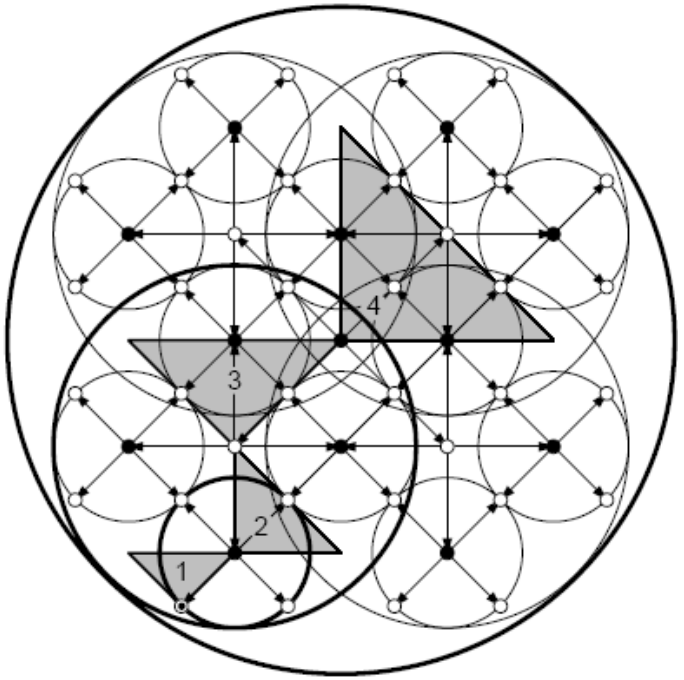
$$\frac{s_T}{\frac{h}{2}} = \frac{e_T}{d \cdot \tan \frac{\gamma}{2}}$$



Thresholding Screen Space Error

- Lindstrom 2001 proposed to build nested diamond balls $\mathcal{B}_i$ (spheres of radius r_i around diamond points $\underline{\boldsymbol{p}}_i$) independent of error:

$$r_i = \begin{cases} 0 & \text{if } i \text{ is a leaf node} \\ \max\limits_{j \in C_i} \{\|\mathbf{p}_i - \mathbf{p}_j\| + r_j\} & \text{otherwise} \end{cases}$$

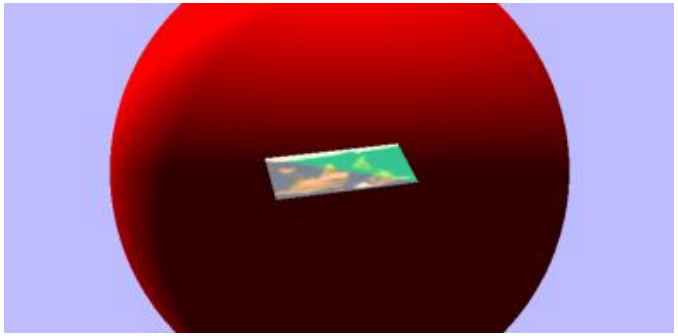


- And to maximize screen space error over nested diamond balls

$$s_i(e_i, \mathcal{B}_i, \underline{\boldsymbol{e}}) = \max_{\underline{\boldsymbol{x}} \in \mathcal{B}_i} c \cdot (1 - |\langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{v}} \rangle|) \frac{e_i}{\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{e}}\|}$$

- If eye is inside $\mathcal{B}_i$ diamond is to be split, otherwise screen maximum is achieved at point closest to eye:

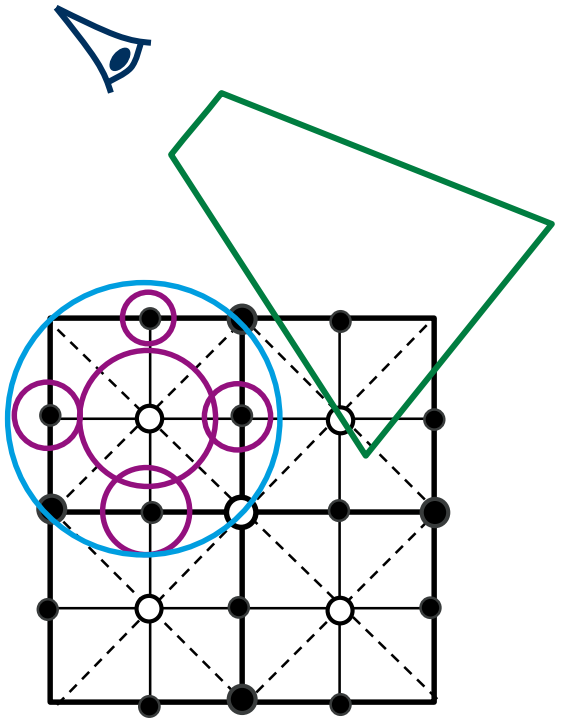
$$s_i(e_i, \mathcal{B}_i, \underline{\boldsymbol{e}}) = c \cdot (1 - |\langle \hat{\boldsymbol{z}}, \hat{\boldsymbol{v}} \rangle|) \frac{e_i}{\|\underline{\boldsymbol{p}}_i - \underline{\boldsymbol{e}}\| - r_i}$$



observation: nested balls become quite huge

View Frustum Culling

- View frustum culling gives significant speed up
- Lindstrom et al. 2001 propose to exploit the diamond balls for this
- The six view frustum planes are passed to the refinement algorithm
- If a ball is completely outside of one of the clipping planes, the whole subtree can be clipped away
- If a ball is completely inside of one clipping planes, all descendants are also inside and therefore no more view frustum culling test need to be performed anymore

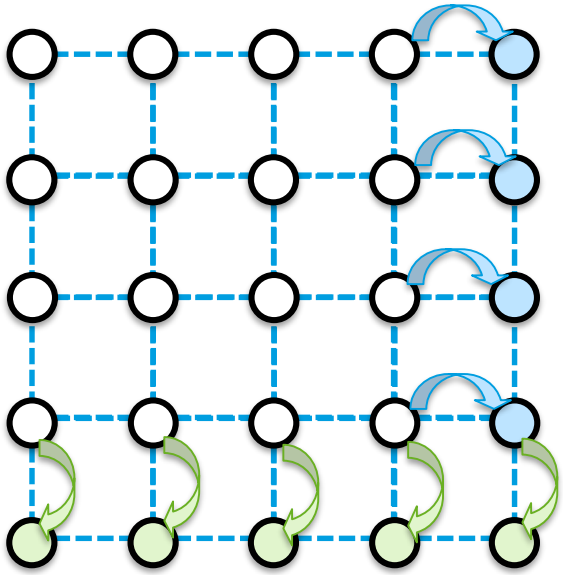


Terrain

IMPLEMENTATION

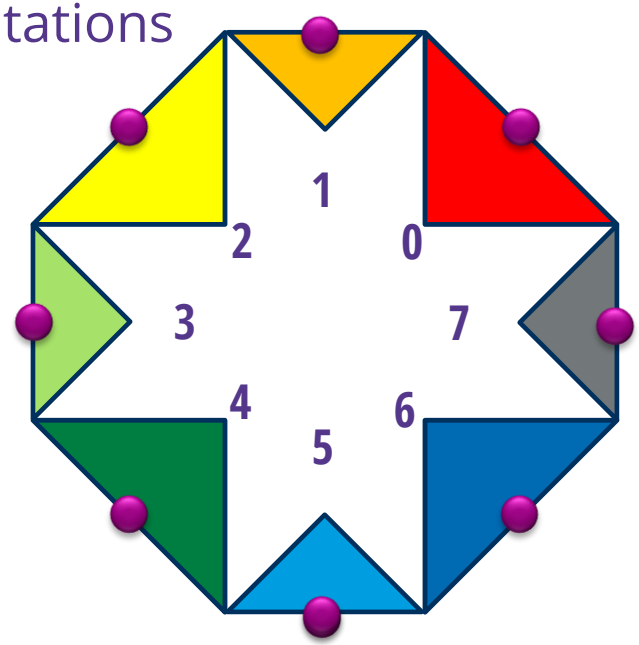
Implementation Details

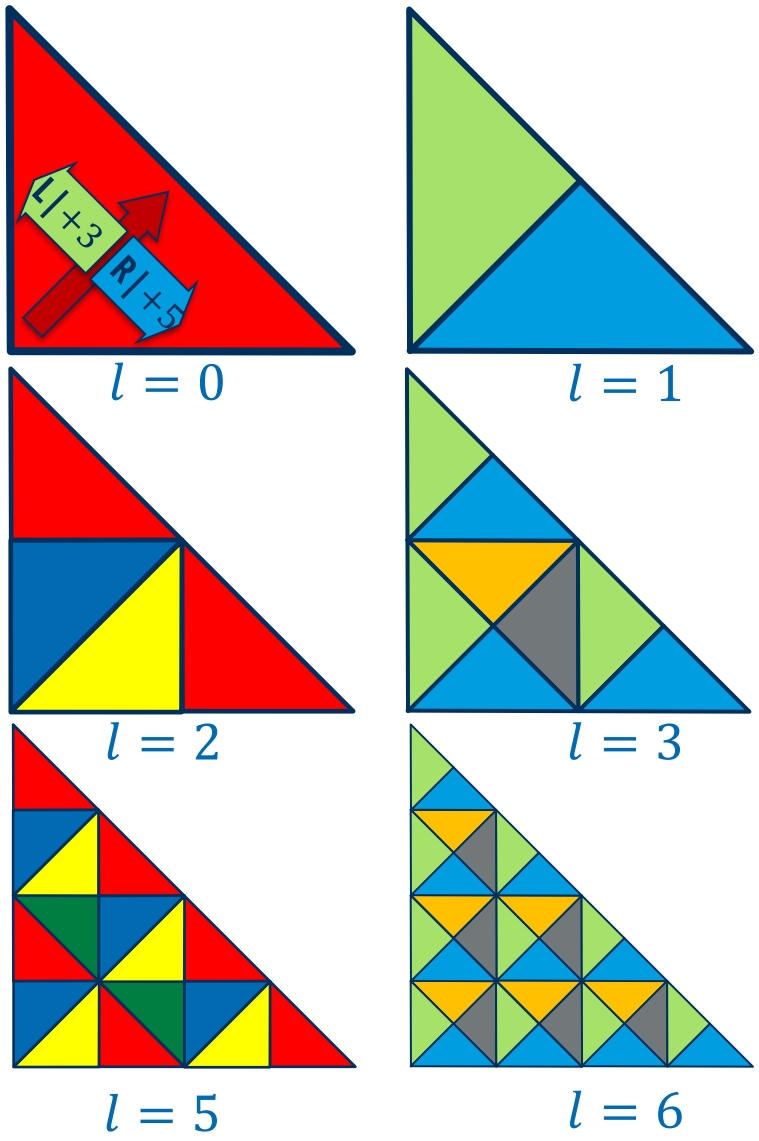
- Texture resolution is assumed to be squared with NxN texels, with N being a power of 2, i.e. 4096
- To make HRT work, texture needs to be padded to dimensions $(N+1) \times (N+1)$ typically by replicating last row and column



Implementation – Triangle Representation

- enumerate 8 different triangle orientations





- $(\underline{\iota}, L, \omega)$ represents triangle node
  - 2D diamond point $\underline{\iota}$ on longest edge measured in texels (i.e. ranging from 0...4096)
  - Edge length L along x or y coordinate axis (i.e. 1, 2, 4, 8, ..., 4096)
  - triangle orientation ω (i.e. 0 ... 7)

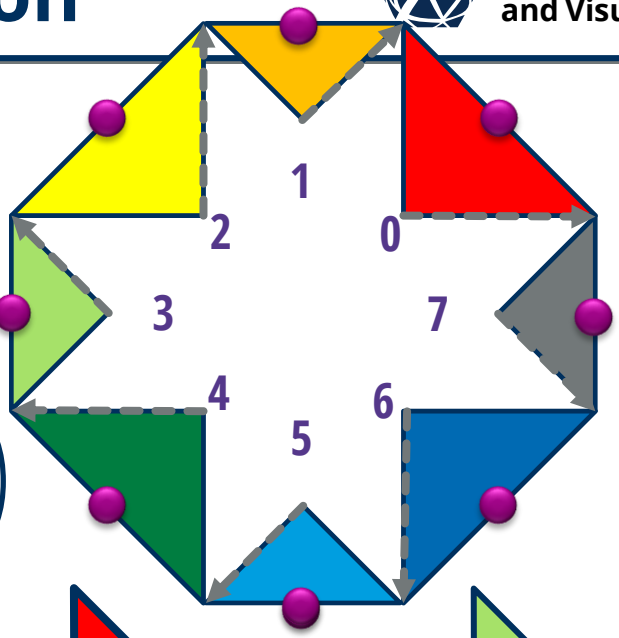
Implementation – Navigation

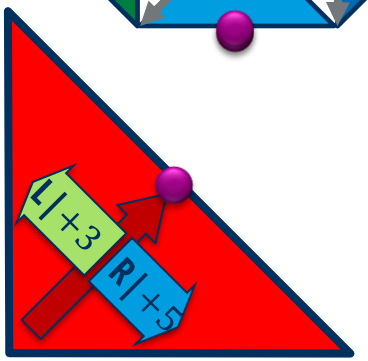
- $\vec{v}(L,\omega) = \left[\begin{pmatrix} L \\ 0 \end{pmatrix}, \begin{pmatrix} L/2 \\ L/2 \end{pmatrix}, \begin{pmatrix} 0 \\ L \end{pmatrix}, \begin{pmatrix} -L/2 \\ L/2 \end{pmatrix}, \dots\right]$
- $\texttt{leftChild}(\underline{\iota}, L, \omega)$

  $\left(\underline{\iota} + \vec{v}(L/2, \omega+4), \begin{cases} L & \omega \text{ even} \\ L/2 & \omega \text{ odd} \end{cases}, \omega+3\right)$
- $\texttt{rightChild}(\underline{\iota}, L, \omega)$

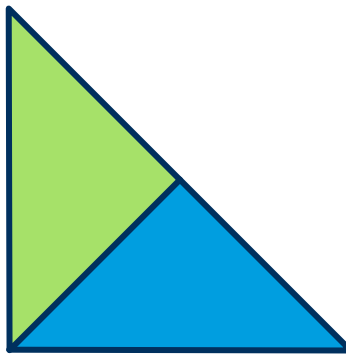
  $\left(\underline{\iota} + \vec{v}(L/2, \omega+6), \begin{cases} L & \omega \text{ even} \\ L/2 & \omega \text{ odd} \end{cases}, \omega+5\right)$
- $\texttt{diamondTopSibling}(\underline{\iota}, L, \omega)$

  $\begin{cases} \text{undef} & \iota_x = 0 \vee \iota_y = 0 \vee \\ & \iota_x = N \vee \iota_y = N \\ (\underline{\iota}, L, \omega+4) & \text{otherwise} \end{cases}$

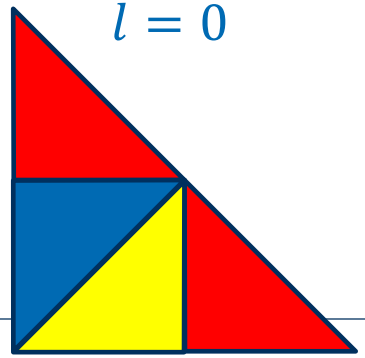




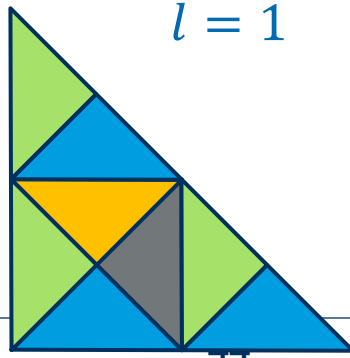
$l = 0$



$l = 1$



$l = 2$

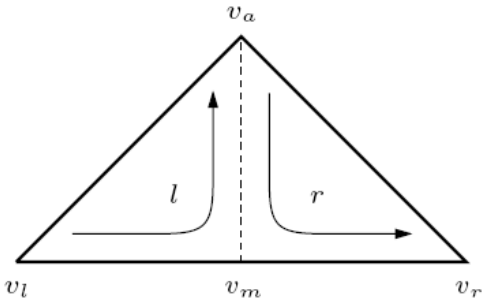


$l = 3$

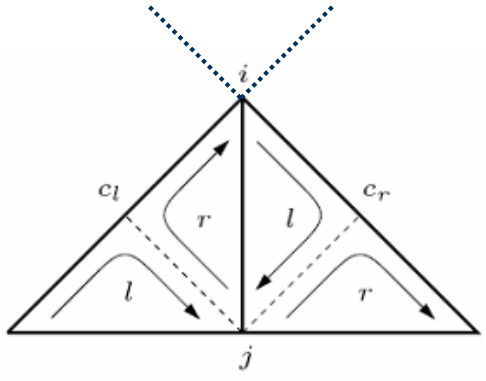
operations on ω are always modulo 8

Implementation – Stripification

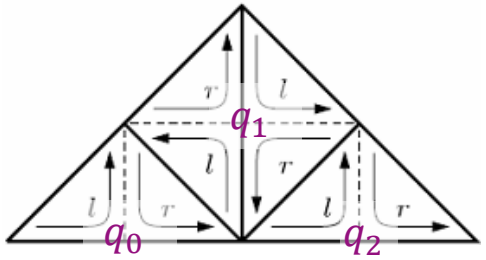
- Lindstrom et al. 2001 propose to directly generate single triangle strip during adaptive refinement
- No data structure is used
- For hierarchical navigation, the triangles are defined by a pair (i, j) of nodes on level l and $l-1$.
- When descending to the left/right partial triangle, the nodes c_l and c_r are needed
- meaning of “left” / “right” alternates from level to level



$v_l v_m v_a v_m v_r$



$v_l c_l v_m c_l v_a c_r v_m c_r v_r$



$v_l q_0 c_l q_0 v_m q_1 c_l q_1 v_a q_1 c_r q_1 v_m q_2 c_r q_2 v_r$

Implementation – Stripification

Computer Graphics and Visualization

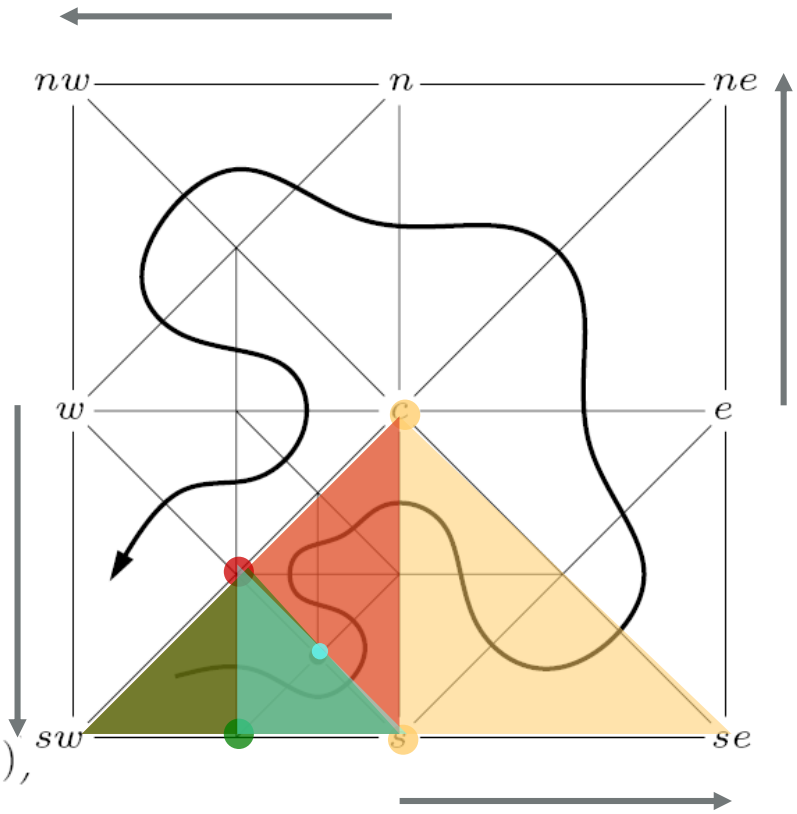
```
tstrip-append(V, v, p)
 1  if v ≠ v_{n-1} and v ≠ v_n then
 2      if p ≠ parity(V) then
 3          parity(V) ← p
 4      else
 5          V ← (V, v_{n-1})
 6      V ← (V, v)

submesh-refine(V, i, j, l)
 1  if l > 0 and active(i) then
 2      submesh-refine(V, j, c_l(i, j), l − 1)
 3      tstrip-append(V, i, l mod 2)
 4      submesh-refine(V, j, c_r(i, j), l − 1)

mesh-refine(V, n)
 1  V ← (i_sw, i_sw)
 2  parity(V) ← 0
 3  for each (j, k) ∈ ((i_s, i_se), (i_e, i_ne), (i_n, i_nw), (i_w, i_sw),
 4      submesh-refine(V, i_c, j, n)
 5      tstrip-append(V, k, 1)
```

level l with 0 for leaf and n for root

inaccurate



SOAR paper:

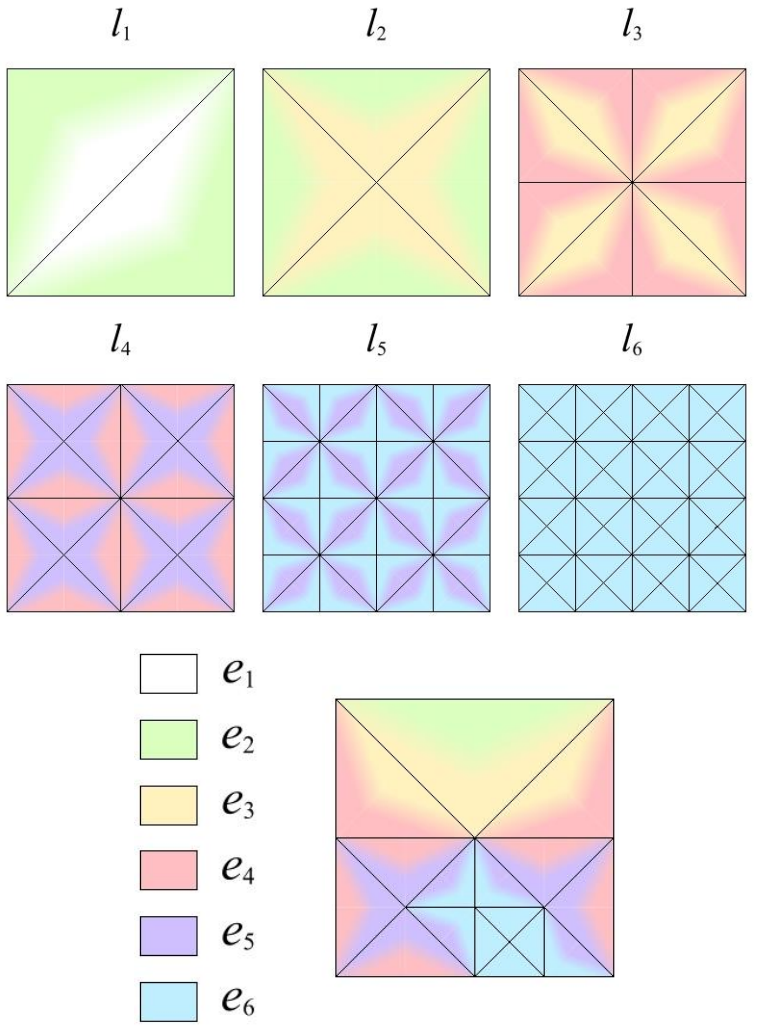
Peter Lindstrom and Valerio Pascucci, *Visualization of Large Terrains Made Easy*
Proceedings of IEEE Visualization 2001, pp. 63-370, 574, October 2001

Terrain

BATCHED METHODS

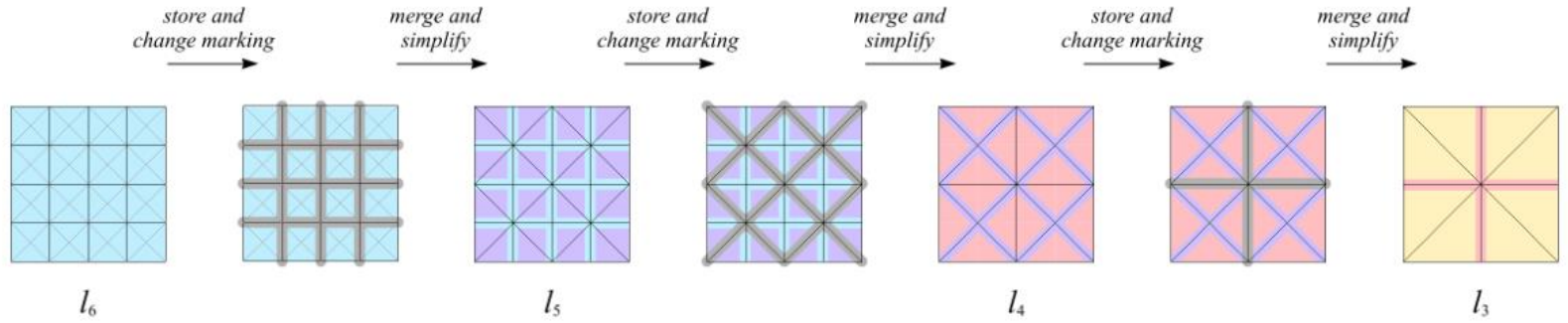
Batched HRT

- Per triangle adaptation on CPU cannot cope with modern GPU speed
- Cignoni et al. propose in BDAM paper to use HRT on triangular batches
- Each triangle in the HRT is replaced by a TIN which is optimized with respect to underlying geometry by a mesh simplification algorithm and for fast rendering by generating triangle strips

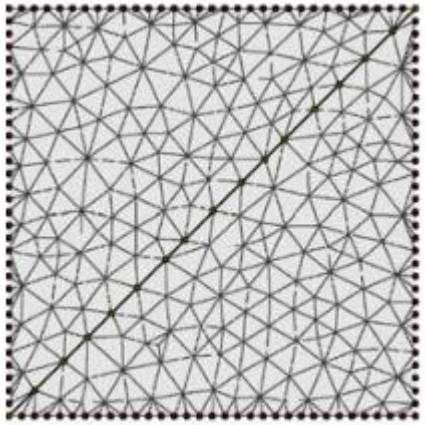


Cignoni et al., BDAM – Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization, EG 2003

Batched HRT



- Batches are built bottom up
- per diamond a simplification algorithm is used to half the triangle count
  - vertices along diamond boundaries (long edges of finer level) are marked to stay untouched
  - vertices along long edges of coarser level are restricted to move on edge only
- simplification errors are used in combination with nested spheres for view-dependent refinement



zoomed in view into a coarsened diamond

Quadtree based refinement

- Color texture is typically organized in quadtree
- Quadtree can also be used for geometry refinement
- Typically adaptation is restricted in a way to ensure that resolution change is nowhere more than one
- tessellation is extracted by splitting quadtree cells into triangle fans according to vertices on cell boundary

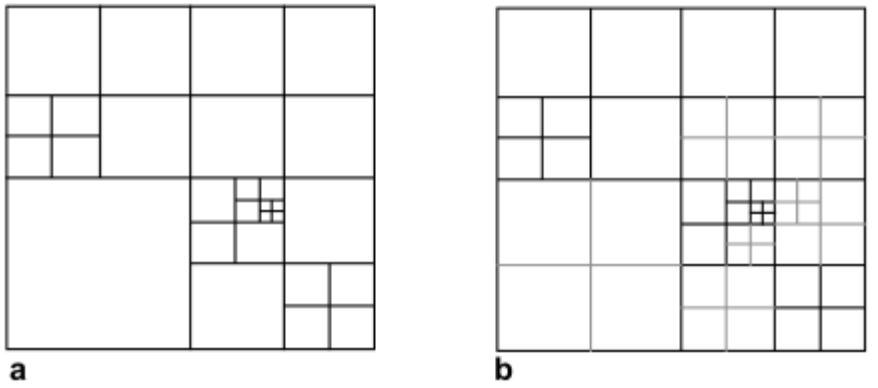


Fig. 4. a Example of an unrestricted quadtree subdivision in parameter space. **b** The restricted subdivision

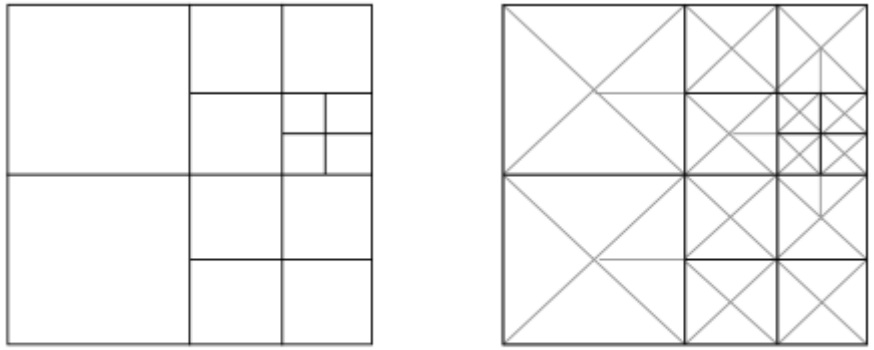


Fig. 7. Improved conforming triangulation of a restricted quadtree subdivision as in [58]

Quadtree based refinement

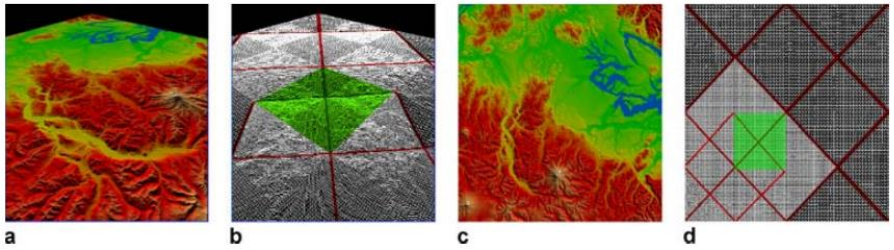


Fig. 1a–d. Terrain rendering using seamless patches. **a** A selected view. **b** The wire-frame of **a**, where the *green* region marks one patch. **c** Top view of **a**. **d** The wire-frame of **c** with the same marked patch

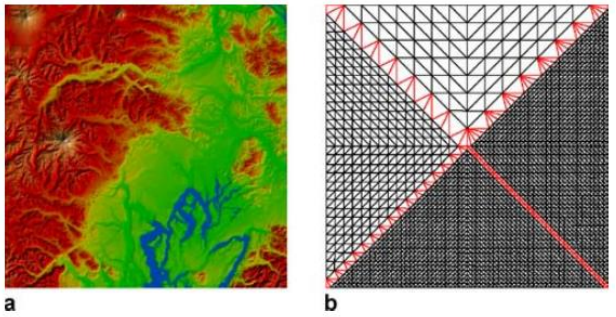


Fig. 2. a An image of one patch. **b** The wire-frame of the same patch which shows the stitching of different resolution tiles

- Again a batched version can better balance between CPU and GPU speed
- Livny et al. propose to use a patch per quadtree cell that can contain triangles of different resolution
- Specialized triangle strips are used to adapt resolution changes

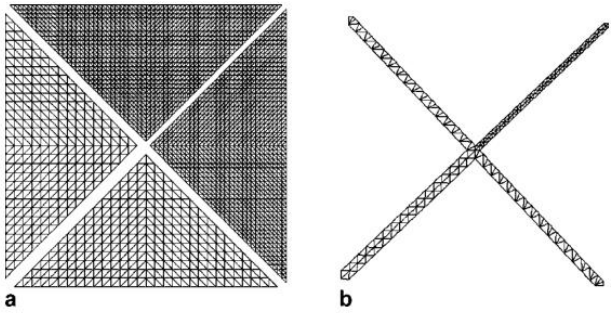


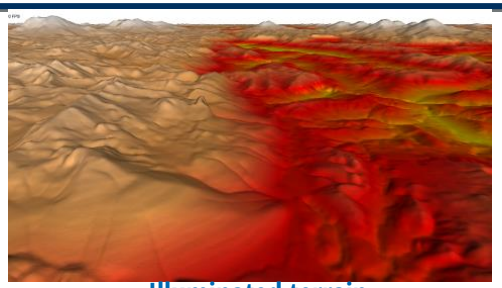
Fig. 3a,b. The components of one patch. **a** The image of four triangular tiles. **b** The image of four strips

Livny, Y., Kogan, Z., & el-Sana, J. (2009). Seamless patches for GPU-based terrain rendering. *The Visual Computer*, 25(3), 197-208.

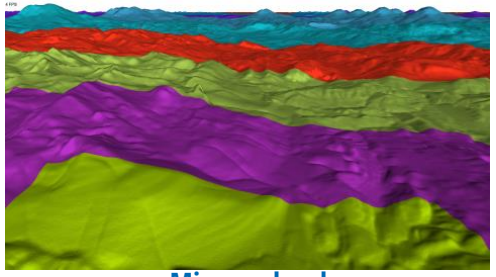
Geometry Clipmaps

- Idea: use mipmap hierarchy to store color and elevation textures such that texture extent increases with decreasing texture level
- Tessellate with resolution adapting triangle strips
- Upload data incrementally when changing the view point
- Blend geometry and color close to level transition
- Supports compressed images
- Suited for GPU implementation
- Combined with texture synthesis

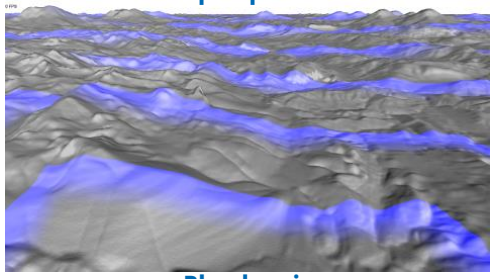
- **F. Losasso, H. Hoppe, Geometry clipmaps: Terrain rendering using nested regular grids. SIGGRAPH 2004.**
  Geometry clipmap framework with compressed and fractal terrains
- **A. Asirvatham, H. Hoppe, Terrain rendering using GPU-based geometry clipmaps. GPU Gems 2, 2005.**
  Efficient GPU implementation
- **S. Lefebvre, H. Hoppe, Parallel controllable texture synthesis. SIGGRAPH 2005**
  Parallel texture synthesis algorithm



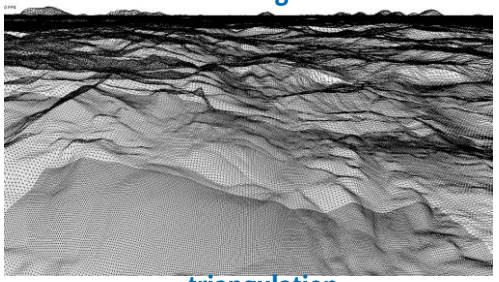
Illuminated terrain



Mipmap levels



Blend regions



triangulation

Demo

Terrain

SHADING

Local Lighting

- For local lighting we need to compute a the terrain normal, which can be computed from the height field gradient according to

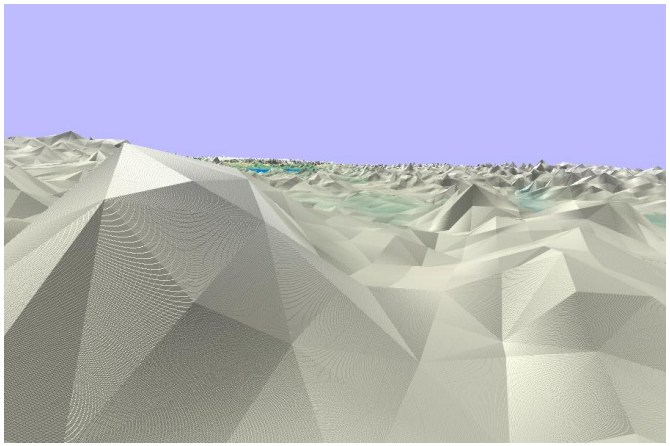
$$\hat{\boldsymbol{n}} = \text{normalize}\left(-h_x, -h_y, 1\right)^T$$

- The normal can either be stored in a normal texture, or (see exer.)
- to avoid additional storage, one can also compute the normal in the fragment shader exploiting the derivative functions dFdx and dFdy of glsl that compute the derivatives in screen space:

```
vec3 estimate_normal(in vec2 tc)
{
    vec2 tx = dFdx(tc);
    vec2 ty = dFdy(tc);
    vec2 hg = vec2(
        texture(dem_tex, tc + 0.5*tx).x - texture(dem_tex, tc - 0.5*tx).x,
        texture(dem_tex, tc + 0.5*ty).x - texture(dem_tex, tc - 0.5*ty).x);

    hg /= (extent.xy*vec2(length(tx), length(ty)));
    hg *= extent.z;
    return normalize(get_normal_matrix()*vec3(-hg, 1.0));
}
```

- When zooming into the normals computed in the shader, on can see artefacts on the bilinear interpolation of the GPU



linear texture magnification imprecision is visible in the lighting artefacts

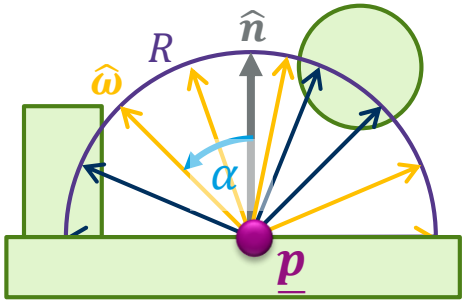
Ambient Occlusion

Zhukov, S., Iones, A., & Kronin, G. (1998). An ambient light illumination model. In *Rendering Techniques' 98* (pp. 45-55). Springer, Vienna.

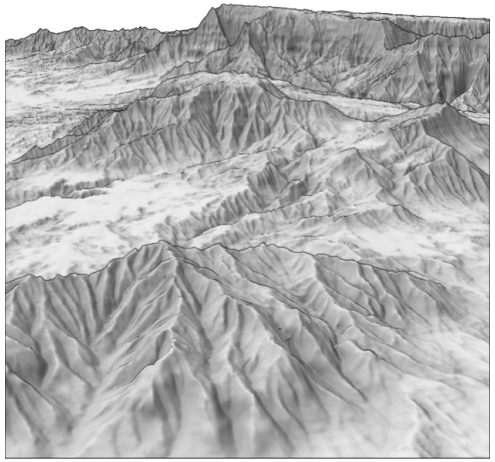
- Ambient occlusion (AO) assumes ambient light everywhere in the scene or coming from the sky.
- At every surface point $\underline{\boldsymbol{p}}$ a scalar factor $A_{\underline{\boldsymbol{p}}} \in [0,1]$ measures the percentage of the arriving light by integrating over all directions $\hat{\boldsymbol{\omega}}$ of the hemisphere around the surface normal $\hat{\boldsymbol{n}}$:

$$A_{\underline{\boldsymbol{p}}} = \frac{1}{\pi} \iint_{\hat{\boldsymbol{\omega}} \in \Omega} \rho\left(L\left(\underline{\boldsymbol{p}}, \hat{\boldsymbol{\omega}}\right)\right) \underbrace{\langle \hat{\boldsymbol{\omega}}, \hat{\boldsymbol{n}} \rangle}_{\cos \alpha} \cdot d\hat{\boldsymbol{\omega}}$$

- With the length $L\left(\underline{\boldsymbol{p}}, \hat{\boldsymbol{\omega}}\right)$ of the free path length to the next occluder clamped to a maximum radius R (typically estimation of L starts at $\underline{\boldsymbol{p}} + \epsilon\hat{\boldsymbol{\omega}}$)
- ρ can be used to emulate different AO models:
  - step function: $\rho(L) = \begin{cases} 0 & L < R \\ 1 & L = R \end{cases}$ used for sky illumination
  - linear function: $\rho(L) = L/R$ used for volumetric illumination
  - Other monotonous function mapping $[0, R]$ to $[0,1]$ possible



AO in 2D hillshading techniques



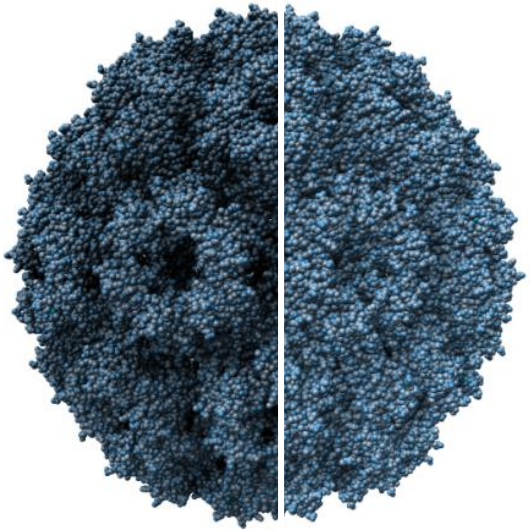
AO in 3D terrain rendering

Ambient Occlusion

Zhukov, S., Iones, A., & Kronin, G. (1998). An ambient light illumination model. In *Rendering Techniques' 98* (pp. 45-55). Springer, Vienna.

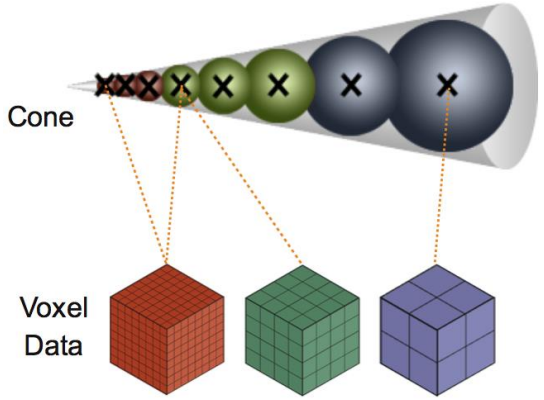
- Ambient occlusion can be evaluated in screen space (SSAO) as a post processing step or in object space (OSAO)
- Object space methods use N sample directions $\hat{\boldsymbol{\omega}}_i$ sampled with probability $p(\hat{\boldsymbol{\omega}}_i)$ and do a Monte Carlo estimate:

$$A_{\underline{\boldsymbol{p}}} = \frac{1}{N} \cdot \frac{1}{\pi} \sum_i \frac{\rho\left(L(\underline{\boldsymbol{p}}, \hat{\boldsymbol{\omega}}_i)\right) \langle \hat{\boldsymbol{\omega}}_i, \hat{\boldsymbol{n}} \rangle}{p(\hat{\boldsymbol{\omega}}_i)}$$



OSAO vs SSAO

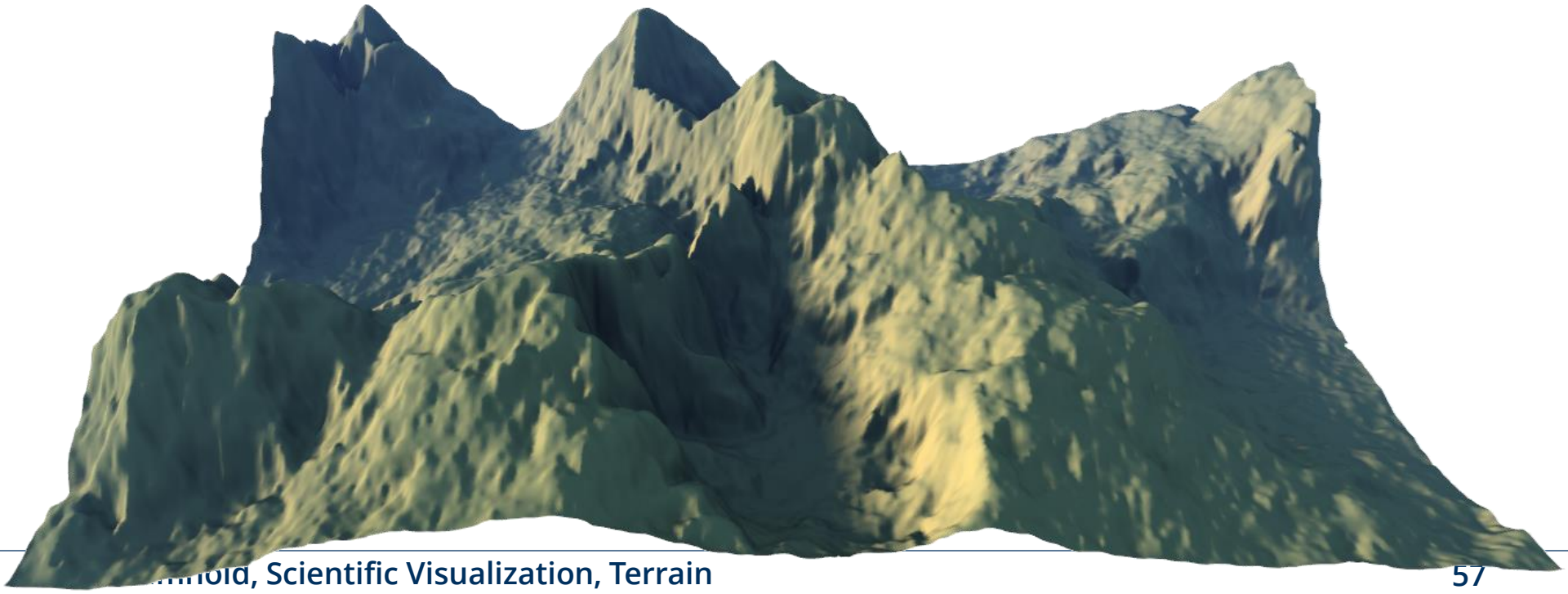
- For a given direction $\hat{\boldsymbol{\omega}}_i$, shadow mapping can be used to compute L for all scene points in one rendering pass in parallel
- With the help of a voxelization of the scene, voxel cone tracing can be used to trace a direction cone very efficiently
- In terrain rendering the terrain is typically static and AO can be "baked" into the color texture in a preprocessing step



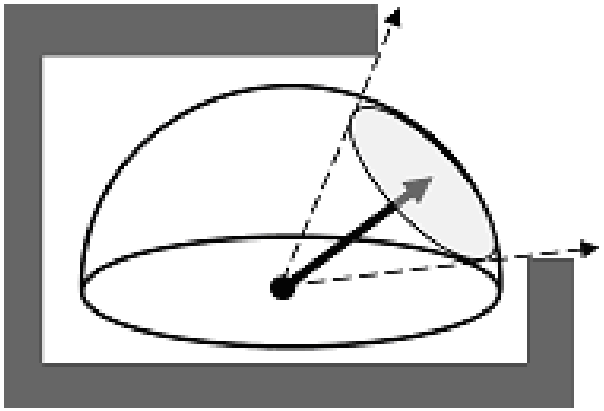
Ambient Aperture Lighting

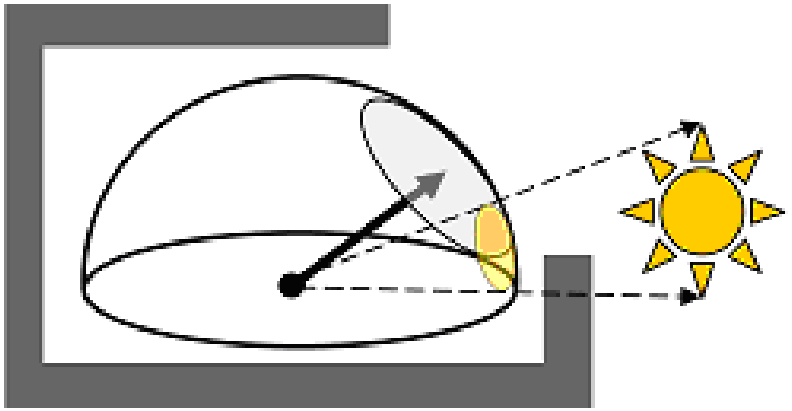
Oat, C., & Sander, P. V. (2007, April). Ambient aperture lighting. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (pp. 61-64). ACM.

- Assumption: terrain (optionally with buildings) is static
- Features: dynamic point & spherical area Light sources with hard & soft shadows
- Approach: pre-compute visibility



Ambient Aperture Lighting





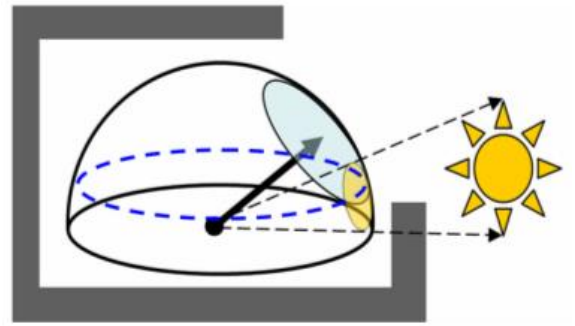
Precomputation Stage

- at every mesh vertex / height field texel compute directional visibility
- approximate visibility function with spherical cap stored as direction and radius

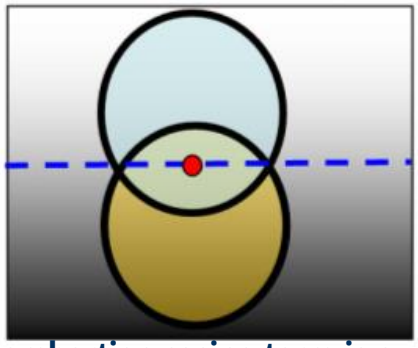
Rendering Stage

- Spherical cap acts as an aperture
- Light sources are projected to sphere and clipped against aperture
- remainder determines incoming light

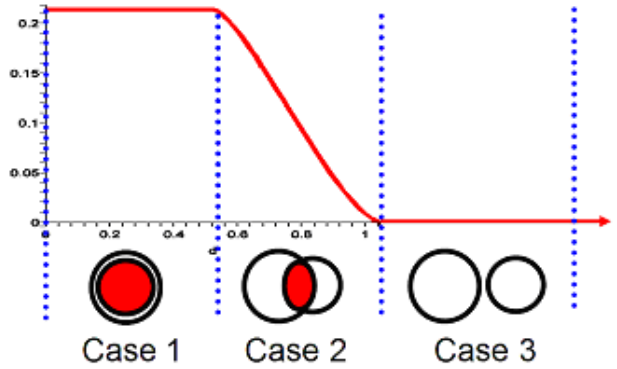
Ambient Aperture Lighting



circular caps of aperture and area light



Lambertian cosine term is evaluated at center of intersection



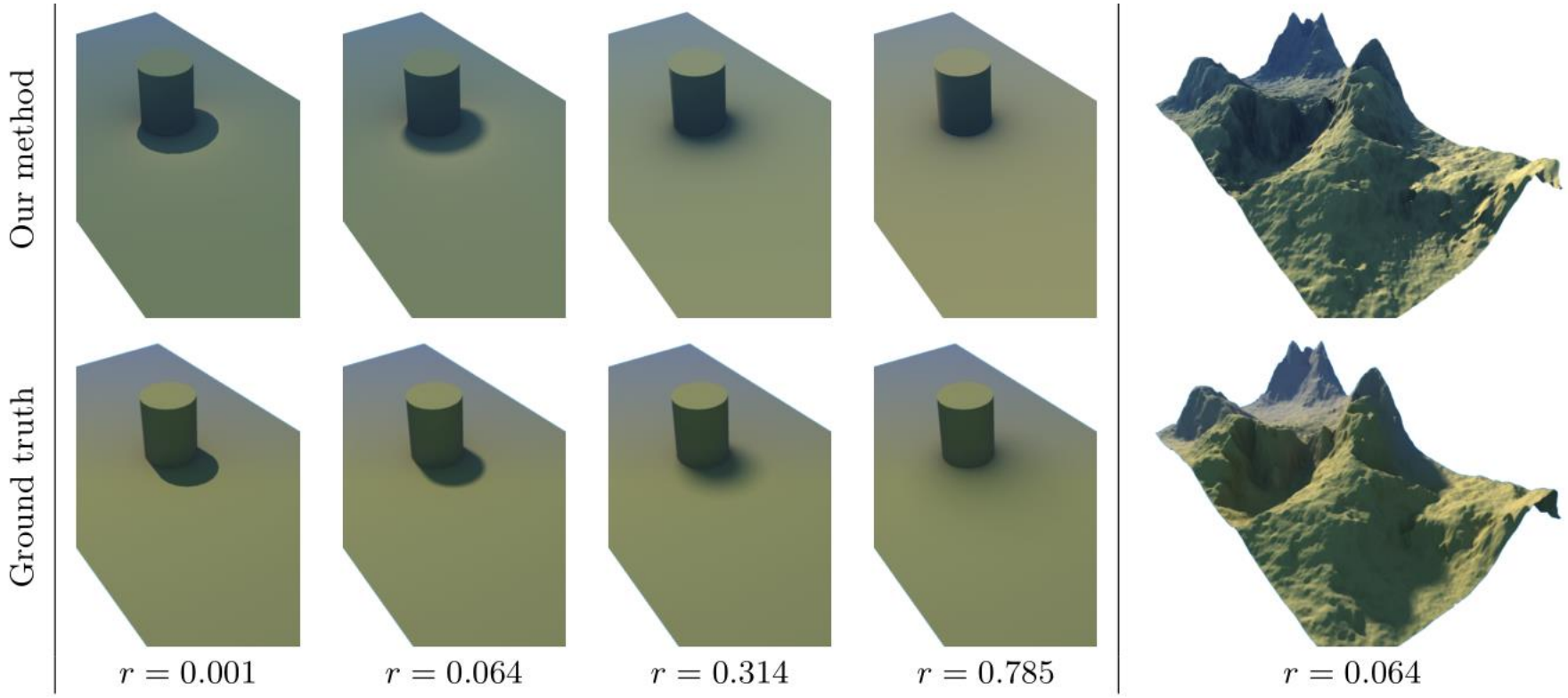
intersection area plotted over angular distance of circle centers

Lighting in Fragment Shader

- area light is assumed to be circular too
- intersect aperture and light circles
  1. one is included in other → result is smaller circle
  2. approximate intersection of circles with smoothstep function
  3. no intersection (full shadow)
- influence of cosine term in diffuse illumination is approximated to be constant and equal to cosine term at intersection center

Ambient Aperture Lighting

- Point light sources are individually clipped at aperture
- Ambient illumination can be computed with an area light source that covers the whole hemisphere



Terrain

REFERENCES

References

- Lindstrom, P., & Pascucci, V. (2001, October). Visualization of Large terrains made easy. In *Proceedings of the Conference on Visualization'01* (pp. 363-371). IEEE Computer Society.
- Losasso, F., & Hoppe, H. (2004, August). Geometry clipmaps: terrain rendering using nested regular grids. In *ACM Transactions on Graphics (TOG)* (Vol. 23, No. 3, pp. 769-776). ACM.
- A. Asirvatham, H. Hoppe, Terrain rendering using GPU-based geometry clipmaps. GPU Gems 2, 2005
- S. Lefebvre, H. Hoppe, Parallel controllable texture synthesis. SIGGRAPH 2005
  Parallel texture synthesis algorithm
- Blow, J. (2000, March). Terrain rendering at high levels of detail. In *Proceedings of the 2000 Game Developers Conference* (Vol. 3). sn.
- Livny, Y., Kogan, Z., & el-Sana, J. (2009). SeamLess patches for GPU-based terrain rendering. *The Visual Computer*, *25*(3), 197-208.
- Pajarola, R., & Gobbetti, E. (2007). Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, *23*(8), 583-605.
- Goswami, P., Makhinya, M., Bösch, J., & Pajarola, R. (2010, May). Scalable Parallel Out-of-core Terrain Rendering. In EGPGV (pp. 63-71).
- Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., & Scopigno, R. (2003, September). BDAM—batched dynamic adaptive meshes for high performance terrain visualization. In Computer Graphics Forum (Vol. 22, No. 3, pp. 505-514). Oxford, UK: blackwell Publishing, Inc.
- Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., & Scopigno, R. (2003, October). planet-sized batched dynamic adaptive meshes (P-BDAM). In Proceedings of the 14th IEEE Visualization 2003 (VIS'03) (p. 20). IEEE Computer Society.

References

- Lario, R., Pajarola, R., & Tirado, F. (2003, September). Hyperblock-quadtin: Hyper-block quadtree based triangulated irregular networks. In *IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2003)* (pp. 733-738).
- Pajarola, R., Antonijuan, M., & Lario, R. (2002, October). Quadtin: Quadtree based triangulated irregular networks. In *Proceedings of the Conference on Visualization'02* (pp. 395-402). IEEE Computer Society.
- Goswami, P., Makhinya, M., Bösch, J., & Pajarola, R. (2010, May). Scalable Parallel Out-of-core Terrain Rendering. In *EGPGV* (pp. 63-71).
- Pajarola, R., & Gobbetti, E. (2007). Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, *23*(8), 583-605.
- Duchaineau, M., Wolinsky, M., Sigeti, D. E., Miller, M. C., aldrich, C., & Mineev-Weinstein, M. B. (1997, October). ROAMing terrain: real-time optimally adapting meshes. In *Proceedings. Visualization'97 (Cat. No. 97CB36155)* (pp. 81-88). IEEE.
- Bao, X., Pajarola, R., & Shafae, M. (2004, December). SMART: An Efficient Technique for Massive Terrain Visualization from Out-of-core. In *VMV* (pp. 413-420).
- Dick, C., Krüger, J. H., & Westermann, R. (2009, March). GPU Ray-Casting for Scalable Terrain Rendering. In *Eurographics (Areas Papers)* (pp. 43-50).
- Dick, C., Schneider, J., & Westermann, R. (2009, March). Efficient geometry compression for GPU-based decoding in realtime terrain rendering. In *Computer Graphics Forum* (Vol. 28, No. 1, pp. 67-83). Oxford, UK: blackwell Publishing ltd.