Illumination and Geometry Techniques

Karljohan Lundin Palmerius

Objectives

Complex geometries

Huge areas

Translucency

Really nice graphics!

Shadows

Graceful degradation

Acceleration

Optimization

Straightforward rendering

Real-time graphics!

Simplicity

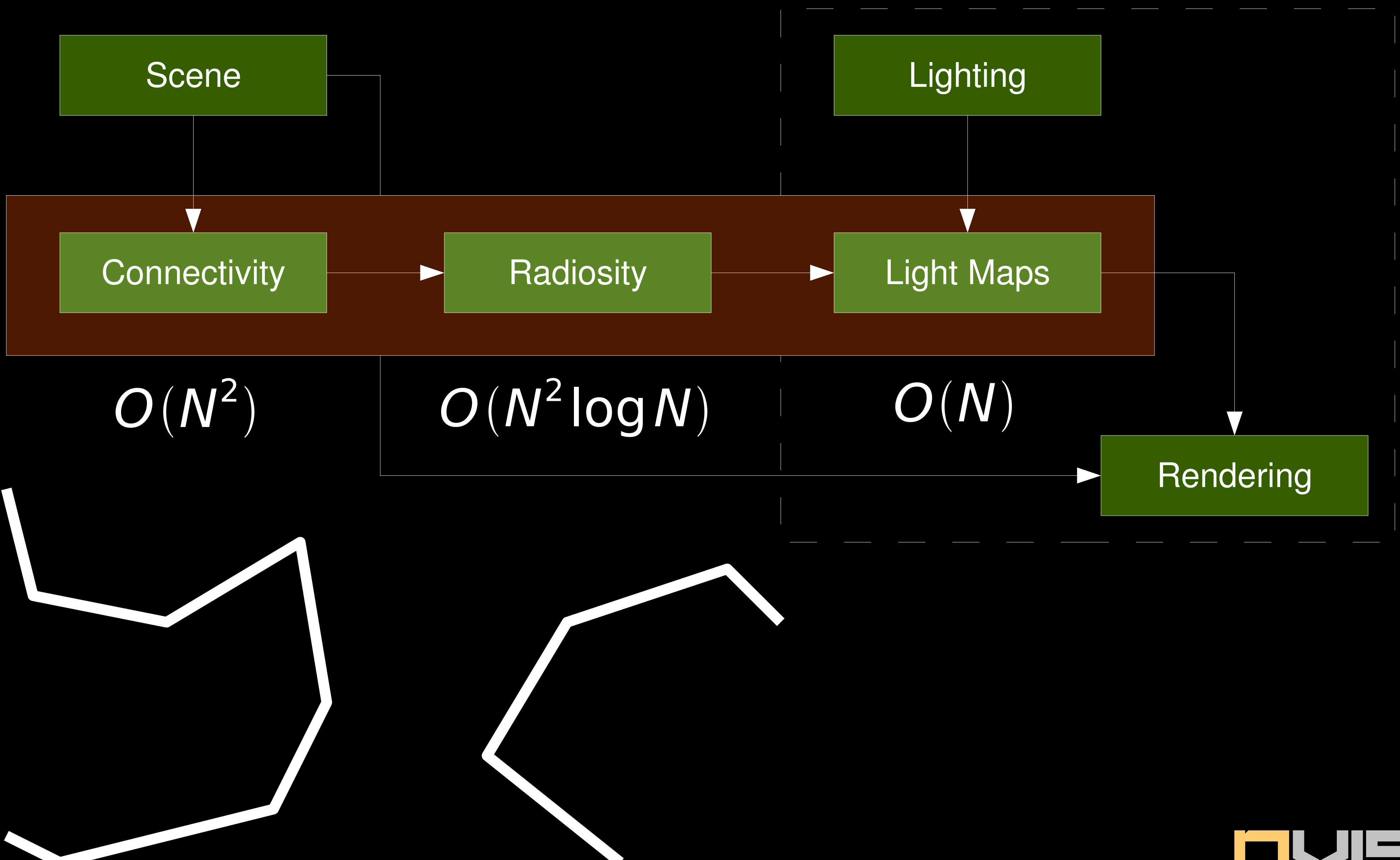
Automatic degradation

Parallelism

Shadowing Algorithms

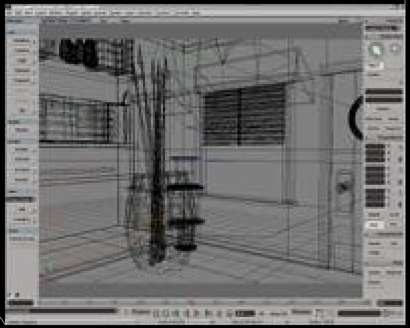
- Global illumination (radiosity)
  - really, really slow
- Raytracing
  - very slow
- Hardware accelerated
  - Shadow Volume, Shadow Mapping
    Shadow Texture, Etc...

Global Illumination



Global Illumination

- Hierarchical radiosity
  - group patches far away
  - group patches in early iterations
- (Quasi) Monte-Carlo radiosity
- Trans-illumination planes
  - intermediate transspatial radiosity map
  - can be hardware accelerated



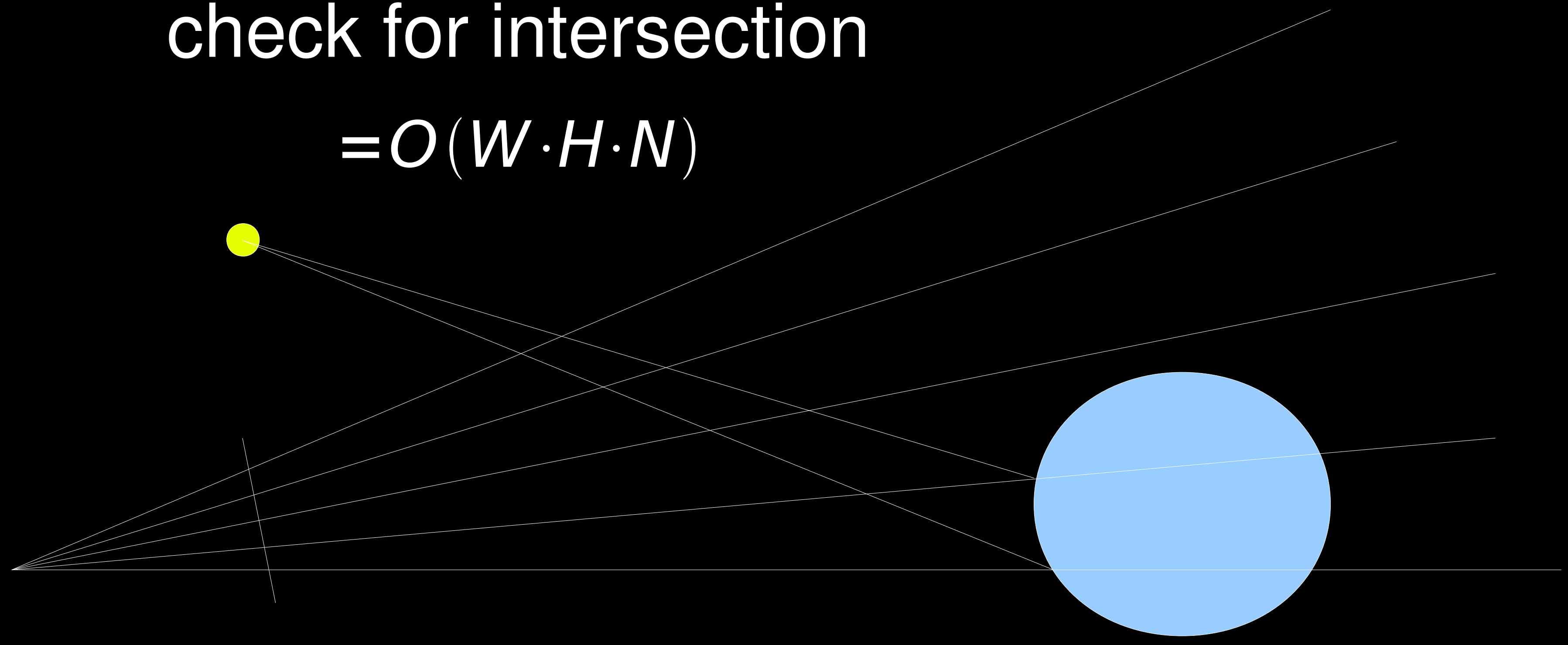
Global Illumination

- Pre-calculate states
  - Turn on/off individual lights
  - Ignore dynamic objects
- Use global illumination in RT rendering
  - Apply pre-calculated light as textures
  - Turn off OpenGL lighting

Raytracing

- for each buffer row
  - for each row pixel
    - for each object
      - check for intersection

$$= O(W \cdot H \cdot N)$$



Raytracing

- Speed-up
  - image-space (trivial) parallelization
  - collision search speed-up
  - frame-to-frame coherency
- Close to interactive with current hardware

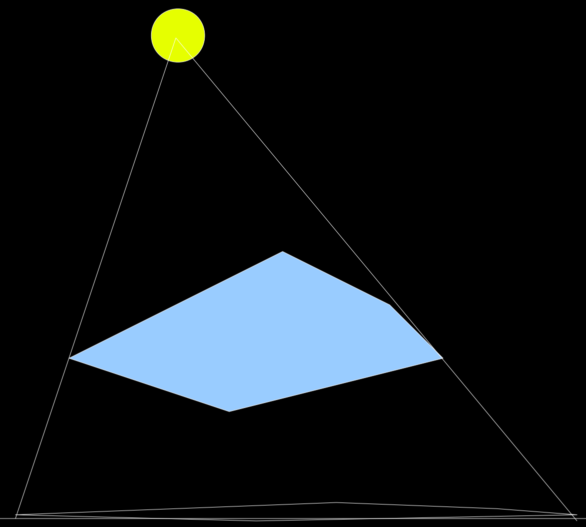
Projected Planar Shadows

- Only planar surfaces

$$\vec{x} \cdot \vec{n} = d$$

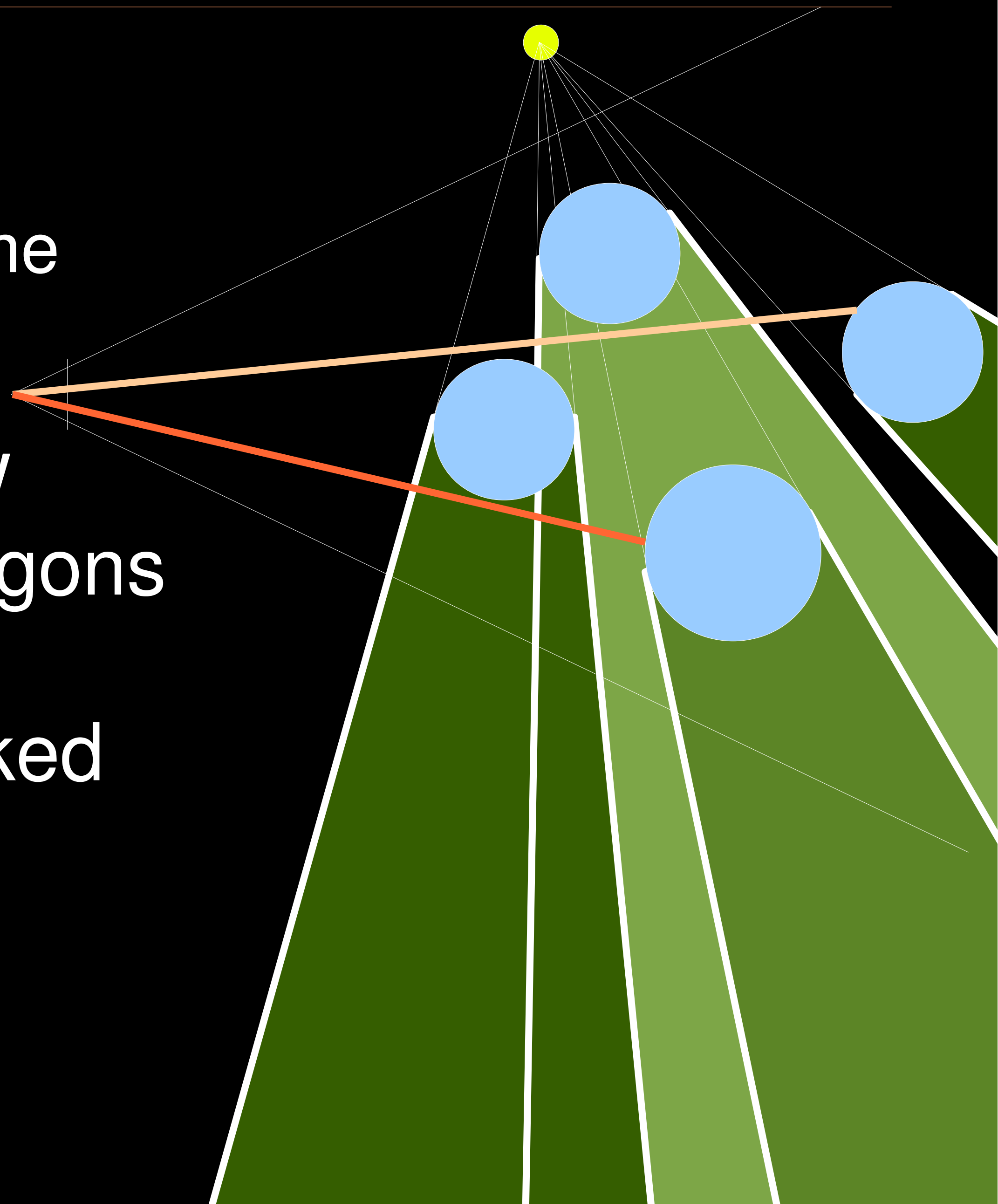
- Project object and colour it black

$$\vec{p}\,' = \vec{x}_L - \frac{d + \vec{n} \cdot \vec{x}_L}{\vec{n} \cdot (\vec{p} - \vec{x}_L)} (\vec{p} - \vec{x}_L)$$



Shadow Volumes

- For each light source
  - polygon shadow volume
- Multi-pass render
  1) render all in shadow
  2) render shadow polygons
     – to mask buffer
  3) render lighting masked



$= O(N)!$

Shadow Volumes

generate shadow polygons

render scene with ambient and emission lighting

for each front face shadow polygon

  if Z-buffer test passes
    then increment stencil buffer value

for each shadow polygon back face

  if Z-buffer test passes
    then decrement stencil buffer value

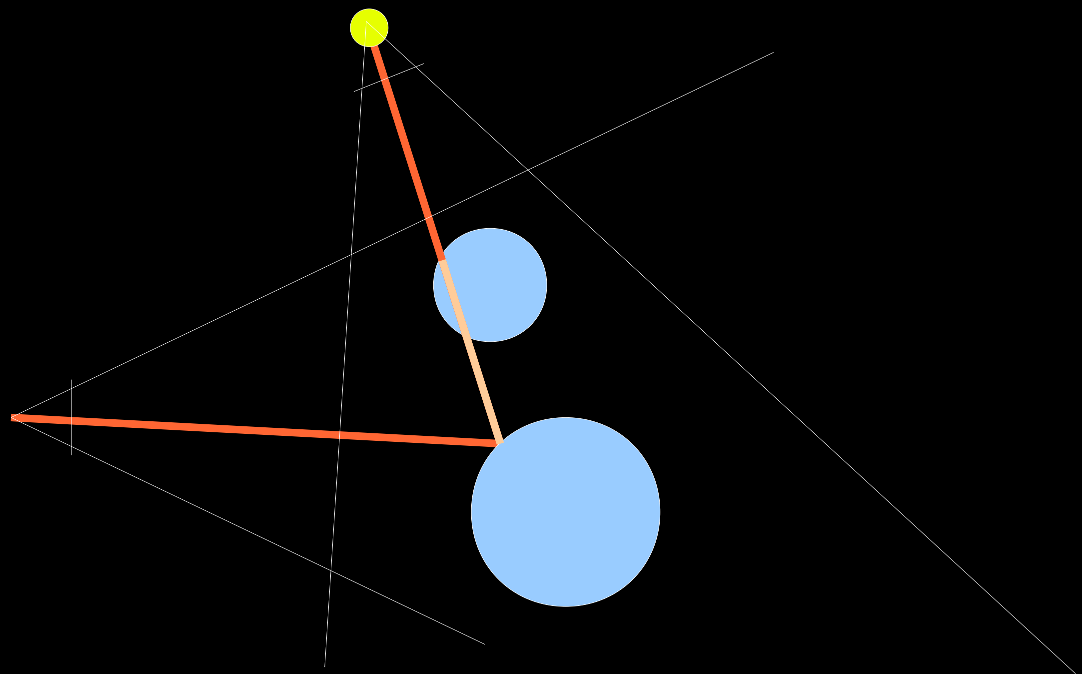
render lit scene with stencil buffer mask

Shadow Volumes

- Shadow volume estimation
  - computationally expensive
  - dependent on scene complexity

Shadow Mapping

- Completely image-space algorithm! $= O(N)$
  1) Render scene from light source -> depth map
  2) Render scene from viewpoint (use map)
  - check each fragment against depth buffer from light



Shadow Mapping

for each light

render scene to light's depth buffer

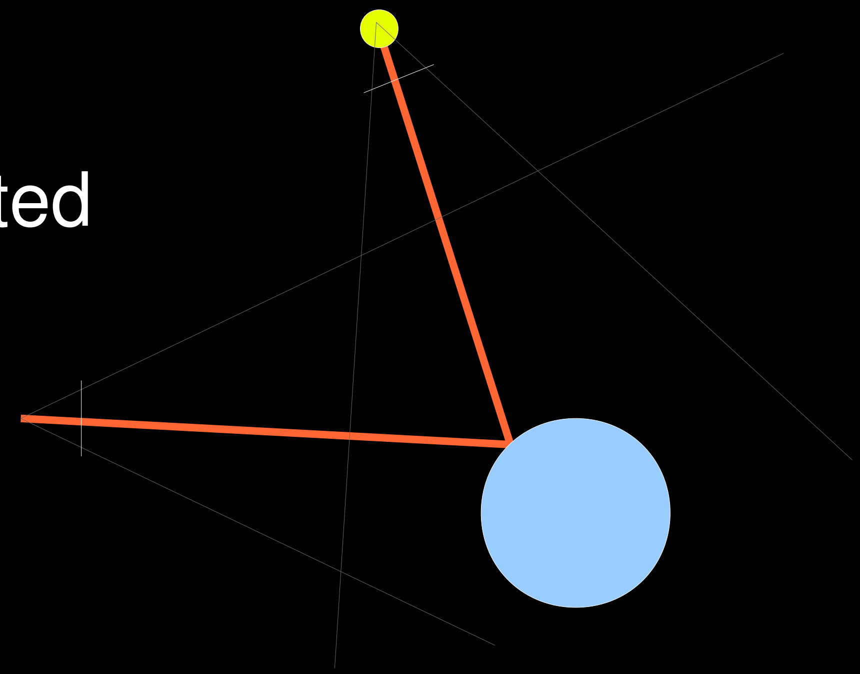
for each fragment

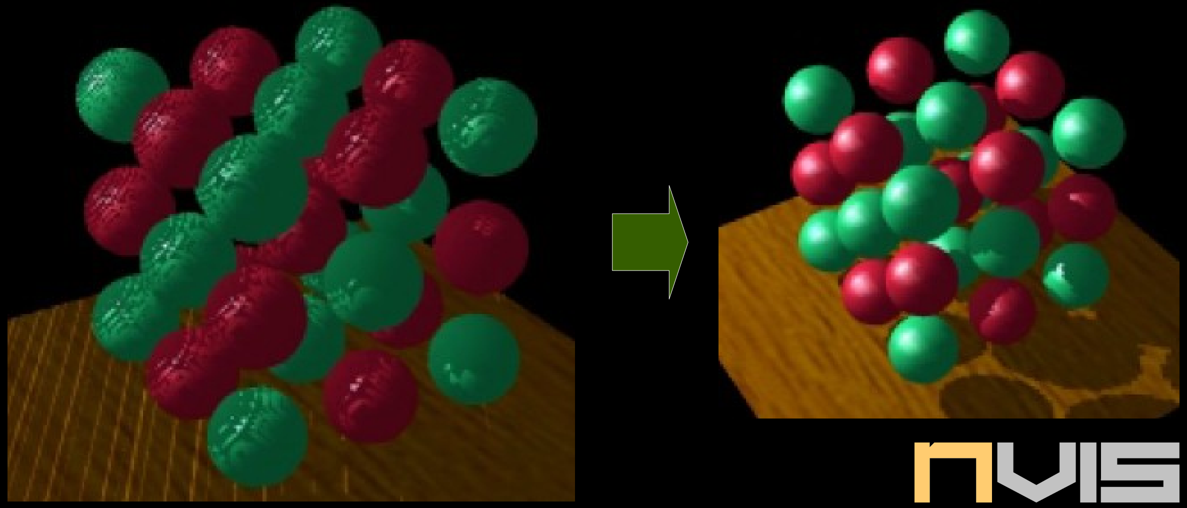
transform fragment into each light's buffer space

if fragment depth < light buffer depth
then render fragment lit by light
else render fragment shadowed

Shadow Mapping

- Precision limit artifacts
  - depth buffer precision is limited
    => self shadowing
  - solution
    - substract bias from Z-value
    - glPolygonOffset



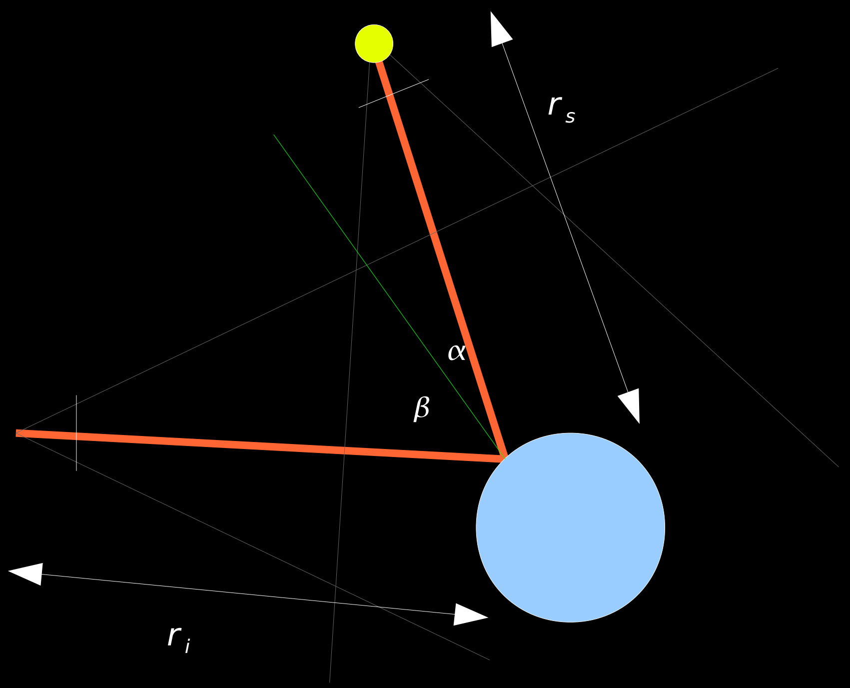


Shadow Mapping

- Resolution mismatch artefacts
  - depth fragment size in image space

$$d = d_s \frac{r_s}{r_i} \frac{\cos\beta}{\cos\alpha}$$

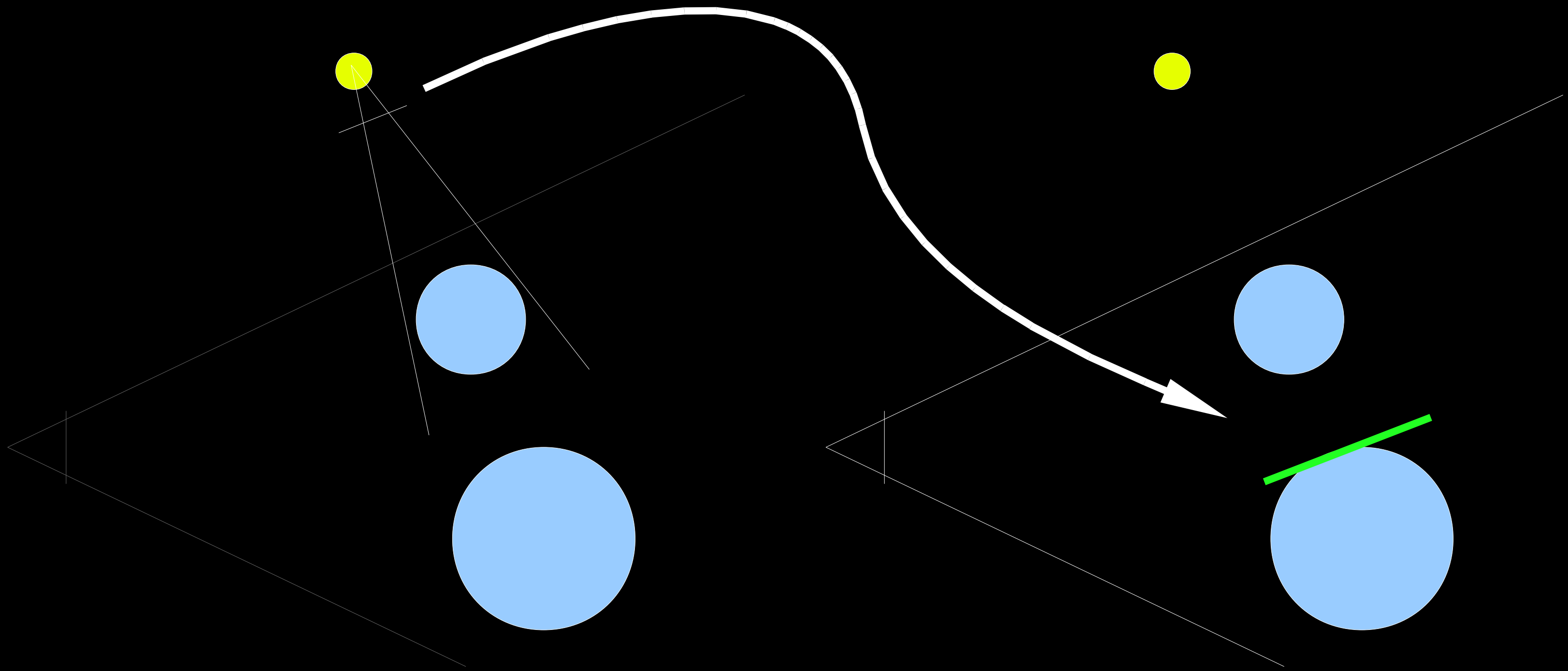
  - solutions
    - shadow bluring
    - percentage closer filtering
    - perspective shadow maps



Shadow Mapping

Shadow Texture

- Hack!
  - texture shadow onto object



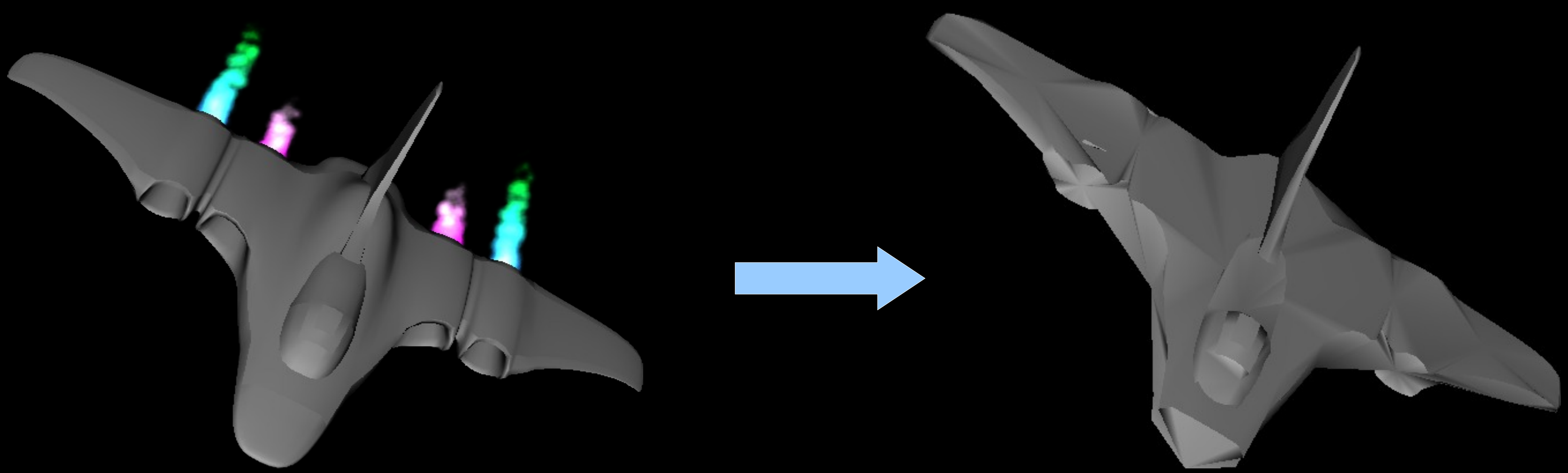
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

Cheating

- Object impact
  - speed, contrast, projected size
  - use bounding volume projected size
  - analyze pre-rendered image of object
- Simplifications
  - polygonal resolution
  - texturing
  - shaders

Object Resolution

- Level-of-detail (LoD)
  - use as low resolution as possible
  - select resolution at render-time



Textures

- Replace geometrical details with texture
  - clothes, faces and ground

Textures

- Image-based rendering
  - Use photographs of things
    - Lighting
    - Material and colours
    - Textures (microstructure)

Fragment Shaders

- Hardware accelerated details
  - each fragment individually estimated
  - real-time updated texturing
  - bumpiness, details, etc.
  - relief texturing

Sprites and Impostors

- Sprite
  - often billboard – polygon facing the camera
  - image/photograph/video instead of object
  - commonly plants, grass, people
- Impostors
  - replace geometry with image
    - pre-rendered / dynamically generated

Level-of-Details

- Adaptive cheating
  - select model resolution at run-time
  - usually distance specific
- Issues
  - generating detail levels
  - error estimation / level selection
  - pop / pop-up artifacts
  - resolution mis-match

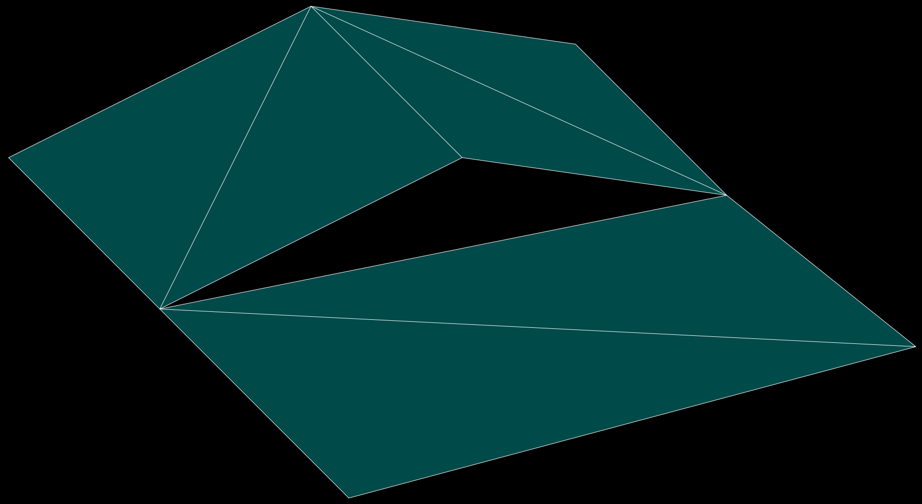
Pop Artifacts

- Too large of a change
  - object morphing – change in shape
  - creeping texture
  - popping attracts eye
- Remidy
  - gradual change is invisible
  - change at sub visible size (resolution)

Resolution Mis-match

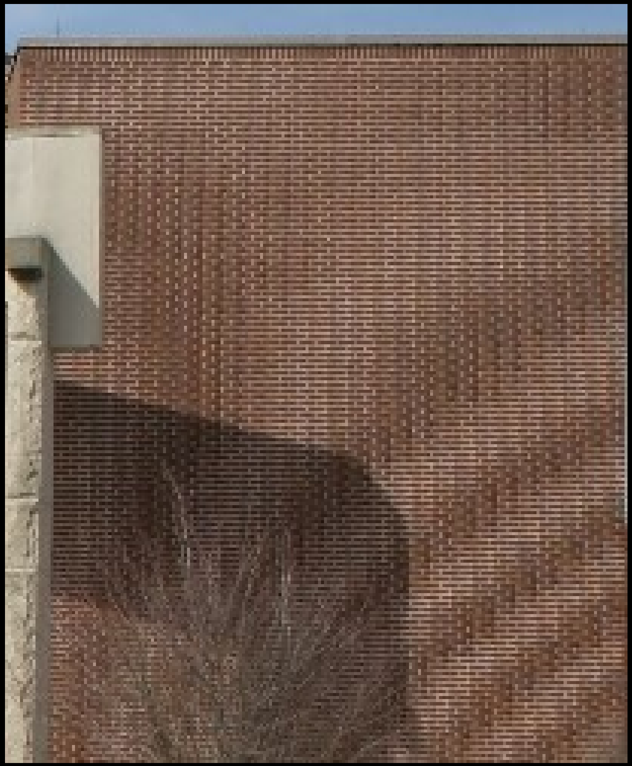
- Part of different resolution don't match
  - Texture resolution change

  - Polygon patches



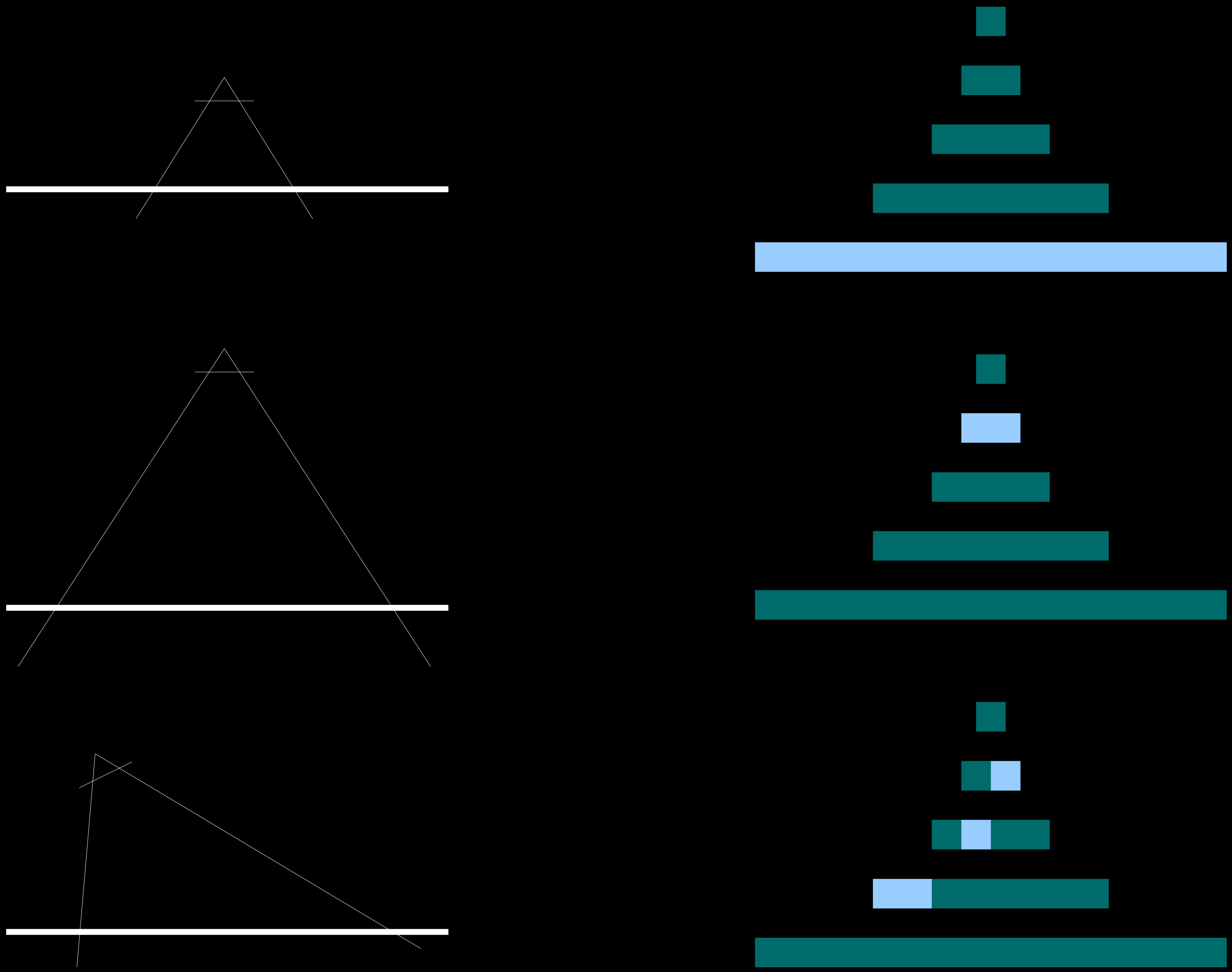
MIP-map

- Texture resolution issues
  - Aliasing at distance – moiré patterns
  - Unnecessarily high resolution
  - Too low resolution
- Solution
  - *Multum In Parvo* (MIP map)
  - Multiple texture resolutions
  - Hardware support in OpenGL



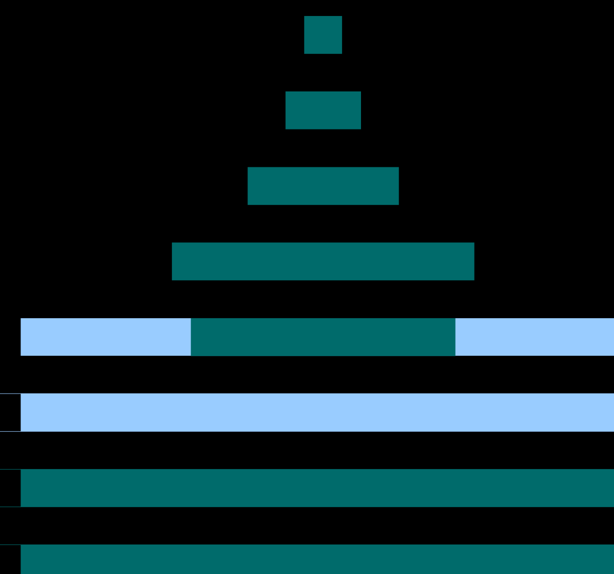
MIP-map

MIP-map



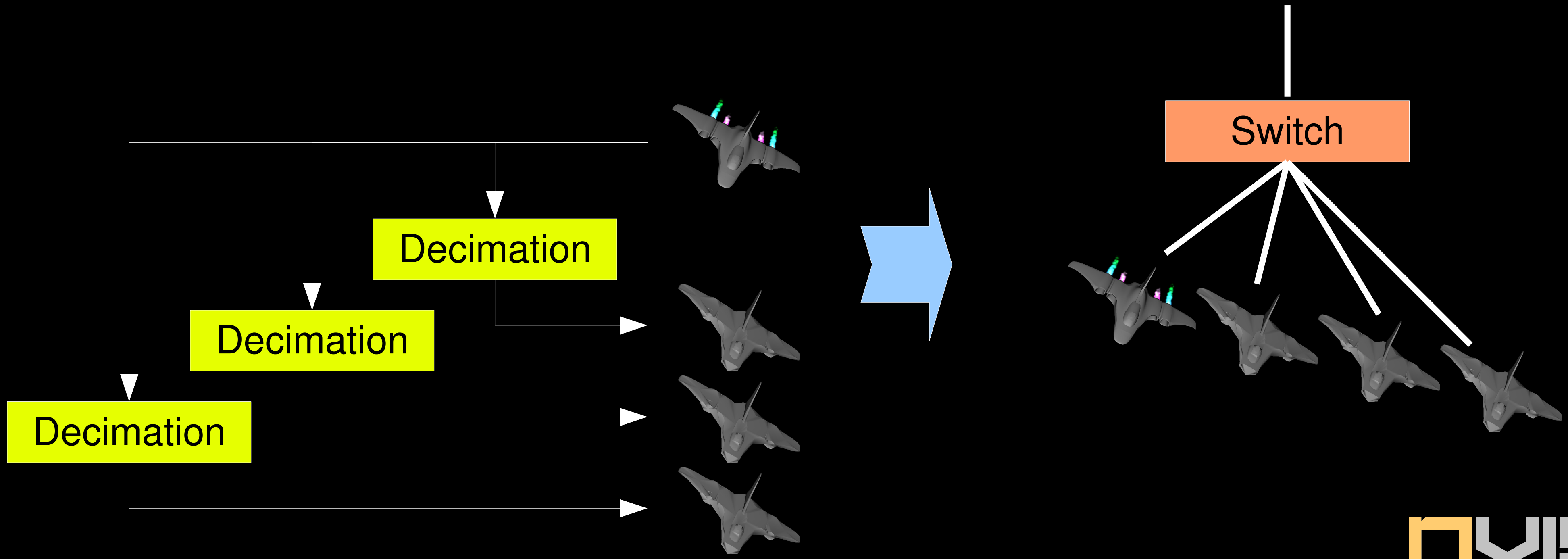
Clip-map

- Clipped MIP-map (SGI)
  - Full texture at low resolution
  - Only partial texture at high resolution



LoD Polygonal Objects

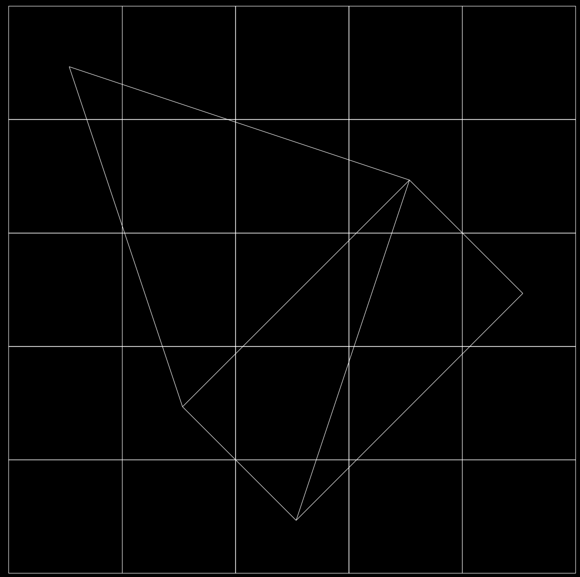
- Multiple resolution versions
  - Generate multiple resolutions
  - Select right resolution



Decimation

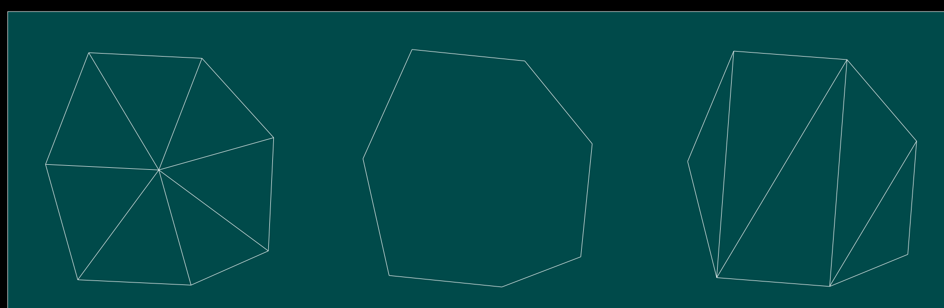
Decimation

- Vertex clustering
  - Cluster vertices using regular grid
    - Replace cluster with single vertex
    - Update faces accordingly
  - Poor result
    - low quality
    - no triangle count specification
    - no error measure

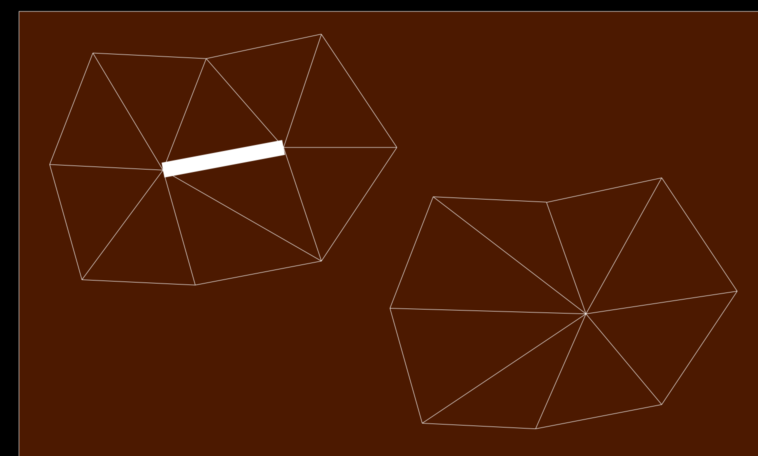


Decimation

- Remove vertices to simplify model
  - Vertex removal, edge contraction



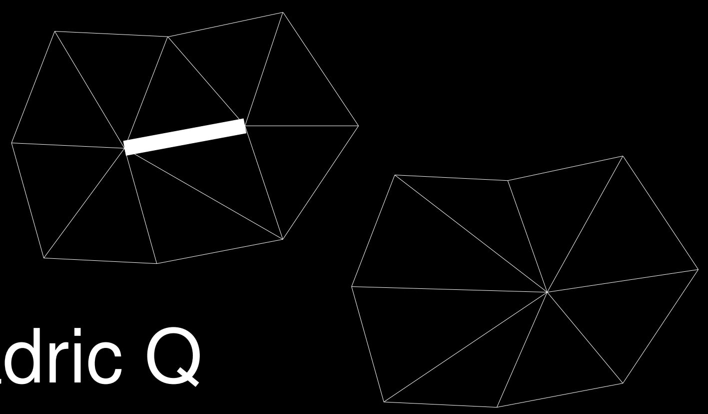
$7 \rightarrow 5 \;\; \sim 28\%$



$10 \rightarrow 8 \;\; \sim 20\%$

  - Remove vertex/edge until
    - exceeding error threshold
    - reaching triangle count

Quadric Error Metric



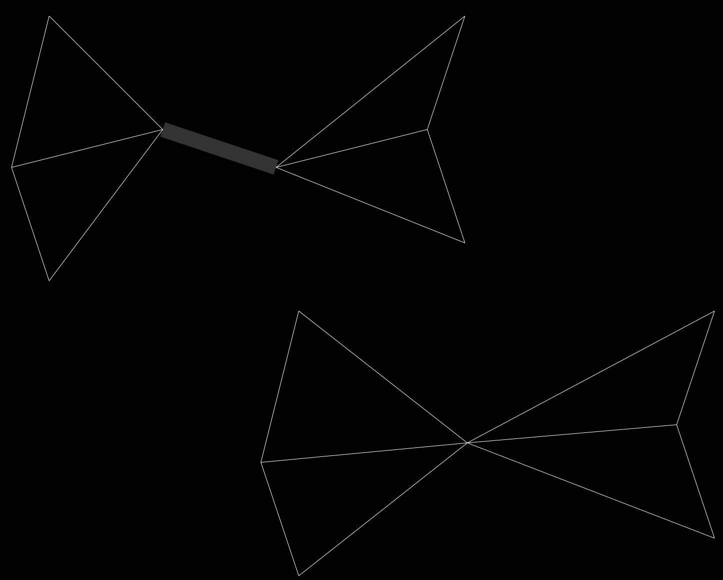
- "Quadric Decimation"
  - Edge / non-edge contraction
  - Associate vertex with error quadric Q

$$\varepsilon(\vec{v}) = \vec{v}^T Q \vec{v}$$

$$Q_{1,2} = Q_1 + Q_2$$

$$\vec{v}\,' = Q_{1,2}^{-1}(0,0,0,1)^T$$

$$cost = \vec{v}\,'^T (Q_1 + Q_2)\vec{v}\,'$$



  - Use heap with vertices and costs

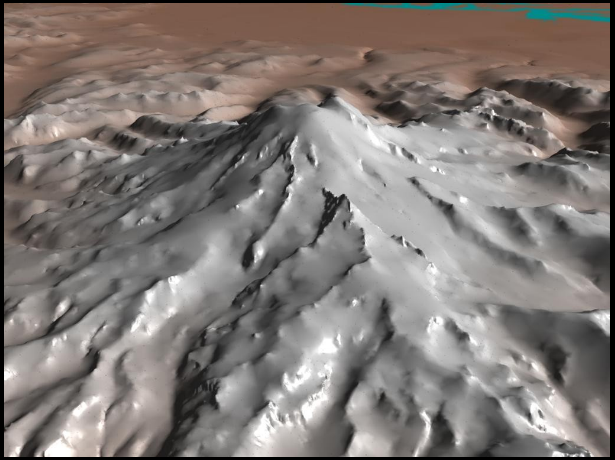
Quadric Error Metric

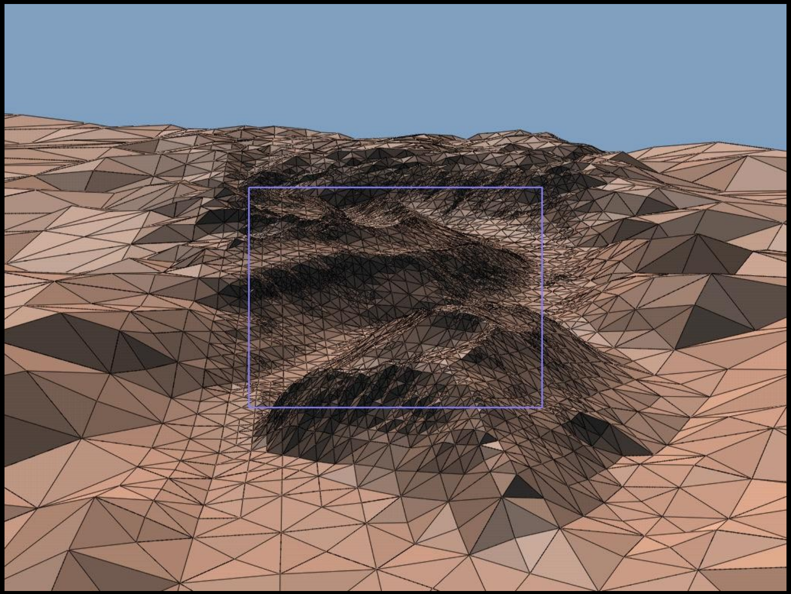
- Initial Quadric Error
  - approximate plane distance square error

$$
\begin{aligned}
\varepsilon(\vec{v}) &= \sum_{\vec{p} \in planes(\vec{v})} \left(\vec{p}^T \vec{v}\right)^2 \\
&= \sum_{\vec{p} \in planes(\vec{v})} \left(\vec{v}^T \vec{p}\right)\left(\vec{p}^T \vec{v}\right) \\
&= \sum_{\vec{p} \in planes(\vec{v})} \vec{v}^T \left(\vec{p}\,\vec{p}^T\right) \vec{v} \\
&= \vec{v}^T \left(\sum_{\vec{p} \in planes(\vec{v})} \vec{p}\,\vec{p}^T \right) \vec{v}
\end{aligned}
\Rightarrow Q = \sum_{\vec{p} \in planes(\vec{v})} \vec{p}\,\vec{p}^T
$$

Adaptive Decimation

- What if different parts are at different distance?
  - Parts of same object requires different resolution
  - MIP-mapping for textures
  - Adaptive decimation for polygonal data
    - e.g. ground meshes

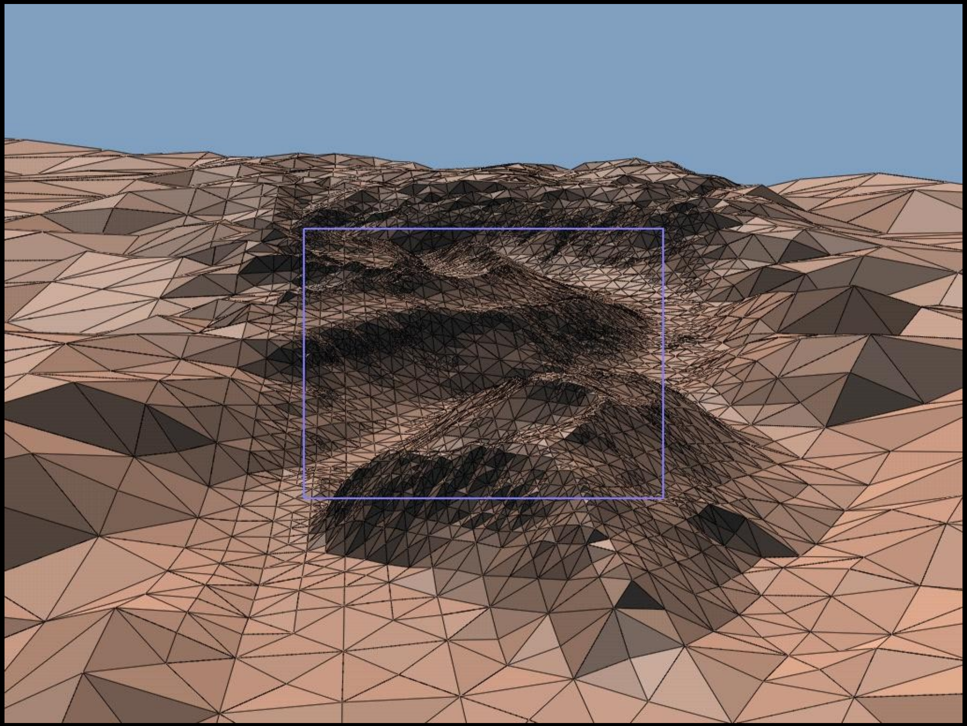




NVIS

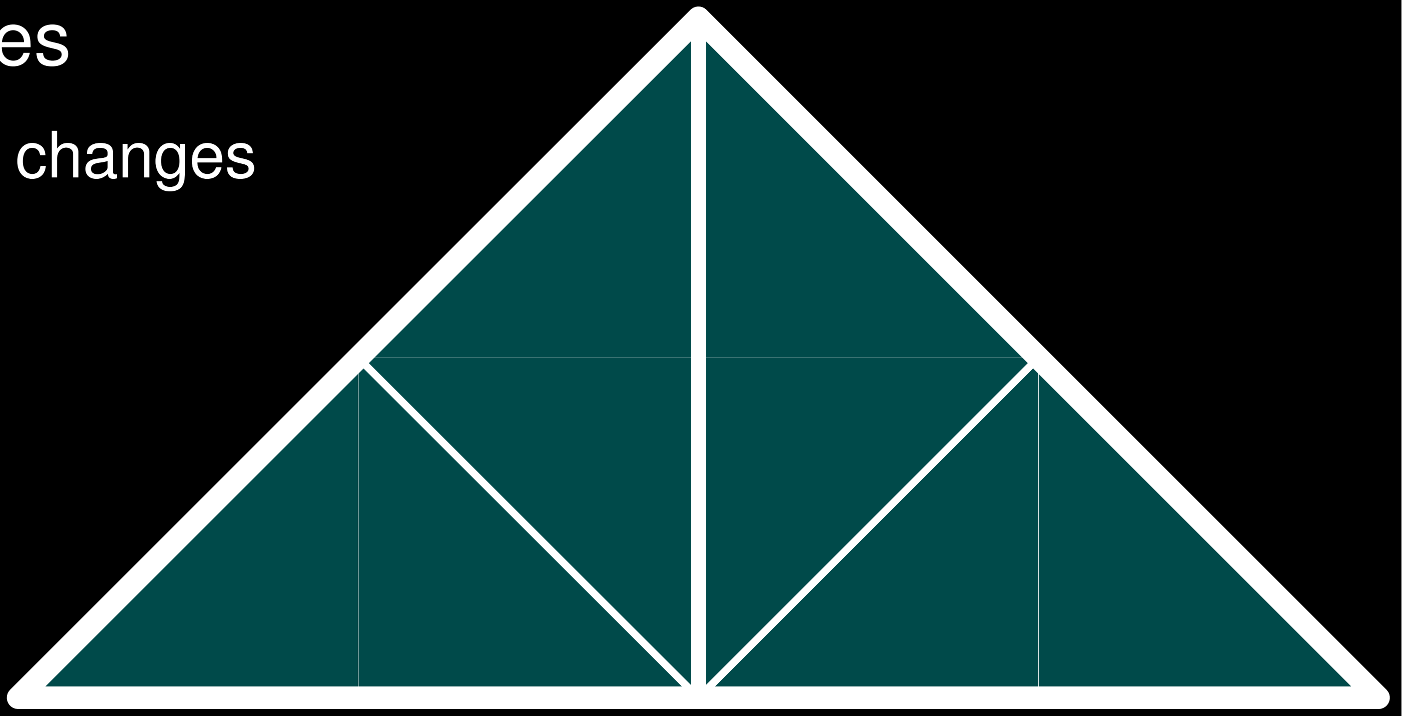
Adaptive Decimation

- Alternatives
  - Patches
  - Adaptive decimation
    - ROAM
    - SOAR
    - etc, etc.
- Criteria
  - distance, frustum culling, decimation error, line-of-sight



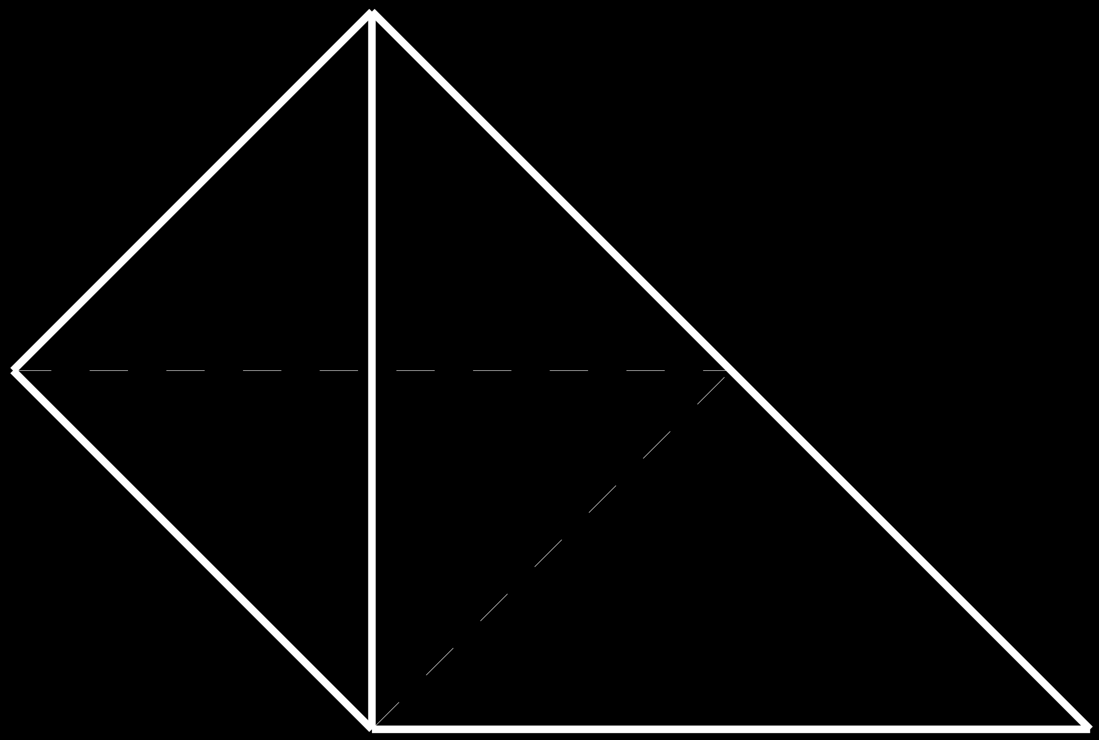
ROAM

- Real-time Optimally Adapting Meshes
  - Triangle bin-tree
  - Time-dependent priority
  - Priority queues
    - Incremental changes
    - Split
    - Merge

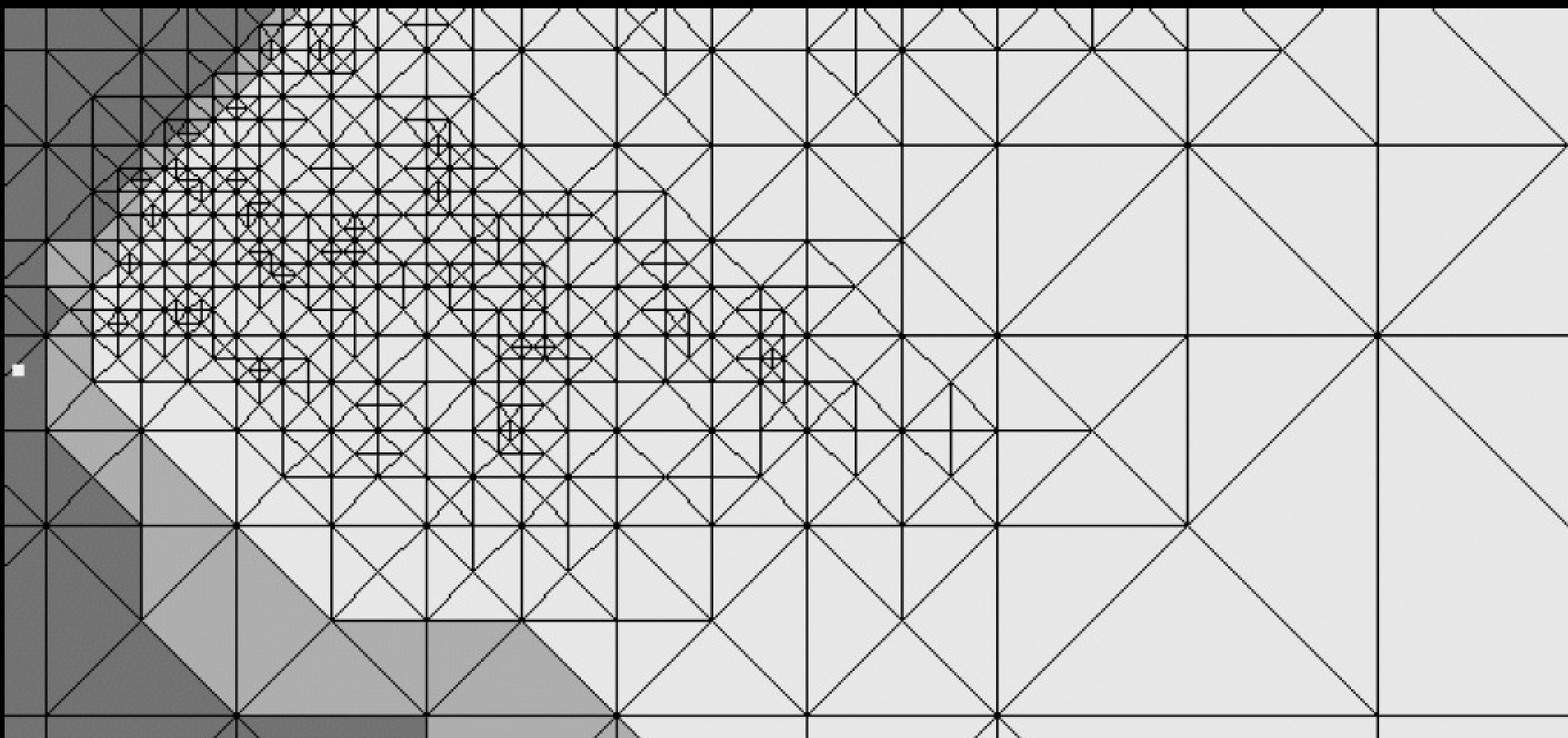


ROAM

- Forced split



ROAM

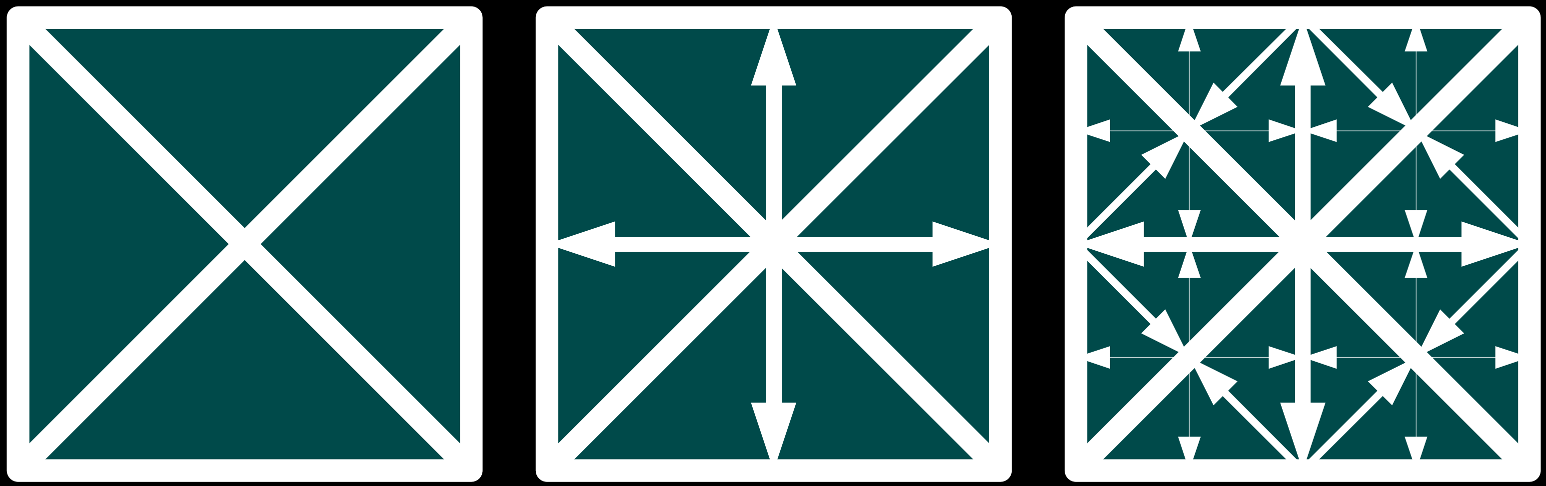


ROAM

- Error Metrics
  - Nested wedges – hierarchical error bound
  - Screen distortions – project wedge on screen
  - Check line-of-sight
    - intersection with wedge – modify triangle priority
  - View frustum, backface detail reduction, specular highlights, silhouette edges, atmospheric effects, object positioning

SOAR

- Stateless, One-pass Adaptive Refinement
  - Longest edge intersection, 4-k mesh, ...
  - Multiresolution mesh forms DAG of vertices
    - children belong to higher resolution level
    - nested error terms ensure active parent of active vertex



SOAR

- Pros
  - Simple to implement
  - No forced split
  - High memory coherency
- Cons
  - Does not use frame-to-frame coherency

Adjusting Degradation at Run-time

- Pre-determine degradation
  - time buffer
  - priority queue
    - depending on distance, size, velocity
  - buy quality for each object
- Dynamic degradation
  - adjust quality if frame-rate is high or low
  - hysteresis

The end