View Factor Calculation for Real-time Scene Generation

Michael L. Stokes, Ph.D.
Trideum Corporation
Huntsville, AL

ITEA 2011 Technology Review Conference
July 19-21, Anapolis, MD

Agenda

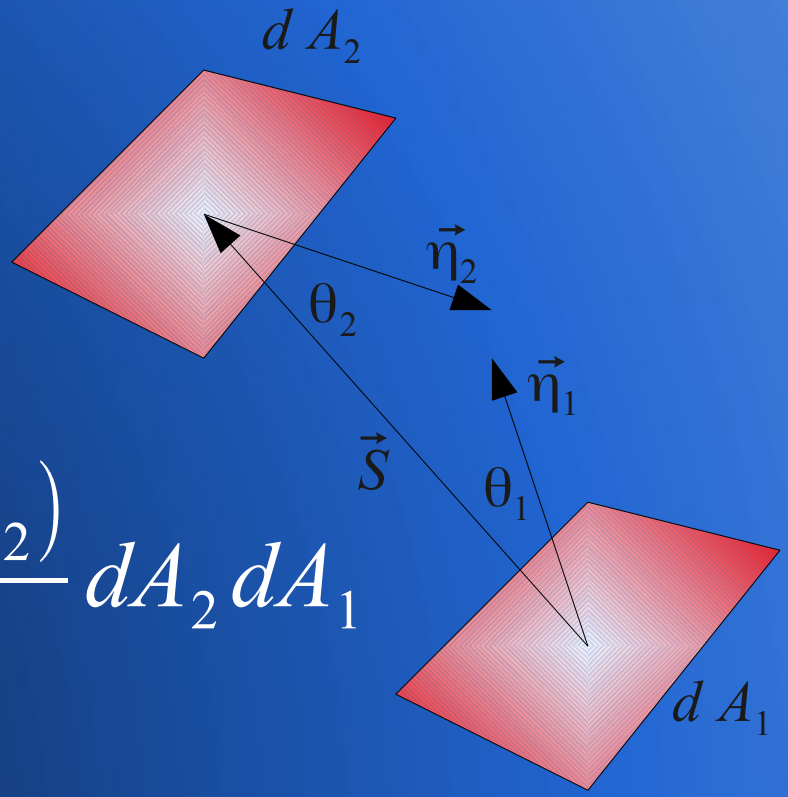
- Acknowledgment
- Review of View Factors
  - Obstructed
  - Unobstructed
- Problematic cases
- Object hierarchies
- Future

View Factors

- In radiative heat transfer, a view factor is the proportion of all the radiation that arrives at surface 1 from surface 2, $F_{1->2}$.
- View factors are also sometimes known as configuration factors, form factors or shape factors.

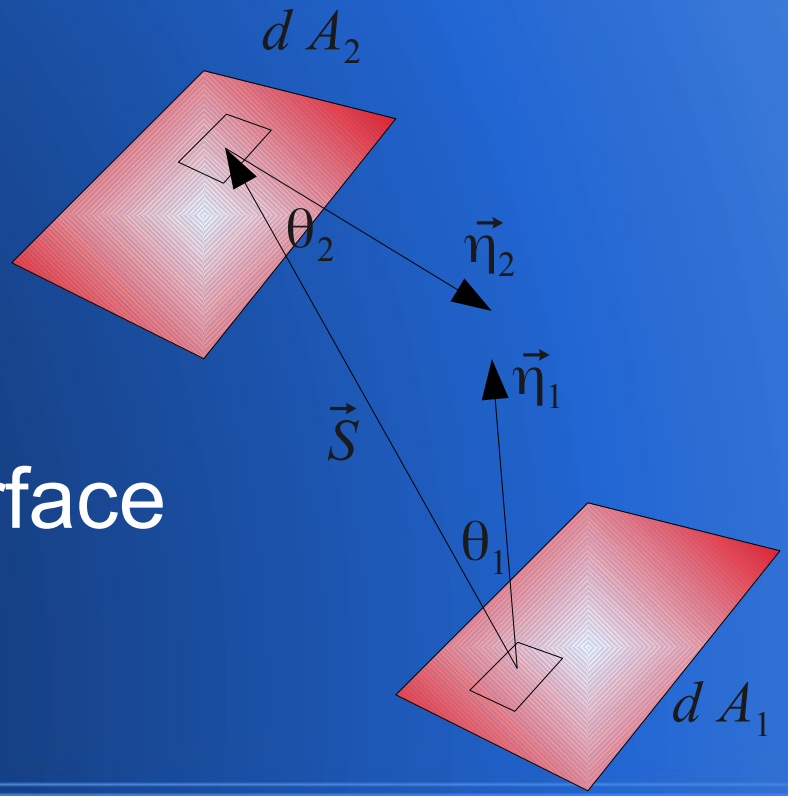
View Factors (Review)

$$F_{1\to 2} = \frac{1}{\pi A_1} \int_{A_1} \int_{A_2} \frac{\cos(\theta_1)\cos(\theta_2)}{S^2} dA_2 dA_1$$



View Factors

- This equation can be numerically integrated by subdividing the two surfaces into small differential areas
- Compute the VFs for each pair of differential areas from each surface
- Sum the VF's across the surface
- Numerical integration $O(N^2)$



View Factors (Review)

- Stokes' Theorem is used to reduce the double area integral to double line contours

$$F_{1\to 2} = \frac{1}{2\pi A_1} \int_{C_1} \int_{C_2} \ln(s)\, d\vec{v}_1\, d\vec{v}_2$$

View Factors (Review)

- If we restrict the line contours to straight edges, then, the previous expression reduces to

$$F_{1\to 2} \approx \frac{1}{4\pi A_1} \sum_{p=1}^{E_1} \sum_{q=1}^{E_2} \cos(\phi_{pq}) \ln(\vec{s} \circ \vec{s}) \Delta v_i \Delta v_j$$

where ϕ is the angle between the two edges

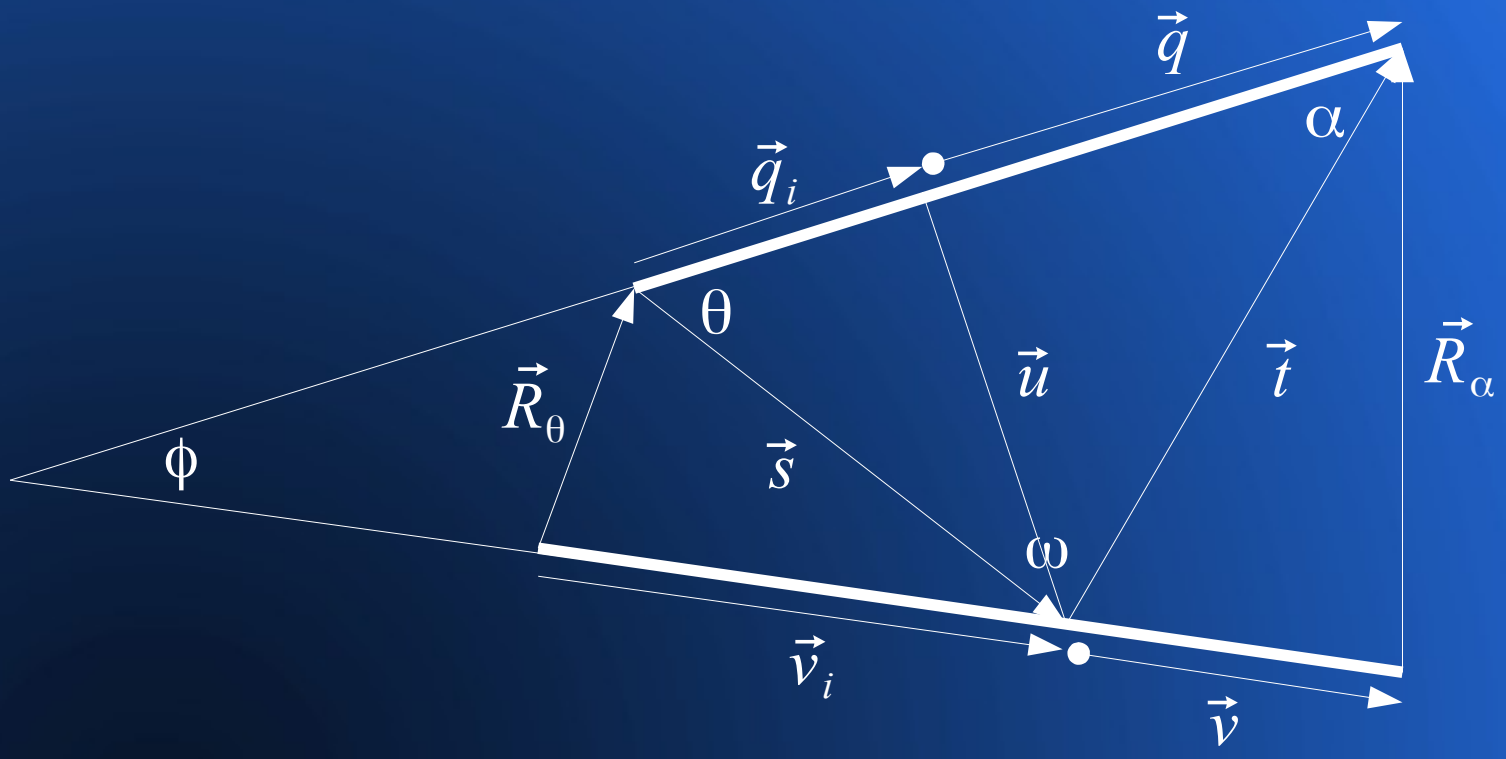
View Factors (Review)

- If we integrate (closed form) one of the two edges, then

$$F_{1\to 2}=\frac{1}{2\pi A_1}\sum_{p=1}^{E_1}\sum_{q=1}^{E_2}\cos(\phi_{pq})\int\left[t\cos(\alpha)\ln(t)+s\cos(\theta)\ln(s)+u\dot{\omega}-q\right]dv_1$$

- *Mitalis and Stephenson[1996]
- Algorithms of this type are O(16N)

View Factors (Review)

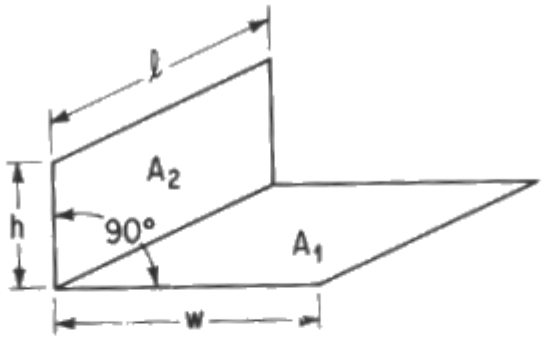


View Factor (Review)

- This form of equation usually provides the highest accuracy when numerically integrated using a Gaussian quadrature using a minimum number of function evaluations.
- Certain configurations can be integrated exactly for common configurations
  - Coincident edges - $F_{1,2} = \frac{L^2}{2}(2 \ln L - 3)$
  - Co linear edges - $F_{1,2} = f(L_1, L_2)$

Problematic Cases

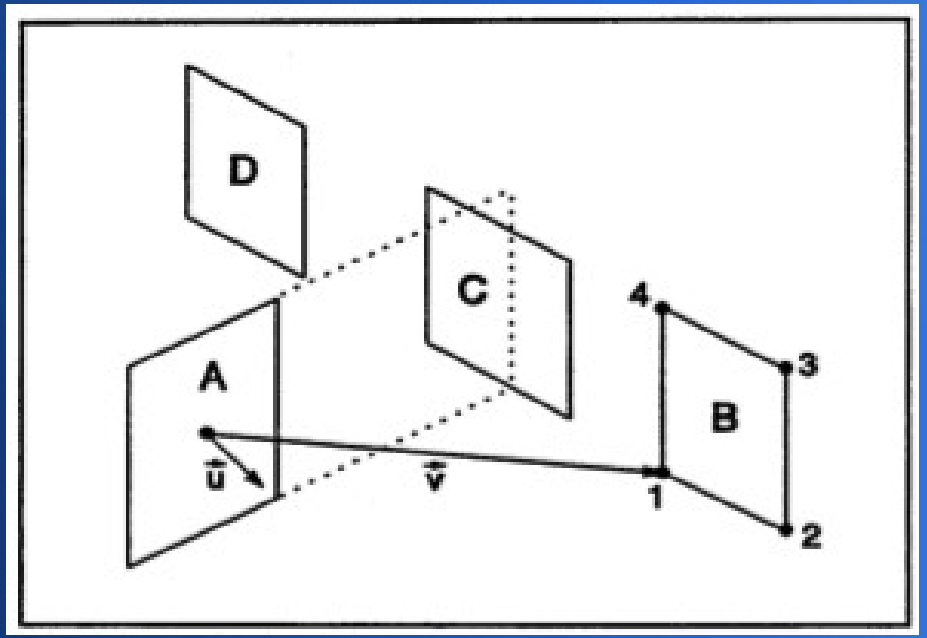
- Shared Boundary or coincident edges
- Solution: Substitute exact solution for shared boundary, numerical quadrature for the other edges



$$F_{1-2} = \frac{1}{W\pi}\left(W\tan^{-1}\frac{1}{W} + H\tan^{-1}\frac{1}{H} - \sqrt{H^2+W^2}\tan^{-1}\sqrt{\frac{1}{H^2+W^2}}\right.$$
$$\left. + \frac{1}{4}\ln\left\{\frac{\left(1+W^2\right)\left(1+H^2\right)}{1+W^2+H^2}\left[\frac{W^2\left(1+W^2+H^2\right)}{\left(1+W^2\right)\left(W^2+H^2\right)}\right]^{W^2}\left[\frac{H^2\left(1+H^2+W^2\right)}{\left(1+H^2\right)\left(H^2+W^2\right)}\right]^{H^2}\right\}\right)$$

Problematic Cases

- Self Obstruction
- Solution: Reconstruct C such that a new edge is co-planar to both A and C



Problematic Cases

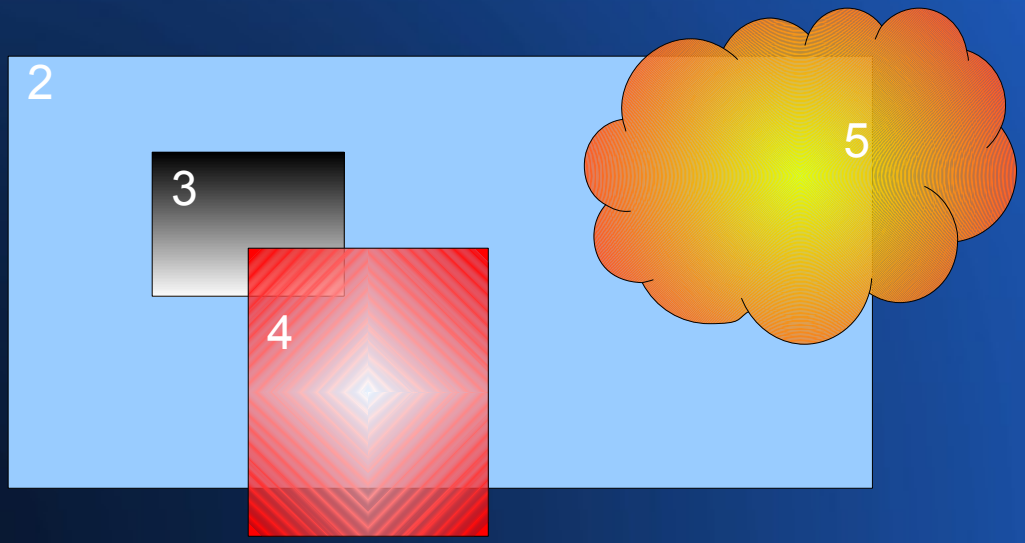
- Complex (crossing edges) or Concave polygons (at least 1 interior angle > 180 degrees)
- Solution: can be decomposed into smaller set of polygons(triangles)

Problematic Cases

- Surface polygons that intersect other polygons other than at polygonal boundaries
- Solution: Don't do it. Most if not facet generators can be optioned not to generate these.

Obstruction/Visibility

- All methods assumed that $F_{1->2}$ had no other polygons that obstructed A_2.



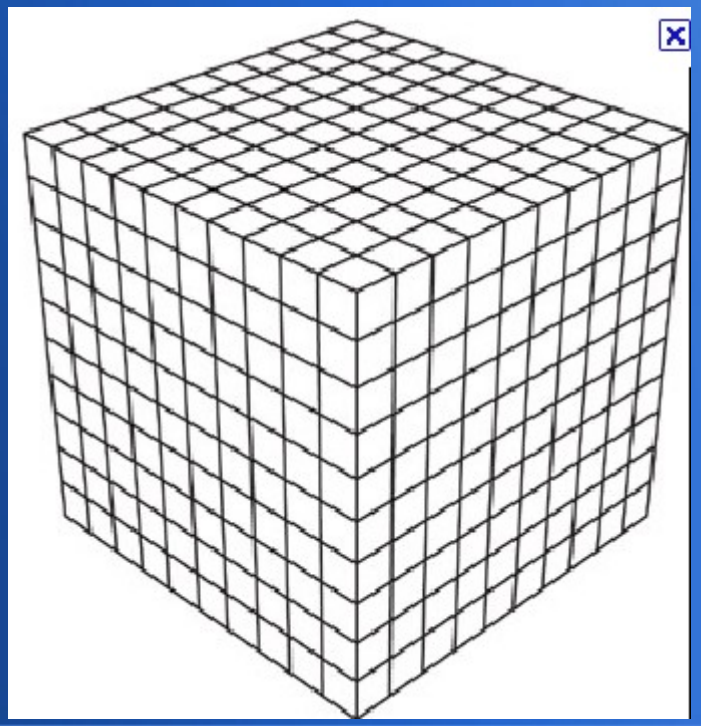
Obstruction/Visibility

Obstruction/Visibility

$$F_{1->2} = F^*_{1->2} - F_{1->3}$$

Obstruction/Visibility

- Overlay a uniform grid of volume elements (voxels) over all components in the scene
- Associate all polygons with a voxel (C++ STL component)
- Given a Ray, compute the set of voxels to visit
- Exhaustive search within voxel to find poly/ray intersection

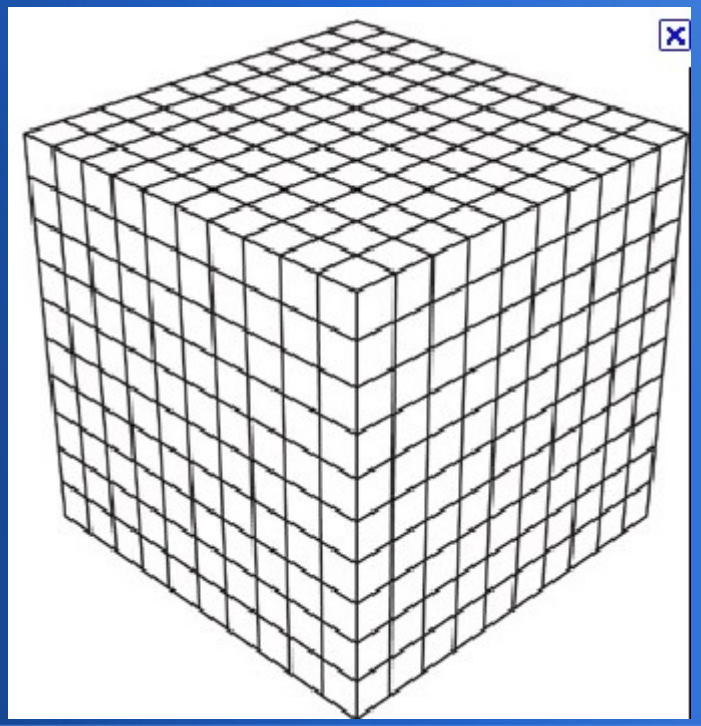


Obstruction/Visibility

- Ray is defined by

$$\vec{R} = \vec{O} + t\,\hat{D}$$

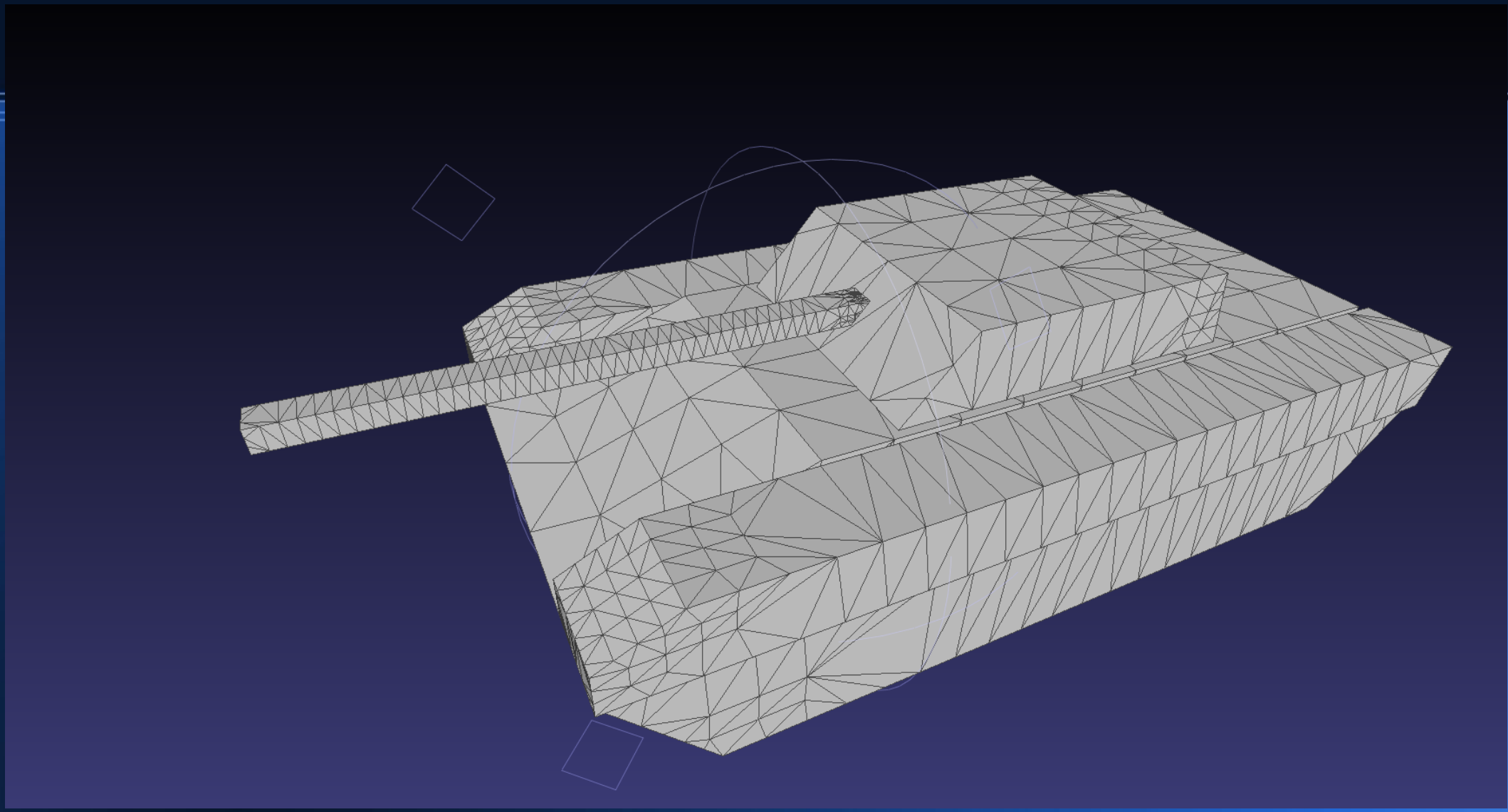
- Search returns an array of “t” values (ascending order) and associated polygons
- These polygons are then used in obstruction tests



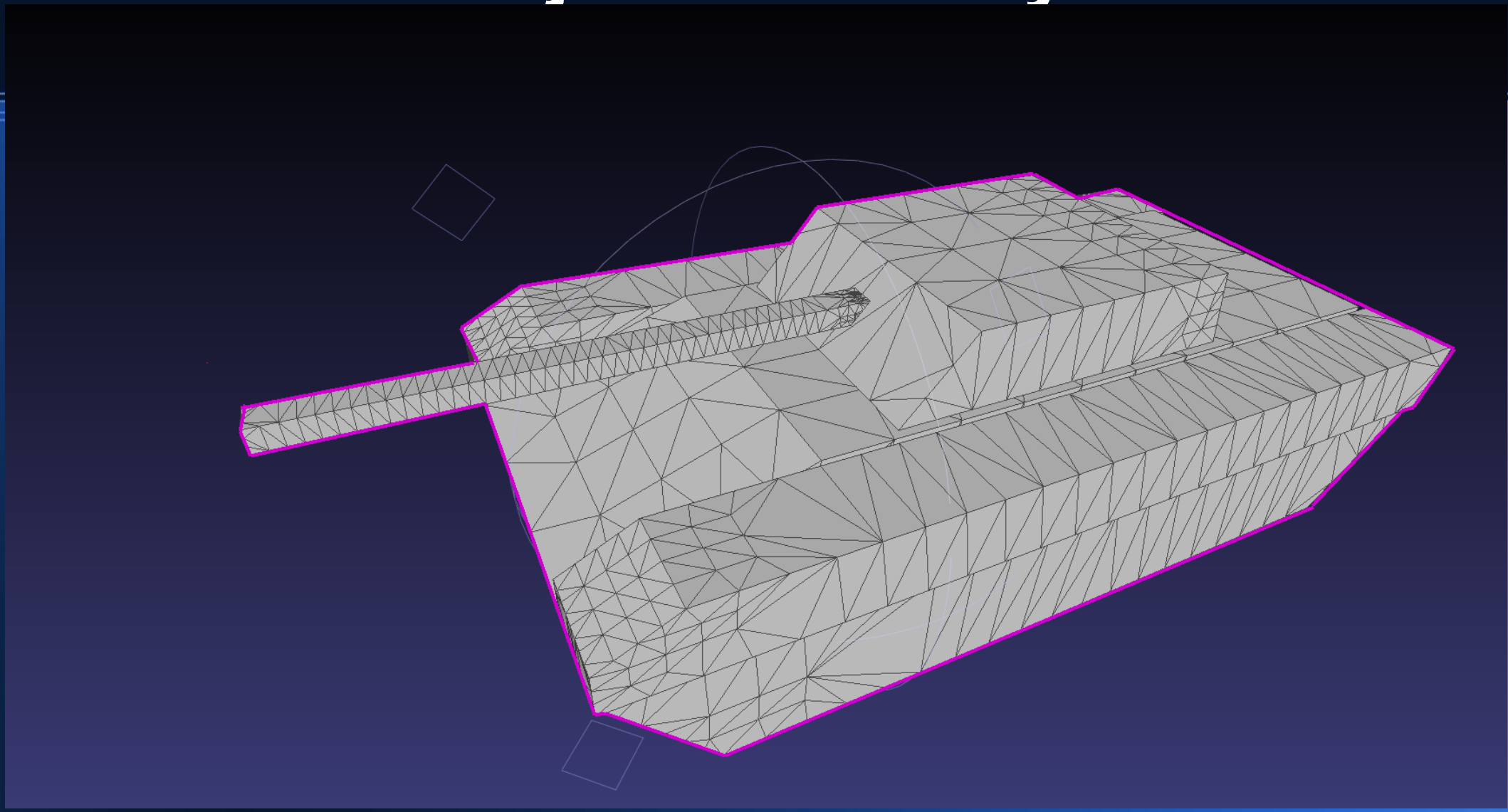
Object Hierarchy

- The Contour integral method allows a vast reduction in operations to compute view factors if large groups of polygons can associated with an object.
- A View Factor for the object can be computed as a single contour integration instead of a summation of the View Factors of the visible polygons.

Object Hierarchy



Object Hierarchy



Future

- Plan to implement algorithms as a “library” for C++
- Handle polygons for any number of edges
- Implement on multi-core and GPU architectures
- For primary use in the STAR S&T/T&E program