Hybrid Forward-Backward Reflection Rendering

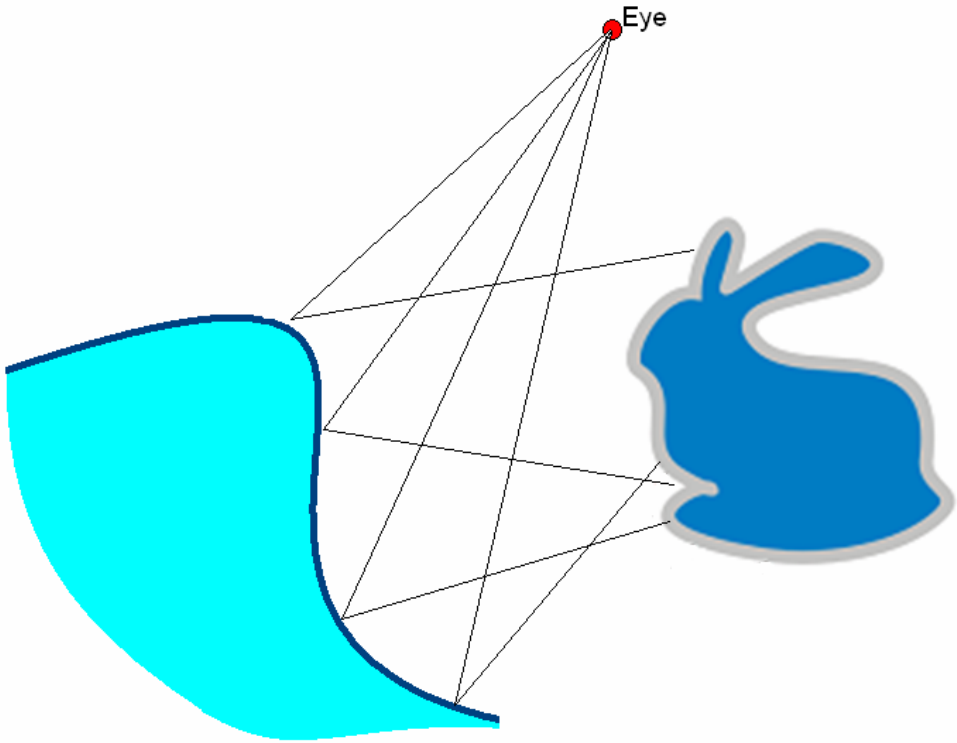
Chunhui Mei

Mar 2006

Reflection Rendering

- Ray tracing – backward, slow
- Environment mapping – fast, not accurate
- Model morphing – forward, only for convex shape
- Approximation – billboard…

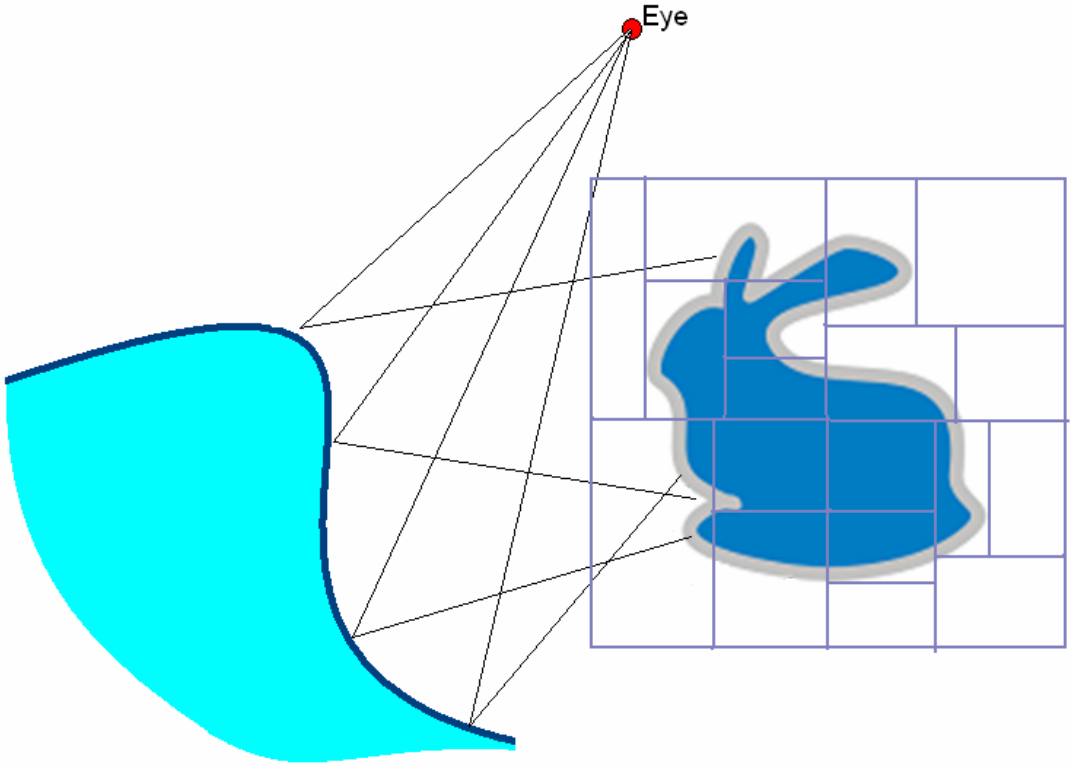
Reflection Rendering



- N_{pixel} =1M
- $N_{ver\text{-}d}$ =1K
- $N_{tri\text{-}d}$ = 2K
- $N_{vex\text{-}r}$ =1K
- $N_{tri\text{-}r}$ =2K
- F_{acc} = 1/1000
- N_{ray} = 1M
- $N_{cluster}$ = 2K

Reflection Rendering

- Ray tracing

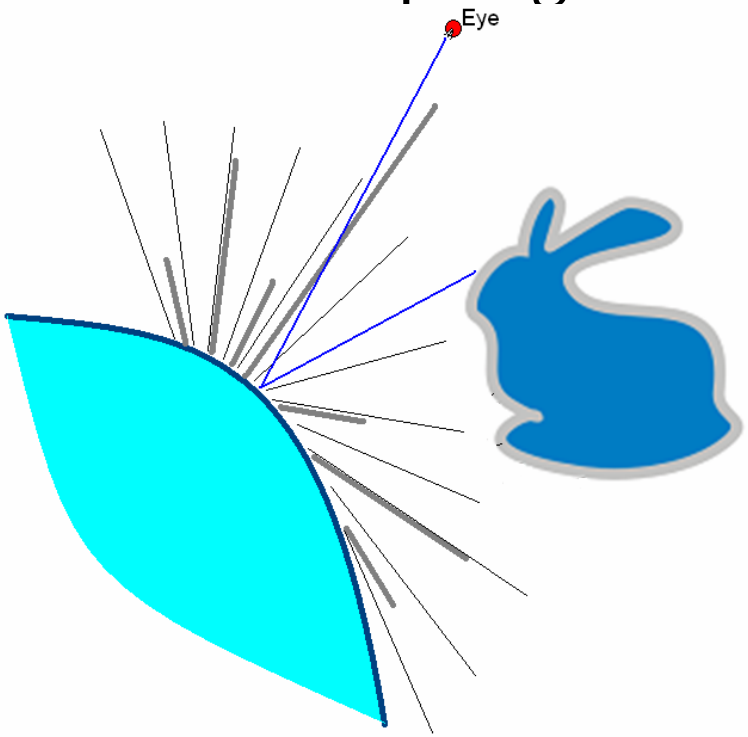


$N_{pixel}*(N_{tri\text{-}d}*F_{acc}*T_{inter}+T_{trav})$

$2M*T_i + 1M*T_t$

Reflection Rendering

- Reflection morphing



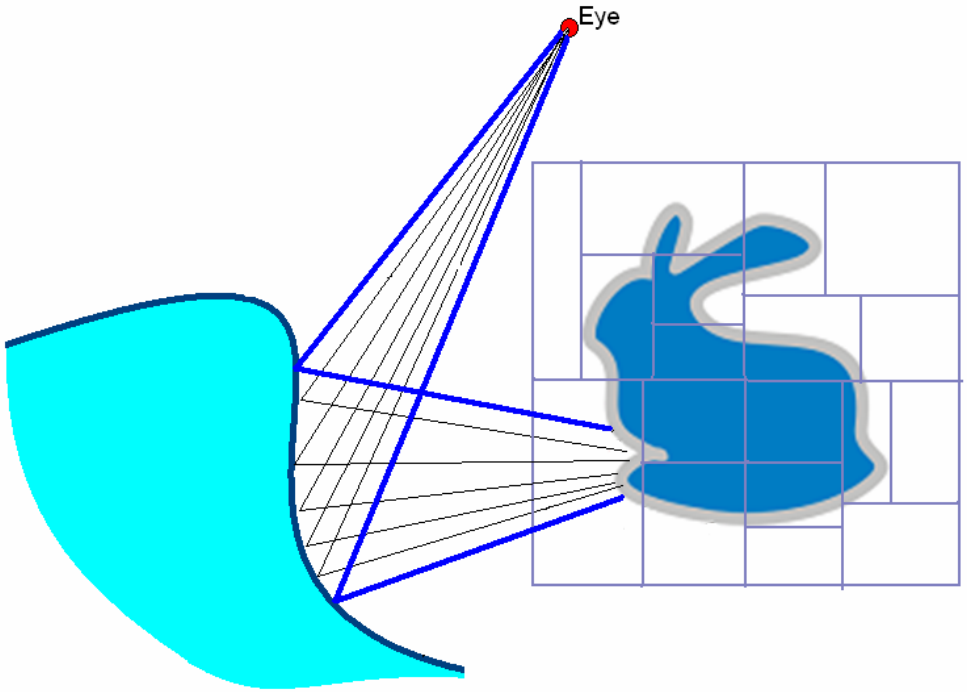
$N_{ver\text{-}d} * (N_{ray} * F_{acc} * T_{inter} + T_{trav})$

$2M * T_i + 1M * T_t$

$1M * Ti + 1K * Tt$

Reflection Rendering

- Ray tracing with clustering



$N_{cluster}*T_{trav}+N_{pixel}*N_{tri\text{-}d}*F_{acc}*T_{inter}$

$2M*T_i + 1M*T_t$

$1M*Ti + 1K*Tt$

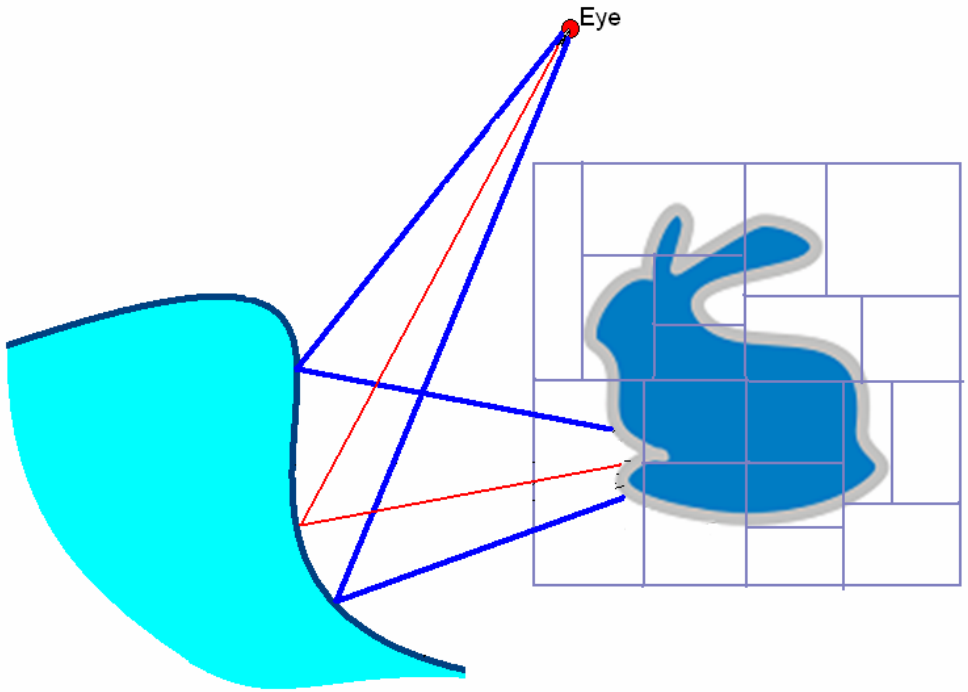
$2M*Ti + 2K*Tt$

Reflection Rendering

- Hybrid backward forward rendering
  - Trace ray-cone in acceleration structure
  - Obtain the vertices which have potential projections
  - Project the vertices by general 3 ray camera
  - Generate reflection image using projected vertices

Reflection Rendering

- Hybrid approach



$N_{cluster}*(T_{trav}+N_{ver\text{-}d}*F_{acc}*T_{proj})$

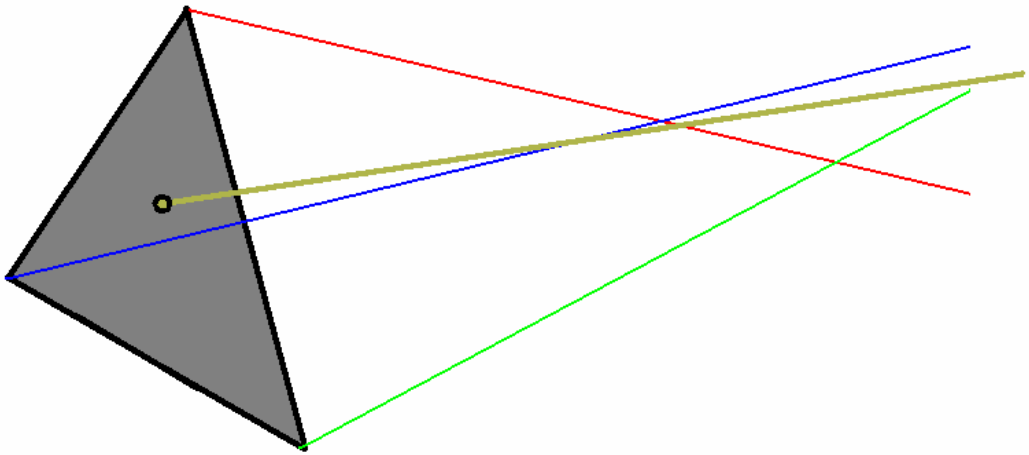
$2M*T_i + 1M*T_t$
$1M*Ti + 1K*Tt$
$2M*Ti + 2K*Tt$
$2K*Tp + 2K*Tt$

Hybrid Reflection Rendering

- Process
  - K-D tree construction of diffuse scene
  - For every triangle on reflective surface
    - Build general 3 ray camera
    - Generate bounding cone of 3 ray camera
    - Trace the cone into K-D tree
    - Obtain vertex set in leaf node intersect with the cone
    - Cull out vertices outside of cone
    - Project the vertices onto the reflective triangle
  - Render reflection image with projection of all vertices

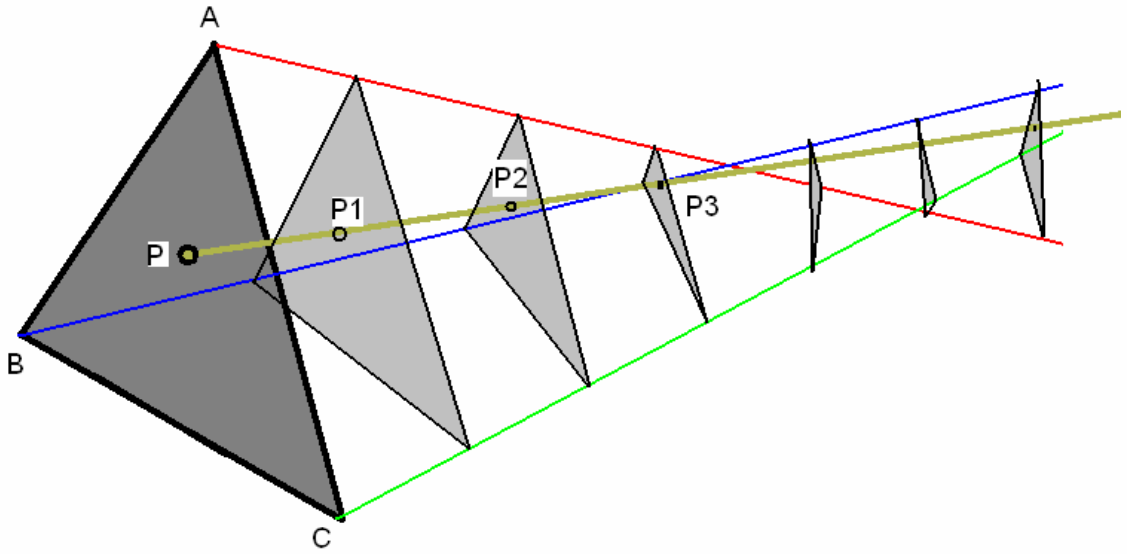
Three Ray Camera

- Ray interpolation inside of triangle



Three Ray Camera

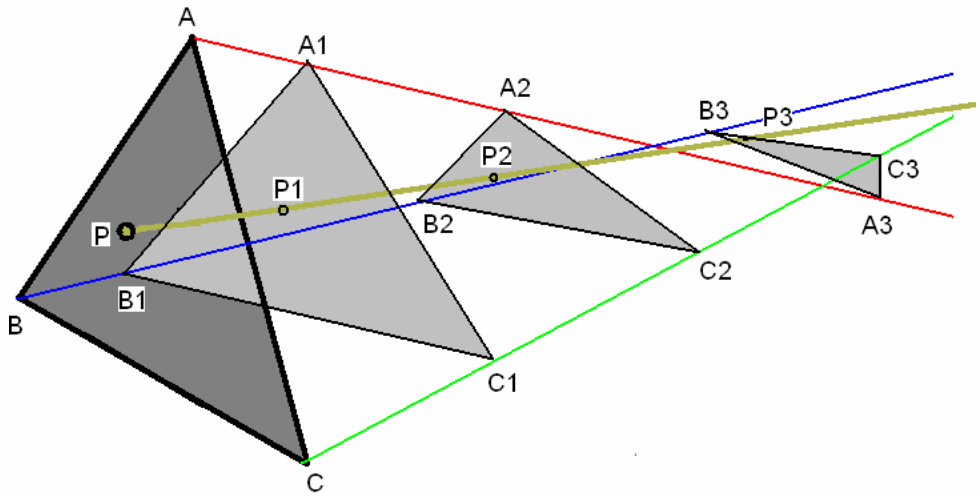
- Parallel 3 ray camera



- Projection using parallel 3 ray camera
- Discontinuity between neighboring parallel 3 ray cameras

Three Ray Camera

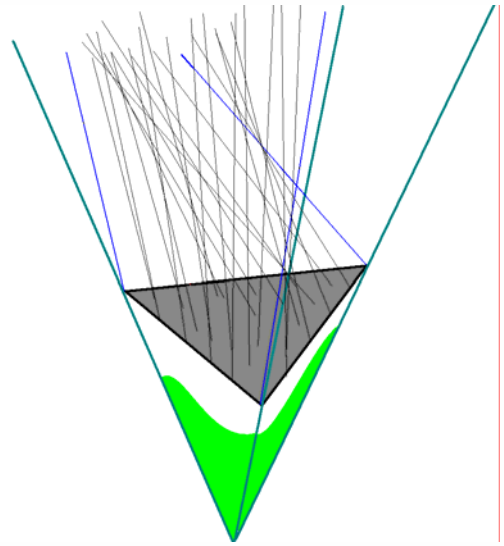
- General 3 ray camera



- Continuous between neighboring triangles
- Nonlinear projection

General 3 ray camera

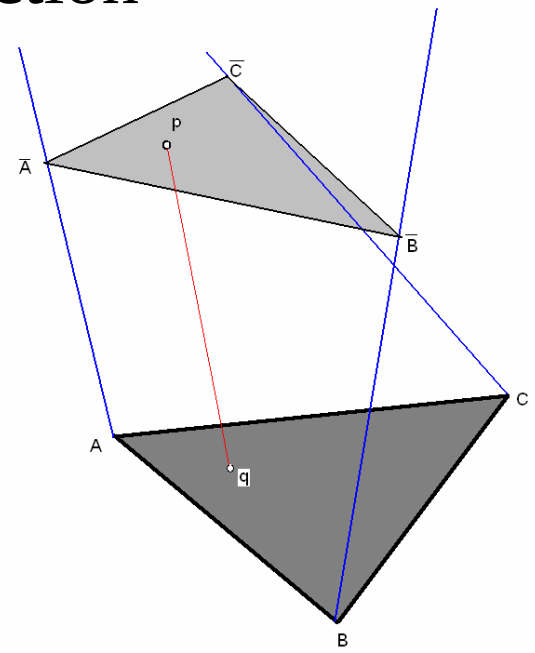
- Bounding cone of 3 ray camera



- Convenient to tracing into K-D tree
- Fast culling for vertex outside of 3 ray camera

General 3 ray camera

- Projection



$$\vec{V}_{A\bar{A}} = l * \vec{r}_A$$
$$\vec{V}_{B\bar{B}} = l * \vec{r}_B$$
$$\vec{V}_{C\bar{C}} = l * \vec{r}_C$$
$$\vec{N} = \vec{V}_{\bar{A}\bar{B}} \times \vec{V}_{\bar{A}\bar{C}}$$

$$\vec{V}_{\bar{A}P} \bullet \vec{N} = 0$$

$$W_A = W_{\bar{A}} = (\vec{V}_{\bar{B}P} \times \vec{V}_{\bar{C}P}) / (\vec{V}_{\bar{A}\bar{B}} \times \vec{V}_{\bar{A}\bar{C}})$$
$$W_B = W_{\bar{B}} = (\vec{V}_{\bar{A}P} \times \vec{V}_{\bar{C}P}) / (\vec{V}_{\bar{A}\bar{B}} \times \vec{V}_{\bar{A}\bar{C}})$$
$$W_C = W_{\bar{C}} = (\vec{V}_{\bar{B}P} \times \vec{V}_{\bar{C}P}) / (\vec{V}_{\bar{A}\bar{B}} \times \vec{V}_{\bar{A}\bar{C}})$$

- Projection speed 3.5M/s

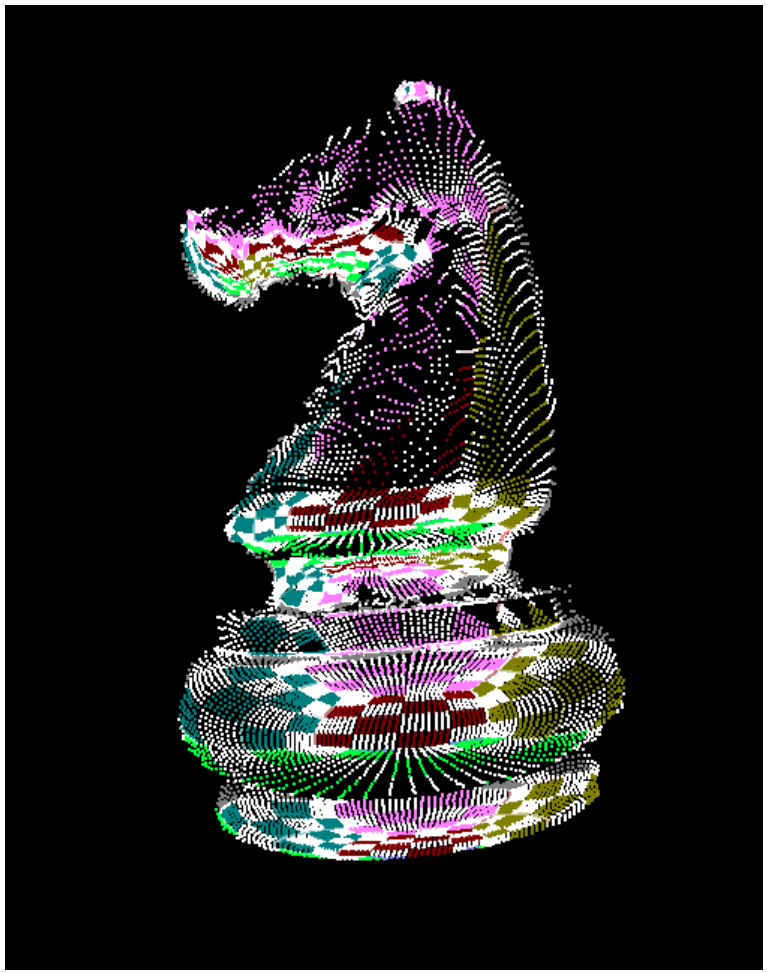
$$Q = W_A * \vec{V}_A + W_B * \vec{V}_B + W_C * \vec{V}_C$$

Hybrid Reflection Rendering

- Projection result

Hybrid Reflection Rendering

- Projection result



Hybrid Reflection Rendering

- Triangulation of projected vertices
  - Connect the nearest vertices for projected triangle
  - Avoid skinning triangles
  - Avoid holes

Hybrid Reflection Rendering

Hybrid Reflection Rendering

- Other approaches in process
  - Point based rendering
  - Clipping for general 3 ray camera
  - Filling the holes with ray tracing

Thank you