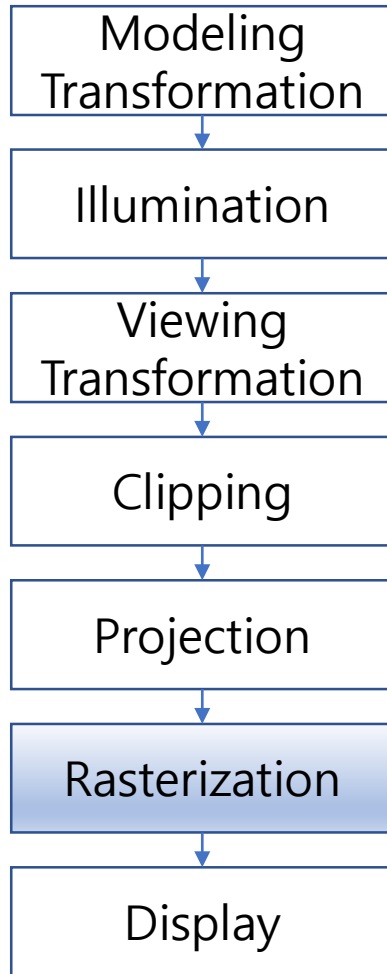


Lecture slides (CT4201/EC4215 – Computer Graphics)

Rasterization

Lecturer: Bochang Moon

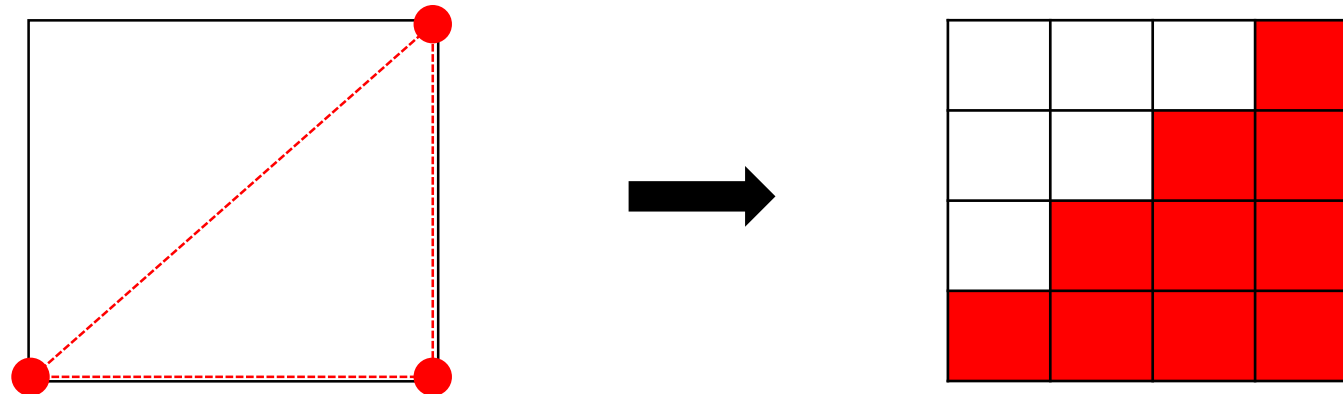
Rasterization



- Determine pixel values covered by primitives

Triangle Rasterization

- Draw a 2D triangles with 2D points p_0, p_1, p_2 in screen coordinates
 - Each vertex can contain multiple properties
 - position, color, etc.
 - For intermediate pixels, we need interpolate them.

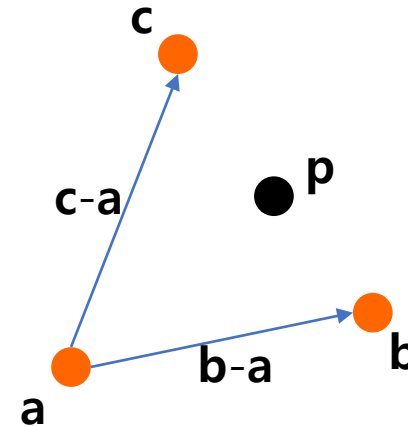


Background: Barycentric Coordinates

- Usually assign some properties (e.g., color) to each vertex in primitives and interpolate those values

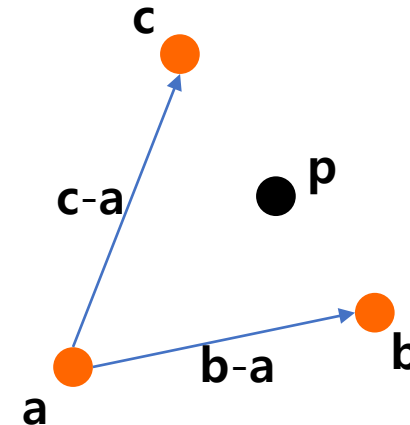
- $\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$
- $= (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$
- $= \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \quad (\alpha \equiv 1 - \beta - \gamma)$

- $\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$
 - with $\alpha + \beta + \gamma = 1$



Background: Barycentric Coordinates

- $p(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
 - with $\alpha + \beta + \gamma = 1$
- Conditions for a point p inside the triangle
 - $0 < \alpha < 1$
 - $0 < \beta < 1$
 - $0 < \gamma < 1$
- Q. Conditions for a point on an edge?
- Q. Conditions for a point at a vertex?



Background: Barycentric Coordinates

- $\mathbf{p}(\alpha, \beta, \gamma) = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c}$

- with $\alpha + \beta + \gamma = 1$

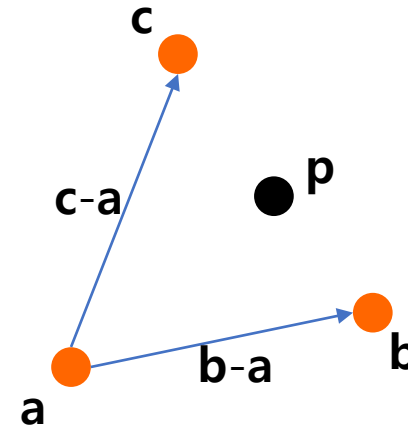
- Q. Given a point \mathbf{p} , how do we compute the coordinates?

- $\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$

- Corresponding matrix form:

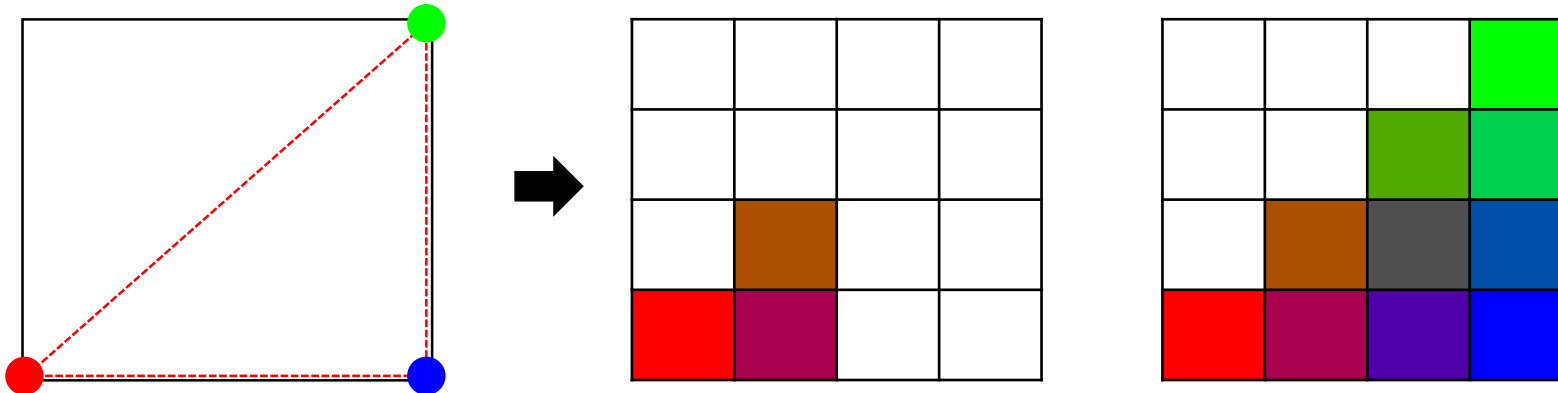
- $$\begin{bmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x_p - x_a \\ y_p - y_a \end{bmatrix}$$

- $\alpha = 1 - \beta - \gamma$



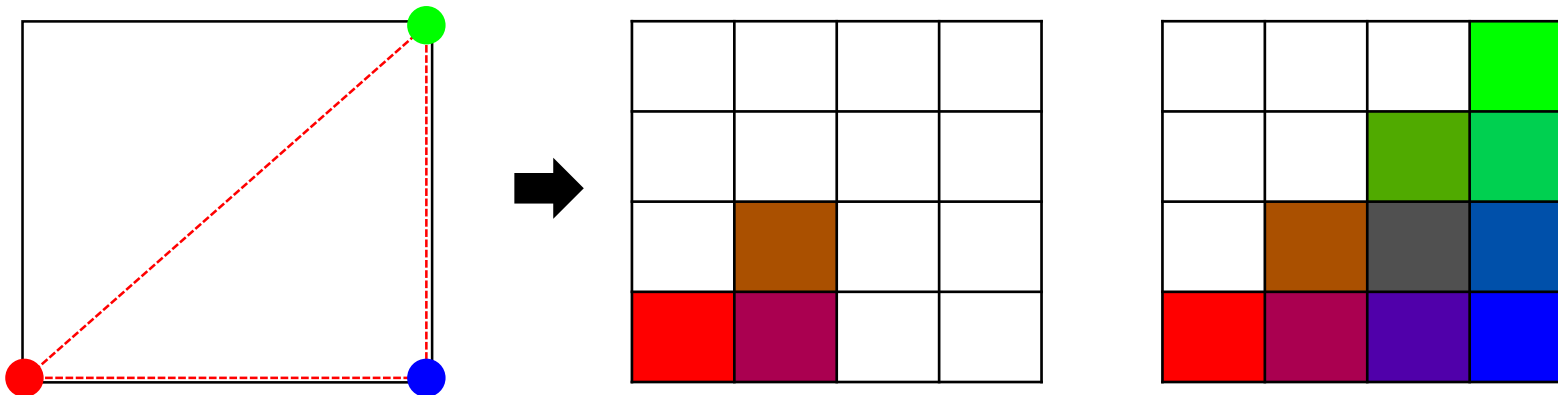
Triangle Rasterization

- For all x do
 - For all y do
 - Compute α, β, γ for (x, y)
 - If $(\alpha \in [0,1] \ \&\& \ \beta \in [0,1] \ \&\& \ \gamma \in [0,1])$ then
 - $c = \alpha c_0 + \beta c_1 + \gamma c_2$ // Gouraud interpolation (Gouraud, 1971)
 - Draw a color c at a pixel (x, y)

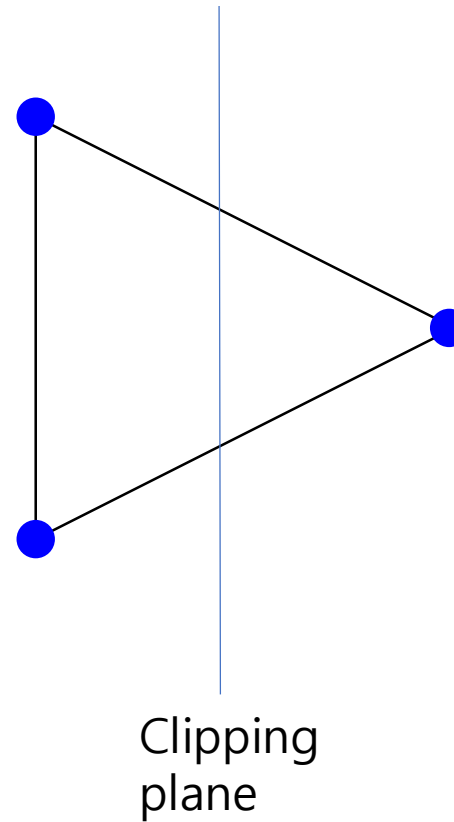
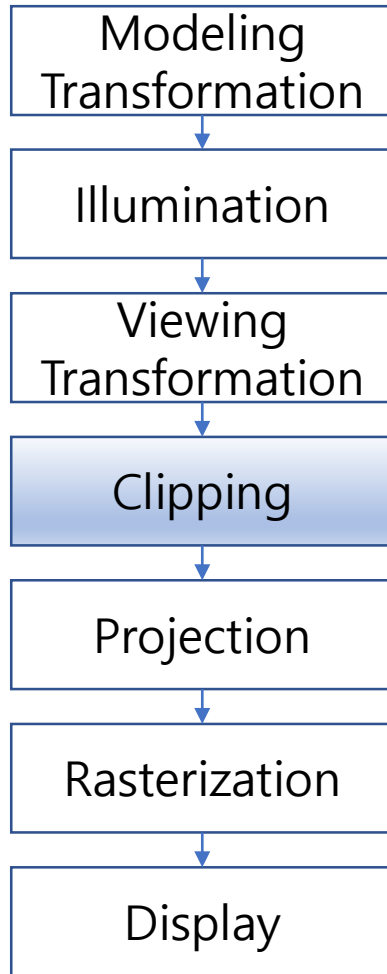


Triangle Rasterization

- For all x **within a bounding box** do
 - For all y **within a bounding box** do
 - Compute α, β, γ for (x, y)
 - If $(\alpha \in [0,1] \ \&\& \ \beta \in [0,1] \ \&\& \ \gamma \in [0,1])$ then
 - $c = \alpha c_0 + \beta c_1 + \gamma c_2$ // Gouraud interpolation (Gouraud, 1971)
 - Draw a color c at a pixel (x, y)

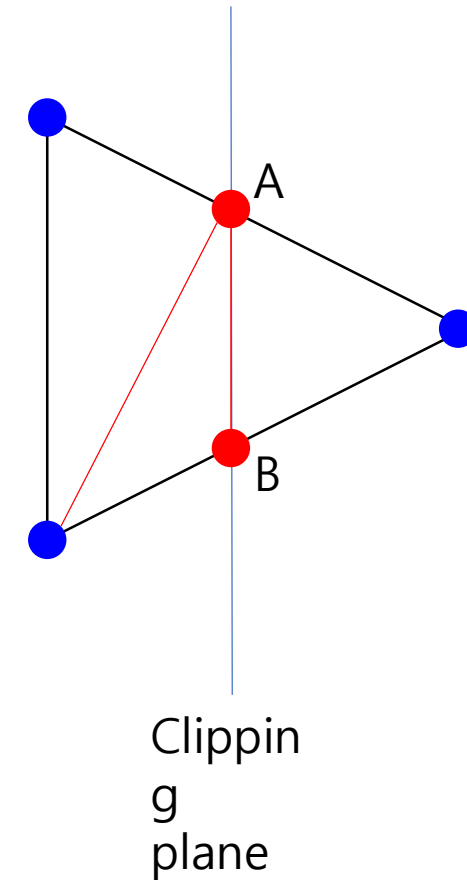


Clipping



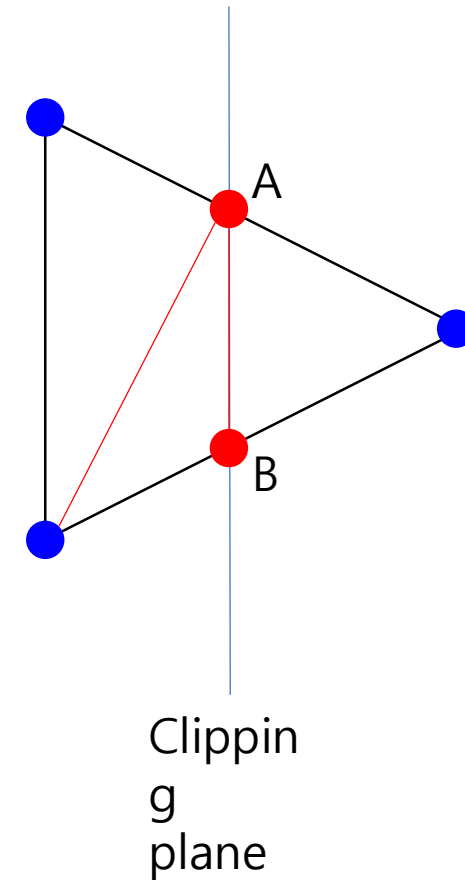
Clipping

- A parametric line connecting \mathbf{a} and \mathbf{c} :
 - $\mathbf{p}(t) = \mathbf{a} + t(\mathbf{c} - \mathbf{a})$
- Implicit plane through point \mathbf{q} with normal \mathbf{n} :
 - $f(\mathbf{p}) = (\mathbf{p} - \mathbf{q}) \cdot \mathbf{n} = 0$
 - Can be written as:
 - $f(\mathbf{p}) = \mathbf{p} \cdot \mathbf{n} + D = 0$
 - Recall that
 - $f(\mathbf{p}) < 0$: \mathbf{p} is inside
 - $f(\mathbf{p}) = 0$: \mathbf{p} is on the plane
 - $f(\mathbf{p}) > 0$: \mathbf{p} is outside



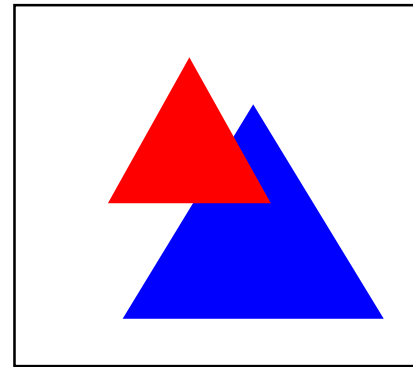
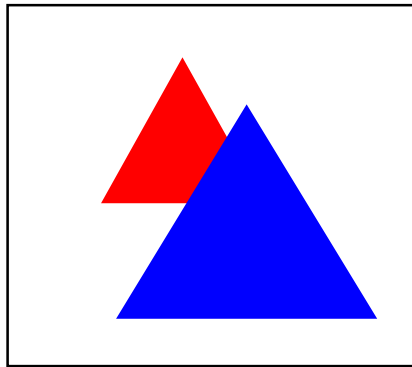
Clipping

- A parametric line connecting \mathbf{a} and \mathbf{c} :
 - $\mathbf{p}(t) = \mathbf{a} + t(\mathbf{c} - \mathbf{a})$
- Implicit plane through point \mathbf{q} with normal \mathbf{n} :
 - $f(\mathbf{p}) = \mathbf{p} \cdot \mathbf{n} + D = 0$
 - If the end points of a line have different signs,
 - we need to find the intersection point between the plane and line segment
 - $(\mathbf{a} + t(\mathbf{c} - \mathbf{a})) \cdot \mathbf{n} + D = 0$
 - Solving for t:
 - $t_A = -\frac{\mathbf{n} \cdot \mathbf{a} + D}{\mathbf{n} \cdot (\mathbf{c} - \mathbf{a})}$
 - $\mathbf{A} = \mathbf{a} + t_A(\mathbf{c} - \mathbf{a})$
 - A similar computation will give B.



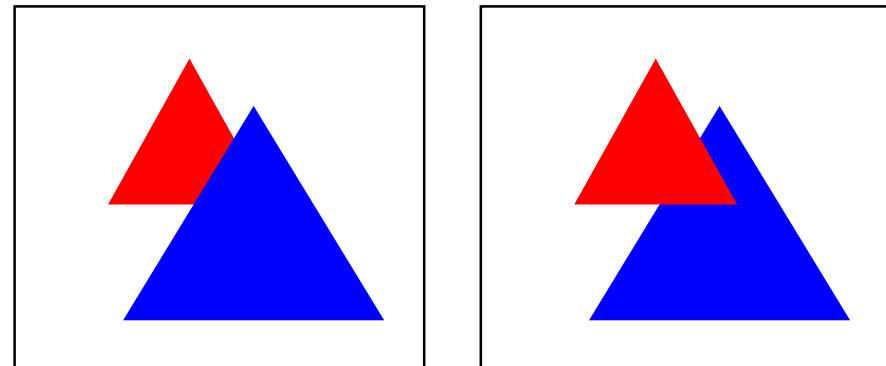
Z-Buffer for Hidden Surfaces

- In graphics, z-buffer (depth buffer) is used to draw closest primitives.
- Z-buffer
 - Store the depth at each pixel

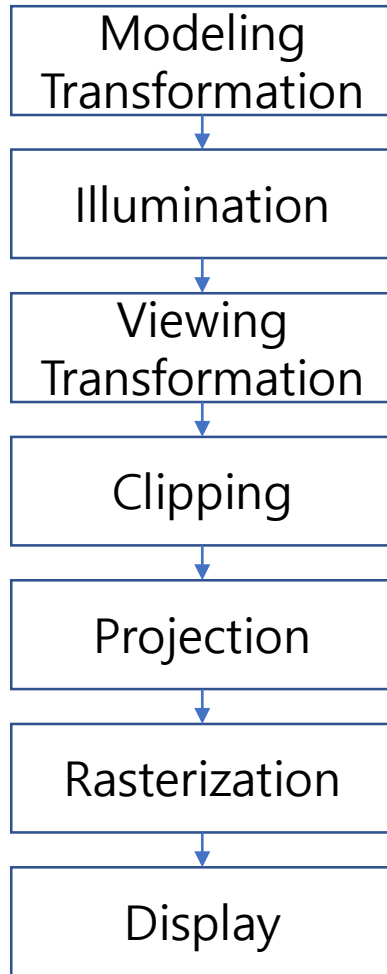


Z-Buffer for Hidden Surfaces

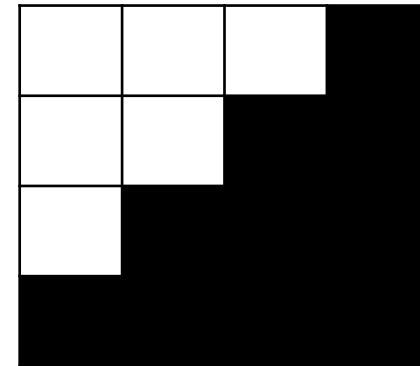
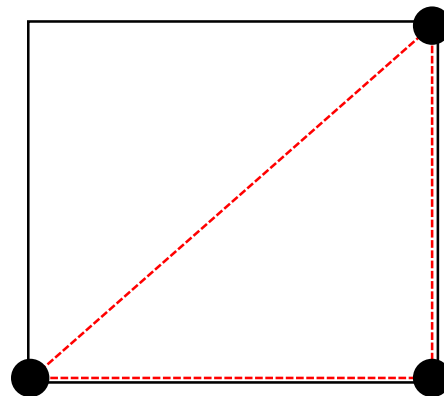
- In graphics, z-buffer (depth buffer) is used to draw closest primitives.
- Z-buffer
 - Store the depth at each pixel
- Algorithm:
 - Initialize z-buffer to the maximum depth (i.e., the depth of the far plane)
 - Interpolate the z-coordinate as a vertex attribute (similar to the color interpolation)
 - If the z value at $(x, y) < z\text{-buffer}(x, y)$
 - Fill the pixel color at (x, y)
 - Update the $z\text{-buffer}(x, y)$



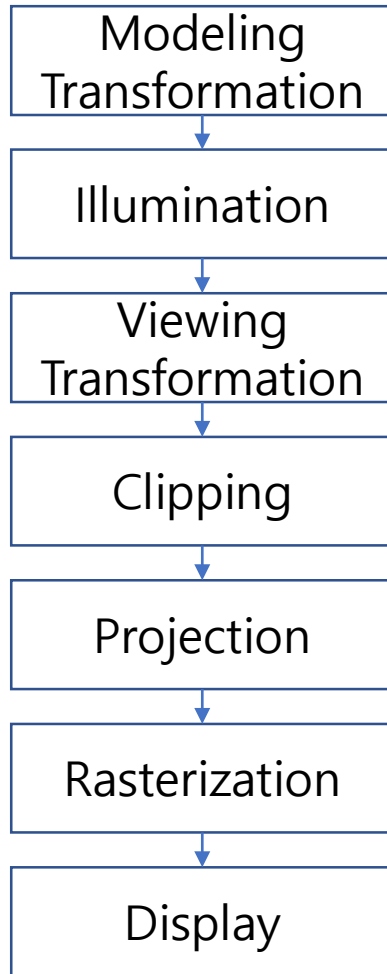
Review of the Rendering Pipeline



- So far, we have covered most pipeline components.
- The illumination part will be discussed later.
- Q. how do we handle aliasing?

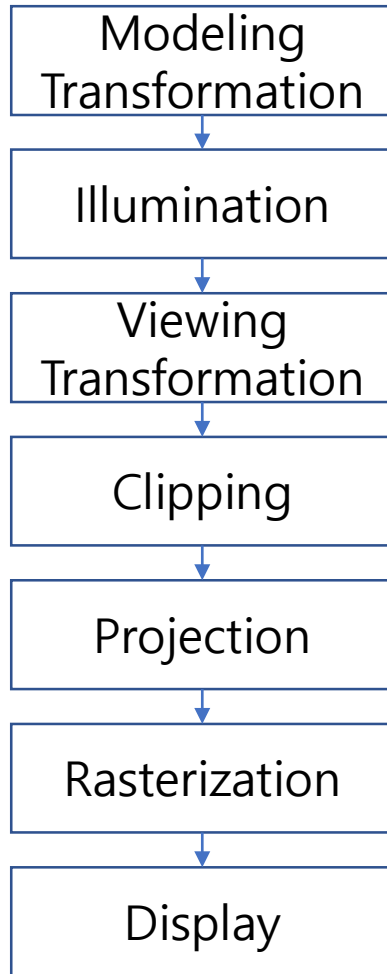


Review of the Rendering Pipeline



- So far, we have covered most pipeline components.
- The illumination part will be discussed later.
- Q. how do we handle aliasing?
 - Super-sampling: draw a larger image than a specified resolution, and average multiple colors with a filter (box filter)
 - Sophisticated filters can be designed.
 - 3D game may provide an option to users for this anti-aliasing.

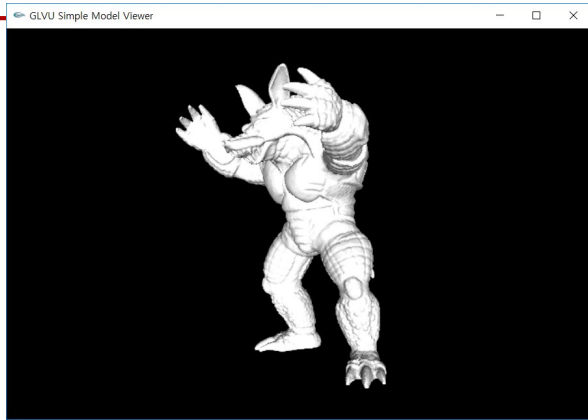
Review of the Rendering Pipeline



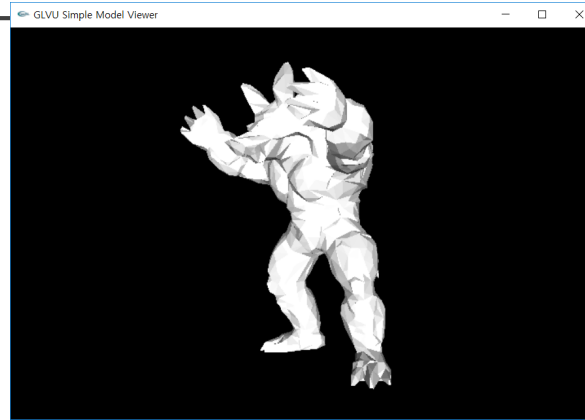
- So far, we have covered most pipeline components.
- The illumination part will be discussed later.
- Q. how do we handle aliasing?
- Q. do we have other optimization techniques?
 - Levels-of-detail (LOD)
 - Static LOD
 - Dynamic LOD
 - Stochastic Simplification of Aggregate Detail, [Cook et al. SIG 07]



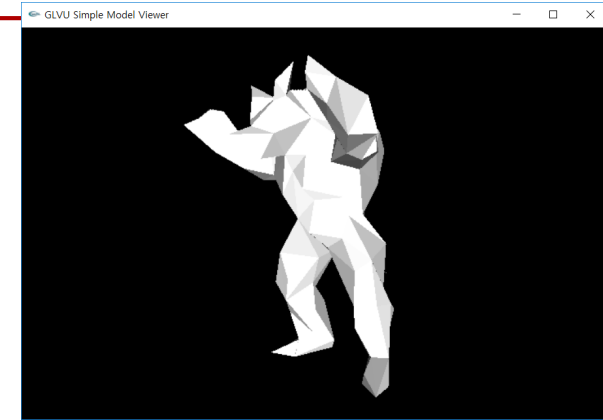
Review of the Rendering Pipeline



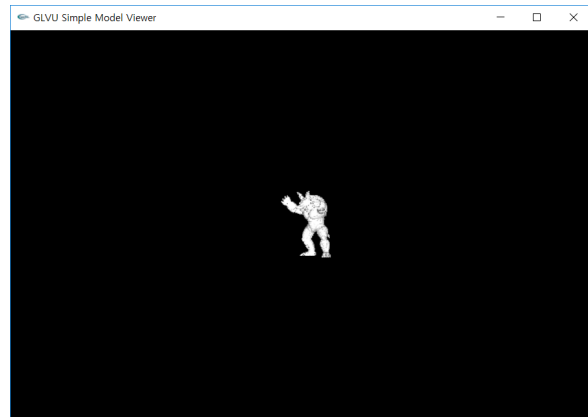
345944 tri.



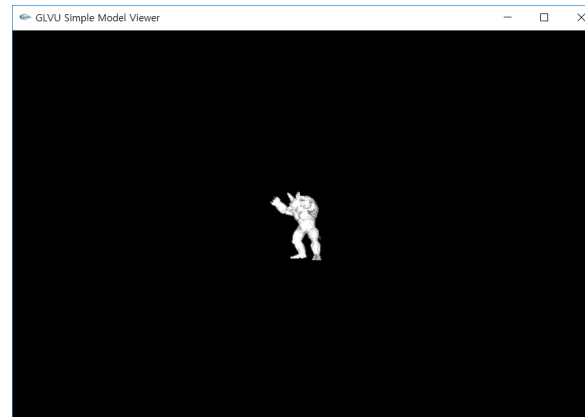
3000 tri.



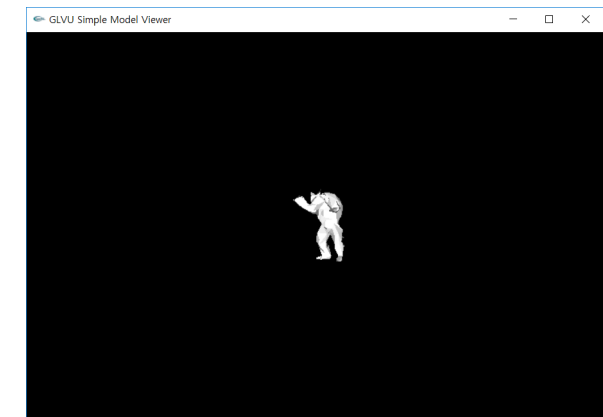
300 tri.



345944 tri.



3000 tri.



300 tri.

Further Readings

- Chapter 8
- Chapter 12.4