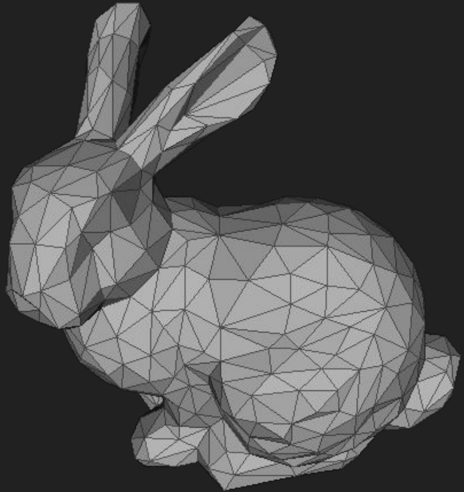
13 - Graphics, Rendering, and Lighting

CS 3160 - Game Programming
Max Gilson

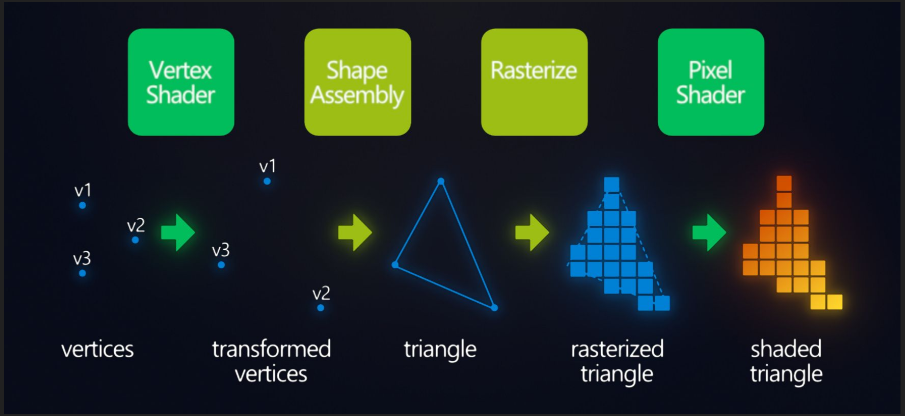
Graphics Rendering Pipeline

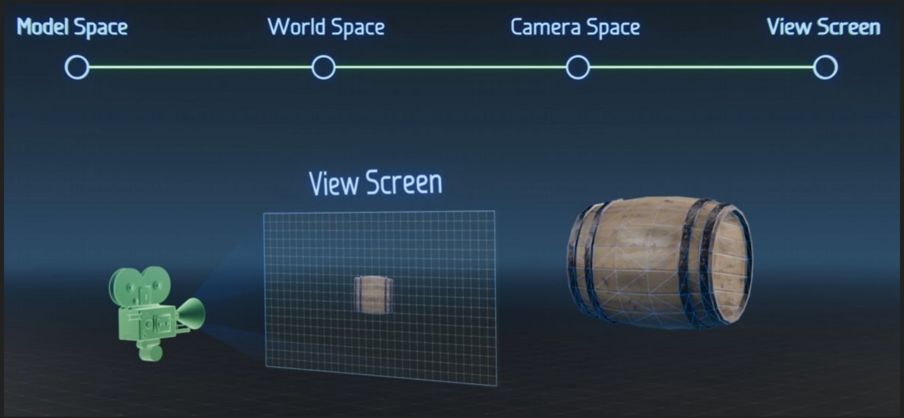
- Your graphics card goes through 3 key steps to display graphics on your screen:
  - Vertex Shading
  - Rasterization
  - Fragment Shading



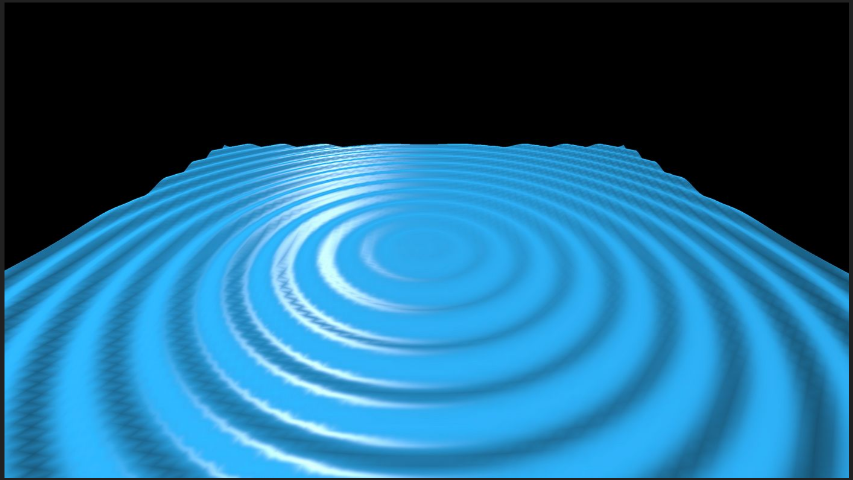
Vertex Shading

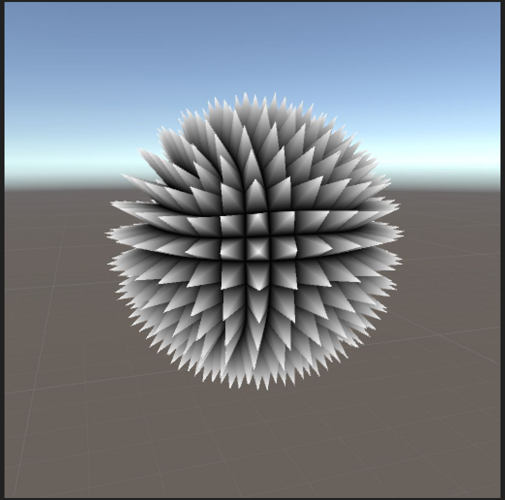
- Vertex Shading translates the vertices of a triangle from 3D space (the game) to 2D space (your screen)
- Using transformation matrices (transforms) all vertices in the model are converted from:
  - Model space (local)
  - World space (game world)
  - Camera space (relative to camera)
  - Screen space (2D screen)





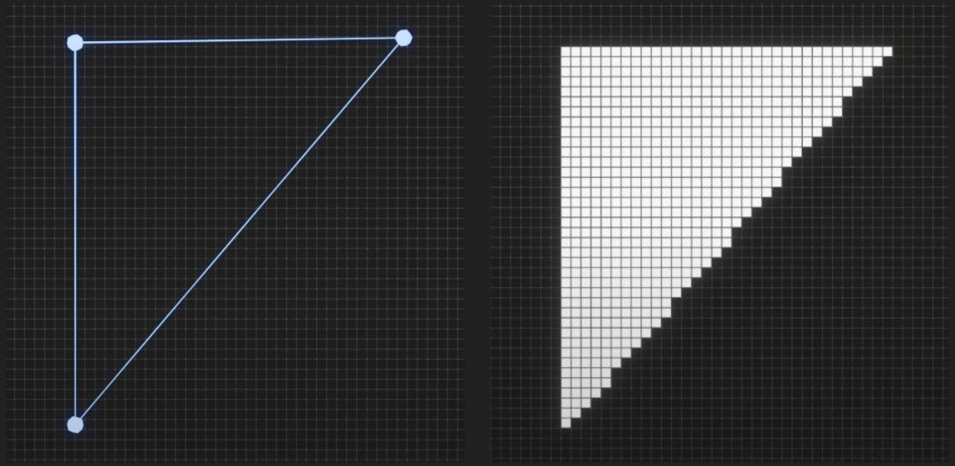
Vertex Shader Examples





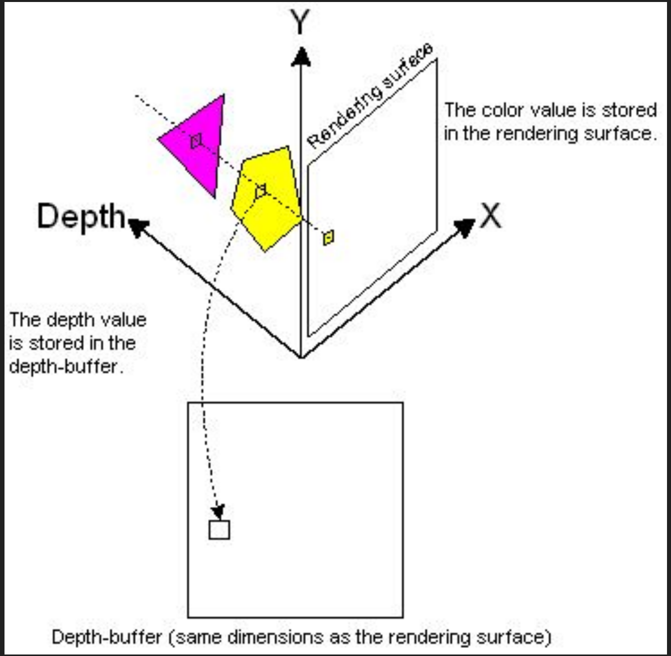
Rasterization

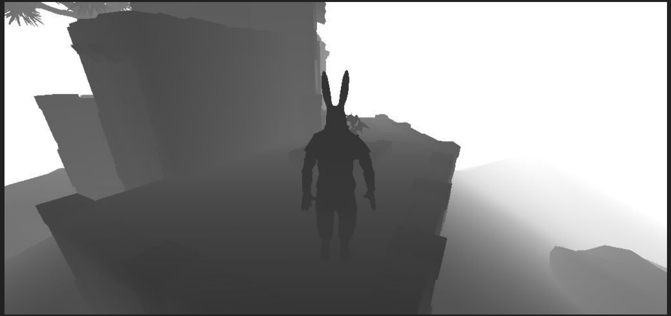
- Rasterization is the “filling in” of the triangle, or finding what pixels make up the triangle
- The color or texture that makes up the triangle must be applied here
- This group of colored pixels are called fragments



Z-Buffer or Depth Buffer

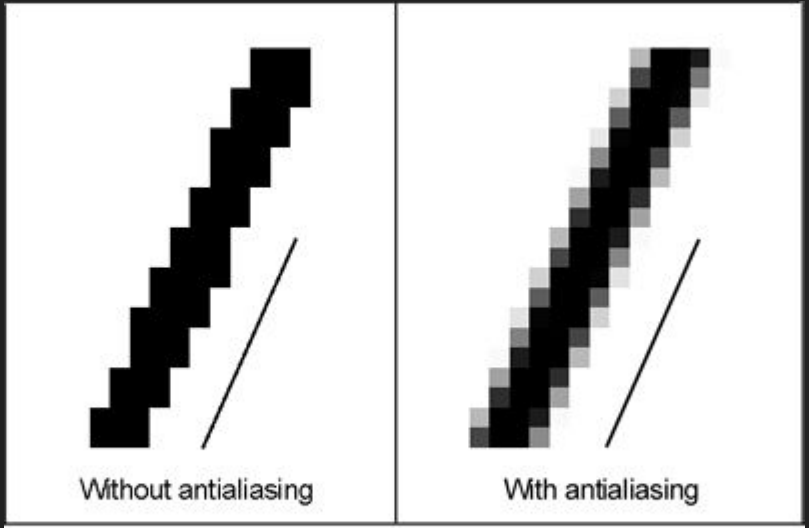
- Often times, triangles will overlap with each other but one triangle will be further away
- The Z-Buffer or Depth Buffer is a depth value for every pixel on the screen
- If a triangle is drawn on top of another triangle, every pixel depth value is compared


Y
Rendering surface
The color value is stored in the rendering surface.
Depth
X
The depth value is stored in the depth-buffer.
Depth-buffer (same dimensions as the rendering surface)



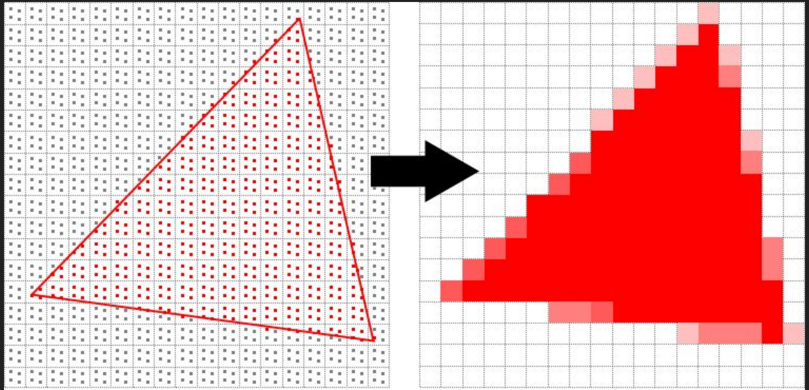
Antialiasing

- The straight lines of the triangles aren’t always perfectly aligned with pixels
- This causes jagged edges
- This is solved with antialiasing
- Antialiasing adds a fractional shade to adjacent pixels
- SSAA samples points within pixels to determine how much of that pixel is occupied by the triangle



Without antialiasing

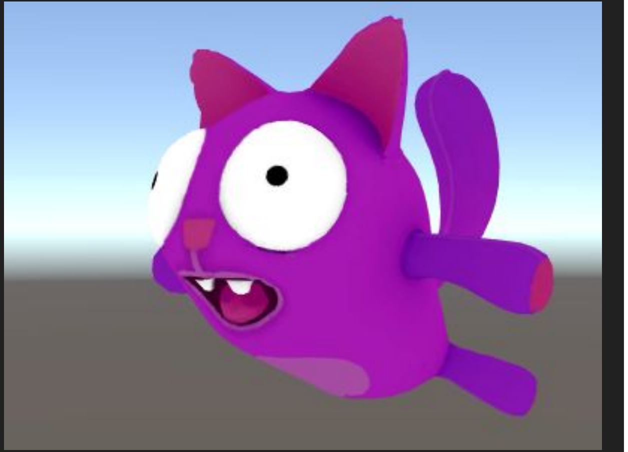
With antialiasing



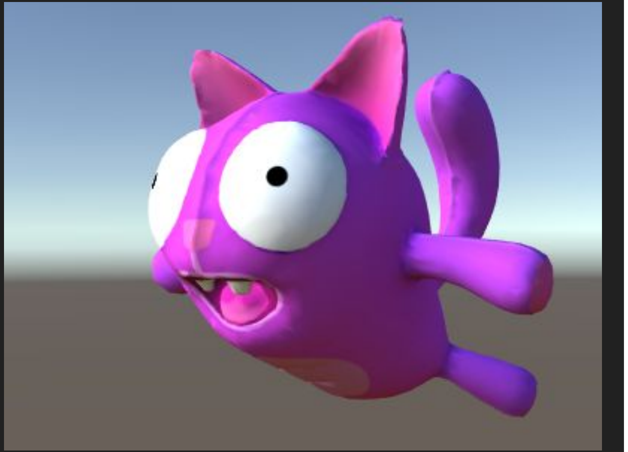
Using 4 samples per pixel (MSAAx4)

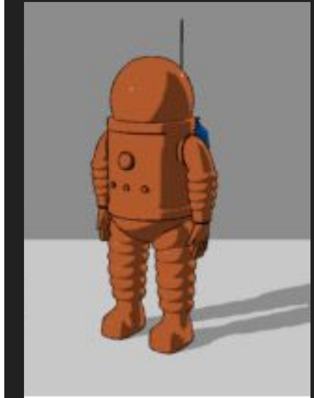
Fragment Shading

- Fragment shading is adding lighting or other shading effects to the fragments
- For lighting, every light that affects the fragment must be accounted for
- For other effects, a fragment shader can alter the appearance of the fragment



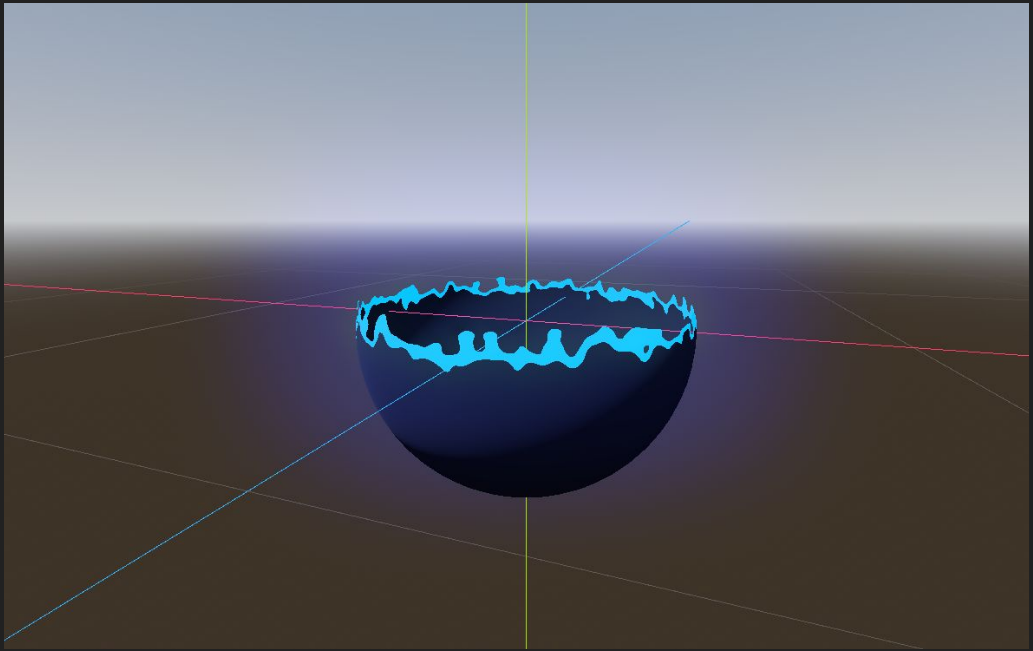
plastic shader

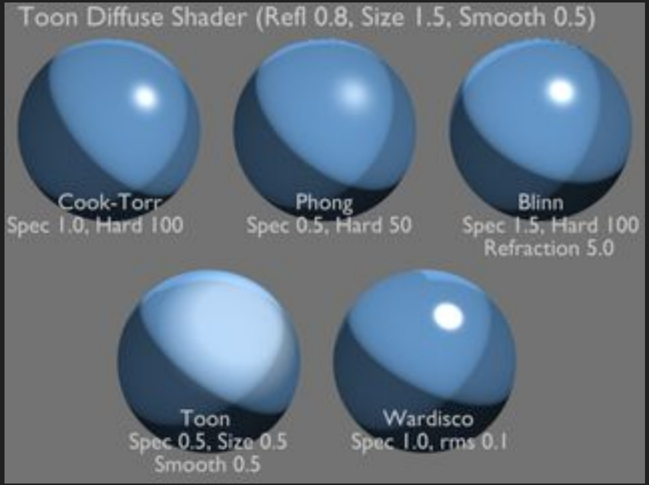


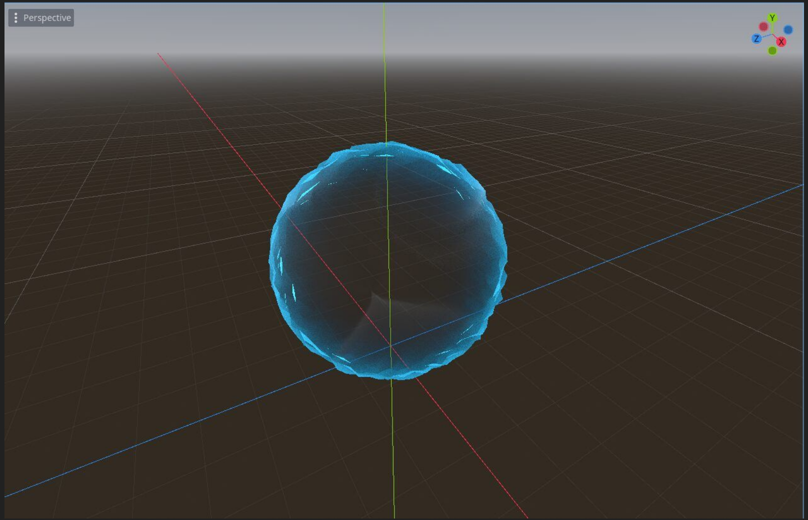


toon shader

Fragment Shader Examples

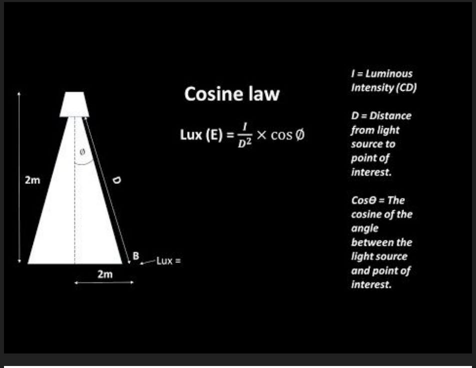






Lighting Angle

- The light intensity of a triangle is dependent on:
  - The intensity of the light
  - The distance from the light
  - The angle of the surface relative to the light
    - Angle between light direction and face normal
- This calculation has to be performed for every triangle affected by light for every light source



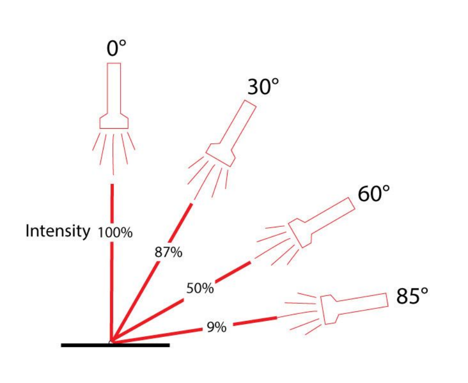
Cosine law

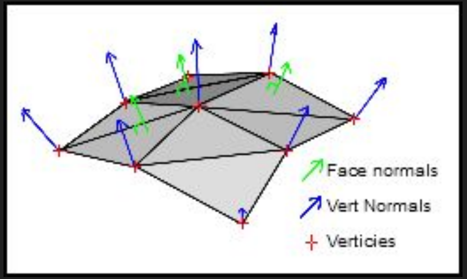
$$\text{Lux (E)} = \frac{I}{D^2} \times \cos \emptyset$$

I = Luminous Intensity (CD)

D = Distance from light source to point of interest.

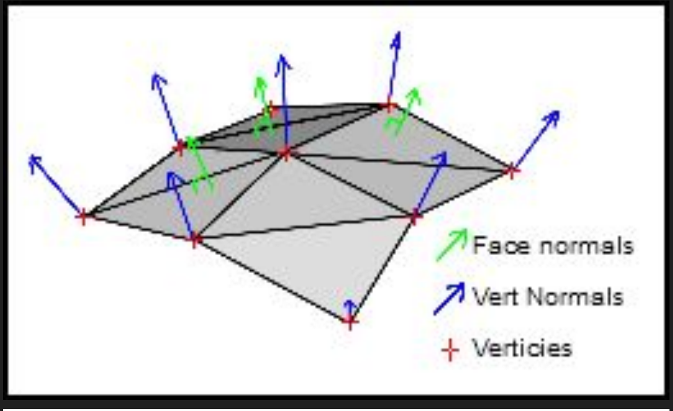
Cosθ = The cosine of the angle between the light source and point of interest.

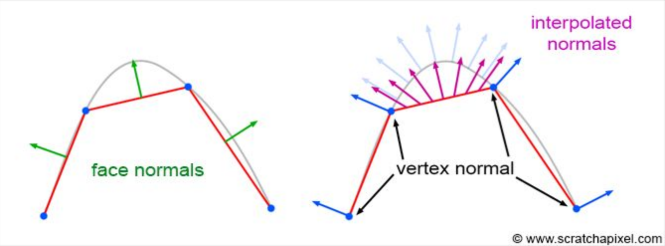




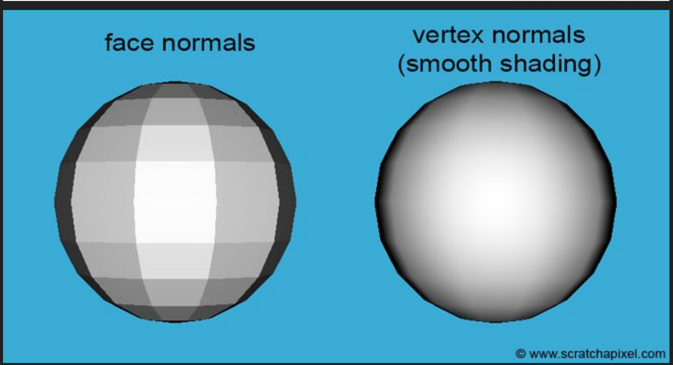
Flat Shading vs Smooth Shading

- If using just the face normal to calculate the light direction, the whole polygon takes on 1 light intensity
  - This is called flat shading
- Instead, we can calculate vertex normals and calculate a smooth gradient of normals between them
  - This is called smooth shading
- In smooth shading, a normal has to be calculated for each pixel, rather than just for the entire fragment





© www.scratchapixel.com



© www.scratchapixel.com