Graphics Pipeline

Interactive 3D Graphics using GPU

Check out WebGL demo by yourself:
https://github.khronos.org/glTF-Sample-Viewer-Release/

Inside Graphics Card

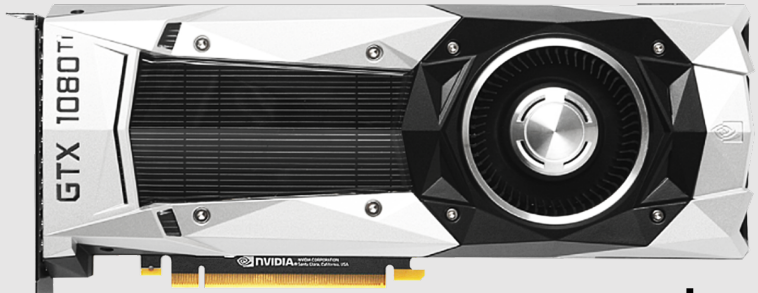


Image Credit: MarcusBurns1977

GPU

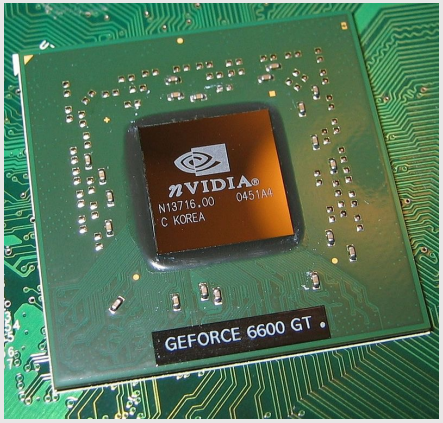
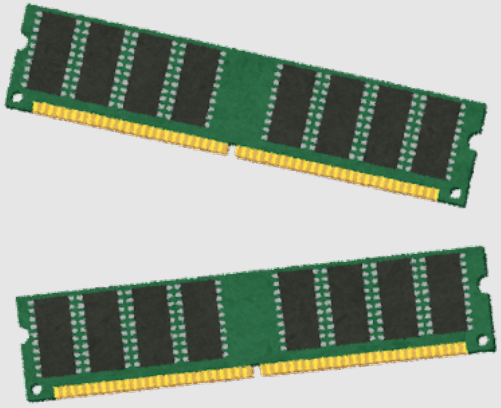


Image Credit: Berkut @ Wikipedia

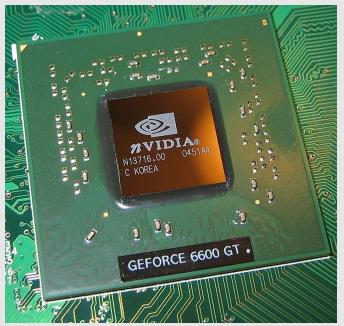
GPU Memory (VRAM)



CPU vs GPU

- Number of cores are very different

Credit: Eric Gaba @ Wikipedia



Credit: Berkut @ Wikipedia

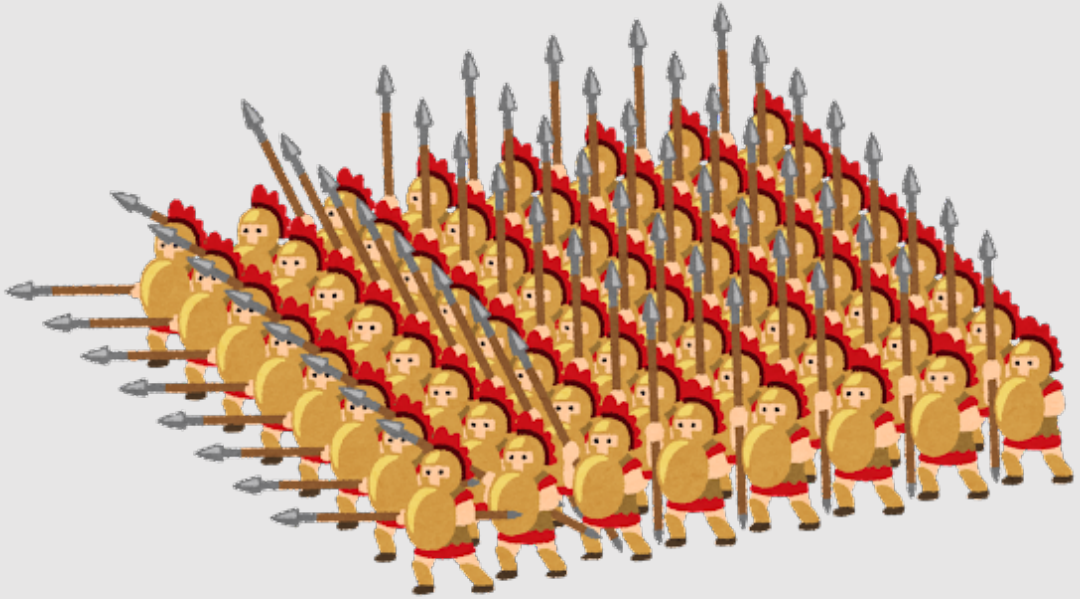
	CPU	GPU
Name	Intel Core i7-6700K	GeForce RTX 2080
Clock	4.2 GHz	1.5 GHz
Number of cores	4	4352
Meomry band width	34.1 GB/s	448.0 GB/s

CPU vs GPU

fast & powerful few threads

VS

Lot of threads doing the same task



CPU vs GPU

One task starts after another task

VS

Multiple tasks run separately

Framebuffer

- Write value at the framebuffer on GPU, you will see image on the screen

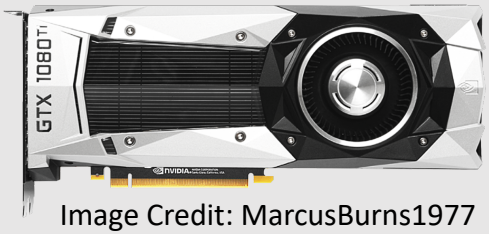


Image Credit: MarcusBurns1977

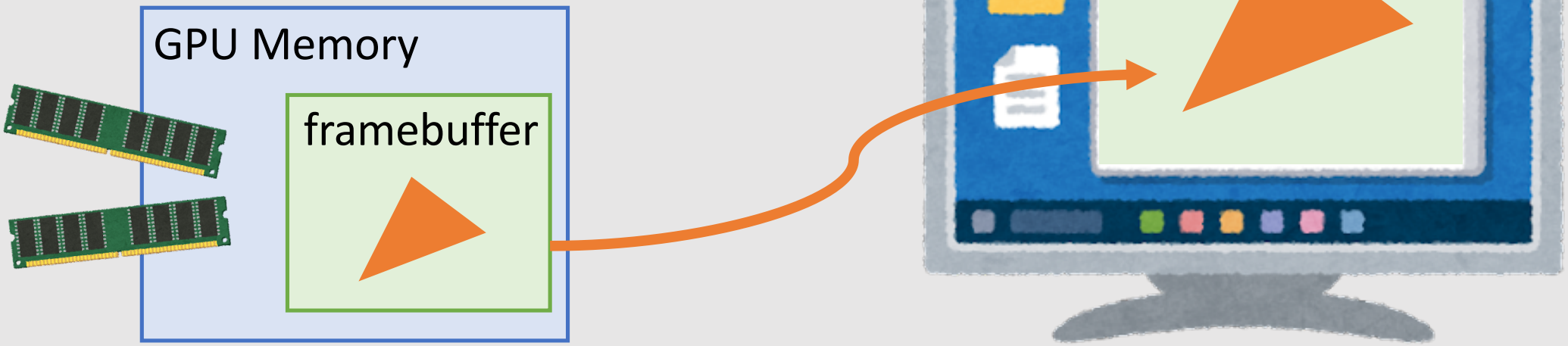
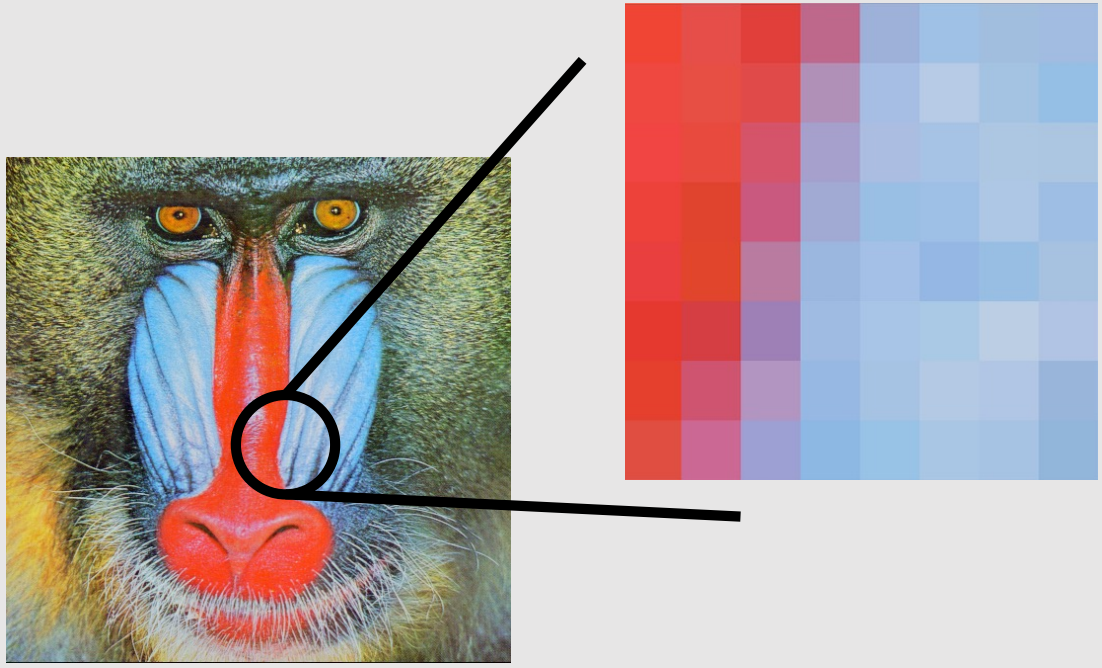


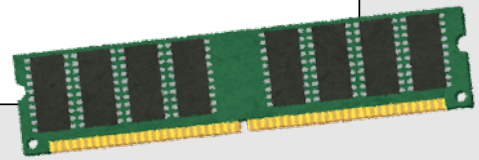
Image on a Memory

- A pixel has RGB (or RGBA) value
- Image is a 3D array of numbers



On memory

RGBRGBRGBRGBRGB…
RGBRGBRGBRGBRGB…
RGBRGBRGBRGBRGB…
RGBRGBRGBRGBRGB…
RGBRGBRGBRGBRGB…
RGBRGBRGBRGBRGB…

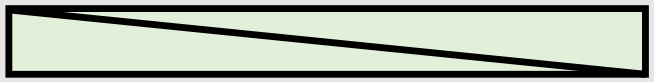


Graphics Pipeline Draw Triangles

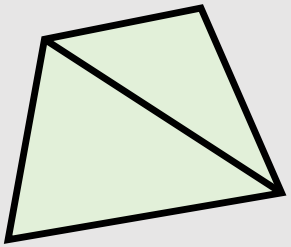
- Graphics pipeline convers all the primitives into triangles

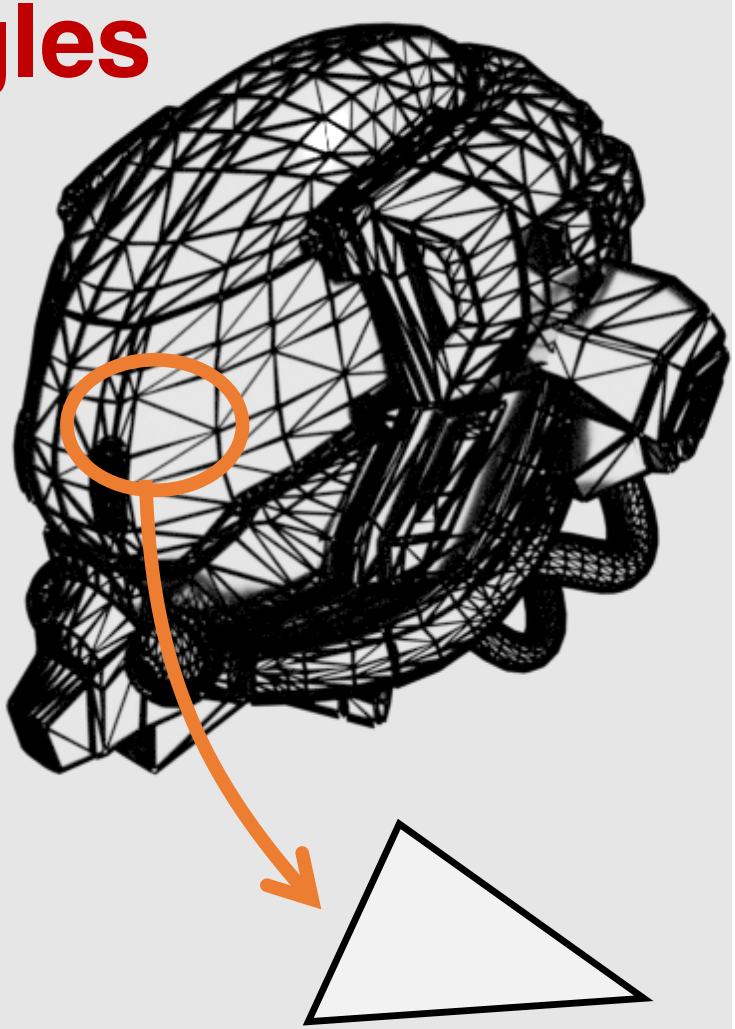
Point:

Line:



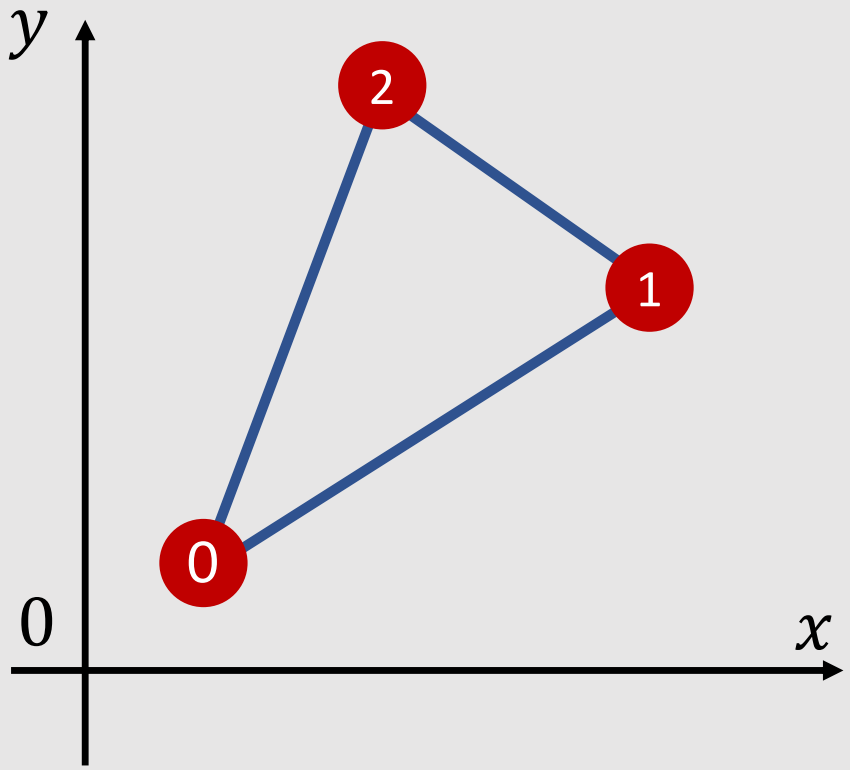
Quad:





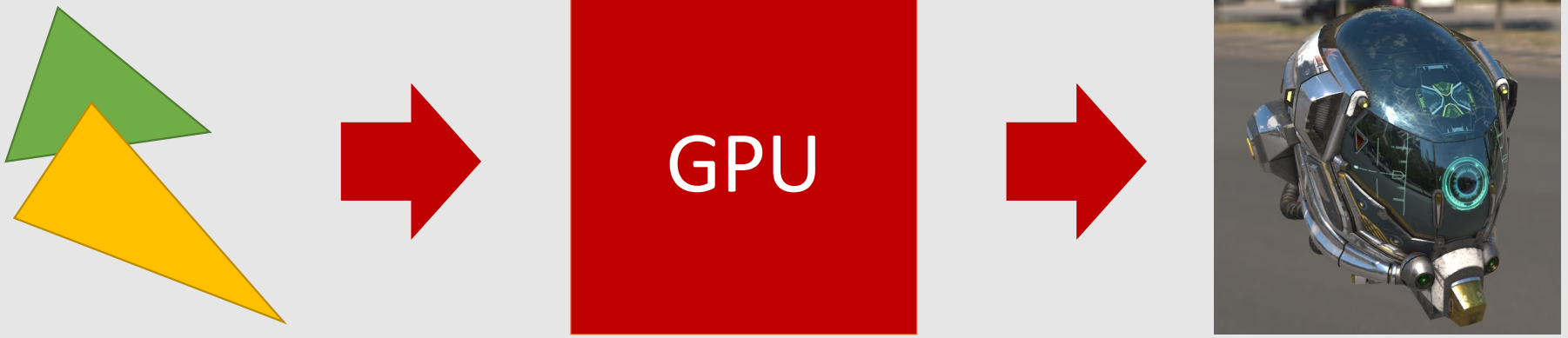
How to Store 3D Shape in Graphics Card

- Triangles = array of coordinates



	x	y
0	1.0	1.0
1	5.0	4.0
2	3.0	5.0

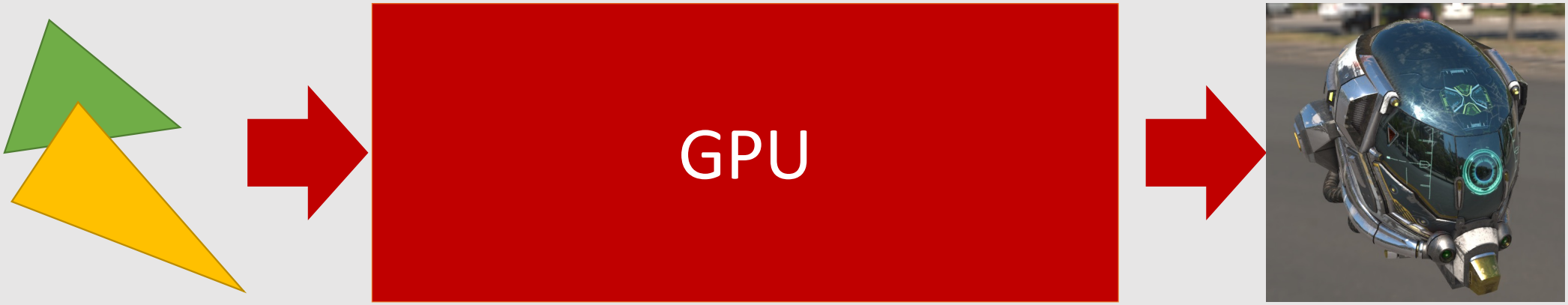
Graphics Pipeline



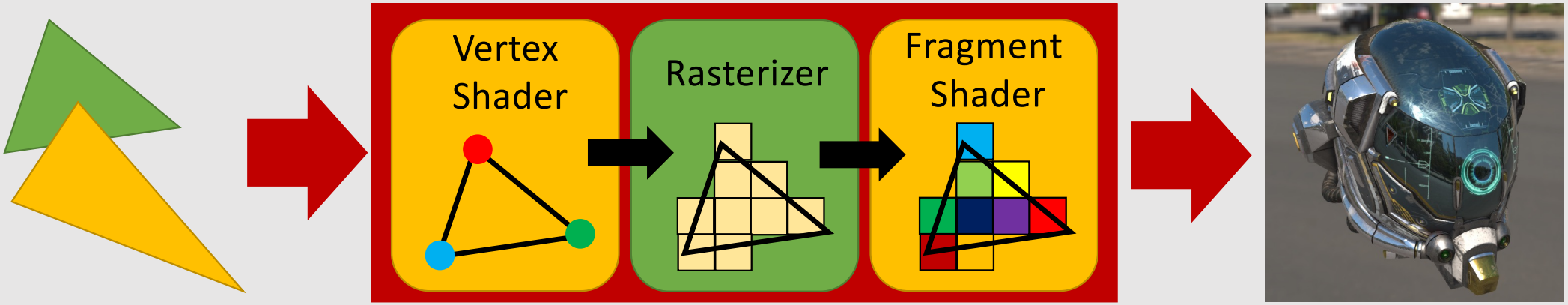
triangles

3D graphics

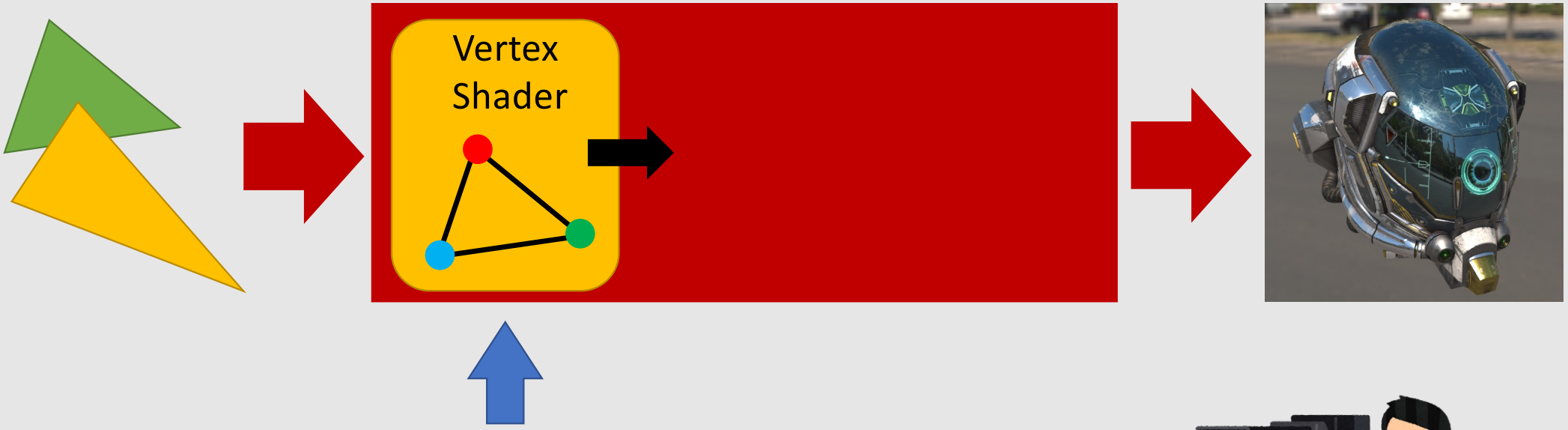
Graphics Pipeline



Graphics Pipeline: Three Main Components



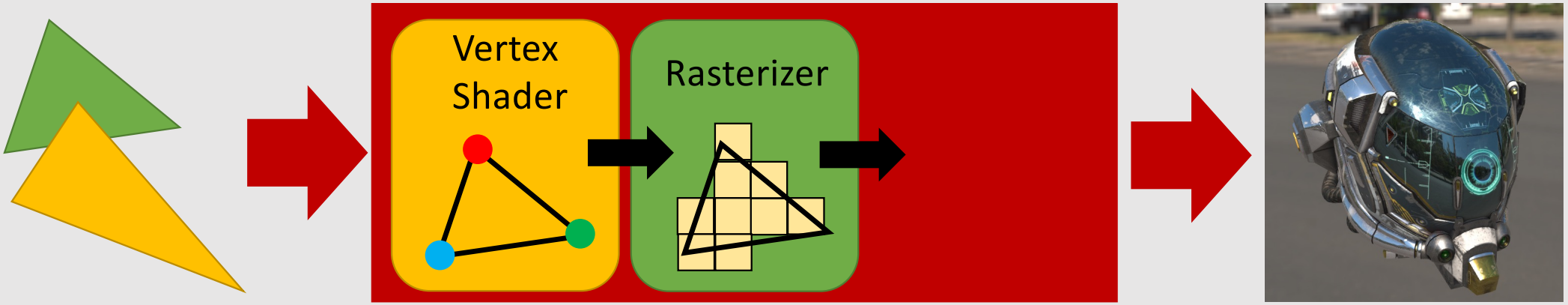
Graphics Pipeline: Vertex Shader

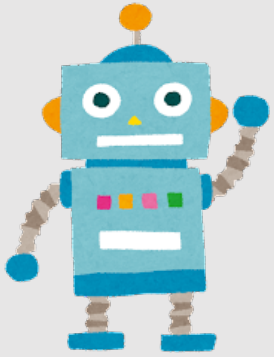


Thread per vertex

- View transformation…etc
- User can write a program

Graphics Pipeline: Rasterizer

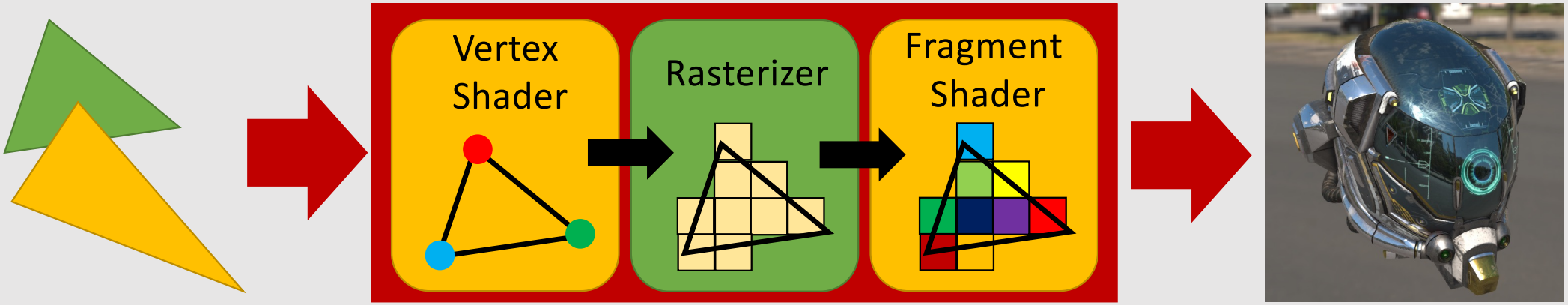




Pixels in the triangle are extracted

- Fully automatic

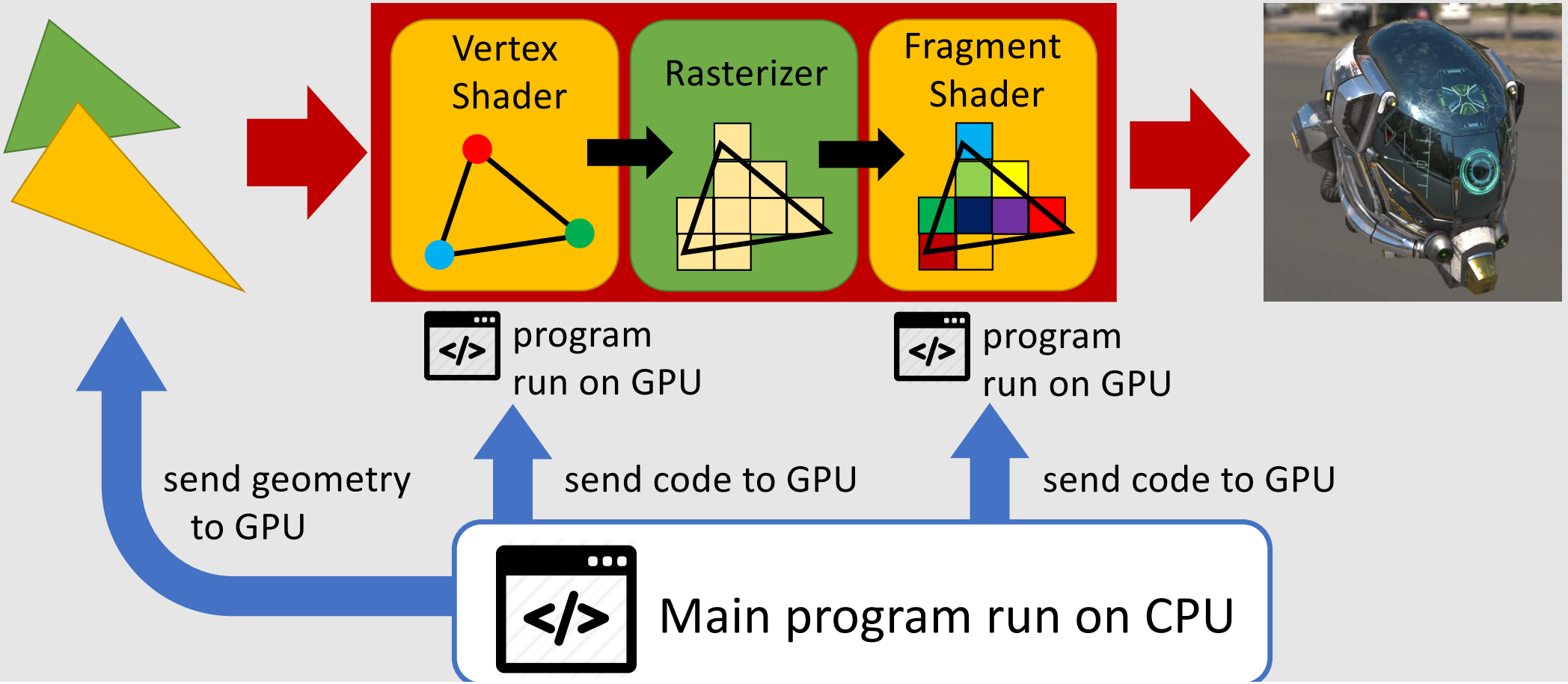
Graphics Pipeline: Fragment Shader



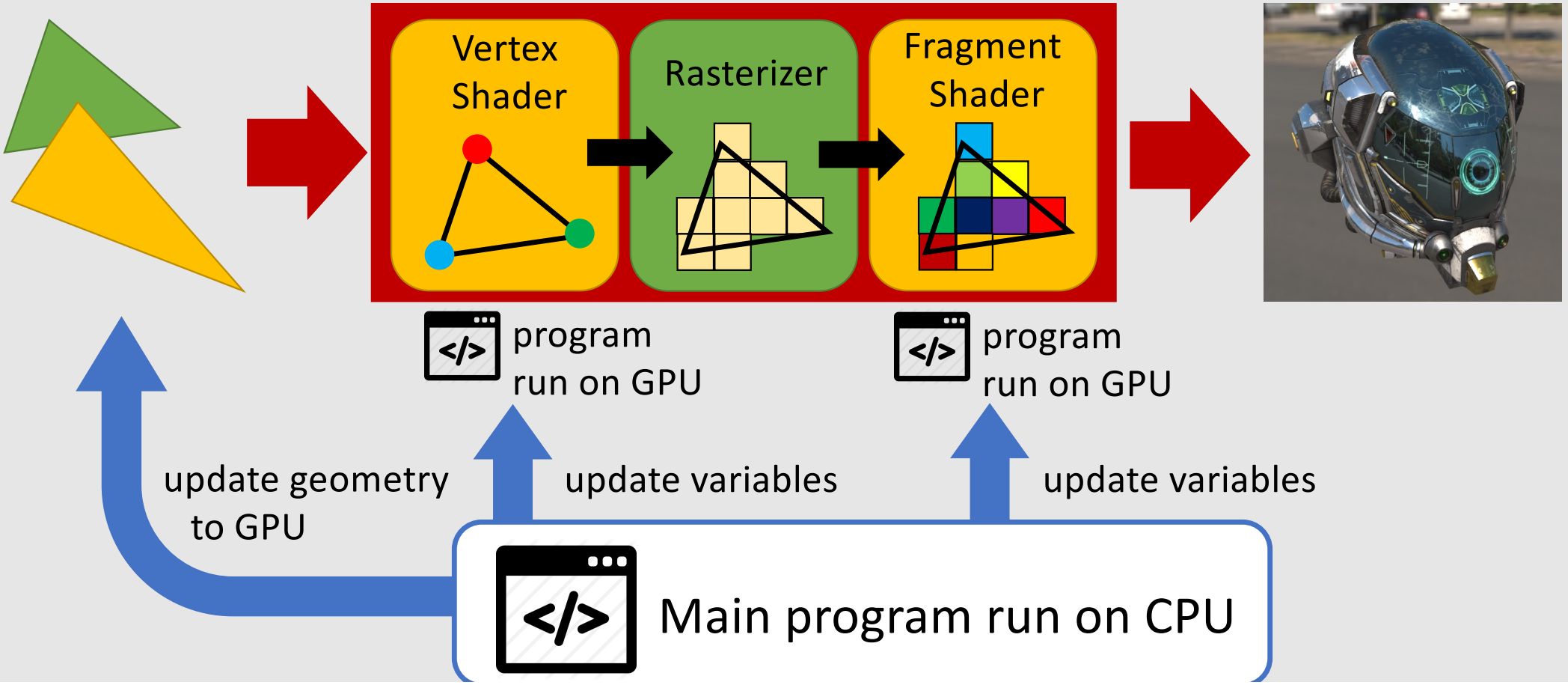
Thread par pixel

- Paint pixel w.r.t. lighting, material
- User can write a program

Graephics Pipeline: CPU & GPU



Graphics Pipeline: CPU & GPU

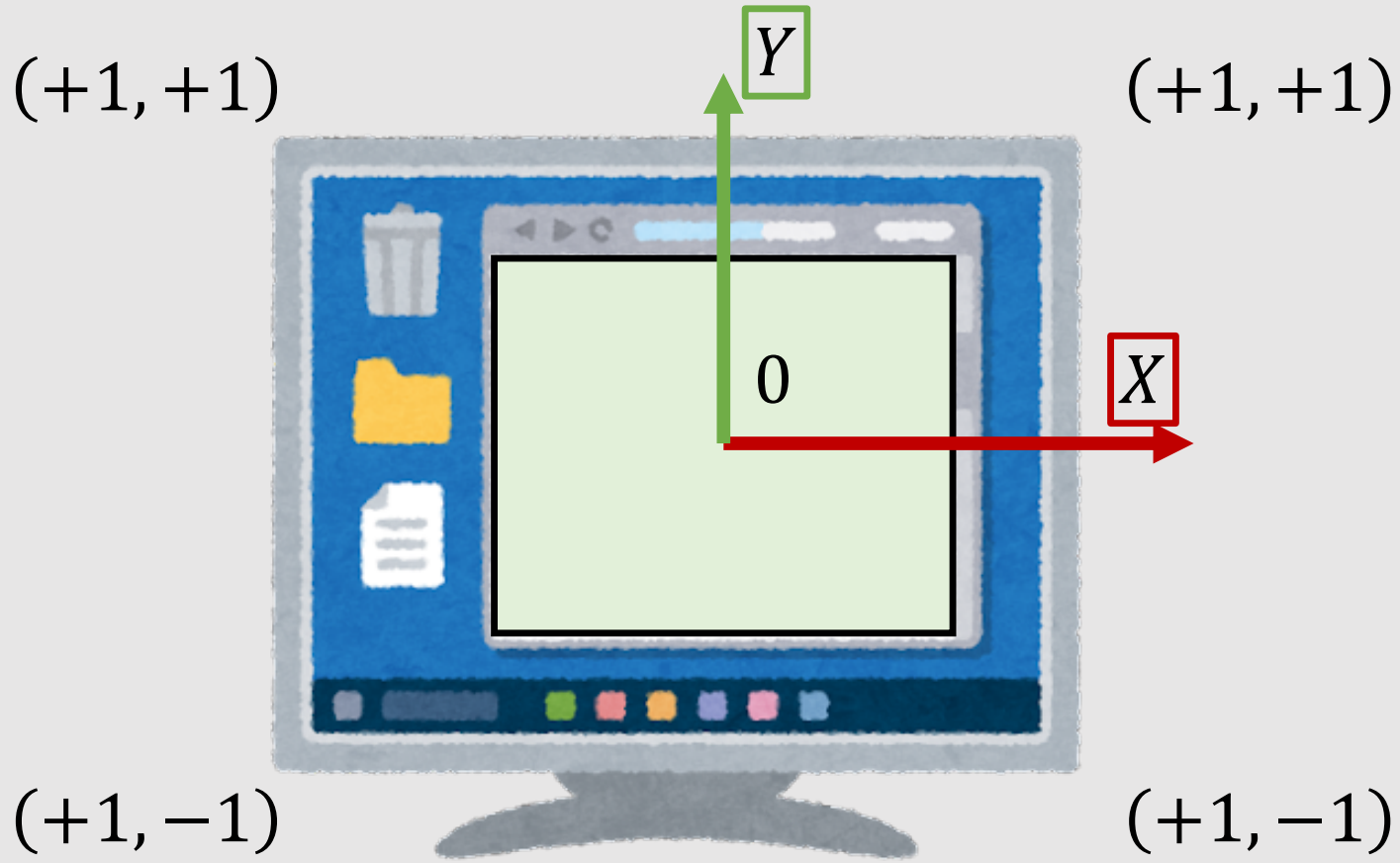


Graphics APIs

- OpenGL
- Vulkan
- DirectX
- Metal

Microsoft® DirectX®

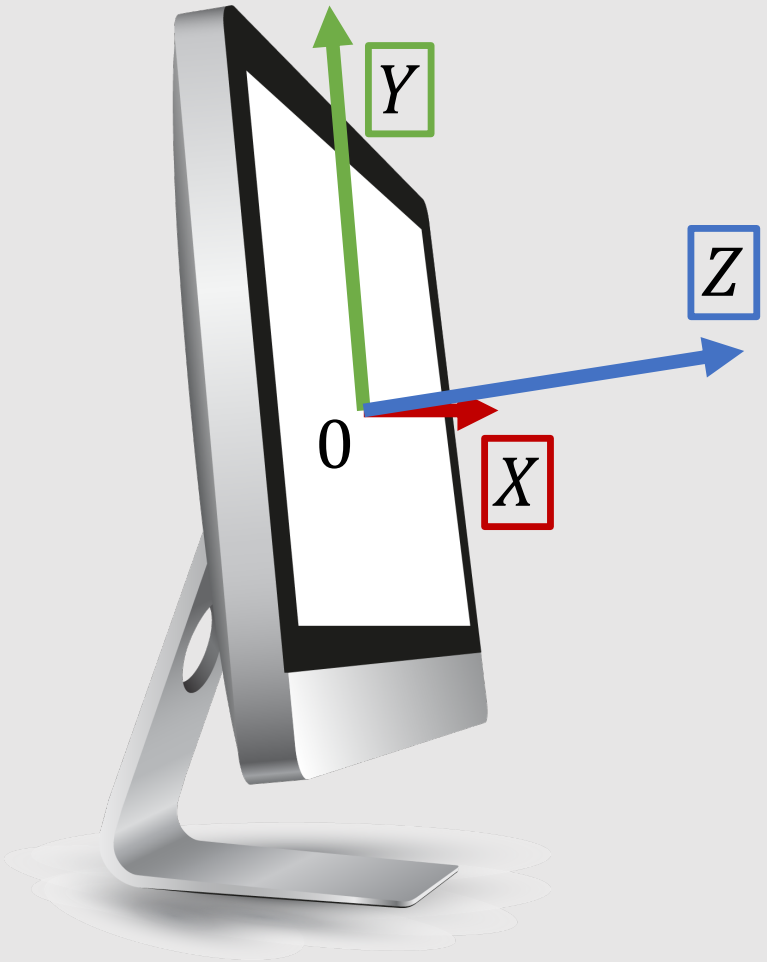
Coordinate on the Screen: [-1,+1]x[-1,+1]



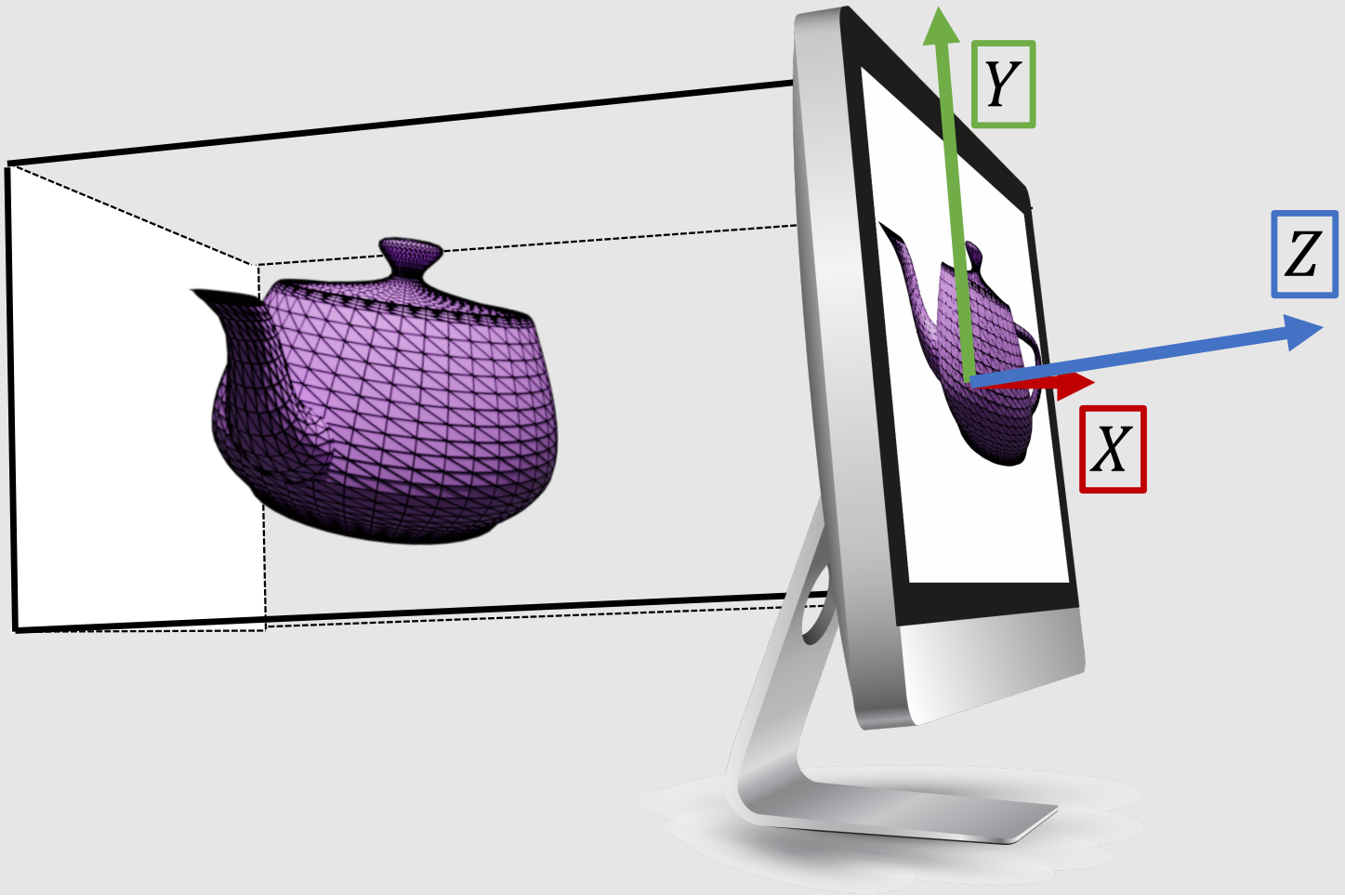
Z-Axis on the Screen

$$\vec{e}_X \times \vec{e}_Y = \vec{e}_Z$$

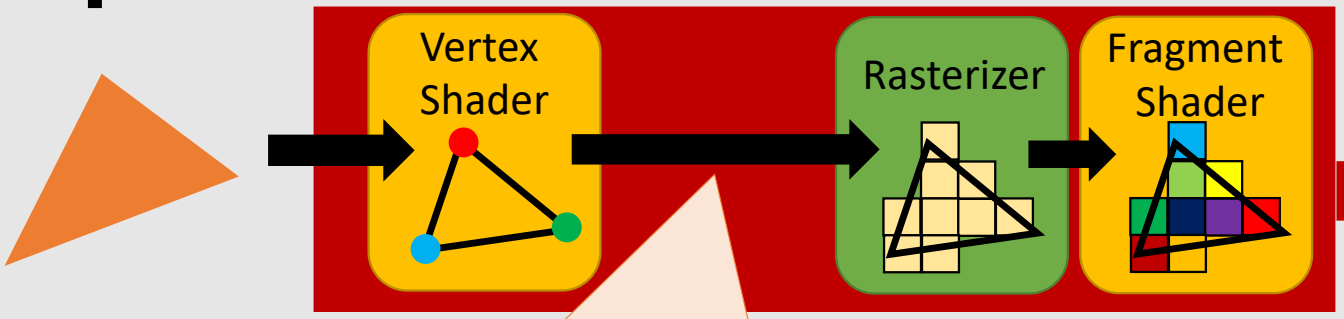
Going counter-clockwise and up!



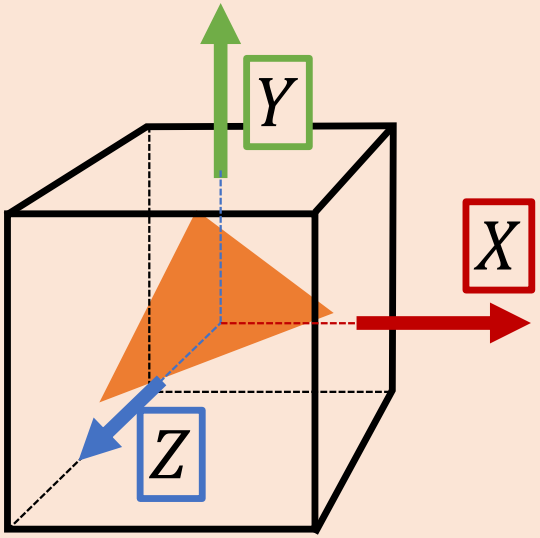
Projection to the Screen

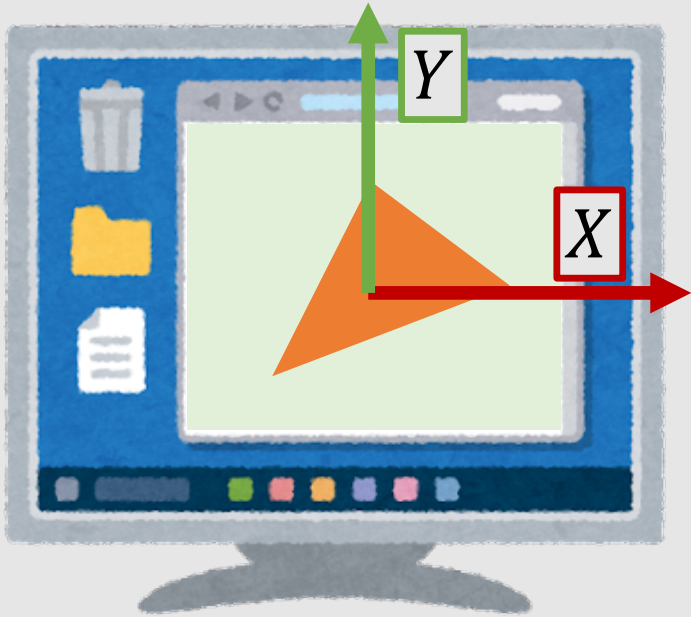


Input of the Rasterizer: Triangle in a Cube

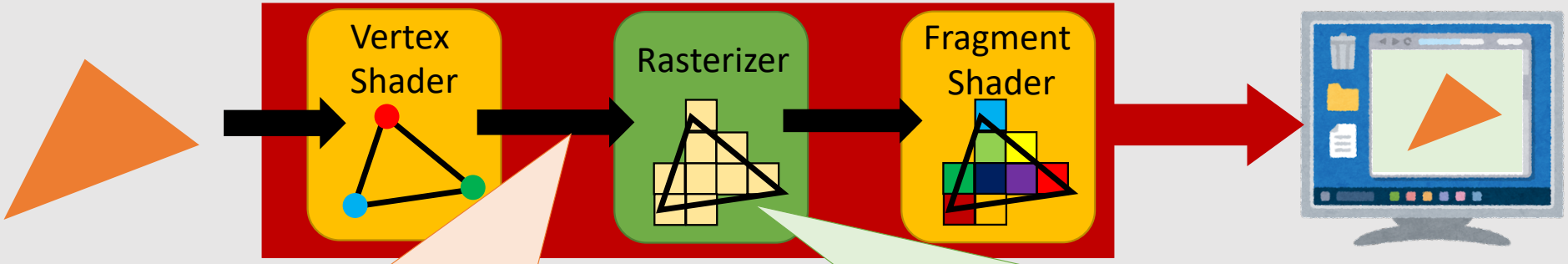


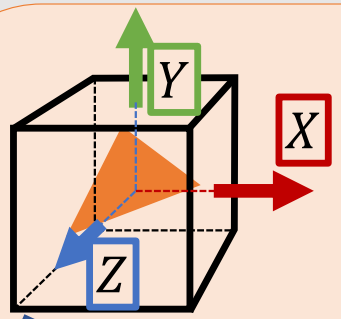
canonical view volume
[-1,+1]x[-1,+1]x[-1,+1]





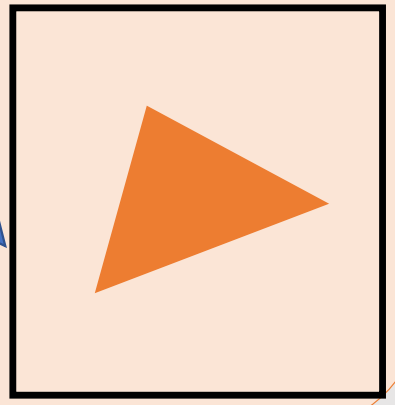
Rasterization



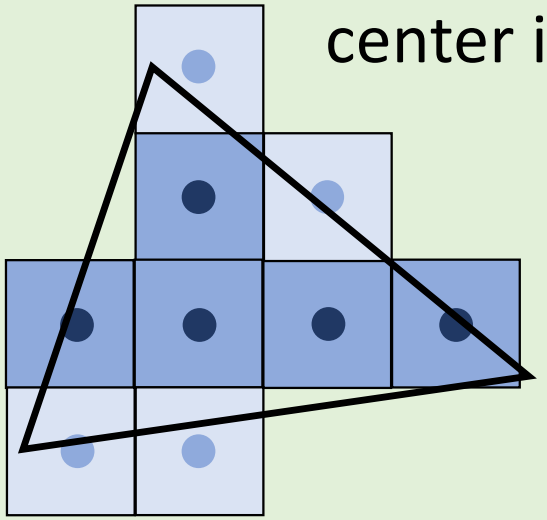


canonical view volume

Looking from $Z^{+\infty}$



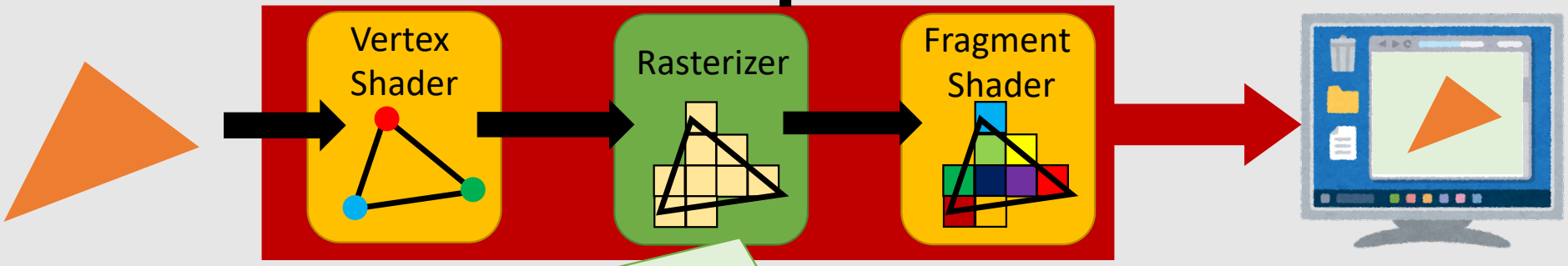
Extract the pixels whose center is inside the triangle

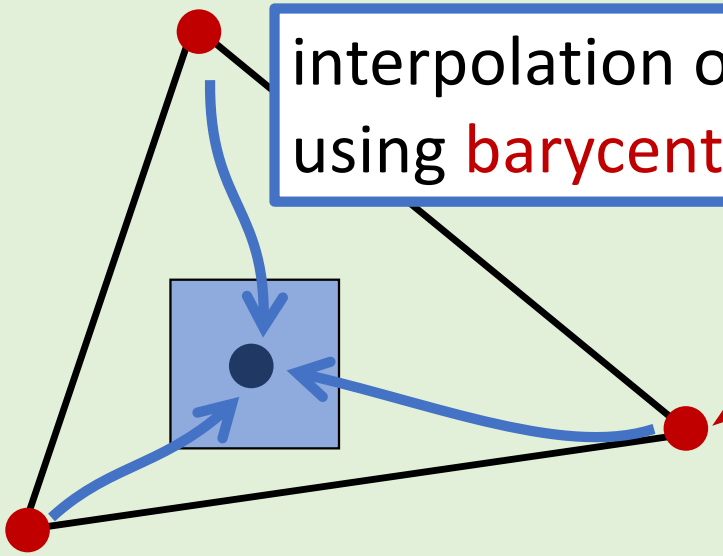


inside

outside

Rasterizer and Interpolation

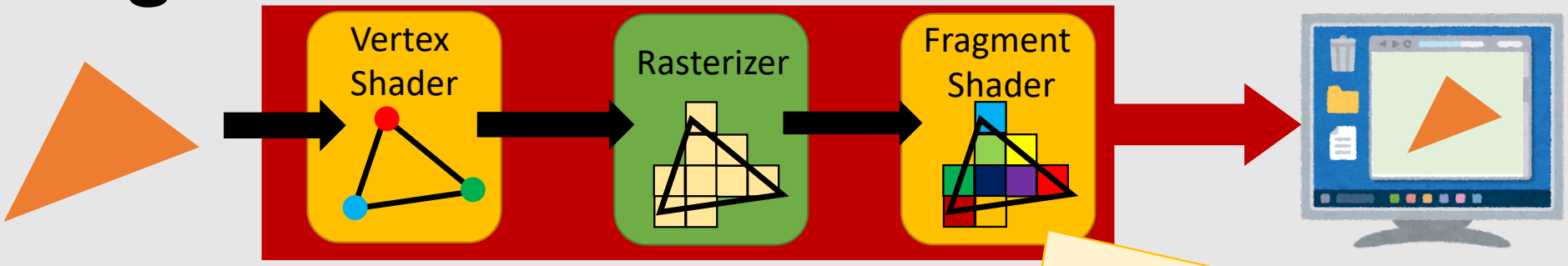




interpolation of vertex attributes
using barycentric coordinate

vertex attributes:
color, normal, UV-coordinate…etc

Fragment Shader



color, normal, UV-coordinate…etc

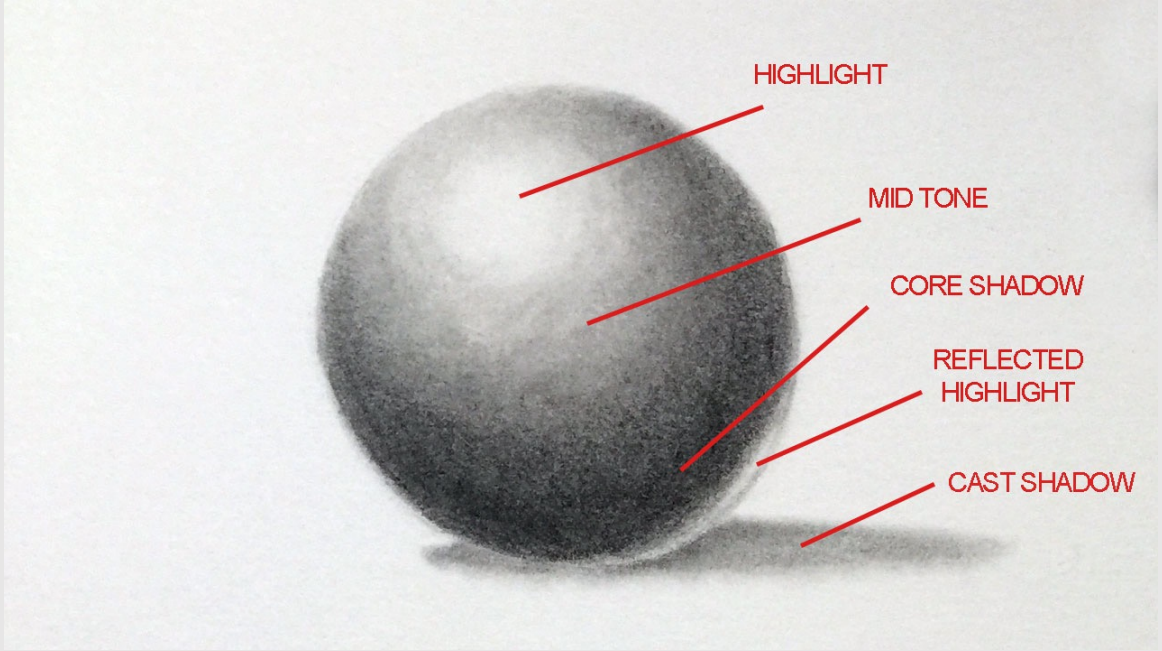
Fragment shader program

R: 15
G: 245
B: 20

Shading

What is Shading?

- Terminology in art
- Changing color w.r.t. material & lighting



Shading Techniques - How to Shade with a Pencil
https://thevirtualinstructor.com/shading-techniques-basics.html

Reflection ≈ Ambient + Diffuse + Specular

- Rough approximation of actual reflection

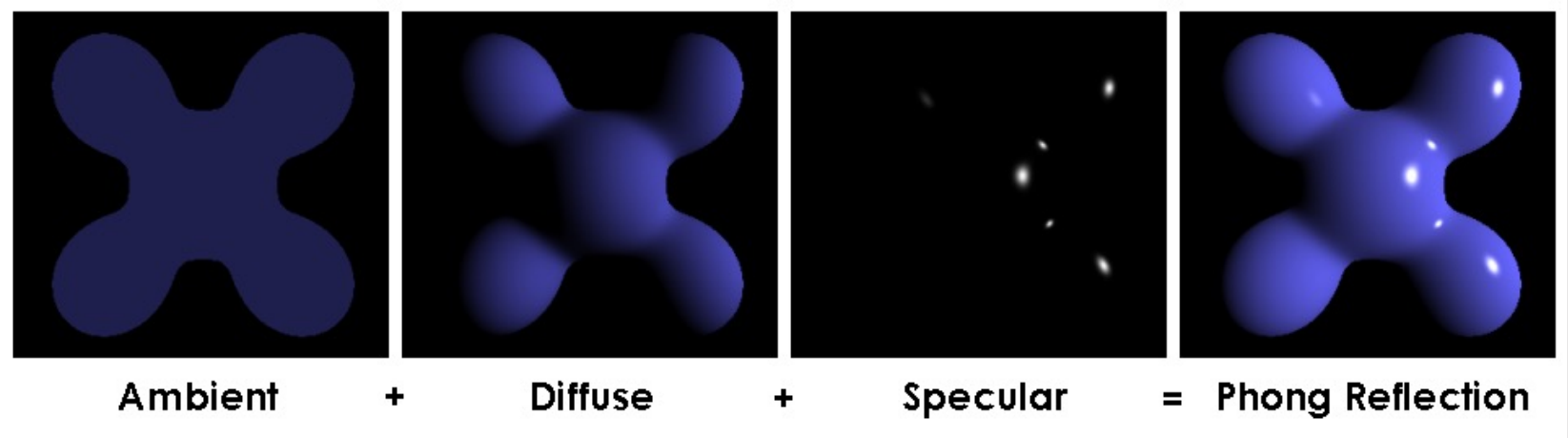
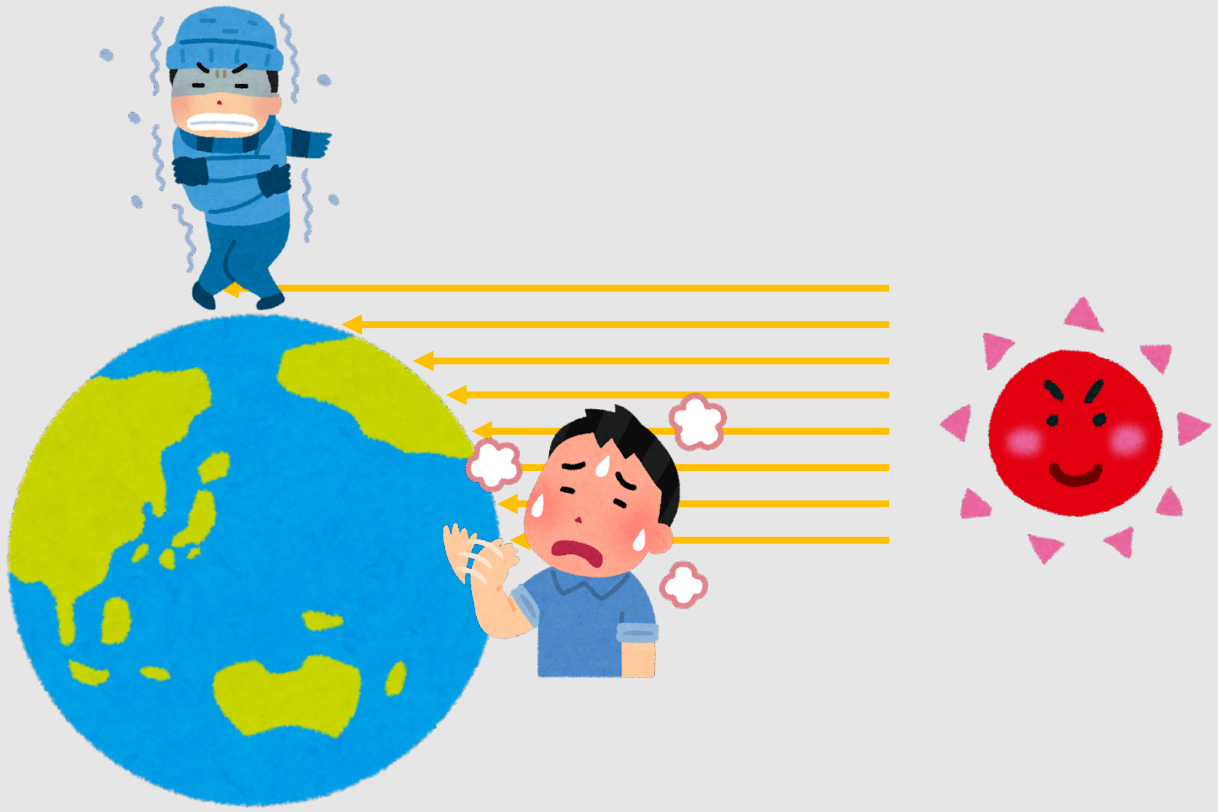


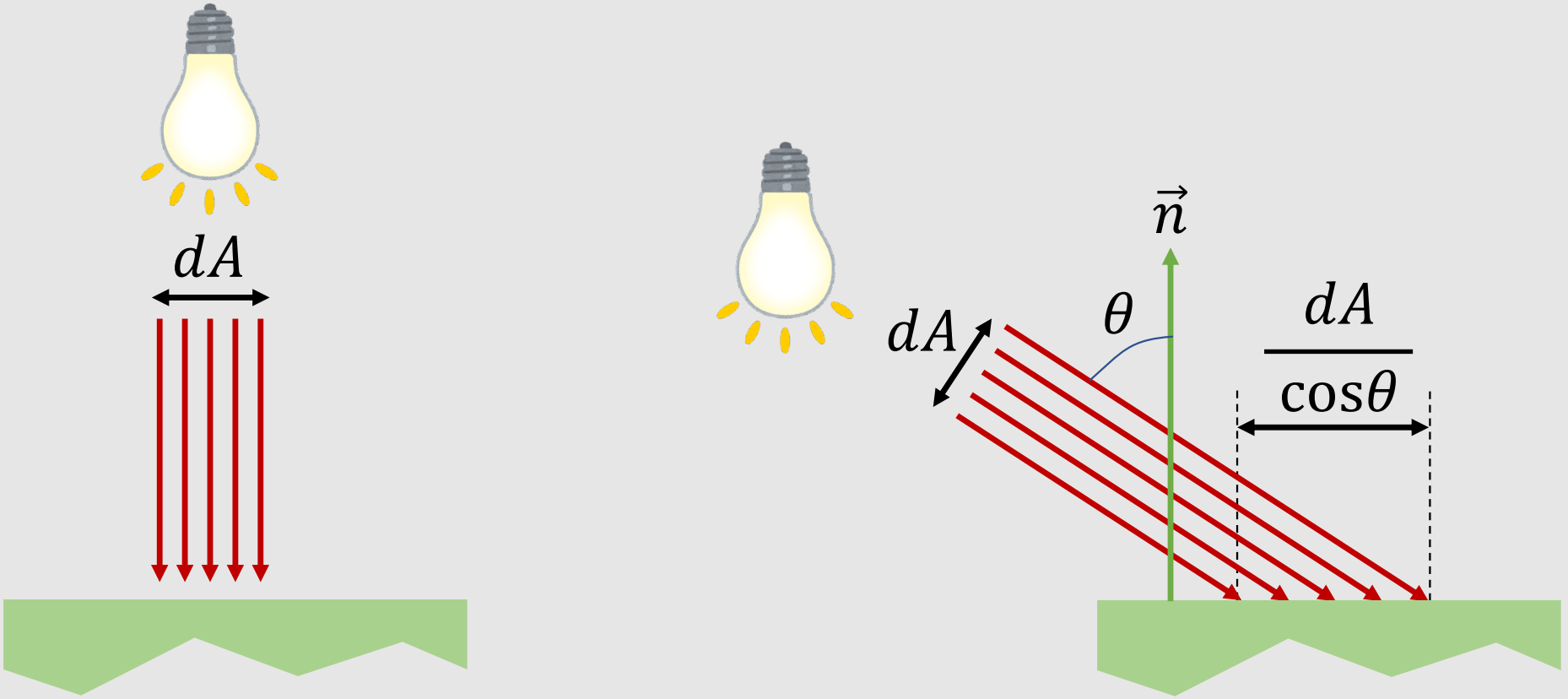
Image credit: Brad Smith @ Wikipedia

Light Weaken on the Oblique Surface



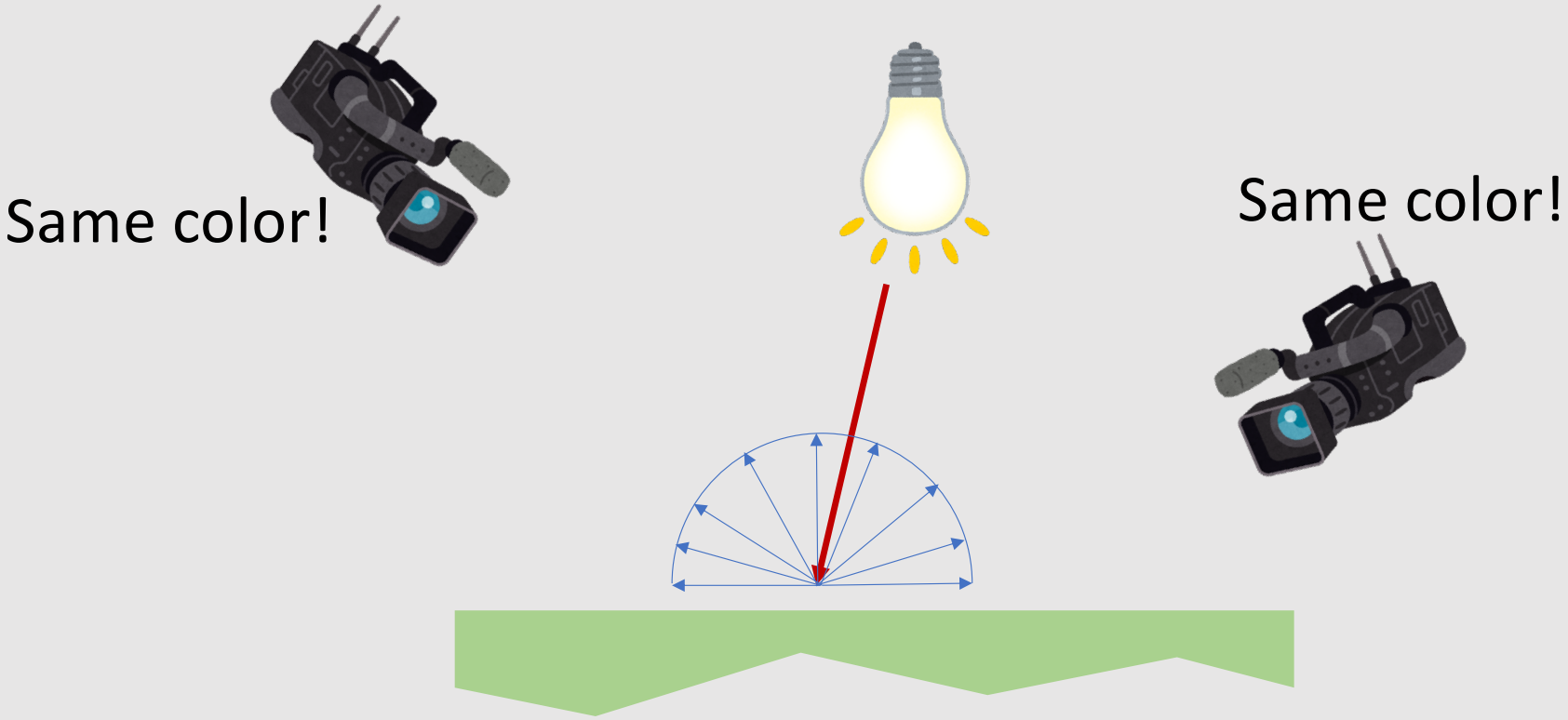
Amount of Incoming Light

- Propotional to the cosine of incident angle θ



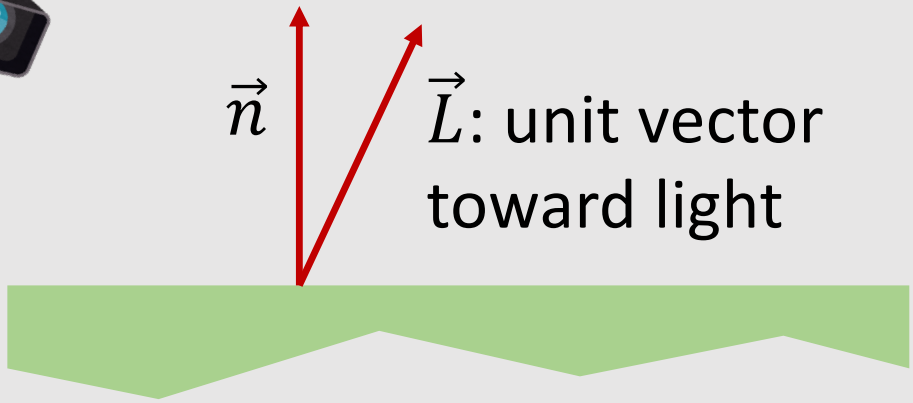
Lambertian Diffusion

- Same brightness regardless of the viewing angle



Lambertian Diffusion

$\vec{c}$: output color



i_d: intensity of light

$$\vec{c} = \vec{k}_d \left(\vec{L} \cdot \vec{n}\right) \mathrm{i_d}$$
$$= \cos\theta$$

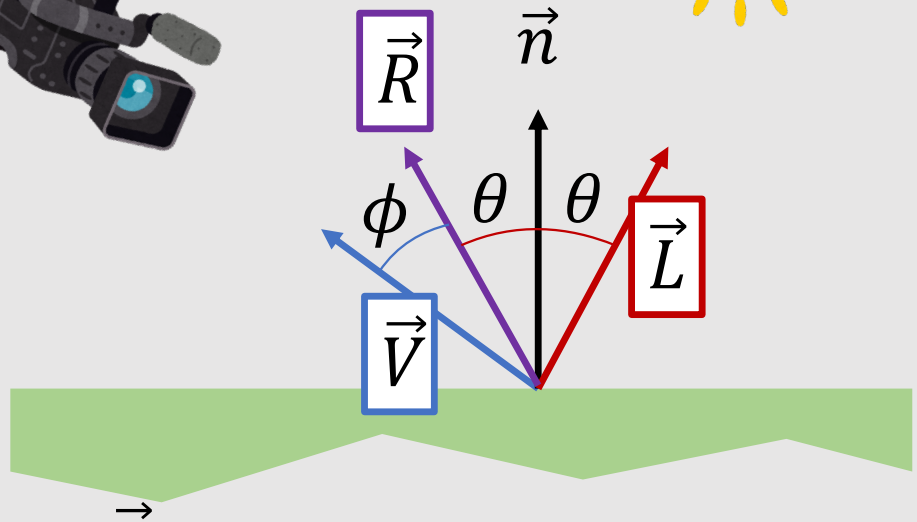
Specular in Phong Shading

$\vec{L}$: unit vector toward light

$\vec{R}$: unit vector perfect reflection

$\vec{V}$: unit vector toward view

$\vec{c}$: output color



$\vec{k}_s$: specular reflection constant

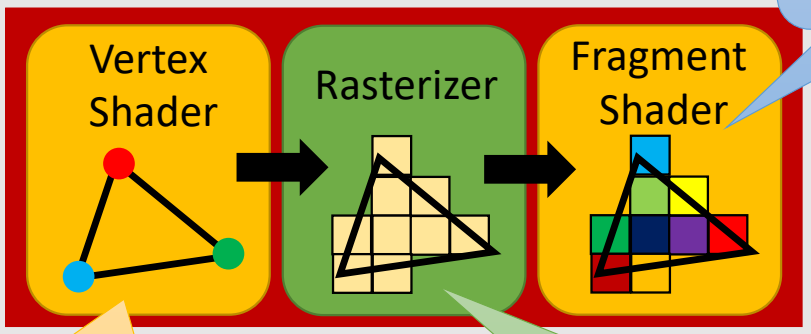
α: shininess constant

$$\vec{c} = \vec{k}_s \left(\vec{R} \cdot \vec{V}\right)^{\alpha} \mathrm{i_s}$$
$$= \cos\phi$$

i_s: intensity of light

Shading Methods

Gourad shading

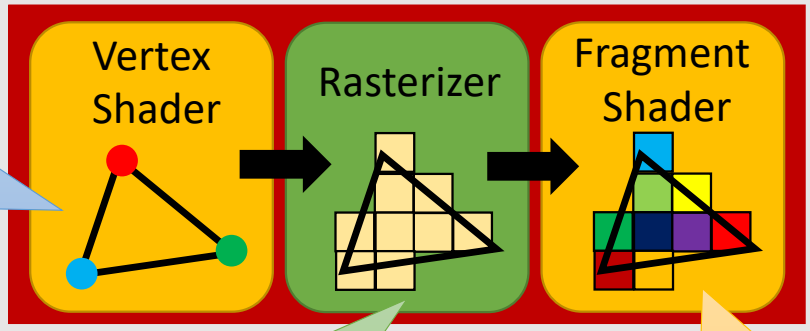


do nothing

computing color at vertex

color interpolation

Phong shading



do nothing

normal interpolation

computing color here

☺ Light weight computation

☹ Color is linear over triangle

Shading Methods

Flat shading | Gourad shading | Phong shading

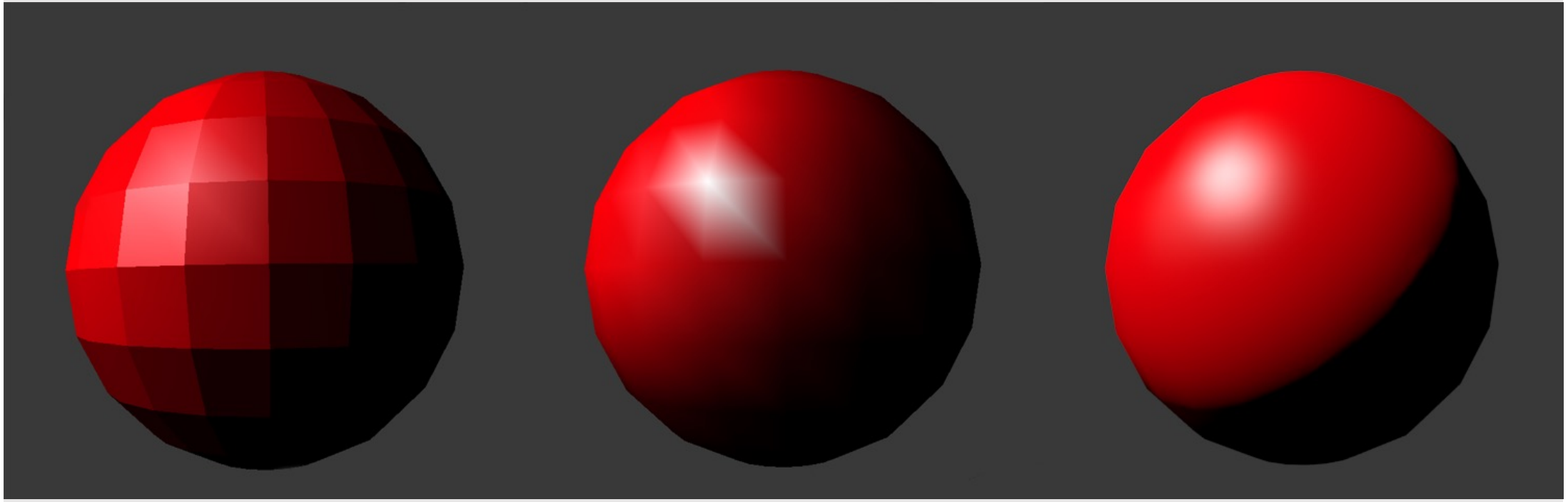
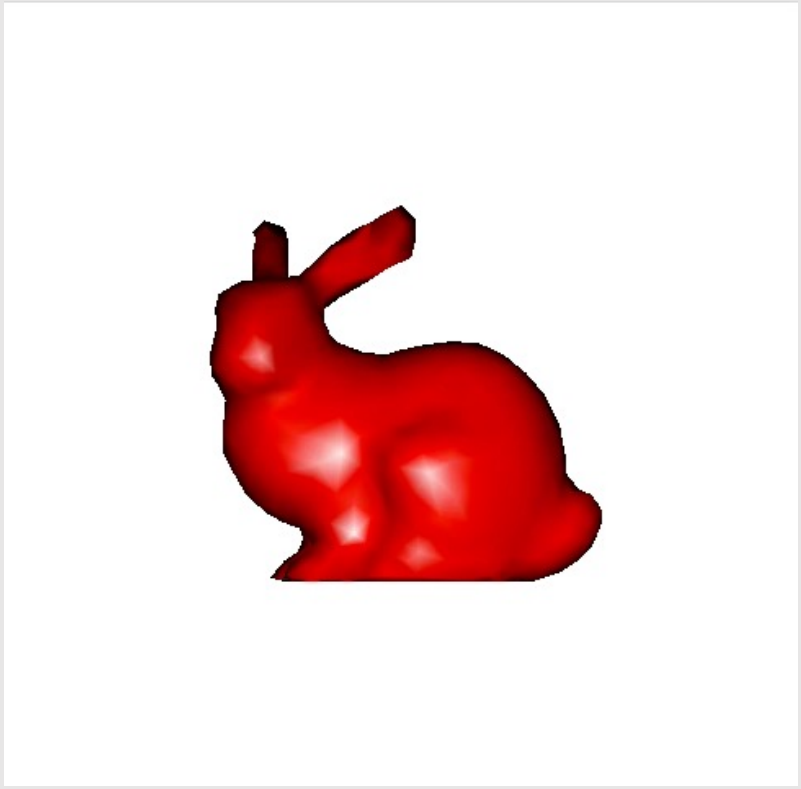


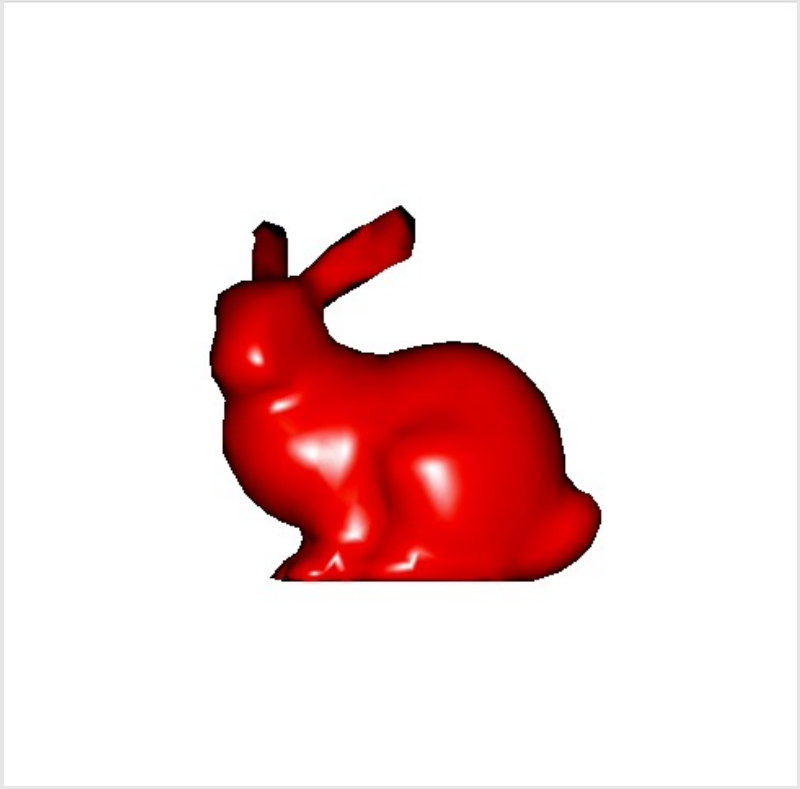
Image credit: Davi.trip @ Wikipedia

Shading Methods

Gourad shading



Phong shading



More Topics

- Coordinate transformation
- Visibility (occlusion)
- Shadow
- Efficient rasterization
- Anti-aliasing