Game Rendering

Ken-Yi Lee

Game Programming, Fall 2020 @ National Taiwan University

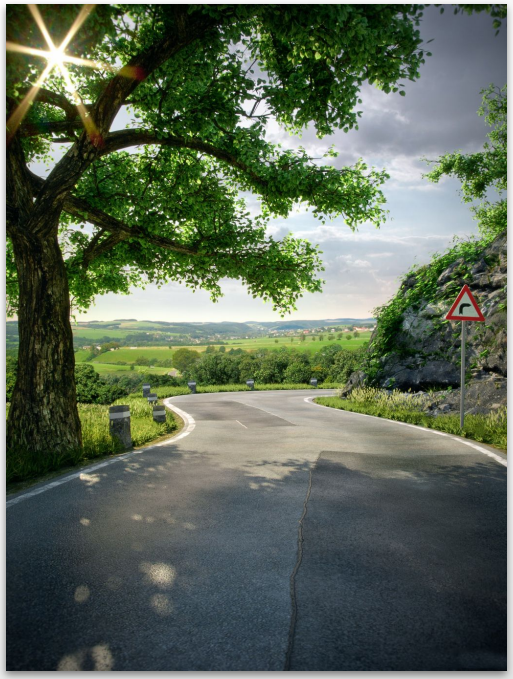

DEVOTION
https://shop.redcandlegames.com/#game

Game Rendering

- 3D or 2D ?

Game Rendering

- 3D or 2D ?
- Realistic ?
  - Photorealistic rendering
    - Physically-based rendering

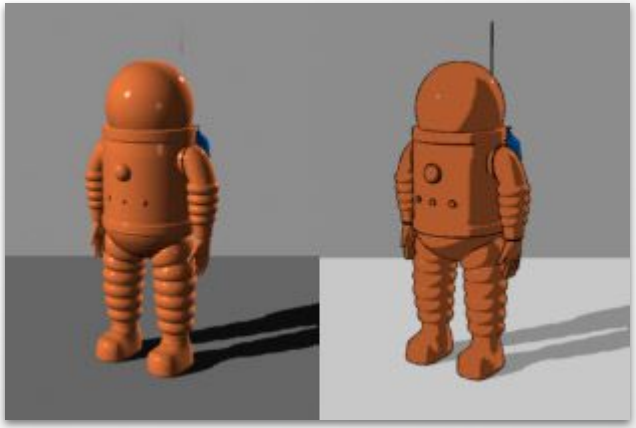


https://en.wikipedia.org/wiki/Unbiased_rendering

FeisStu

Game Rendering

- 3D or 2D ?
- Realistic ?
  - Photorealistic rendering
    - Physically-based rendering
  - Non-photorealistic rendering



https://en.wikipedia.org/wiki/Non-photorealistic_rendering

FeisStu

Game Rendering

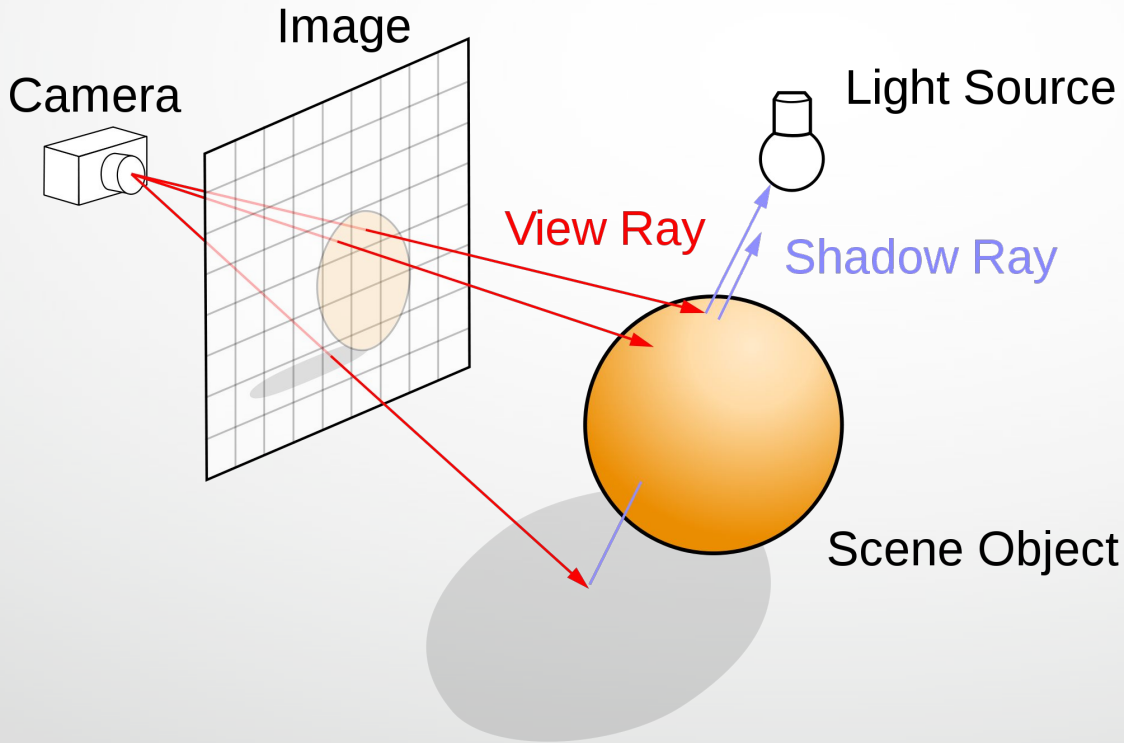
- 3D or 2D ?
- Realistic ?
  - Photorealistic rendering
    - Physically-based rendering
  - Non-photorealistic rendering
- Real-time ?
  - Performance, performance, and performance


Unreal Paris 2020 on Steam
https://store.steampowered.com/app/1280890/Unreal_Paris_2020/


Unreal Paris 2020 on Steam
https://store.steampowered.com/app/1280890/Unreal_Paris_2020/


Unreal Paris 2020 on Steam
https://store.steampowered.com/app/1280890/Unreal_Paris_2020/

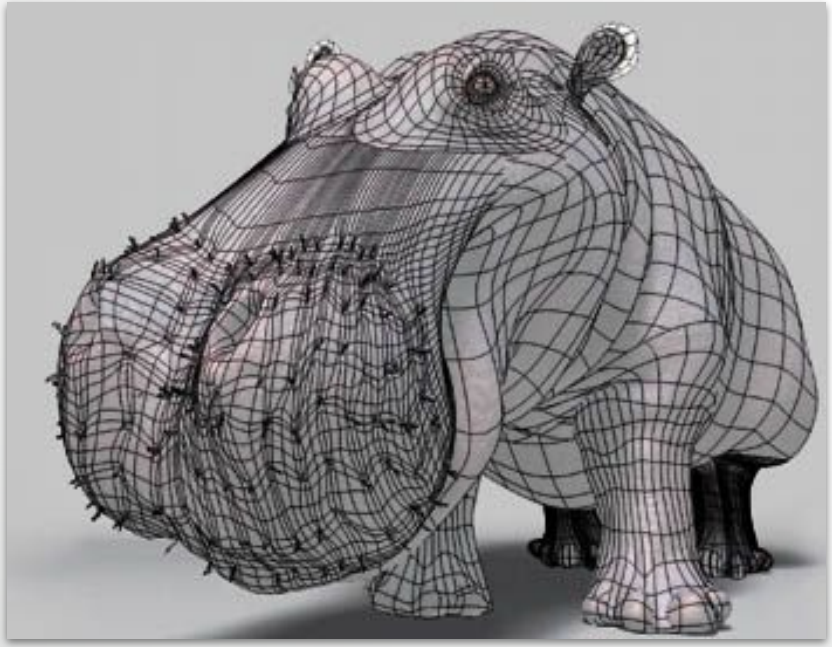
Ray tracing



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Model



http://3drender.com/jbirn/hippo/hairyhipponose.html

Mesh

- Vertices
  - Positions, Normals, UVs, ...
- Faces
  - Triangles or Quads ?

FeisStu

Graphics hardware

Hardware

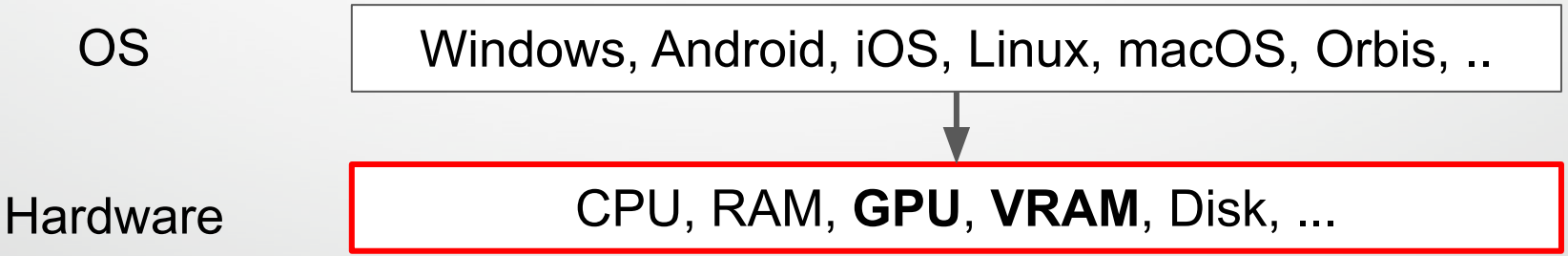
CPU, RAM, GPU, VRAM, Disk, ...

Graphics hardware

Hardware

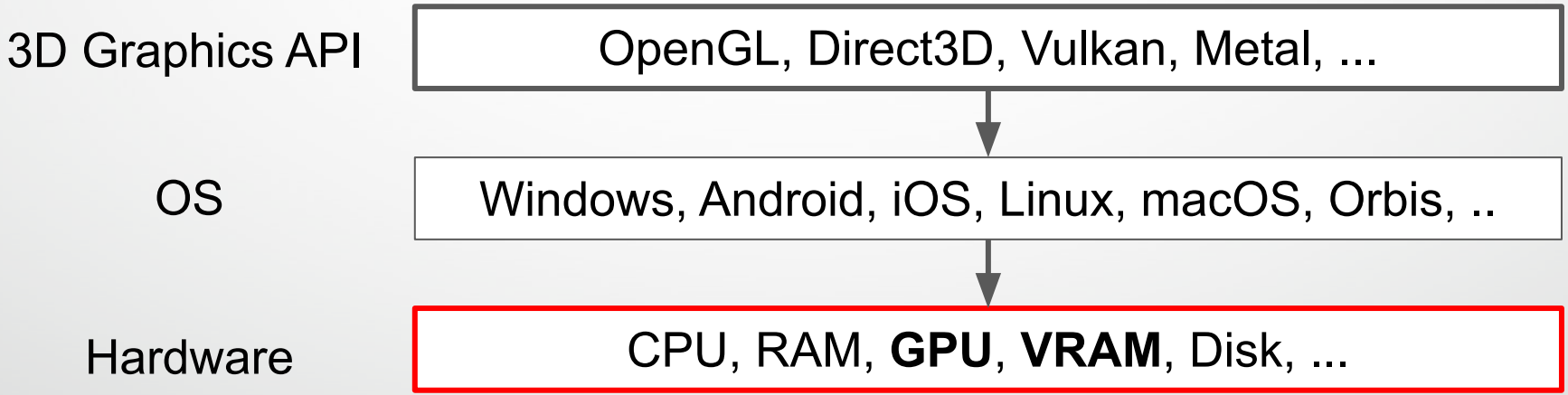
CPU, RAM, **GPU**, **VRAM**, Disk, **...**

Graphics hardware



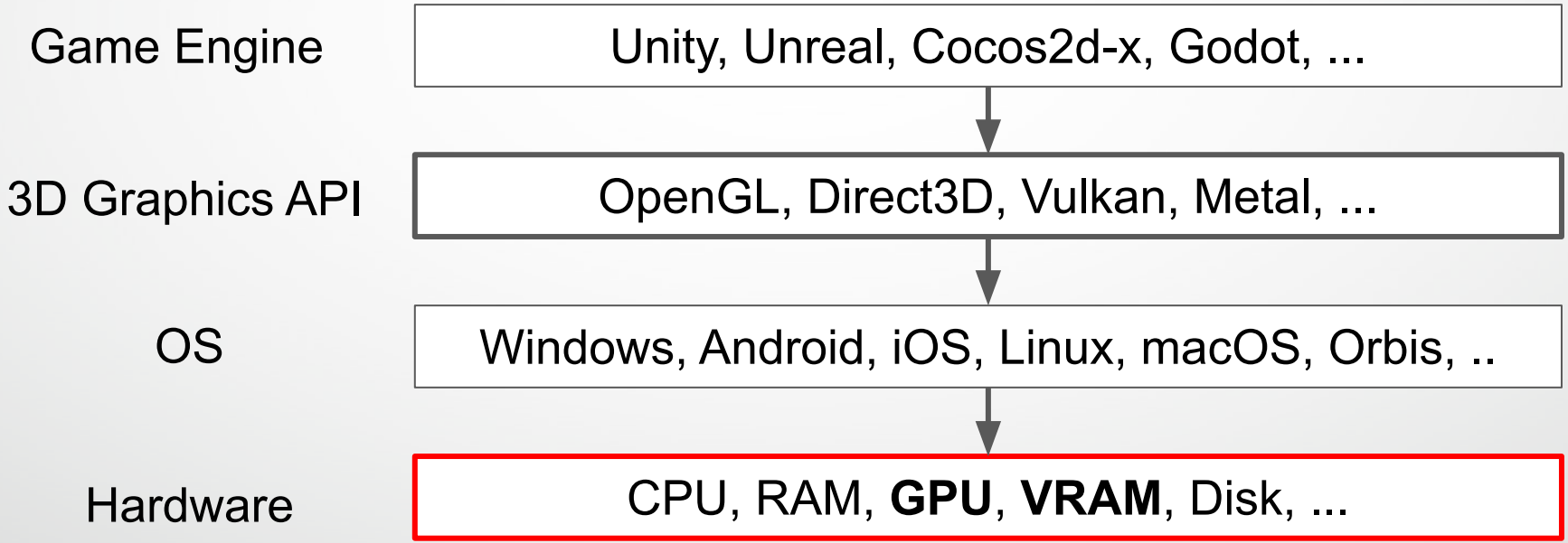
FeisStu

Graphics hardware



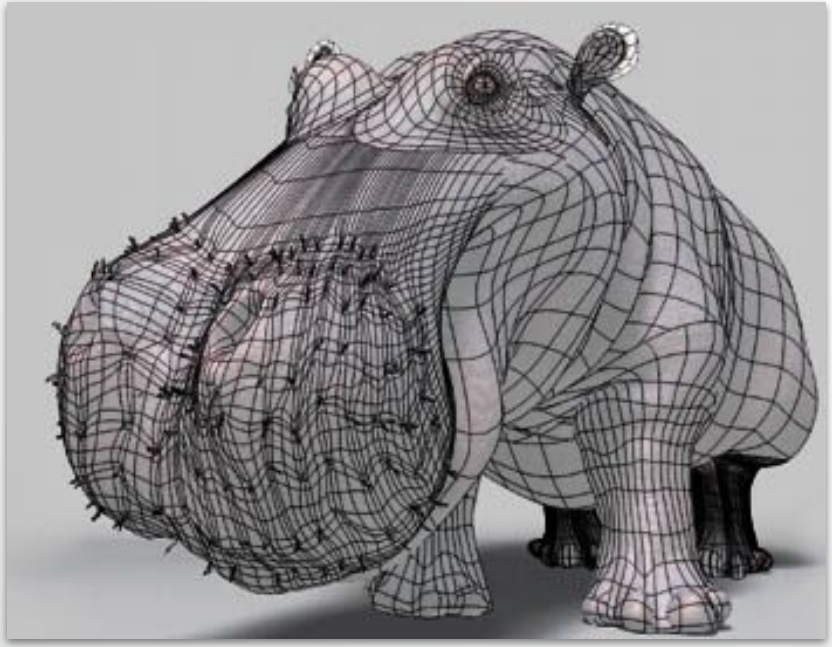
FeisStu

Graphics hardware



FeisStu

Model



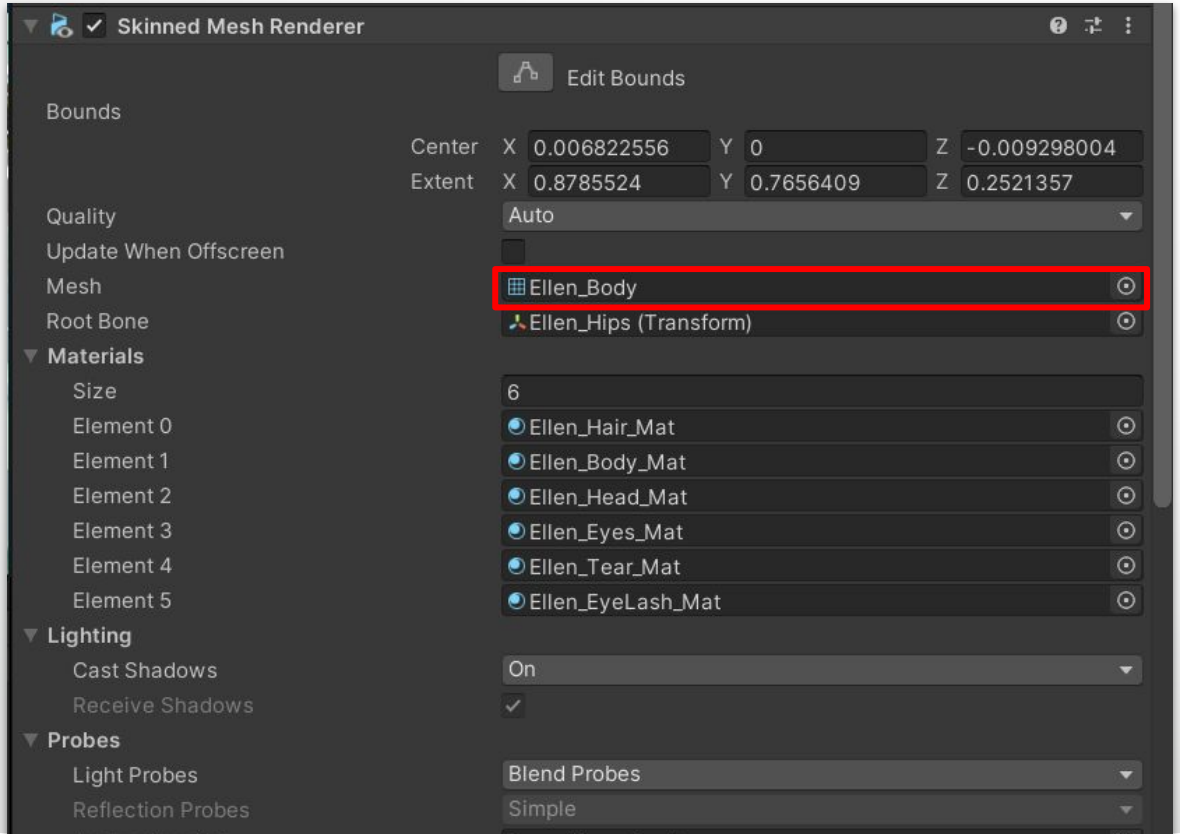
http://3drender.com/jbirn/hippo/hairyhipponose.html

Mesh

- Vertices
  - Positions, Normals, UVs, ...
- Faces
  - Triangles (hardware)
  - Quads (artist)

FeisStu

Mesh Filter / Mesh Renderer



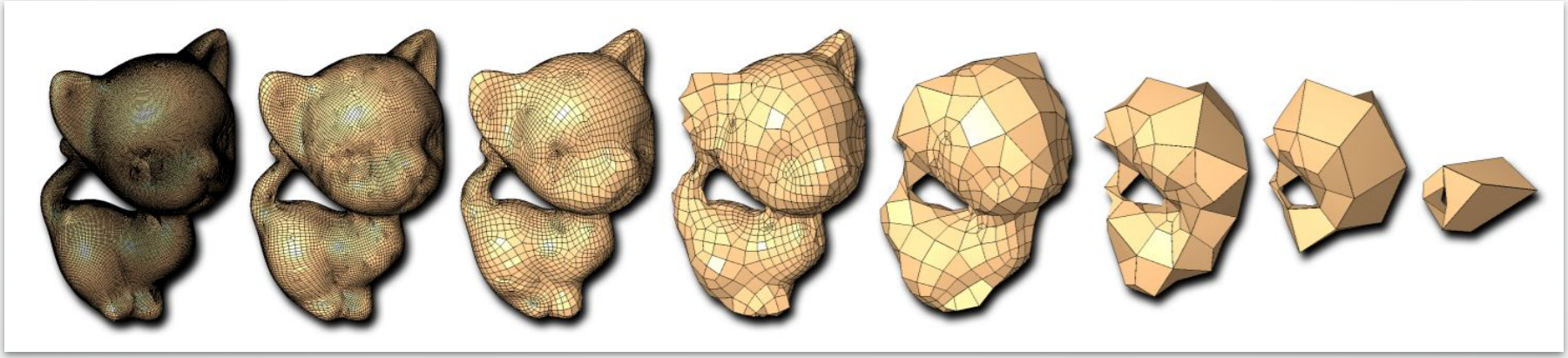
FeisStu

"How many vertices / faces can we use ? "

https://www.zbrushcentral.com/t/headshot-series/360990

FeisStu

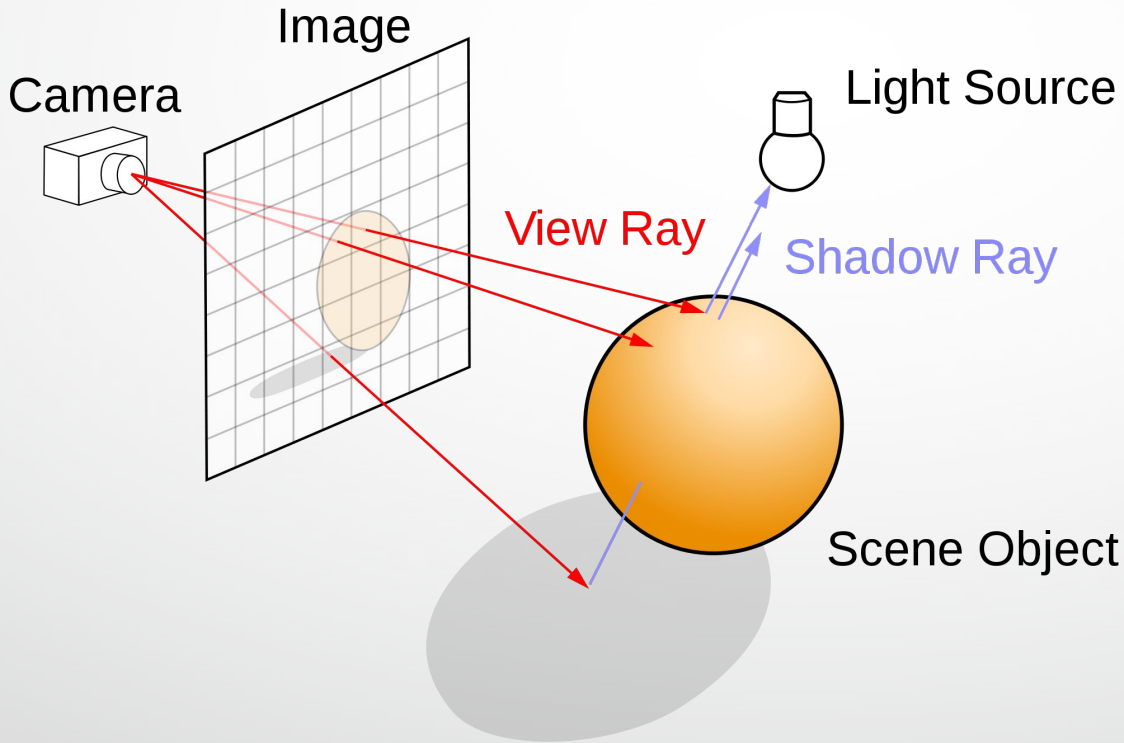
Mesh simplification



https://vgc.poly.edu/~csilva/papers/siggraph-asia2008.pdf

FeisStu

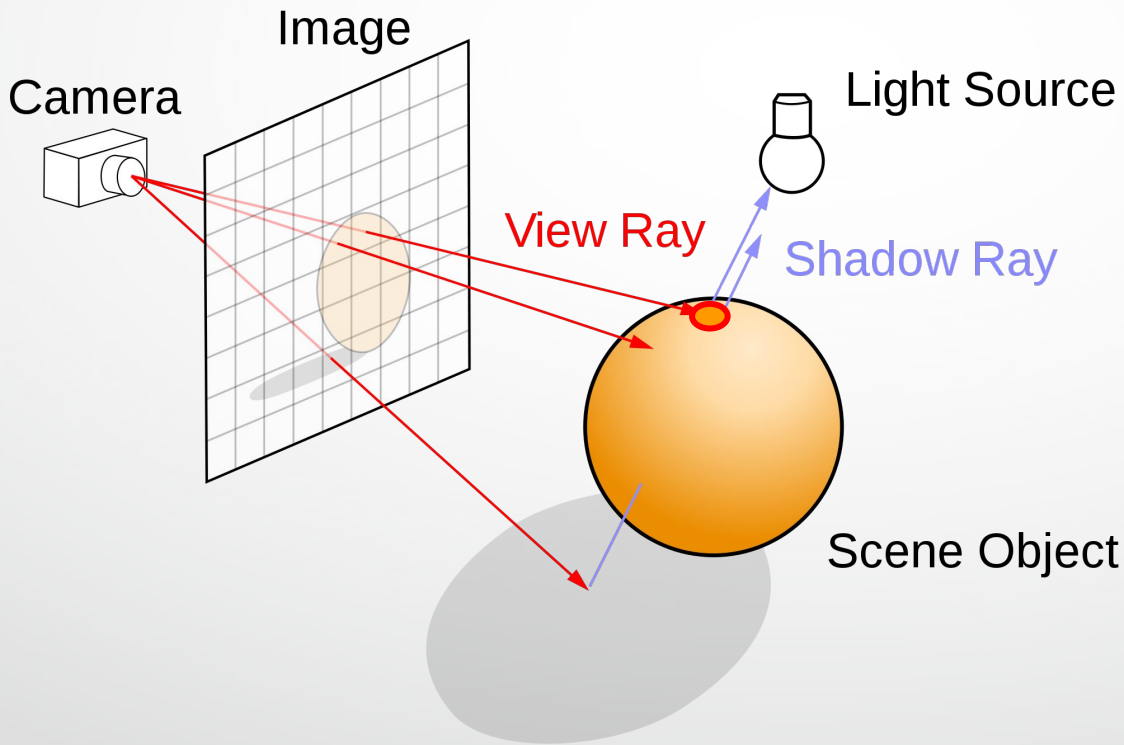
Ray tracing



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

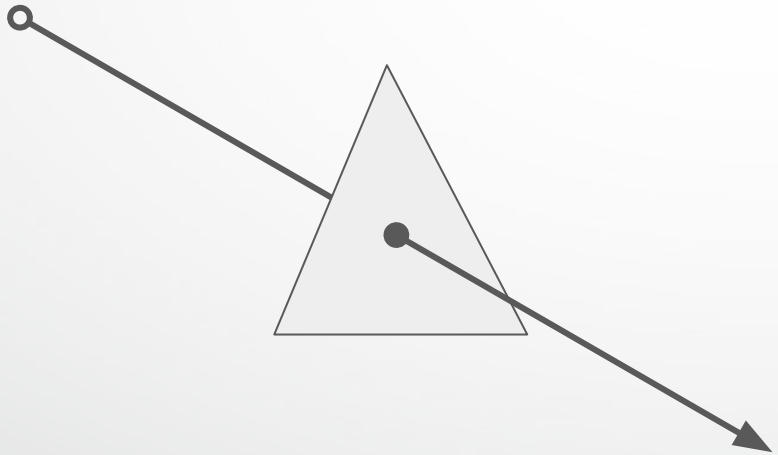
Ray-triangle intersection



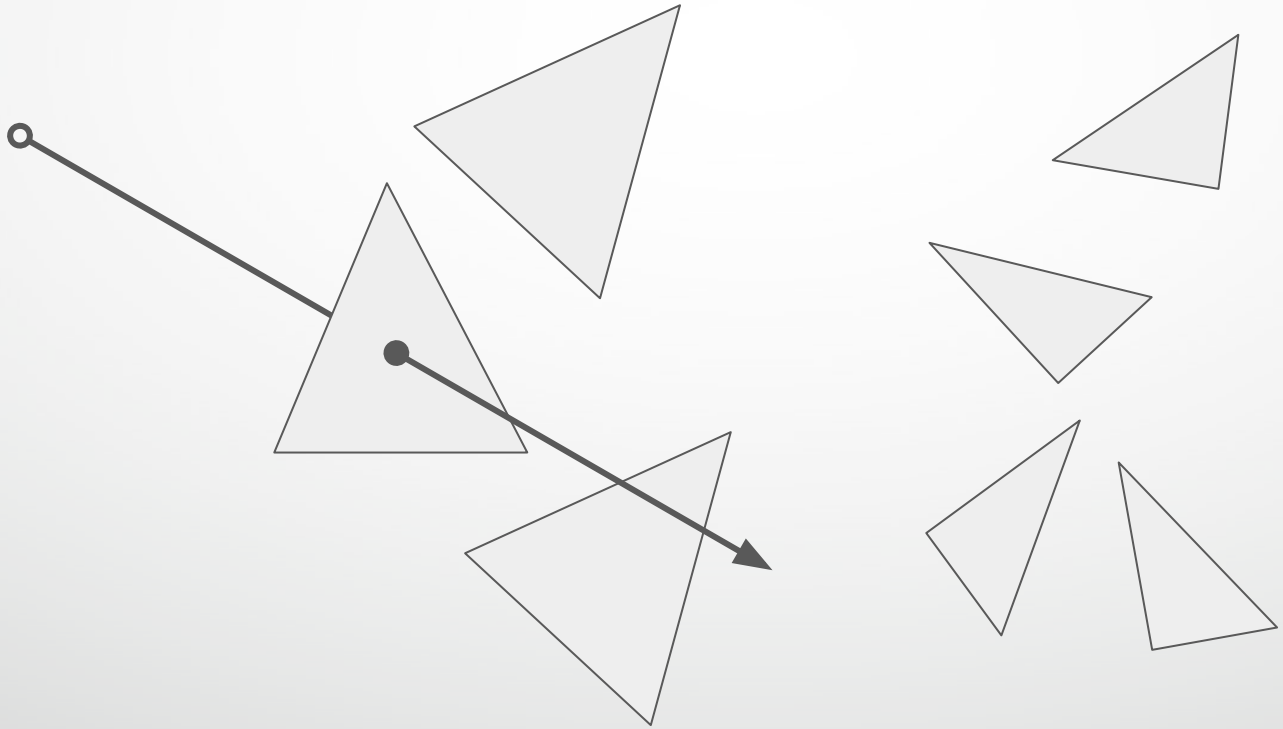
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

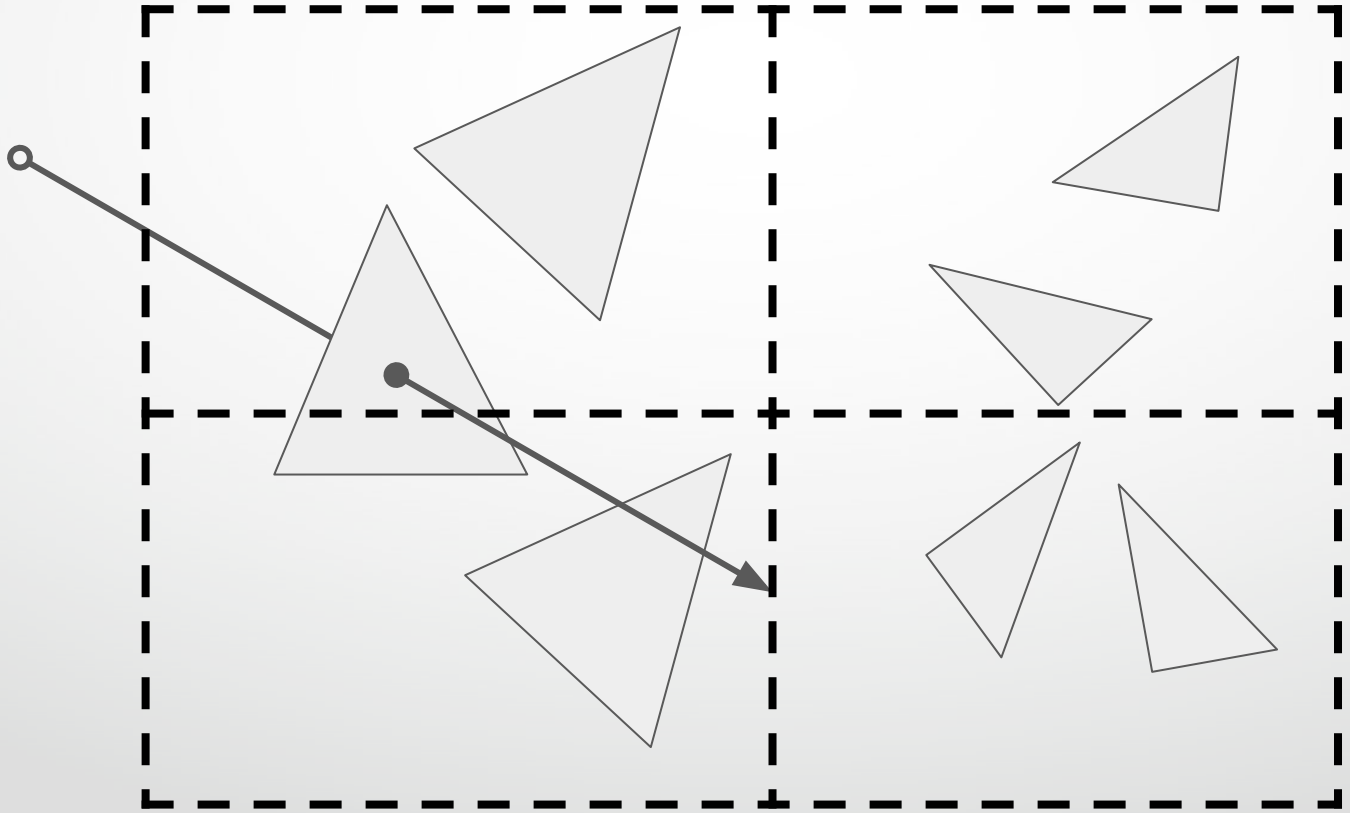
Ray-triangle intersection (cont'd)



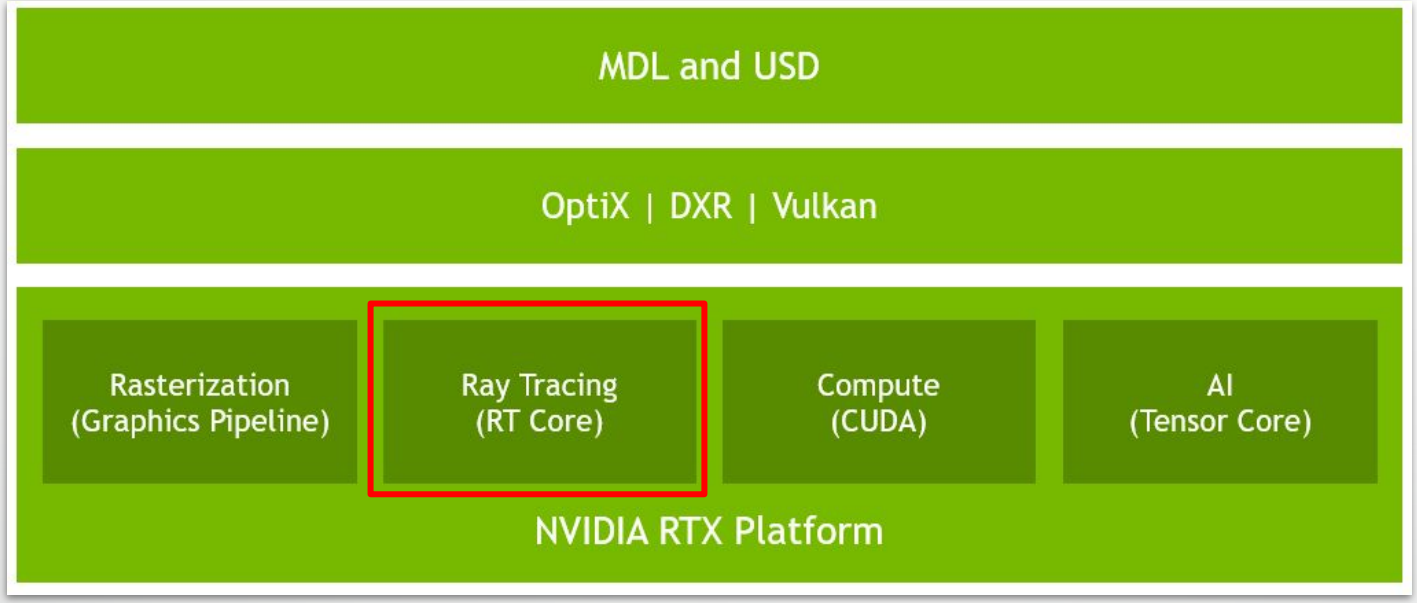
Ray-triangle intersection (cont'd)



Space-partitioning data structures



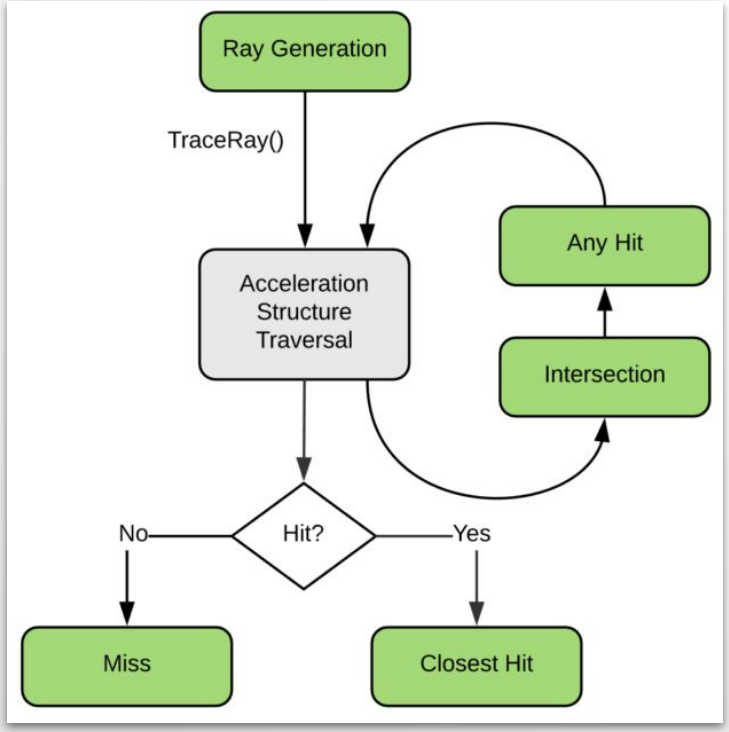
Case study: NVIDIA RTX™ Platform

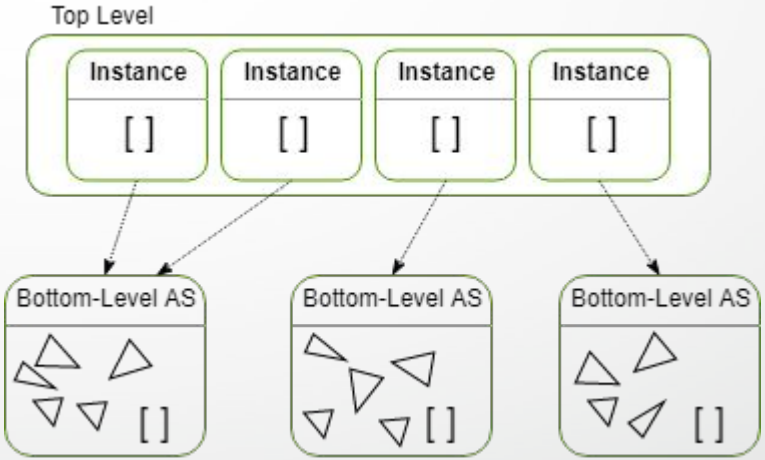


https://developer.nvidia.com/rtx

FeisStu

Case study: DXR (DirectX Raytracing)

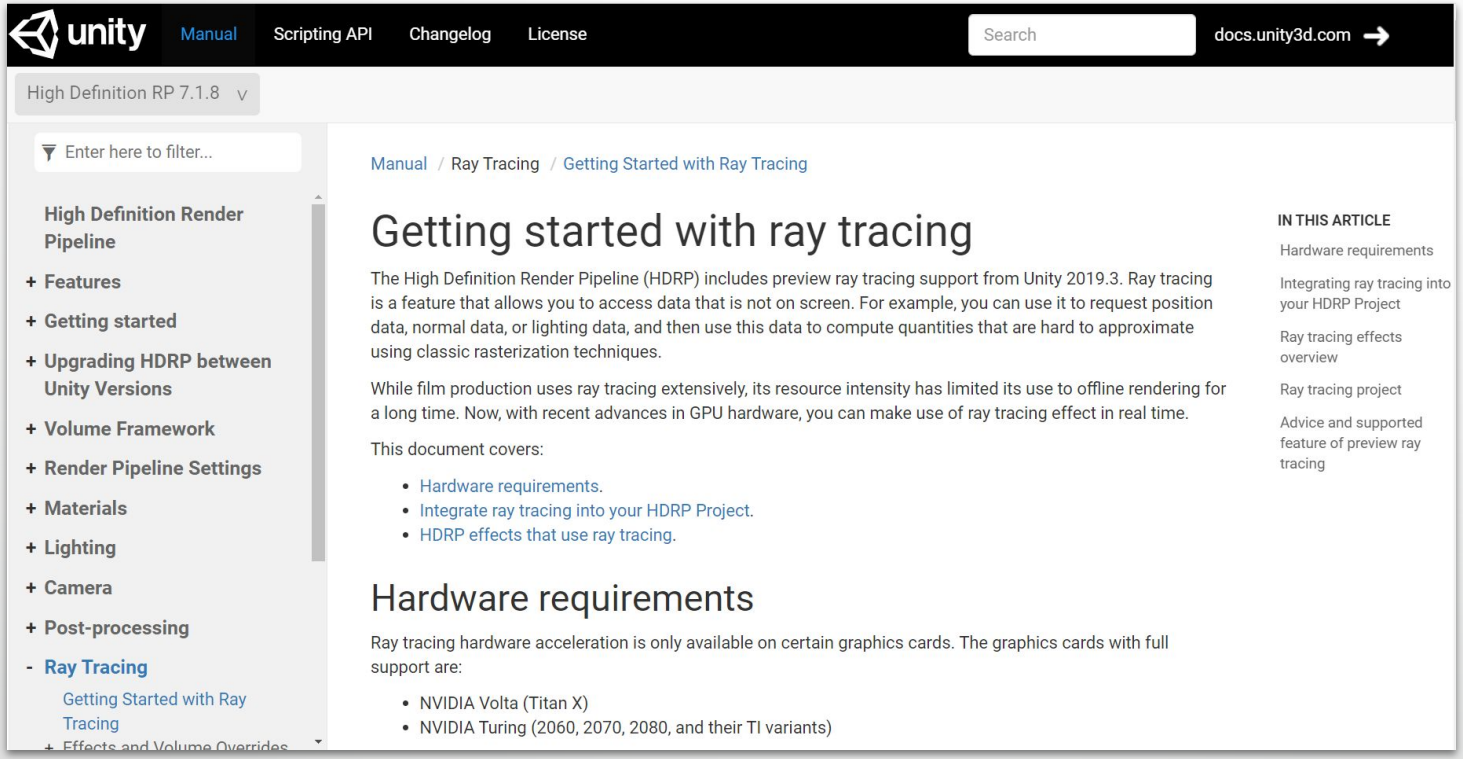




FeisStu


ILMxLAB
IMMERSIVE ENTERTAINMENT

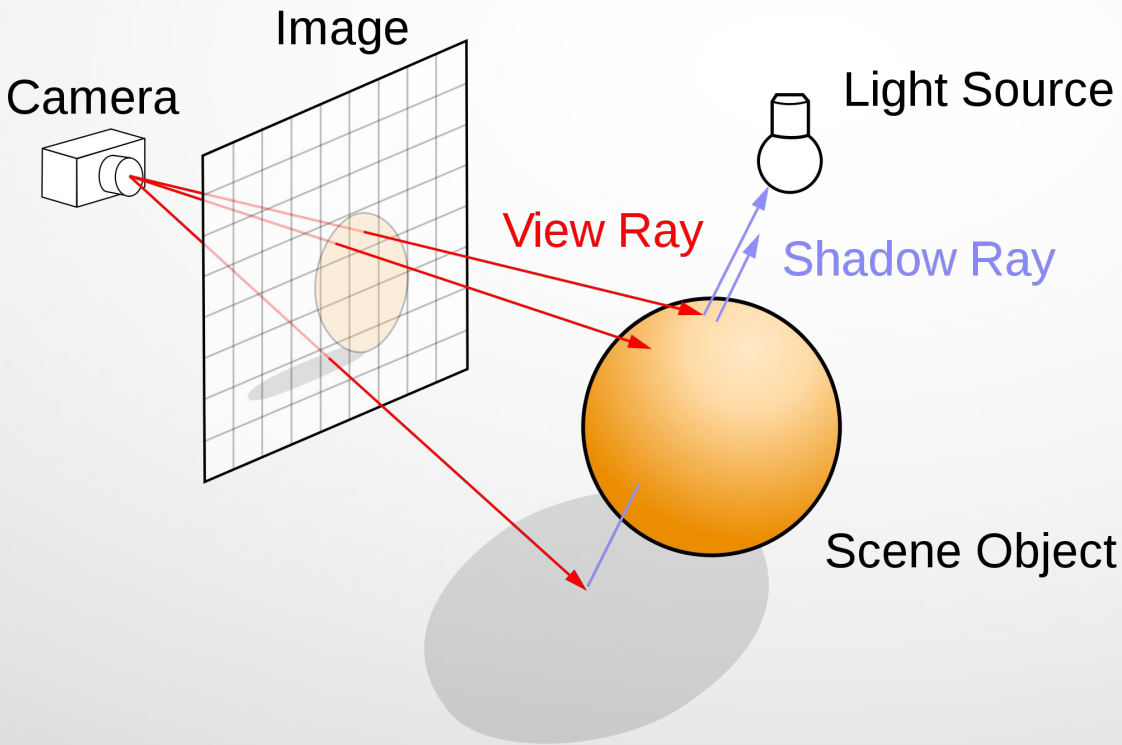
Ray Tracing with HDRP

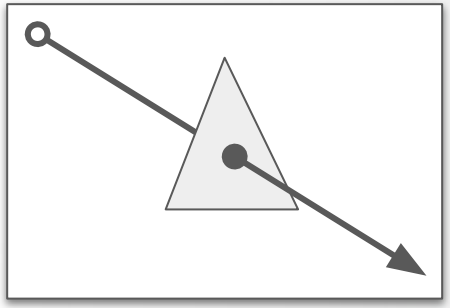


https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Ray-Tracing-Getting-Started.html

FeisStu

Ray tracing

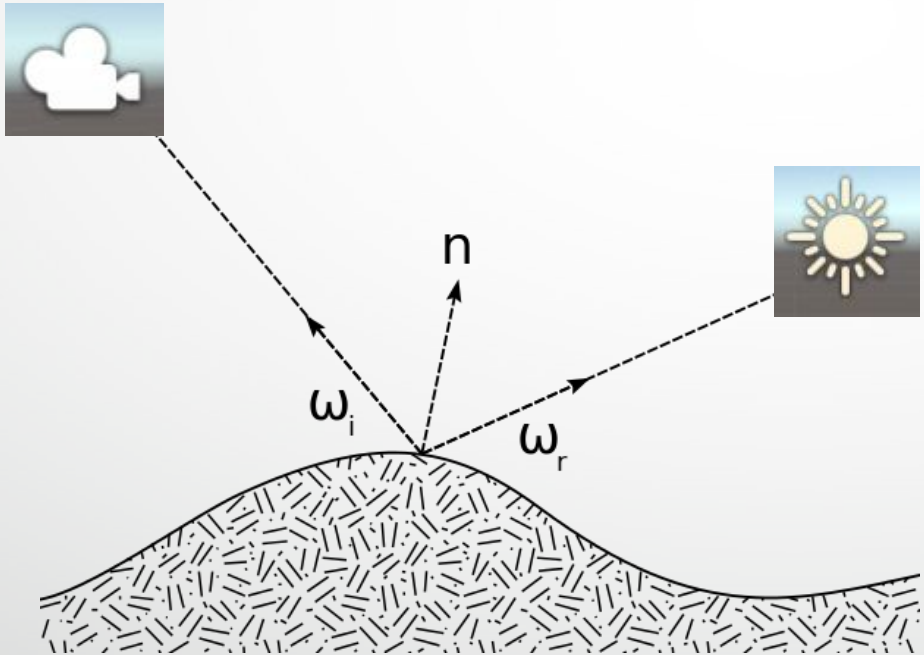




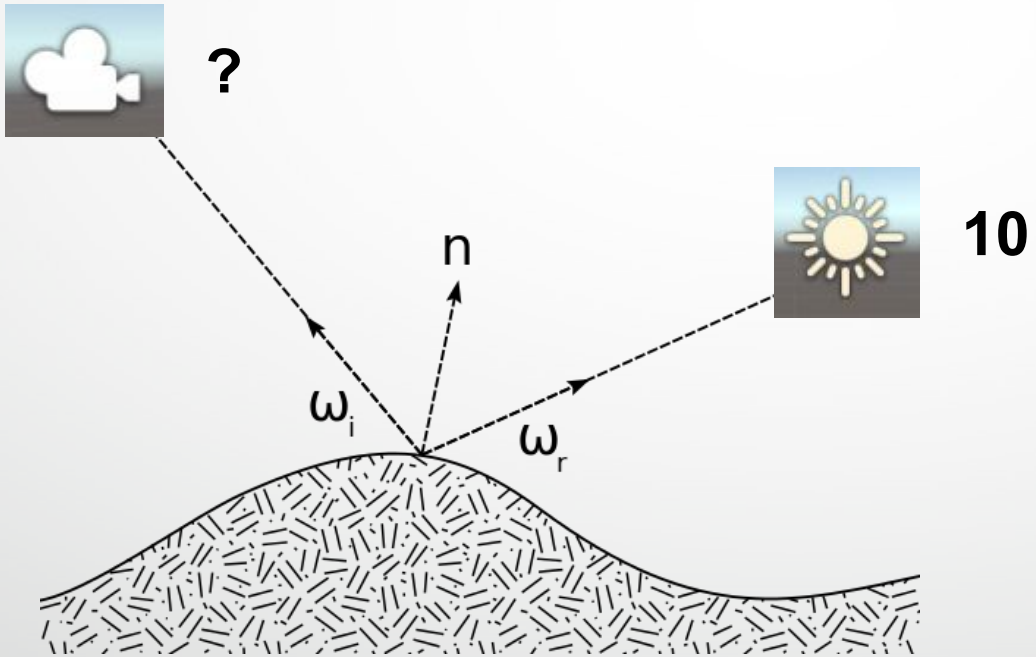
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Material

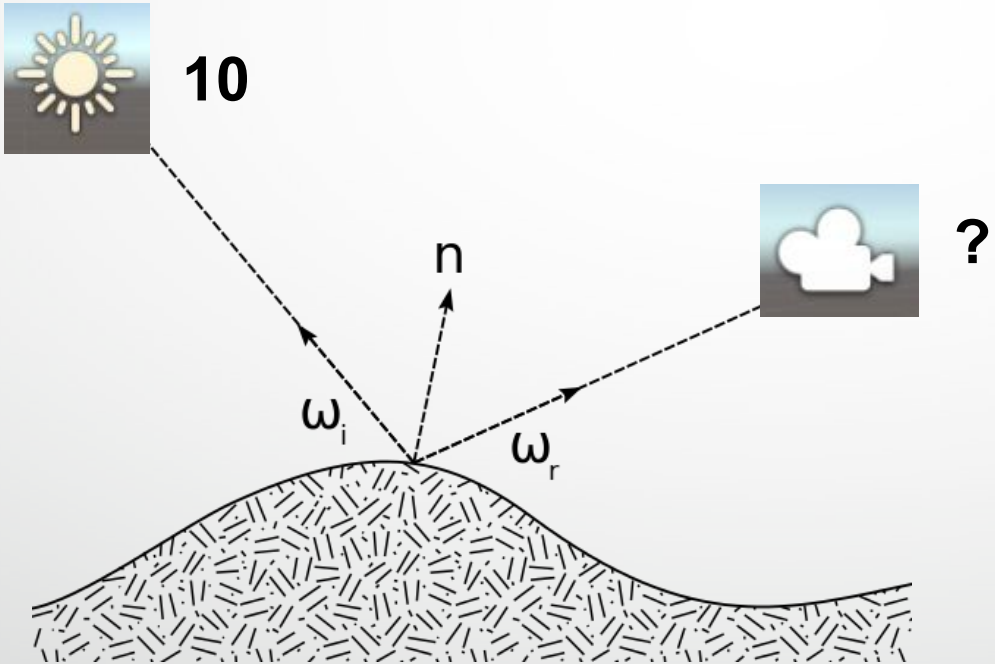


Material



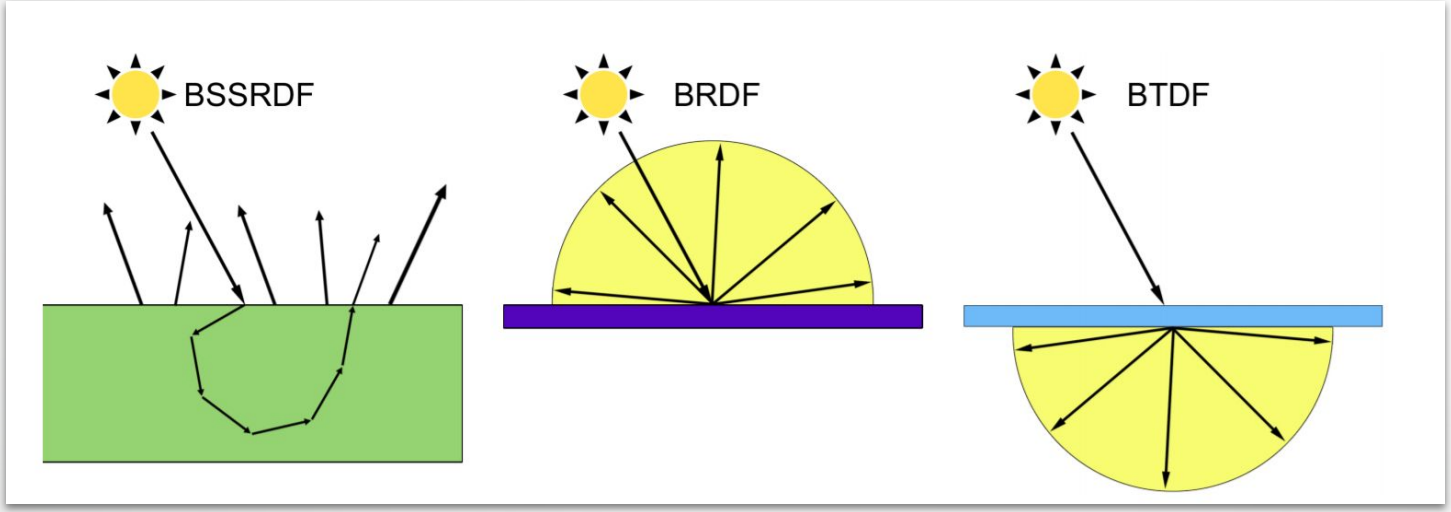
$\mathbf{f}\ (\omega_i, \omega_r)$

Material



$\mathbf{f}\ (\omega_r, \omega_i)$

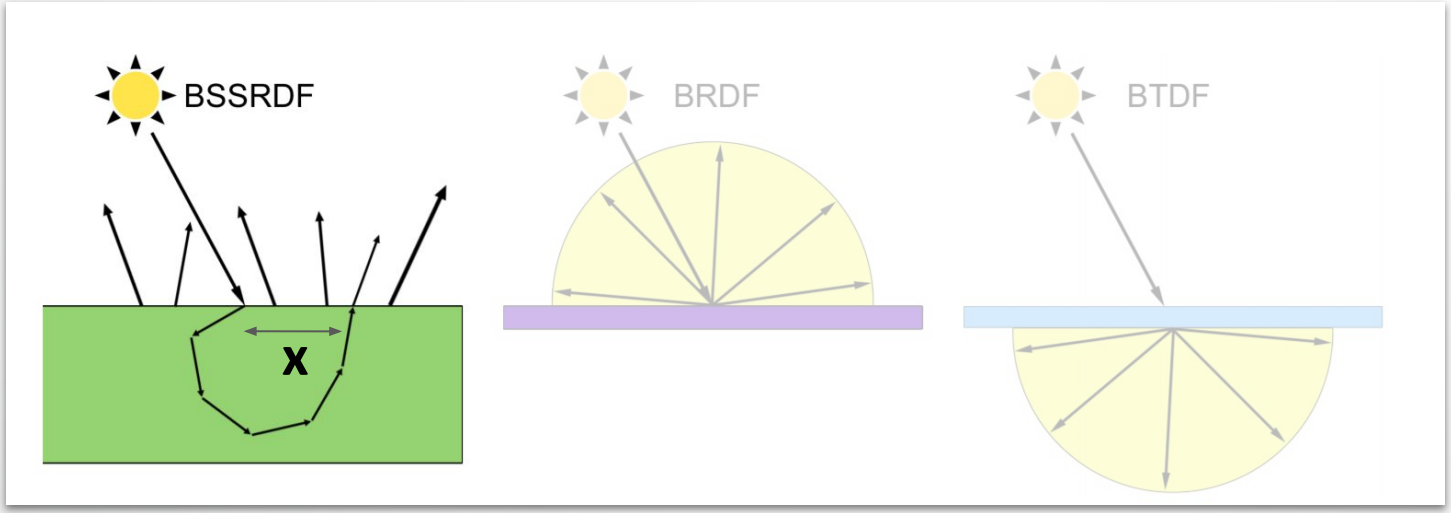
Material (cont'd)



http://collagefactory.blogspot.com/2010/04/brdf-for-diffuseglossyspecular.html

Material (cont'd)

BSSRDF (Bidirectional scattering distribution function)

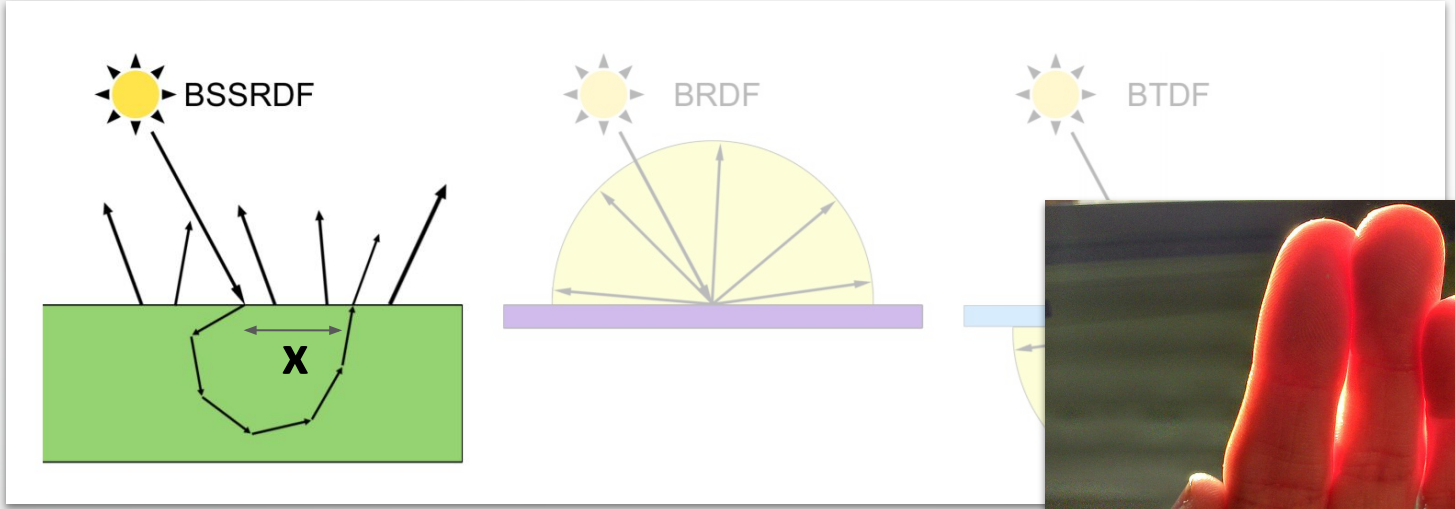


http://collagefactory.blogspot.com/2010/04/brdf-for-diffuseglossyspecular.html

$$\mathbf{f}_{BSSRDF}(x, \omega_i, \omega_r)$$

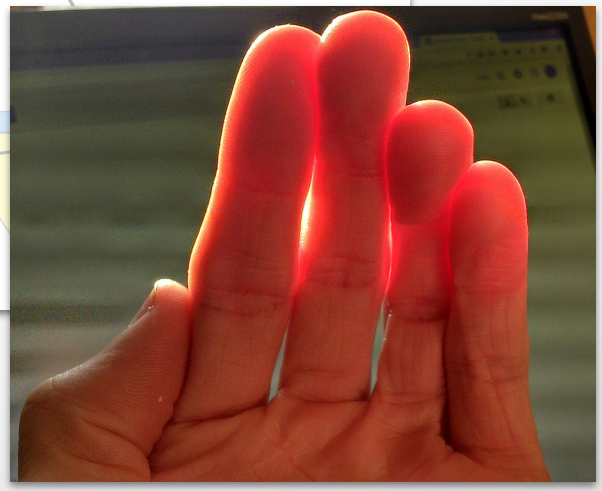
Material (cont'd)

BSSRDF (Bidirectional scattering distribution function)



http://collagefactory.blogspot.com/2010/04/brdf-for-diffuseglossyspecular.html

$$f_{BSSRDF}(x, \omega_i, \omega_r)$$

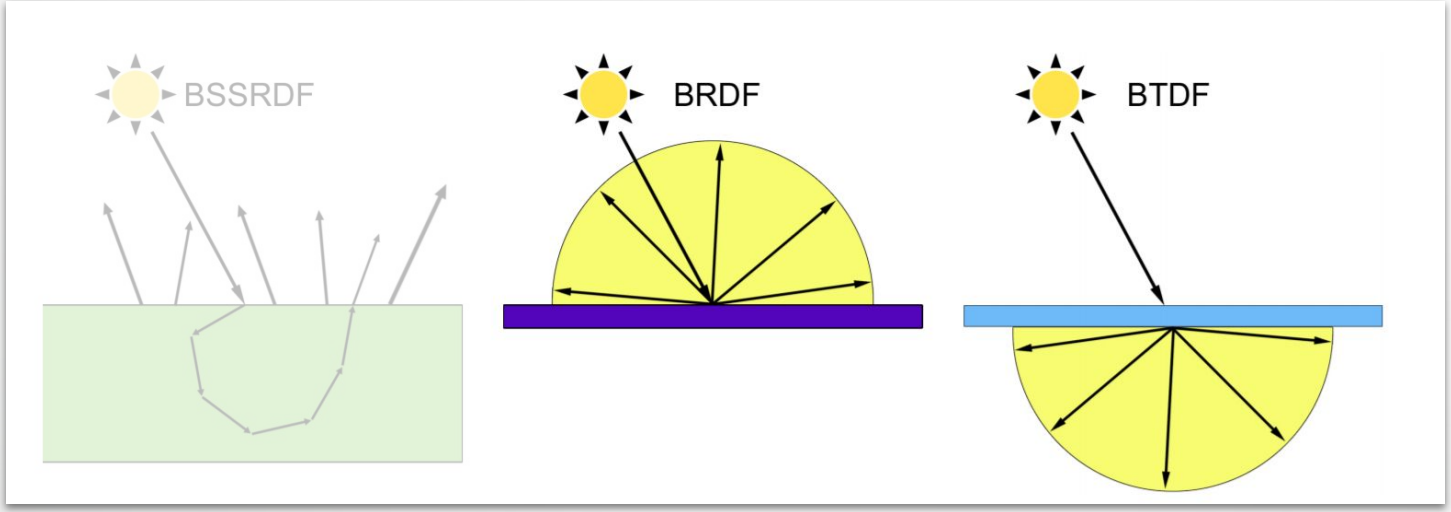


https://en.wikipedia.org/wiki/Subsurface_scattering

FeisStu

Material (cont'd)

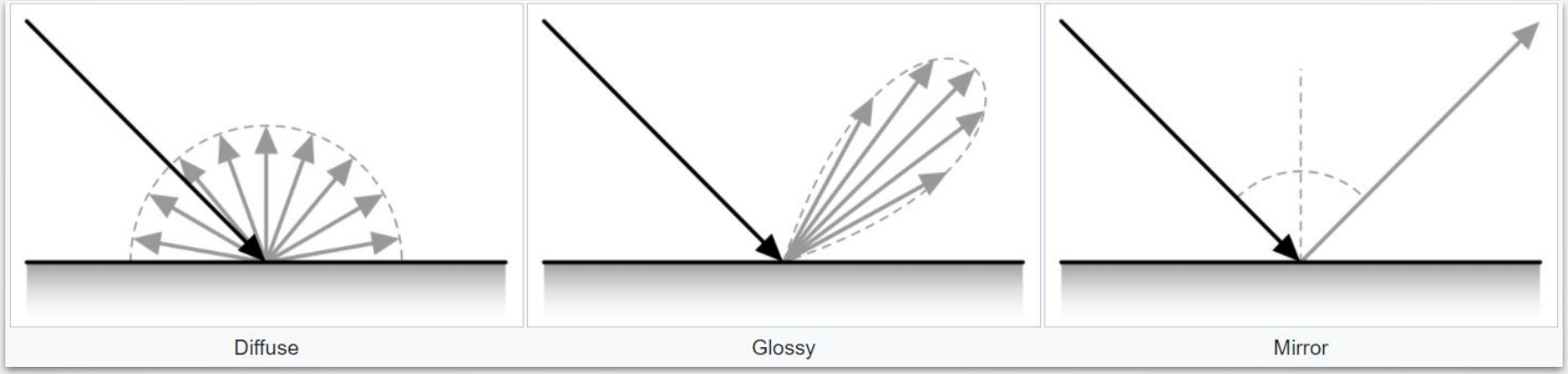
BSDF (Bidirectional scattering distribution function) =
BRDF(Bidirectional reflectance distribution function) +
BTDF(Bidirectional transmittance distribution function)



http://collagefactory.blogspot.com/2010/04/brdf-for-diffuseglossyspecular.html

$$\mathbf{f}_{BRDF}(\omega_i, \omega_r) \quad \mathbf{f}_{BTDF}(\omega_i, \omega_r)$$

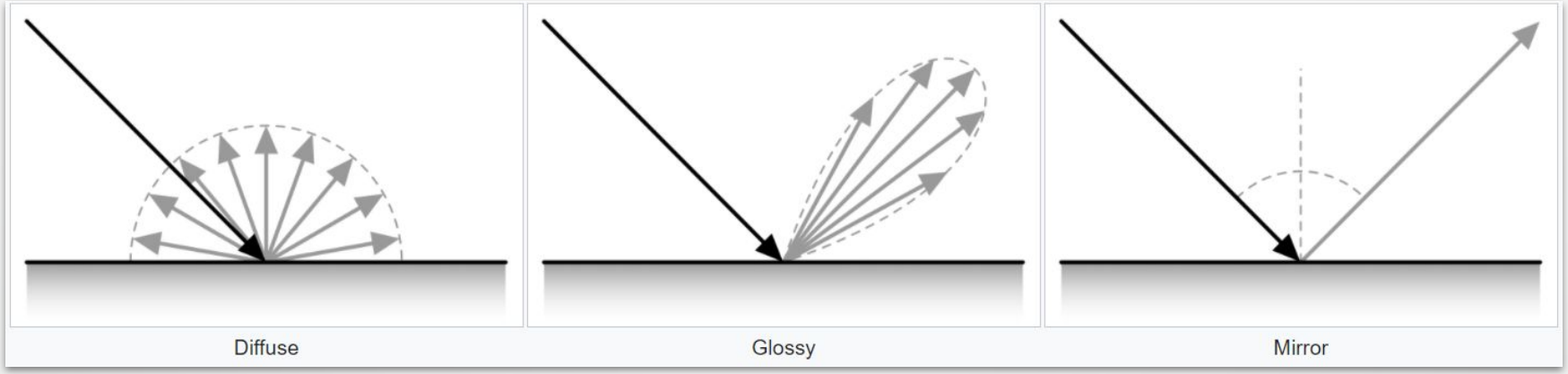
BRDF (Bidirectional reflectance distribution function)



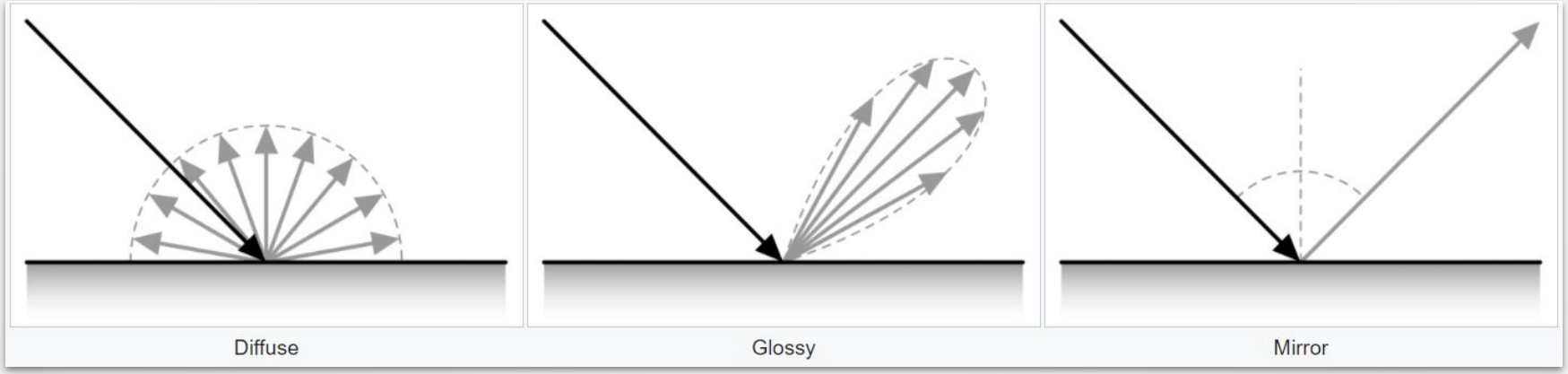
https://en.wikipedia.org/wiki/Bidirectional_reflectance_distribution_function

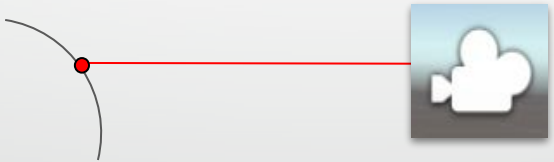
FeisStu

BRDF (Bidirectional reflectance distribution function)

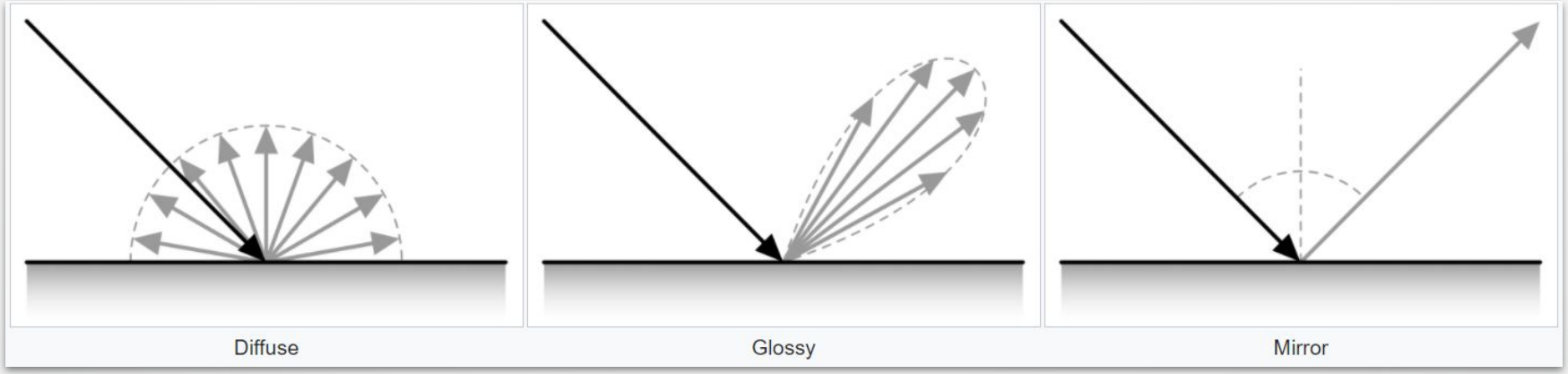


BRDF (Bidirectional reflectance distribution function)

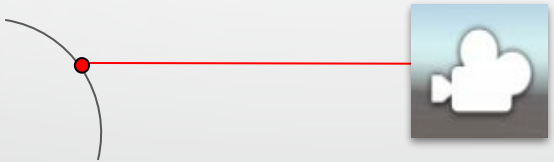




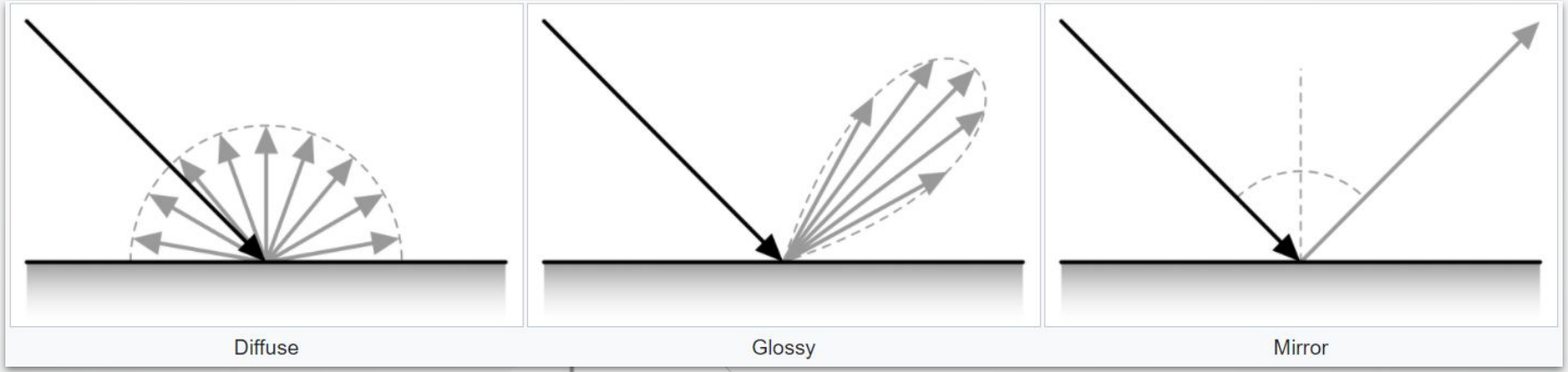
BRDF (Bidirectional reflectance distribution function)

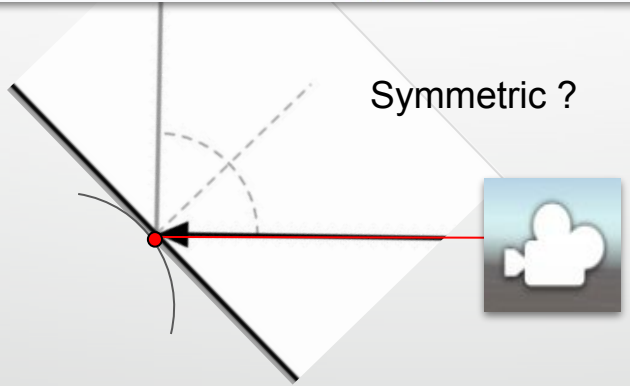


Emissive ?

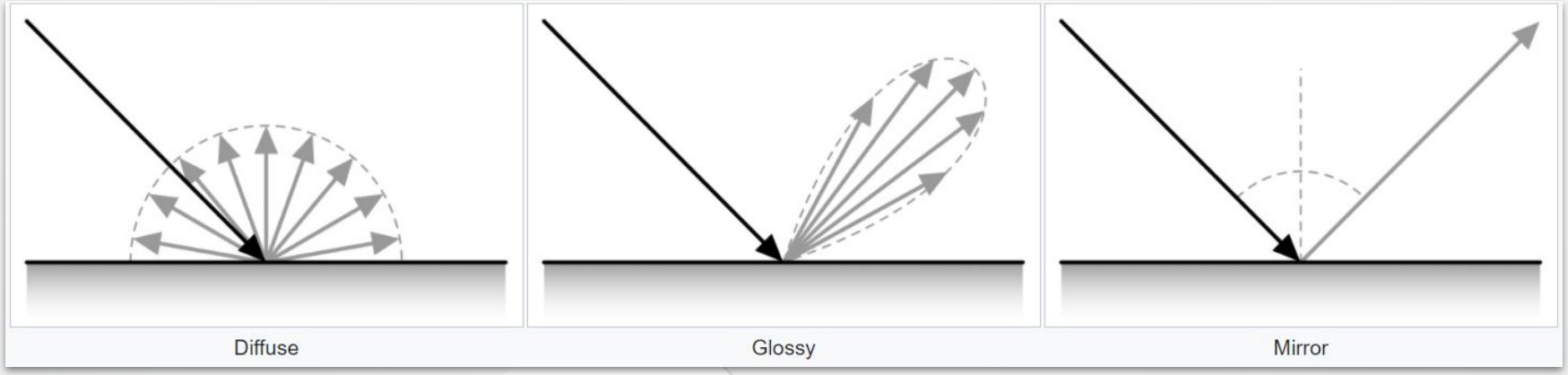


BRDF (Bidirectional reflectance distribution function)

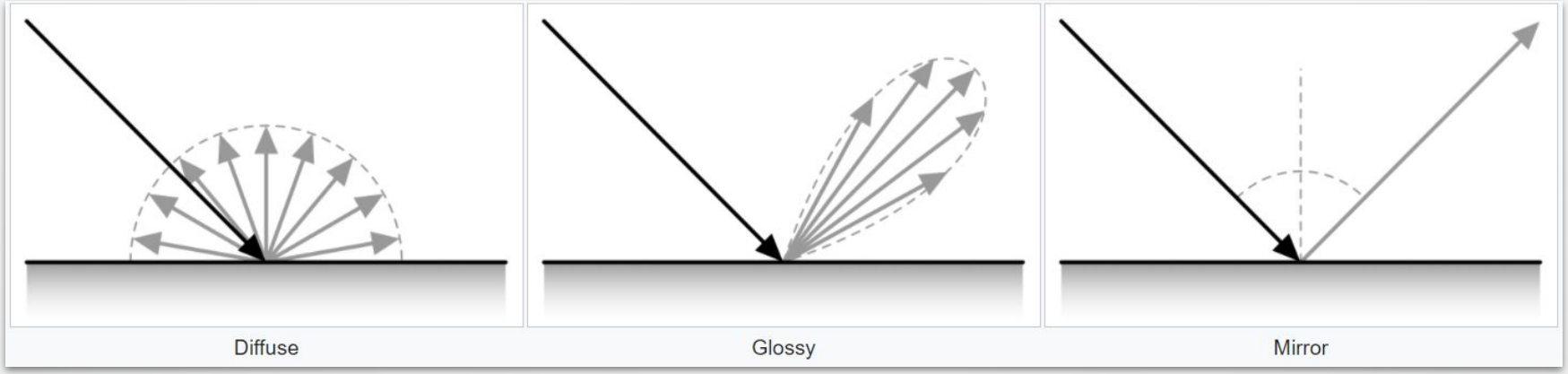


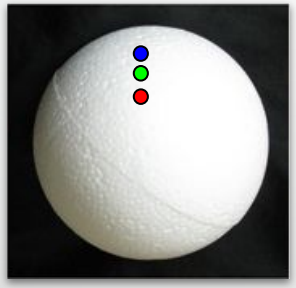


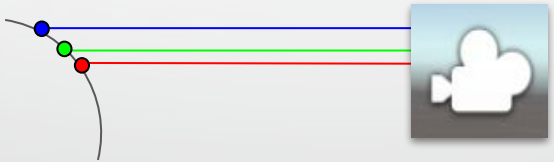
BRDF (Bidirectional reflectance distribution function)



BRDF (Bidirectional reflectance distribution function)

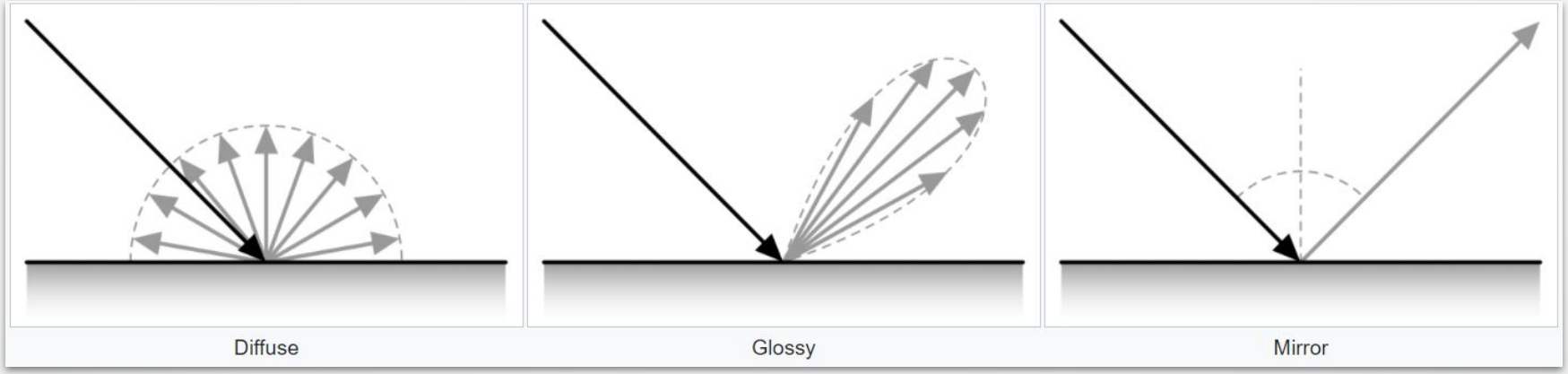


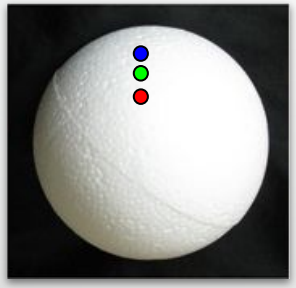


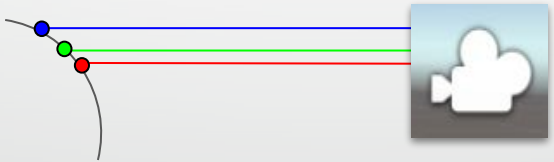


- Different **Normal**s

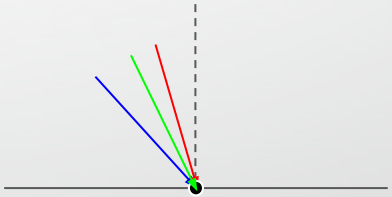
BRDF (Bidirectional reflectance distribution function)





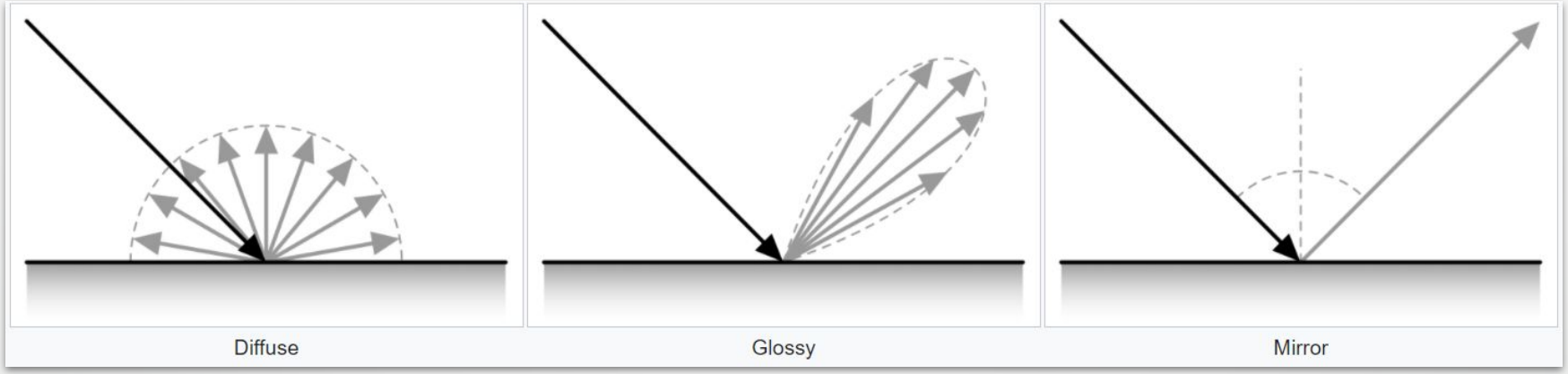


- Different **Normal**s



FeisStu

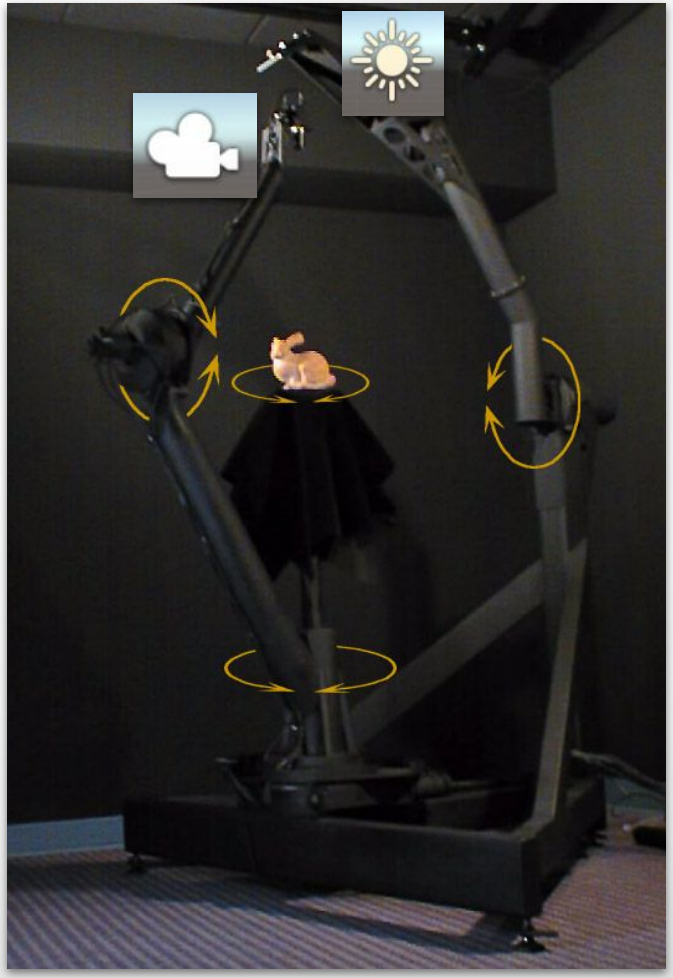
BRDF (Bidirectional reflectance distribution function)



BRDF Measurement

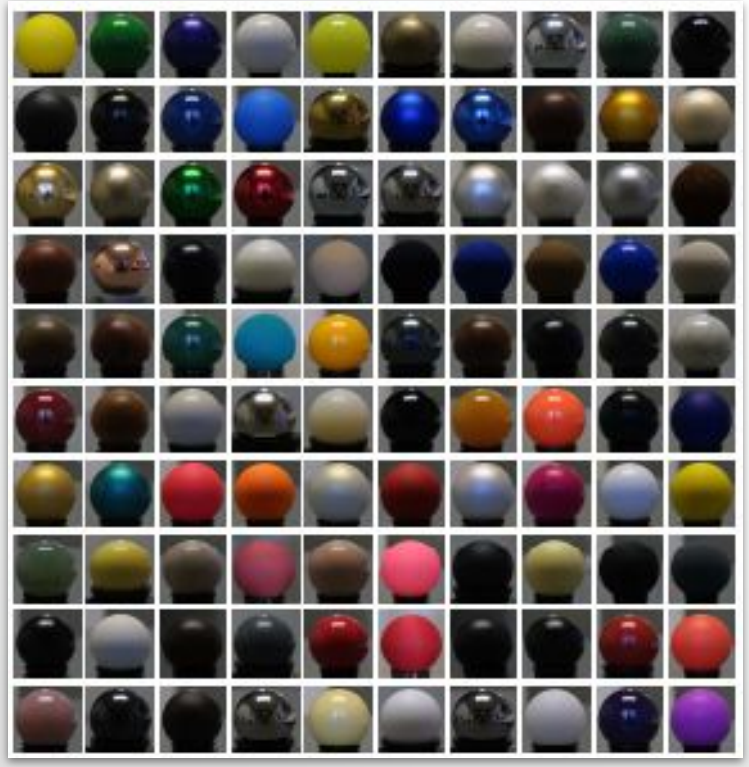
https://graphics.stanford.edu/projects/gantry/

BRDF Measurement

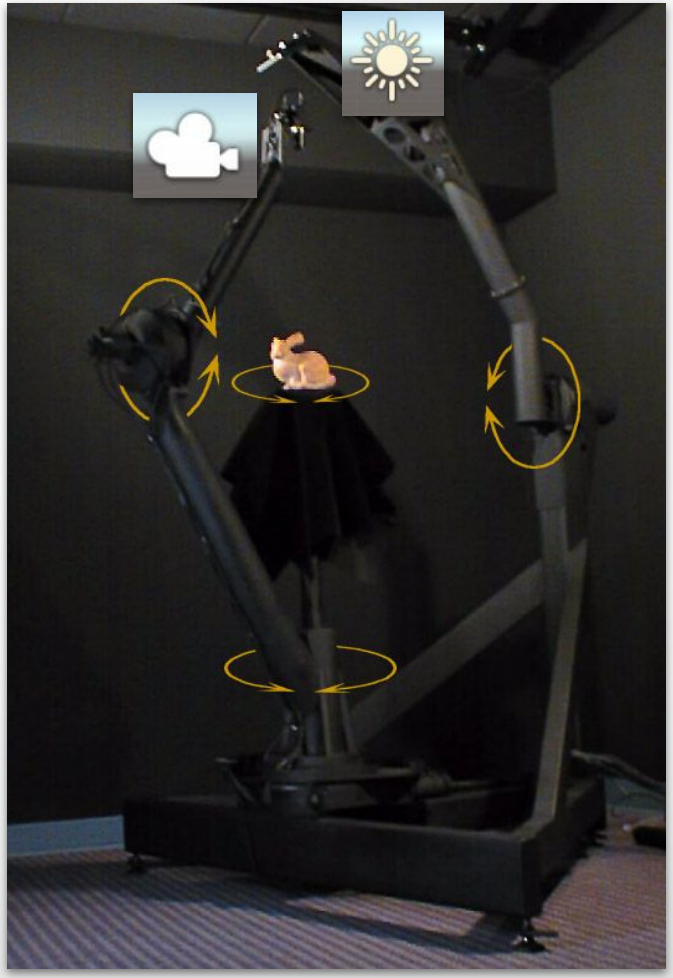


https://graphics.stanford.edu/projects/gantry/

BRDF Measurement

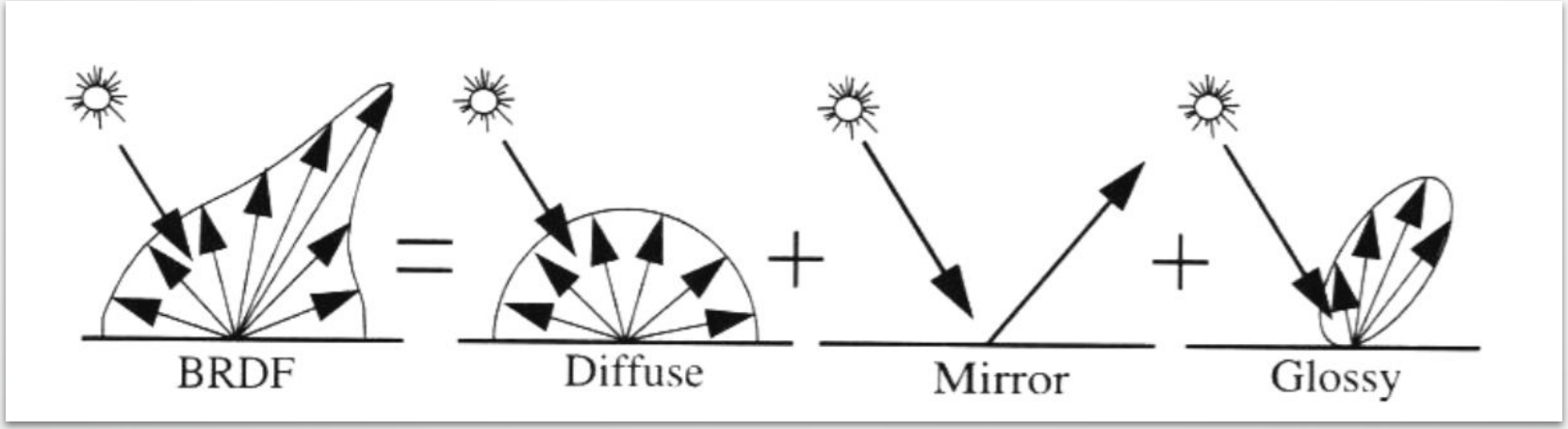


https://www.merl.com/brdf/



https://graphics.stanford.edu/projects/gantry/

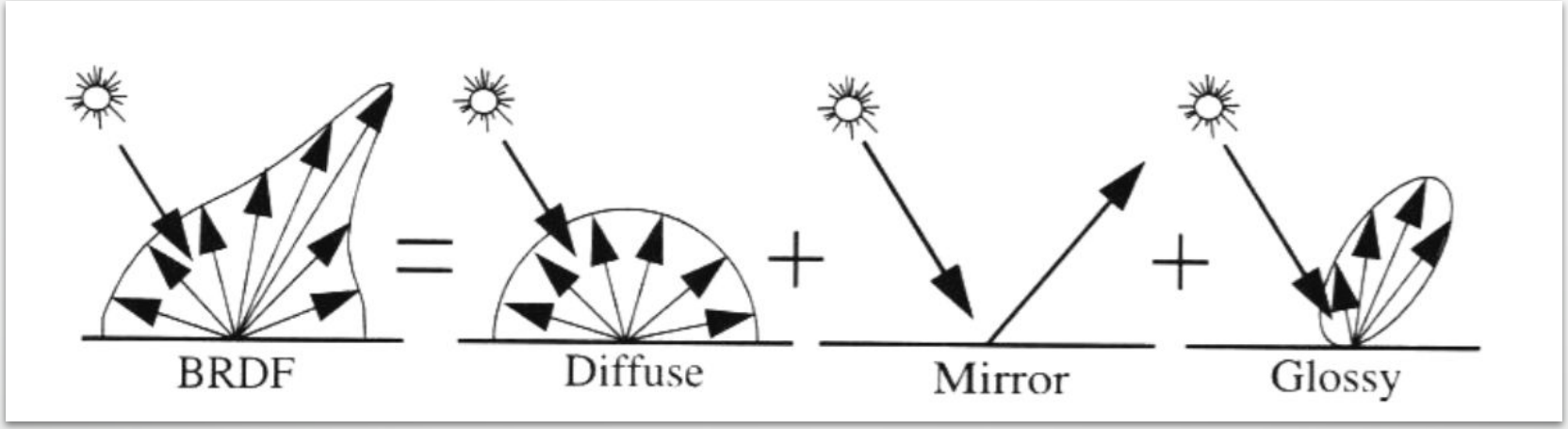
Analytic BRDF (Bidirectional reflectance distribution function)



http://resources.mpi-inf.mpg.de/departments/d4/teaching/ws200708/cg/slides/CG07-Brdf+Texture.pdf

FeisStu

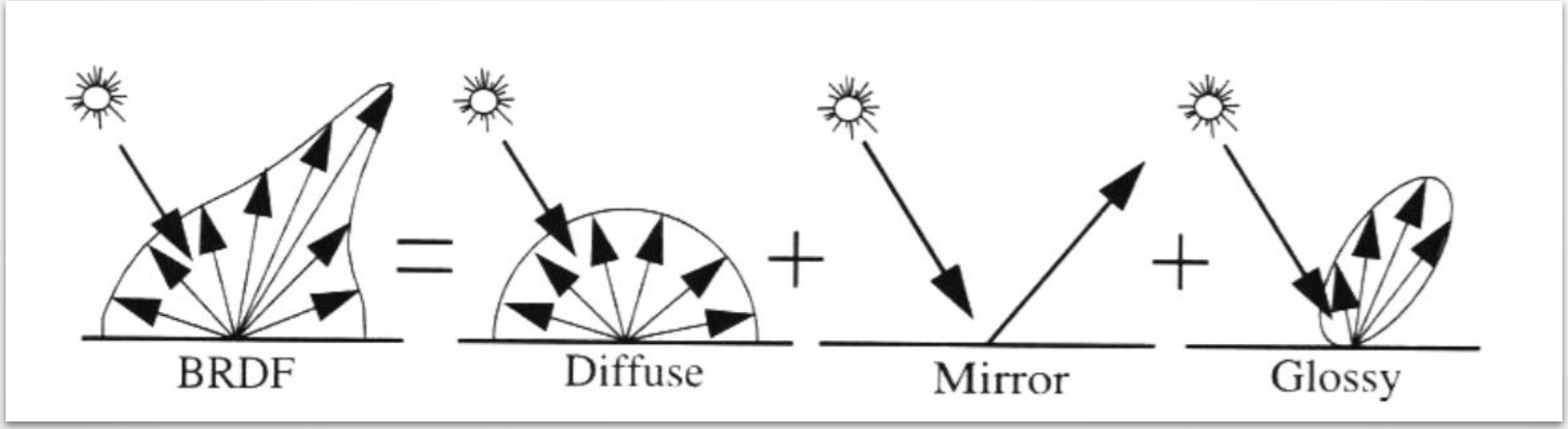
Analytic BRDF (Bidirectional reflectance distribution function)



http://resources.mpi-inf.mpg.de/departments/d4/teaching/ws200708/cg/slides/CG07-Brdf+Texture.pdf

$$= f_D(\quad) + f_M(\quad) + f_G(\quad)$$

Analytic BRDF (Bidirectional reflectance distribution function)

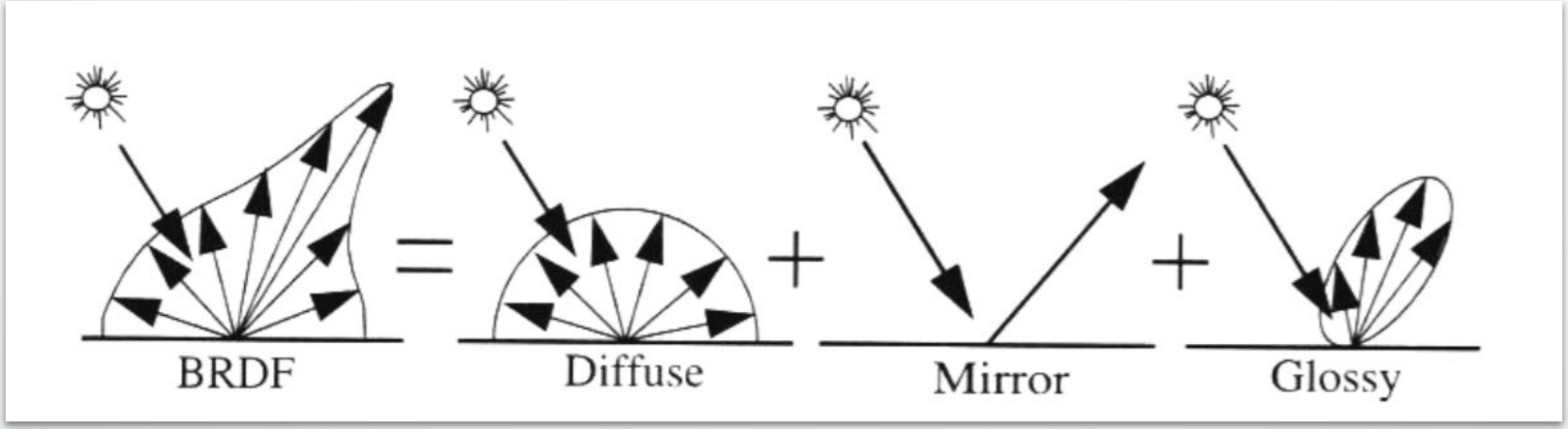


http://resources.mpi-inf.mpg.de/departments/d4/teaching/ws200708/cg/slides/CG07-Brdf+Texture.pdf

$= f_D(0.8) + f_M(2) + f_G(0.1)$

FeisStu

Analytic BRDF (Bidirectional reflectance distribution function)

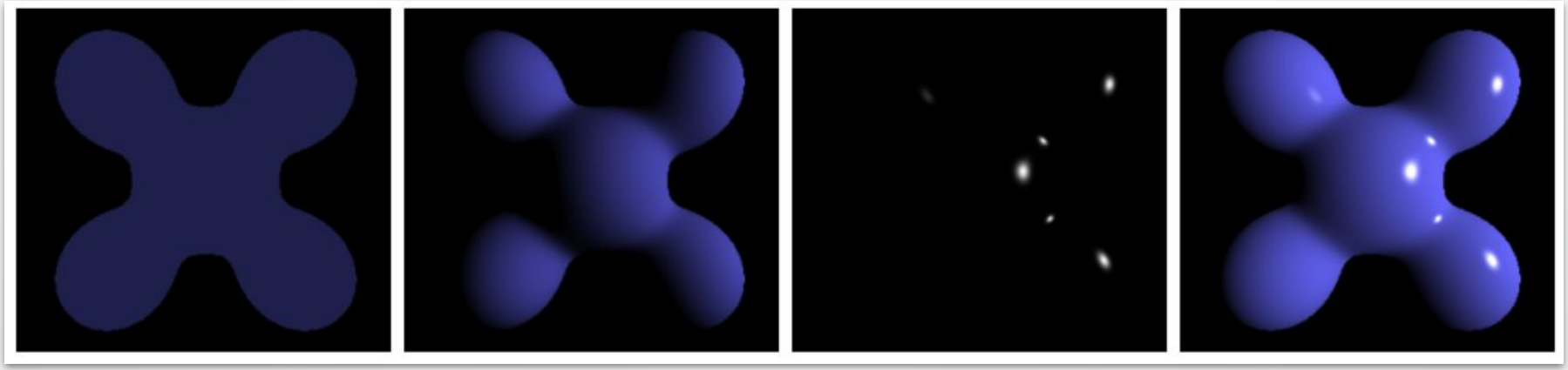


http://resources.mpi-inf.mpg.de/departments/d4/teaching/ws200708/cg/slides/CG07-Brdf+Texture.pdf

$= f_D(0.8) + f_M(2) + f_G(0.1)$

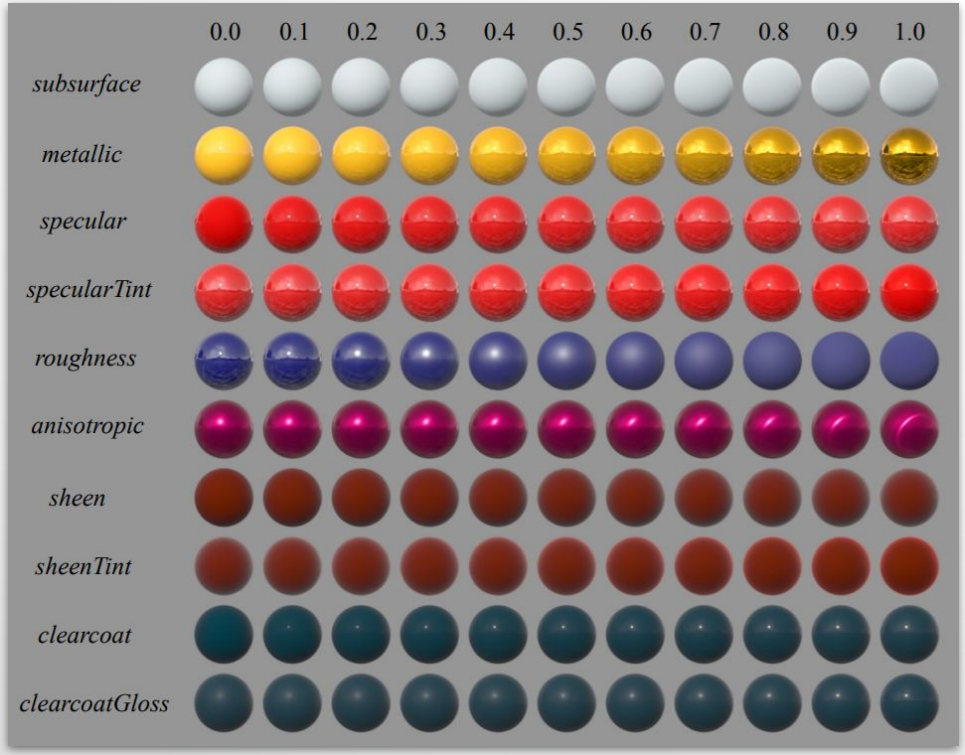
FeisStu

Phong illumination model



$$I_{Ambient}(k_a) + \left[I_{Diffuse}(k_d) + I_{Specular}(k_s) \right] = I$$

Disney Principled BRDF



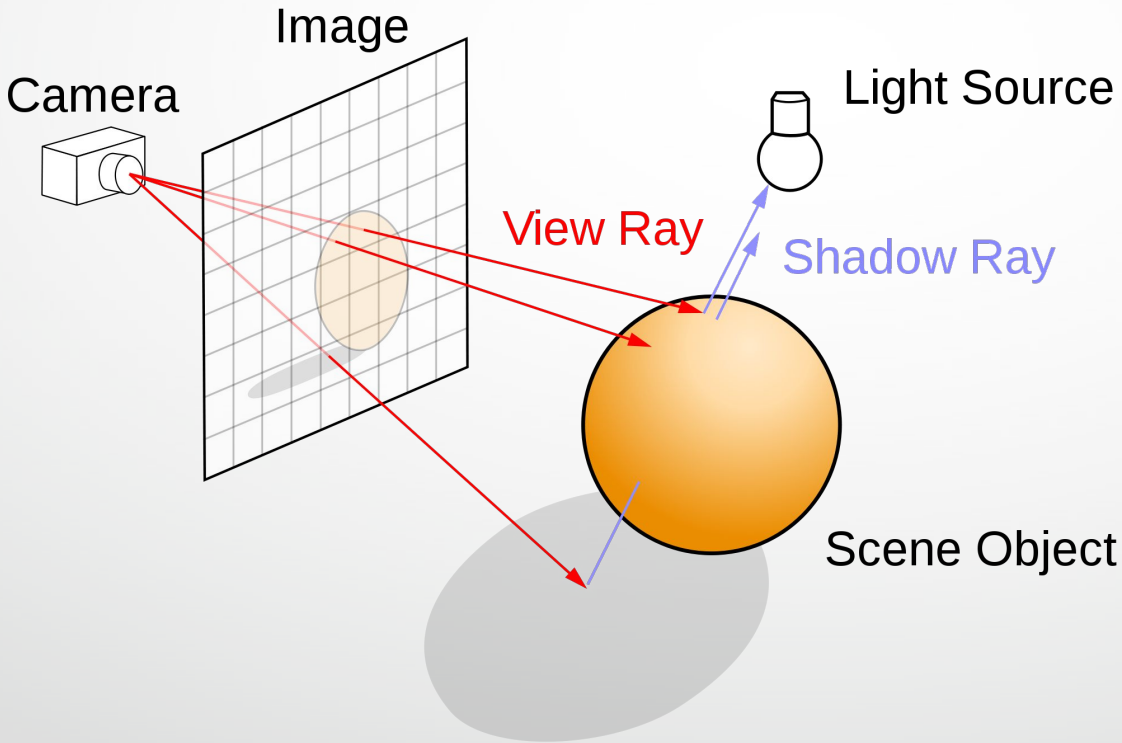
Github :

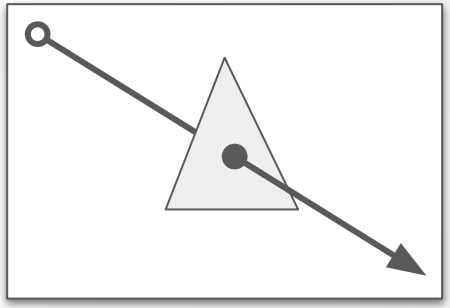
https://github.com/wdas/brdf/blob/main/src/brdfs/disney.brdf

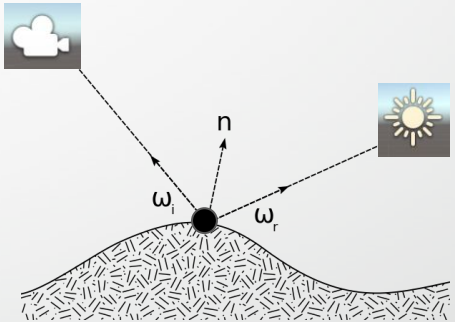
https://blog.selfshadow.com/publications/s2012-shading-course/burley/s2012_pbs_disney_brdf_notes_v3.pdf

FeisStu

Ray tracing



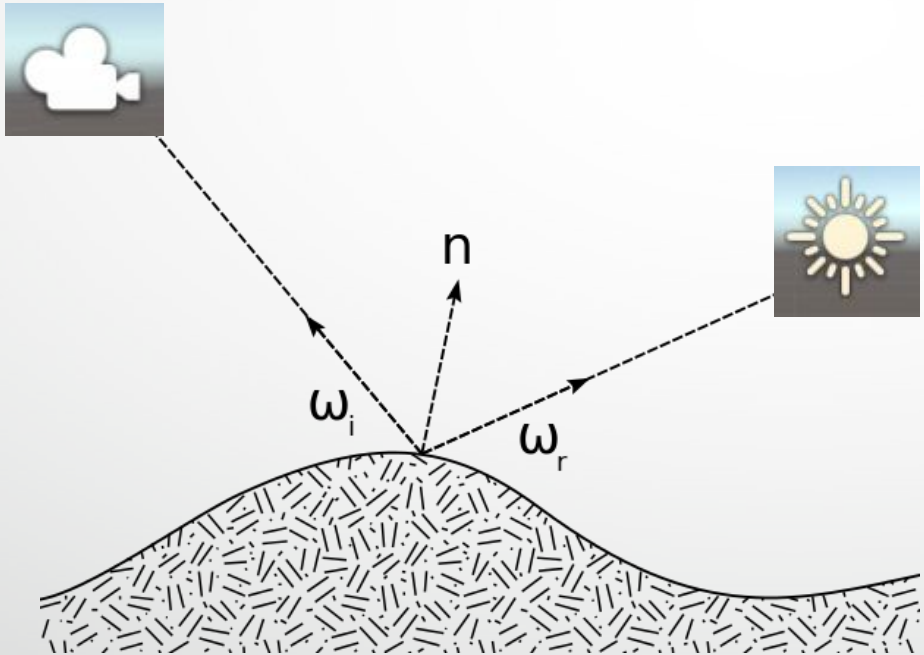




https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

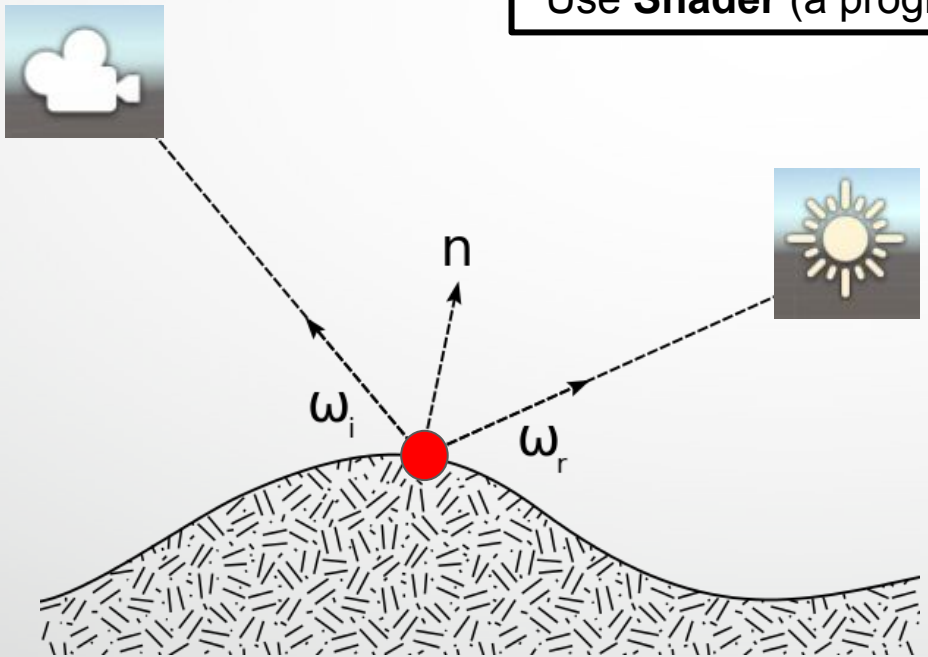
FeisStu

Material



Material

Use **Shader** (a program) to calculate the result

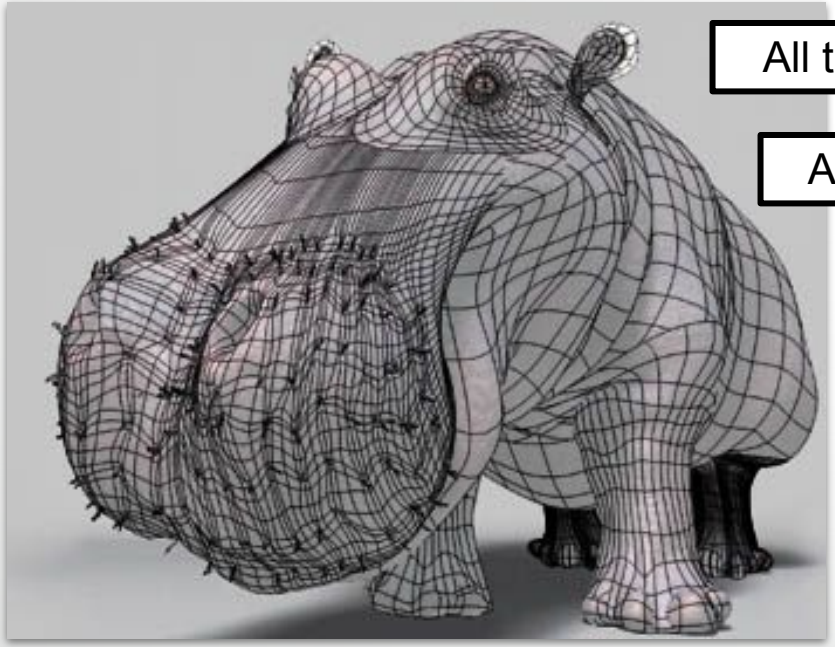


Given :

- Light
- View
- Normal
- BRDF parameters
- Position
- ...

FeisStu

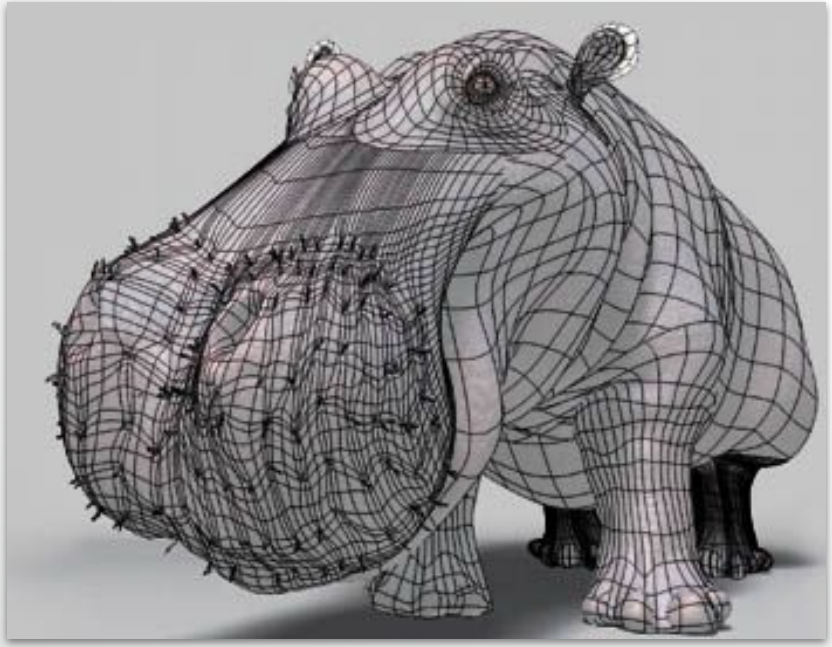
Material (cont'd)



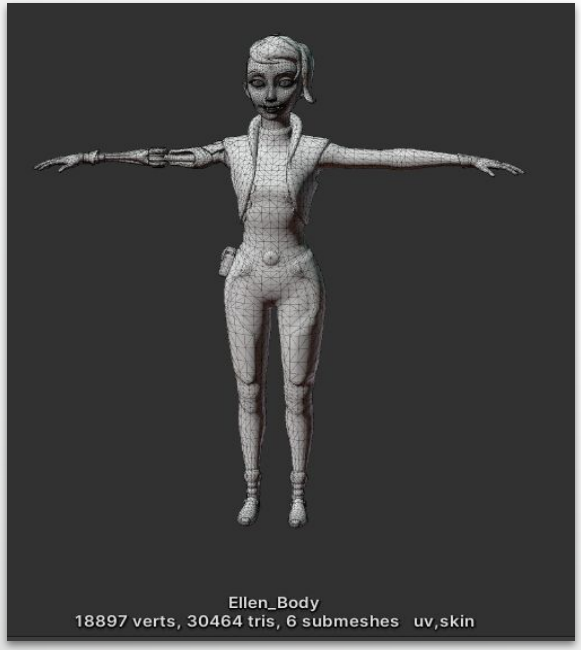
All triangles/quads have the same **Material** ?

All triangles/quads have the same **BRDF** ?

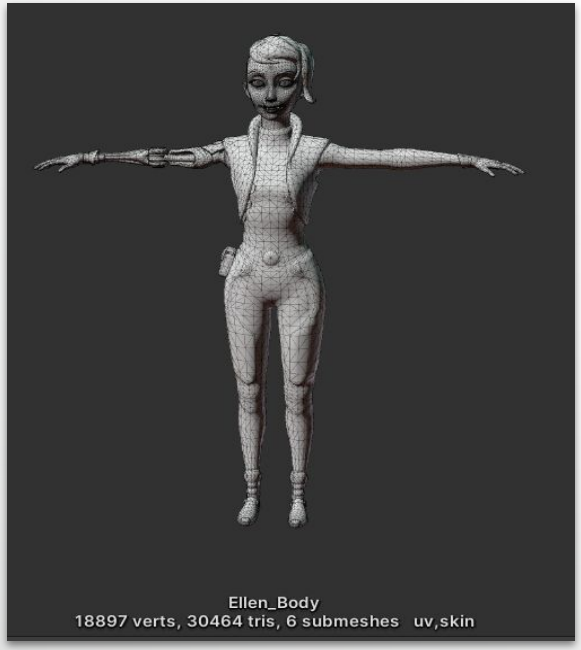
Texture

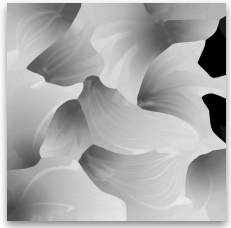


Texture



Texture



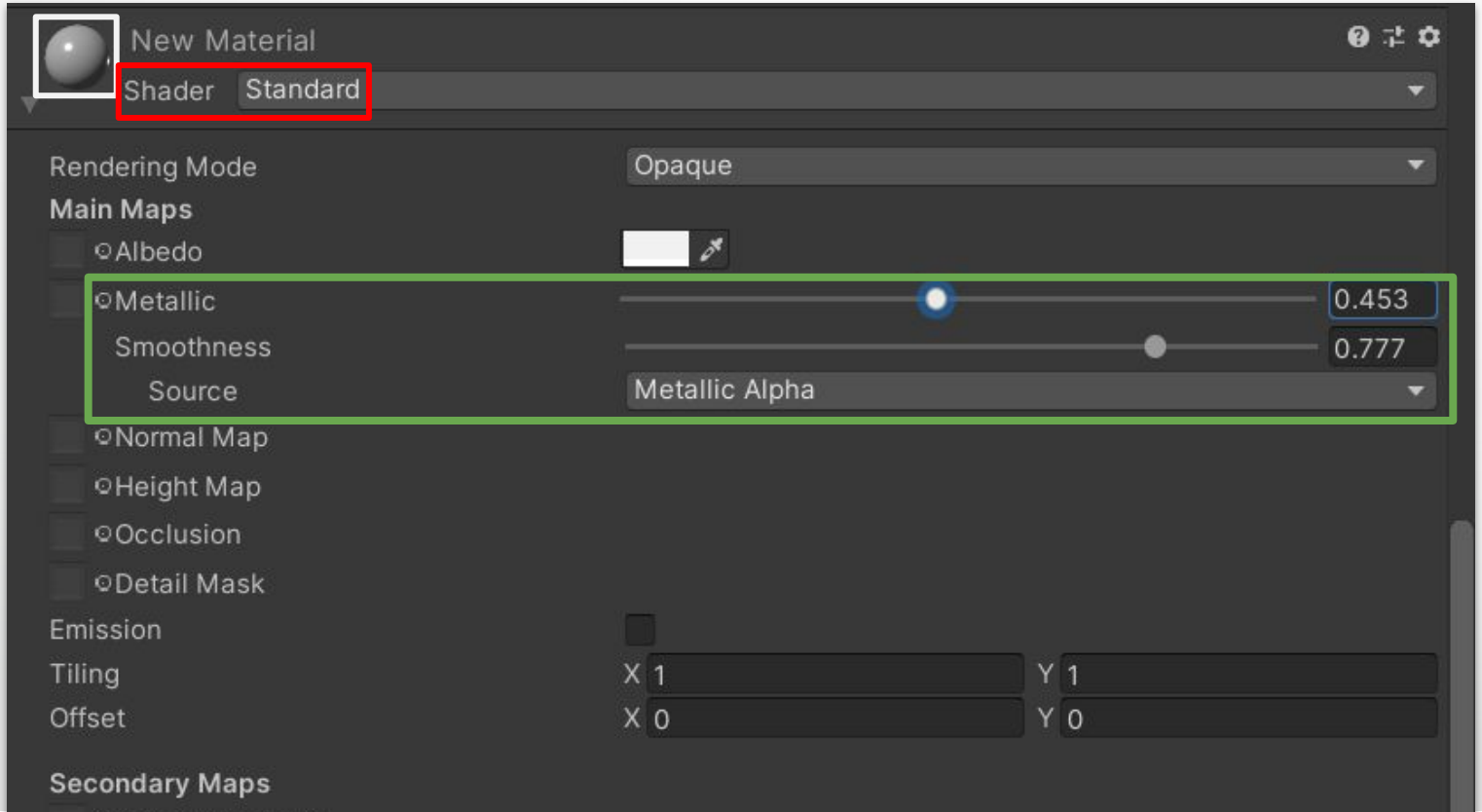


u

v

FeisStu

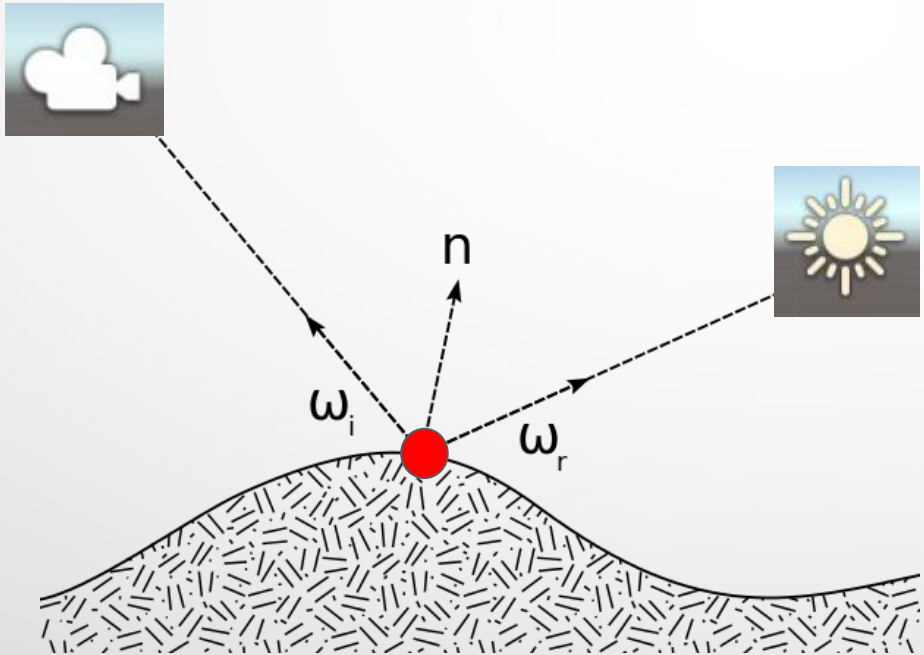
Material



FeisStu

Case Study: Concept Art

Shader

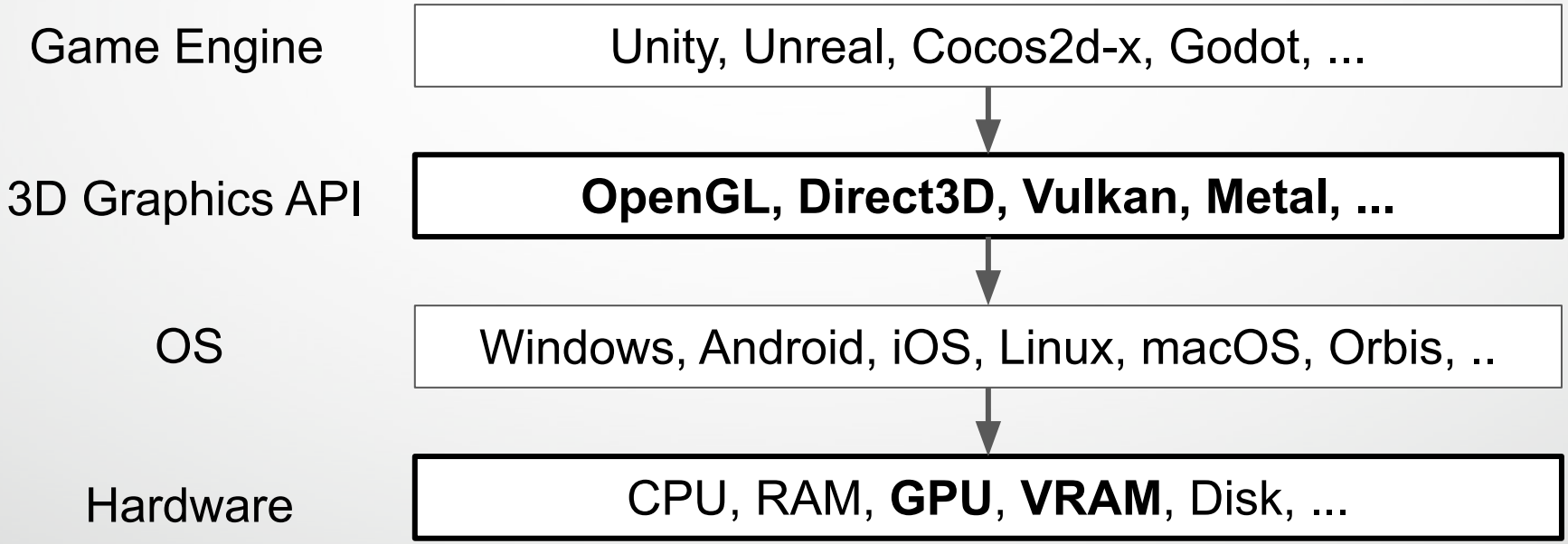


Given :

- Light
- View
- Normal
- BRDF parameters
- Position
- UVs
- Textures
- ...

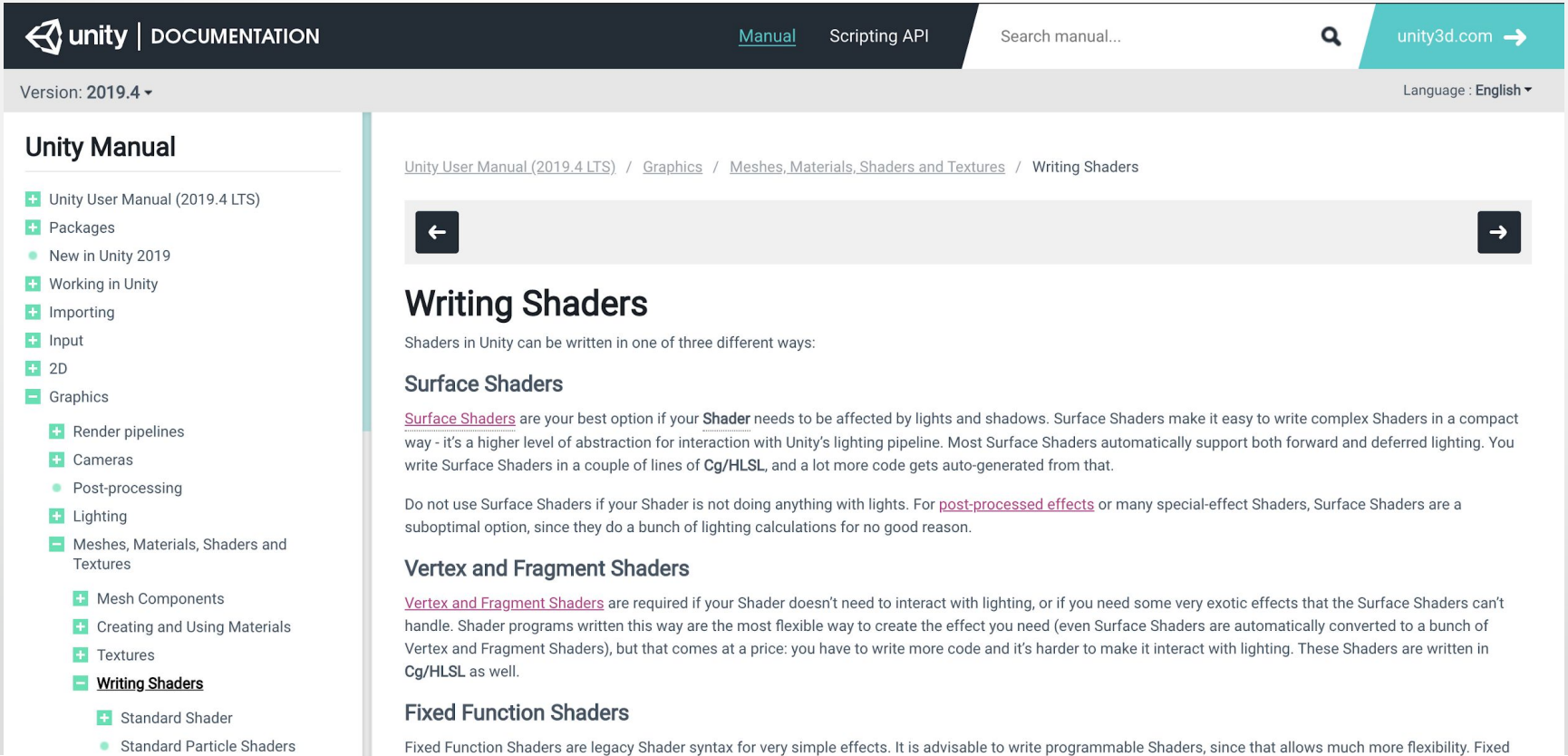
FeisStu

Graphics hardware



FeisStu

Writing shaders



unity | DOCUMENTATION Manual Scripting API Search manual... unity3d.com

Version: 2019.4 Language : English

Unity Manual

- Unity User Manual (2019.4 LTS)
- Packages
- New in Unity 2019
- Working in Unity
- Importing
- Input
- 2D
- Graphics
  - Render pipelines
  - Cameras
  - Post-processing
  - Lighting
  - Meshes, Materials, Shaders and Textures
    - Mesh Components
    - Creating and Using Materials
    - Textures
    - Writing Shaders
      - Standard Shader
      - Standard Particle Shaders

Unity User Manual (2019.4 LTS) / Graphics / Meshes, Materials, Shaders and Textures / Writing Shaders

Writing Shaders

Shaders in Unity can be written in one of three different ways:

Surface Shaders

Surface Shaders are your best option if your **Shader** needs to be affected by lights and shadows. Surface Shaders make it easy to write complex Shaders in a compact way - it's a higher level of abstraction for interaction with Unity's lighting pipeline. Most Surface Shaders automatically support both forward and deferred lighting. You write Surface Shaders in a couple of lines of **Cg/HLSL**, and a lot more code gets auto-generated from that.

Do not use Surface Shaders if your Shader is not doing anything with lights. For post-processed effects or many special-effect Shaders, Surface Shaders are a suboptimal option, since they do a bunch of lighting calculations for no good reason.

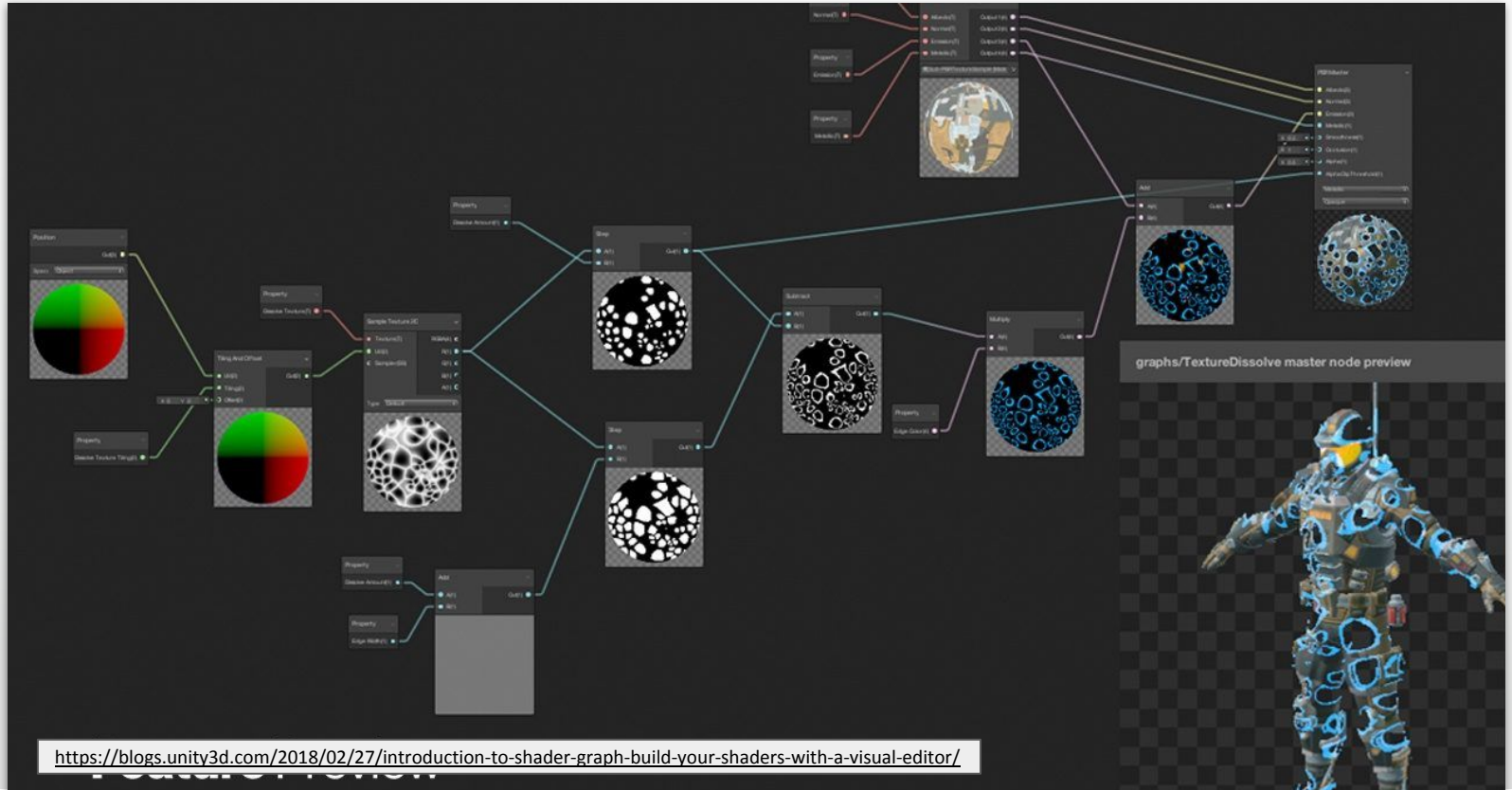
Vertex and Fragment Shaders

Vertex and Fragment Shaders are required if your Shader doesn't need to interact with lighting, or if you need some very exotic effects that the Surface Shaders can't handle. Shader programs written this way are the most flexible way to create the effect you need (even Surface Shaders are automatically converted to a bunch of Vertex and Fragment Shaders), but that comes at a price: you have to write more code and it's harder to make it interact with lighting. These Shaders are written in **Cg/HLSL** as well.

Fixed Function Shaders

Fixed Function Shaders are legacy Shader syntax for very simple effects. It is advisable to write programmable Shaders, since that allows much more flexibility. Fixed

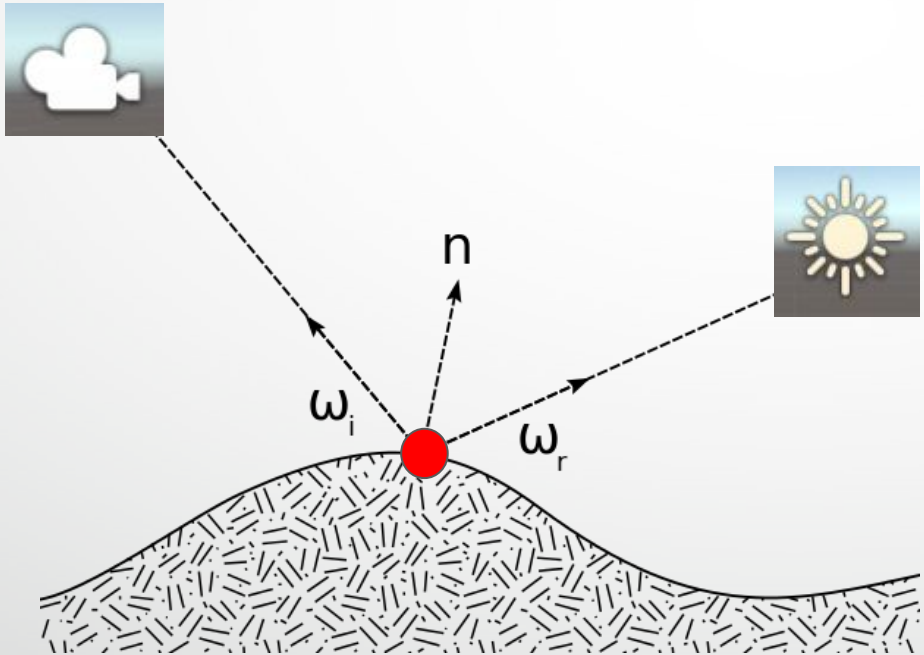
Shader Graph



https://blogs.unity3d.com/2018/02/27/introduction-to-shader-graph-build-your-shaders-with-a-visual-editor/

FeisStu

Shader

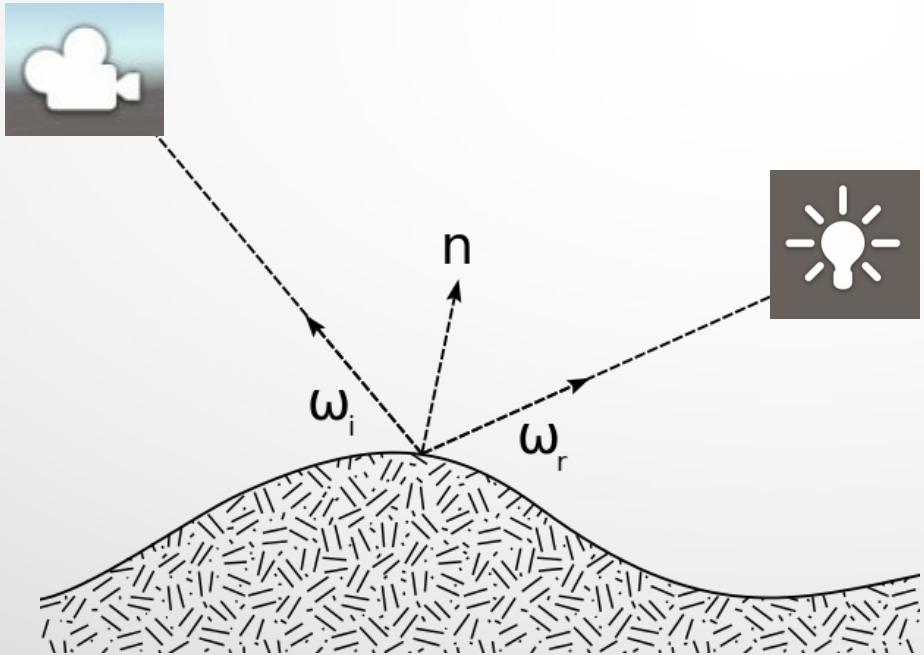


Considers :

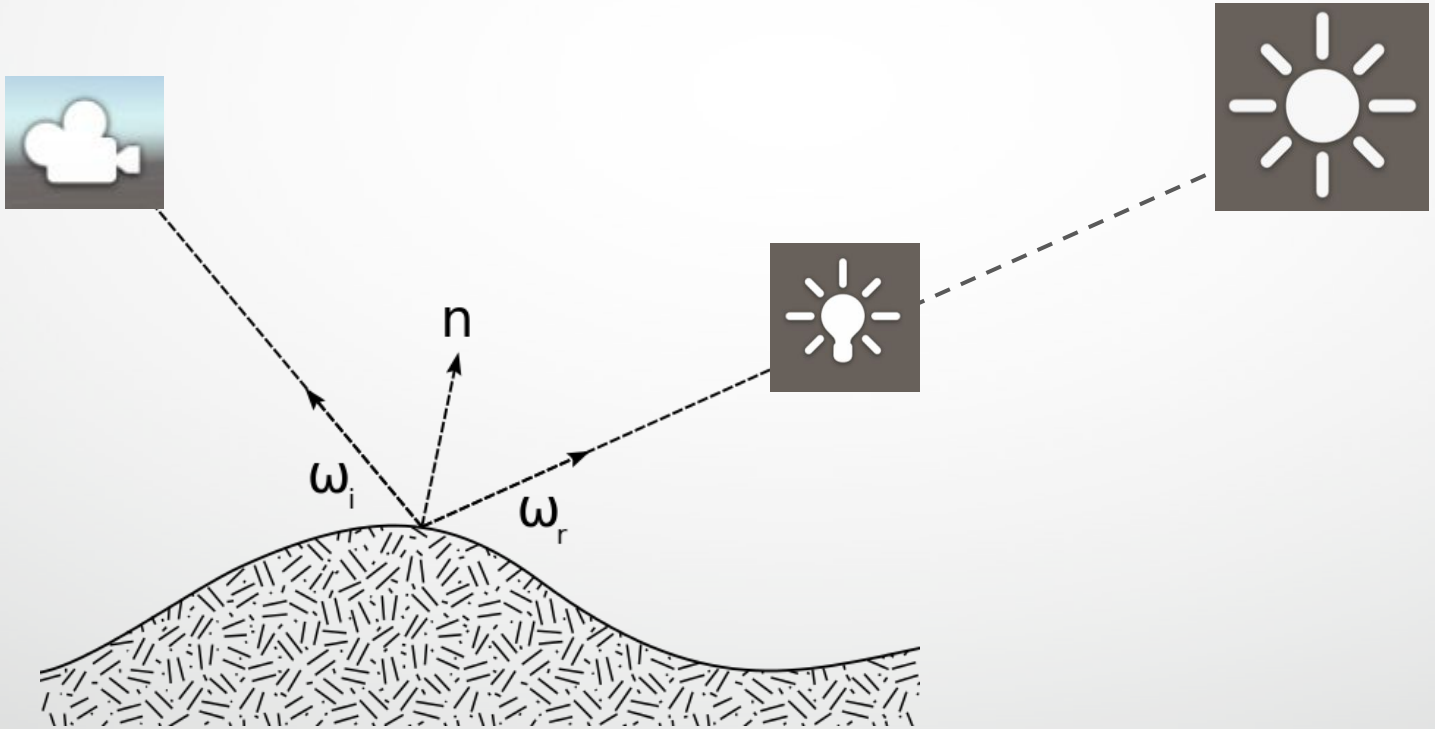
- Light
- View
- Normal
- BRDF
- Position
- UVs
- Textures
- ...

FeisStu

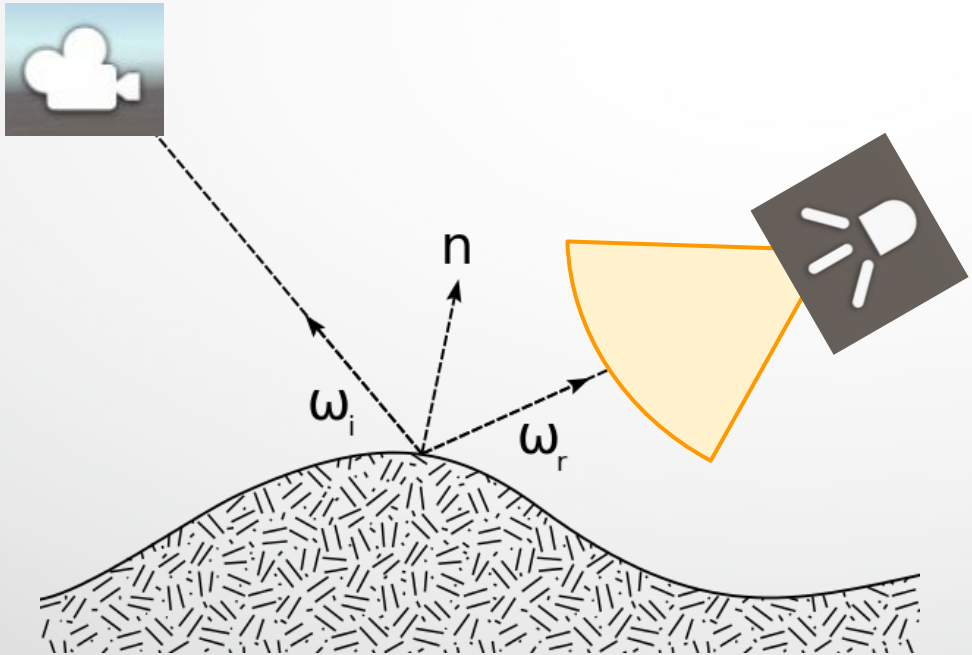
Point light



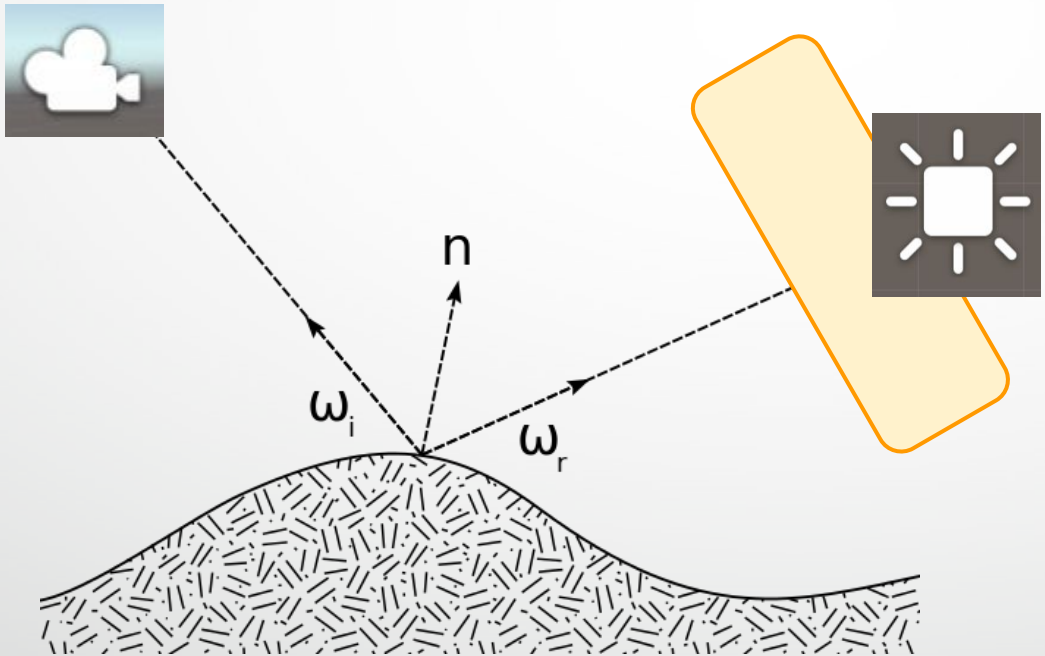
Point light / Directional light



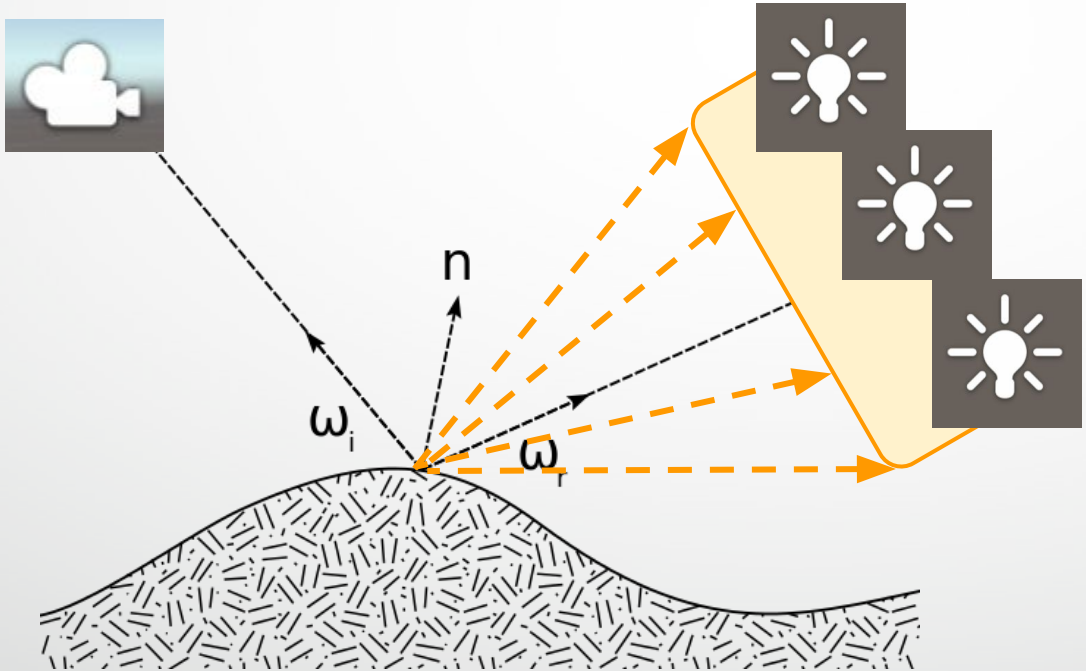
Spot light



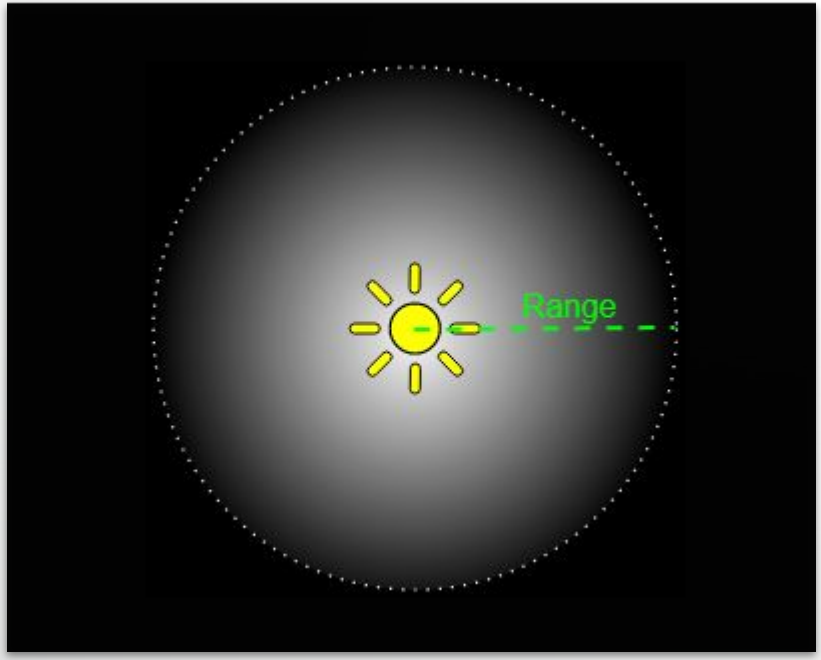
Area light ?

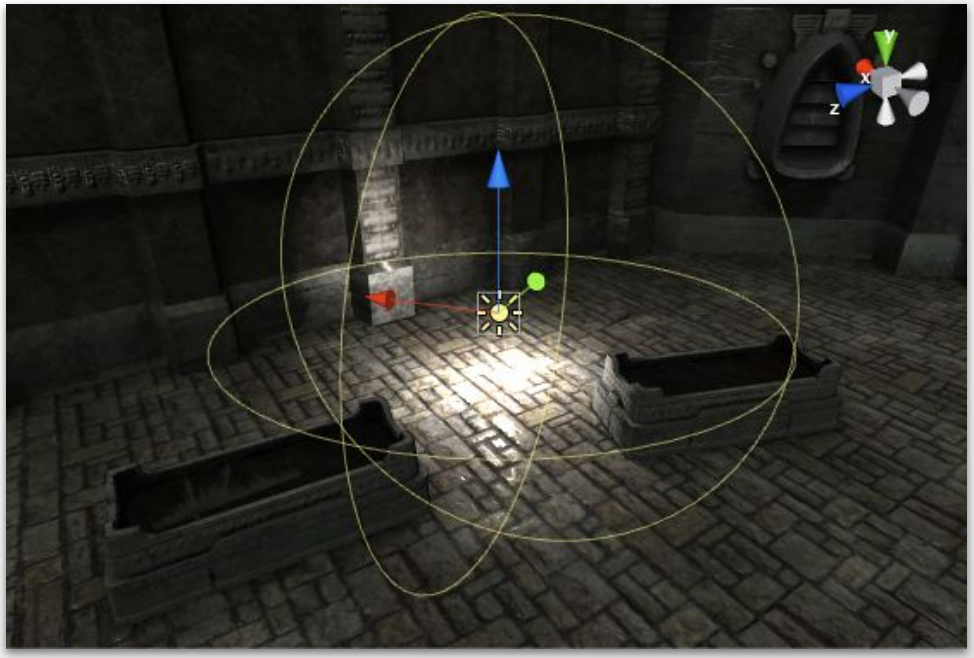


Lots of point lights ? integral ?

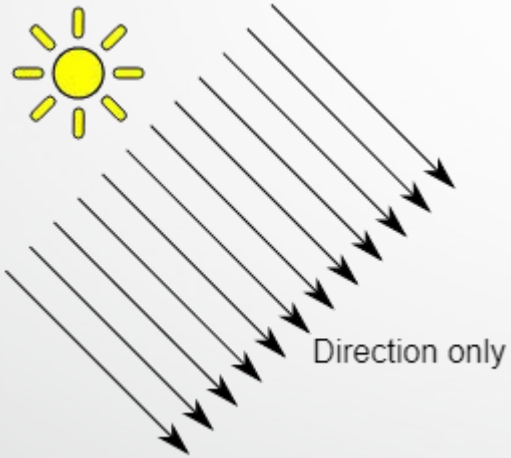


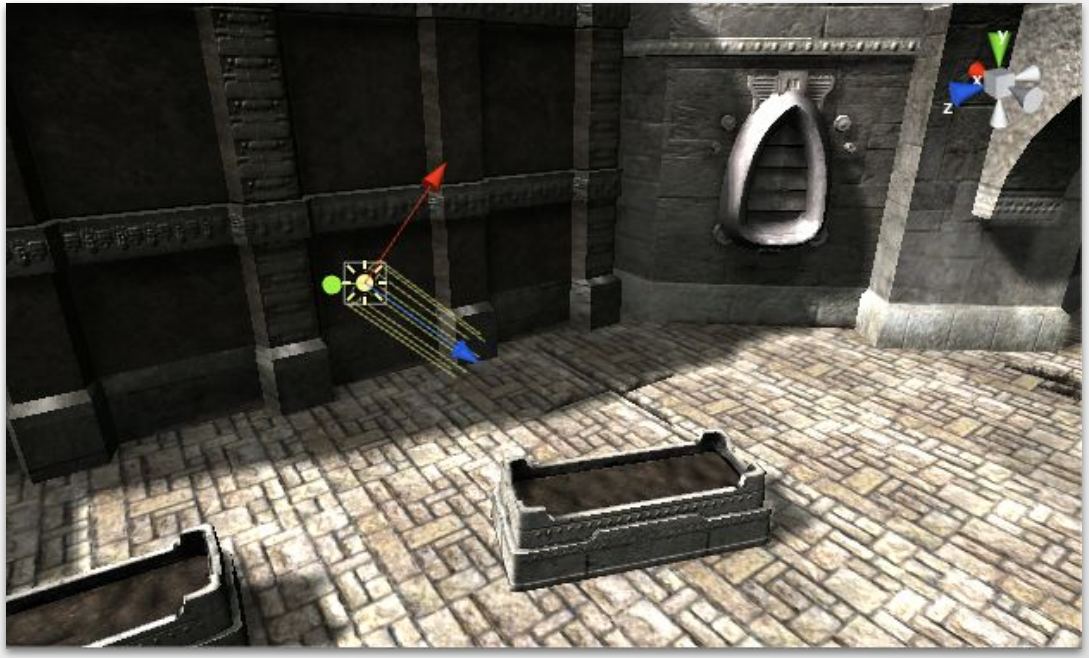
Point light



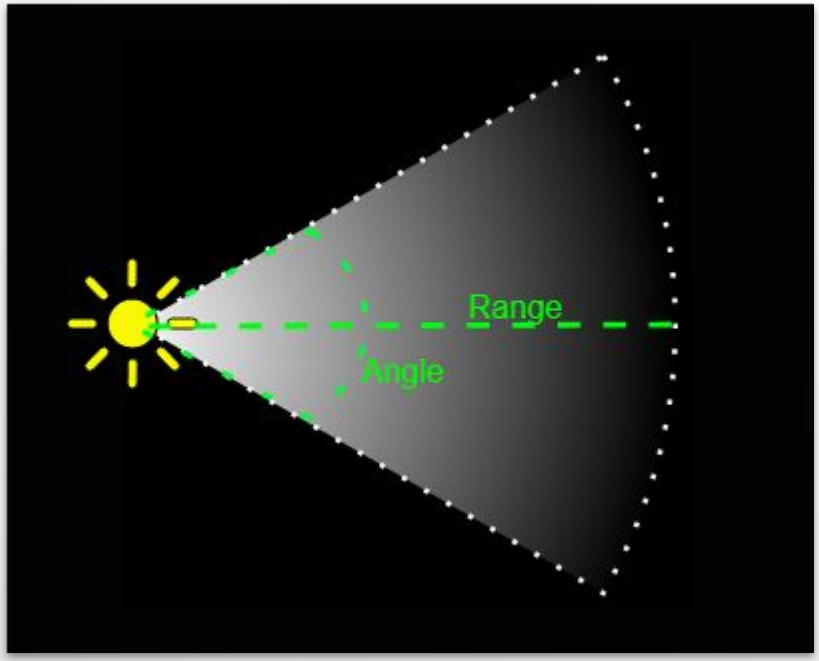


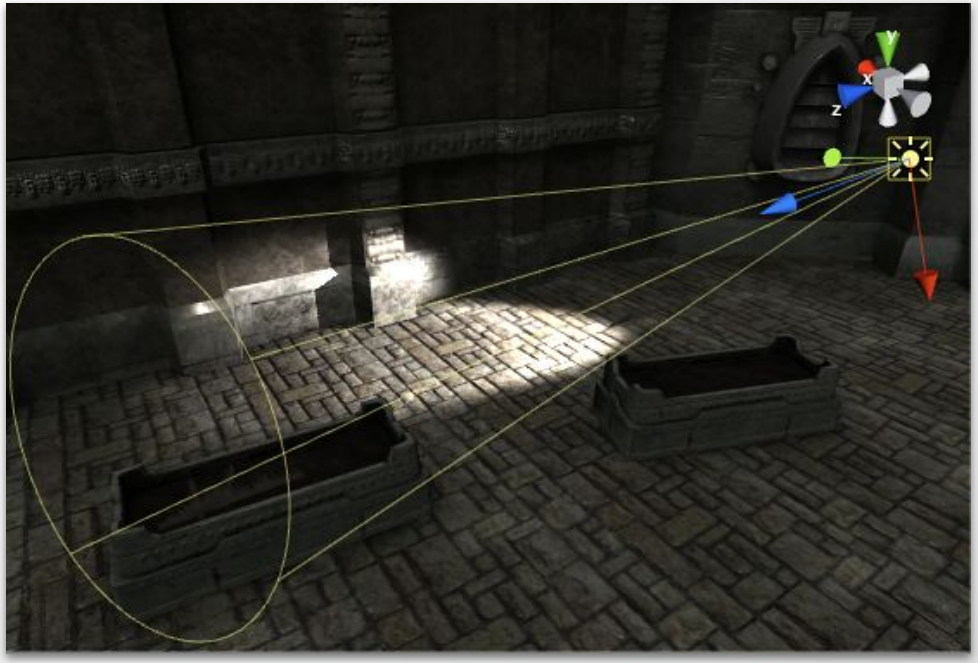
Directional light





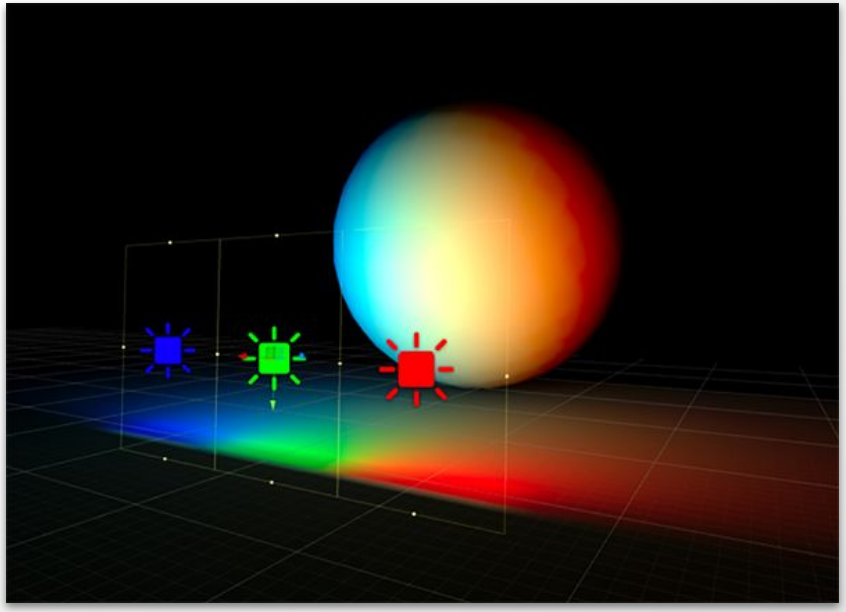
Spot light



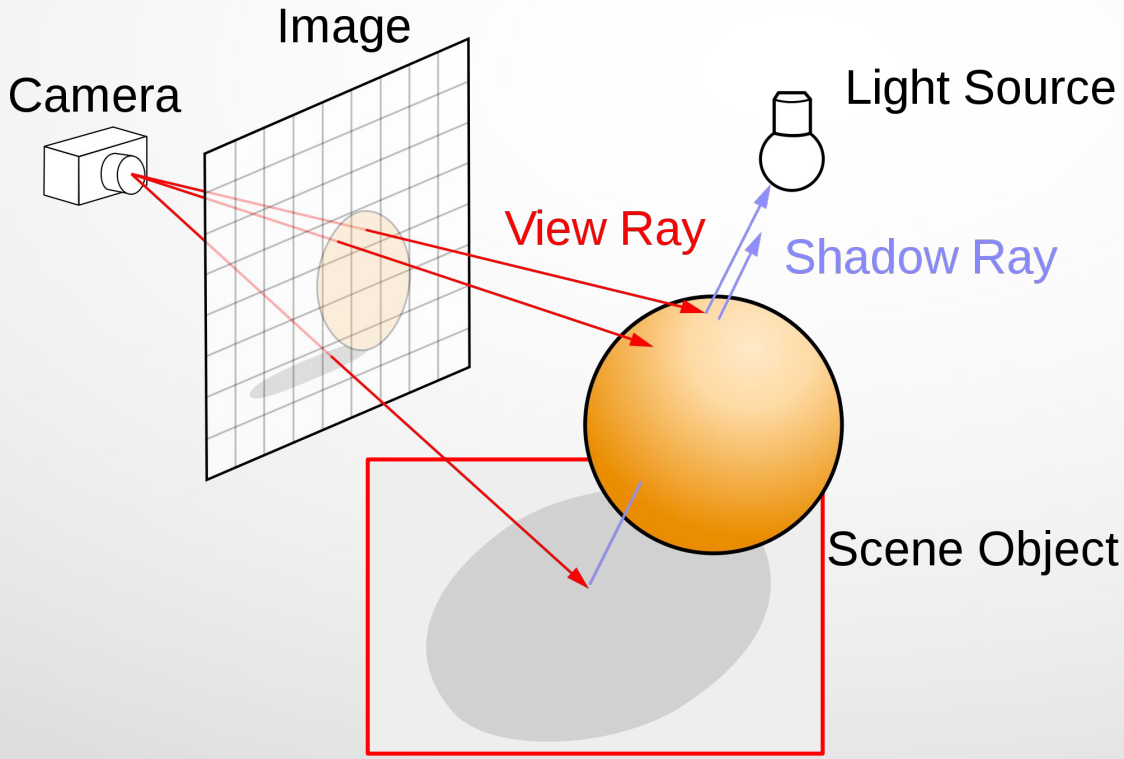


FeisStu

Area light



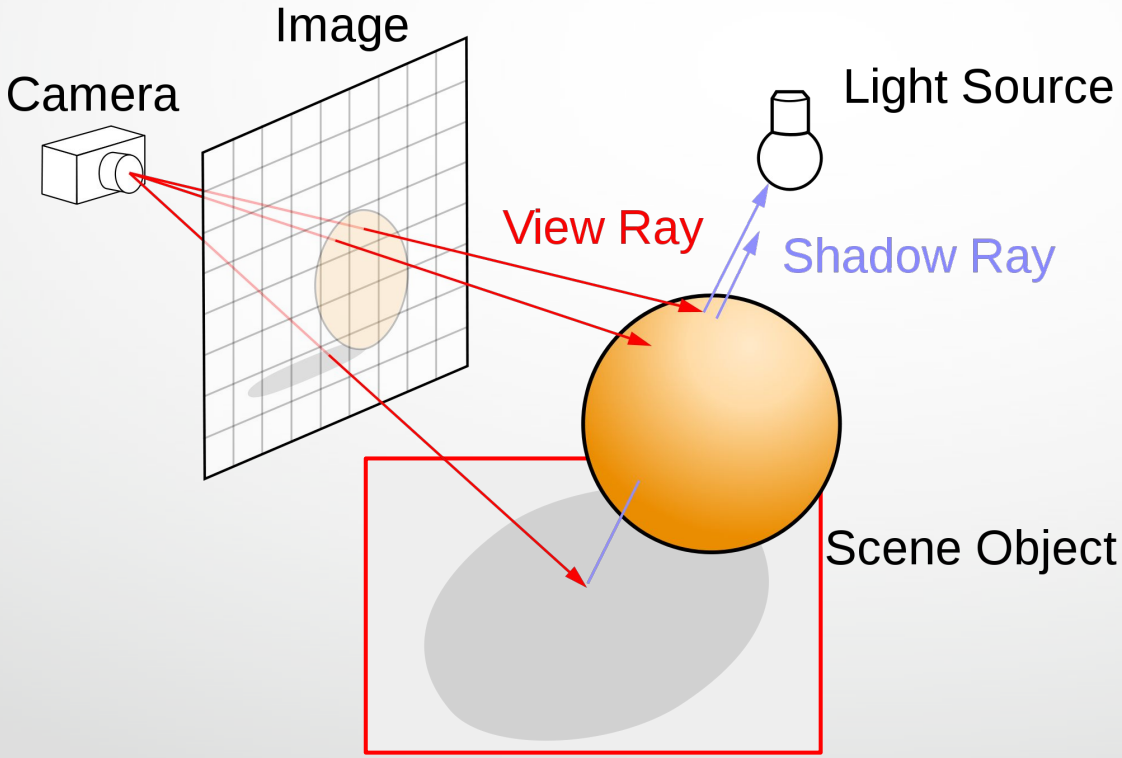
Shadow ?



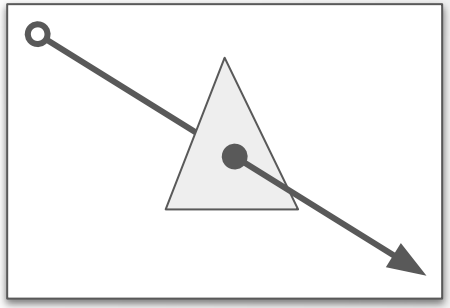
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Shadow ?



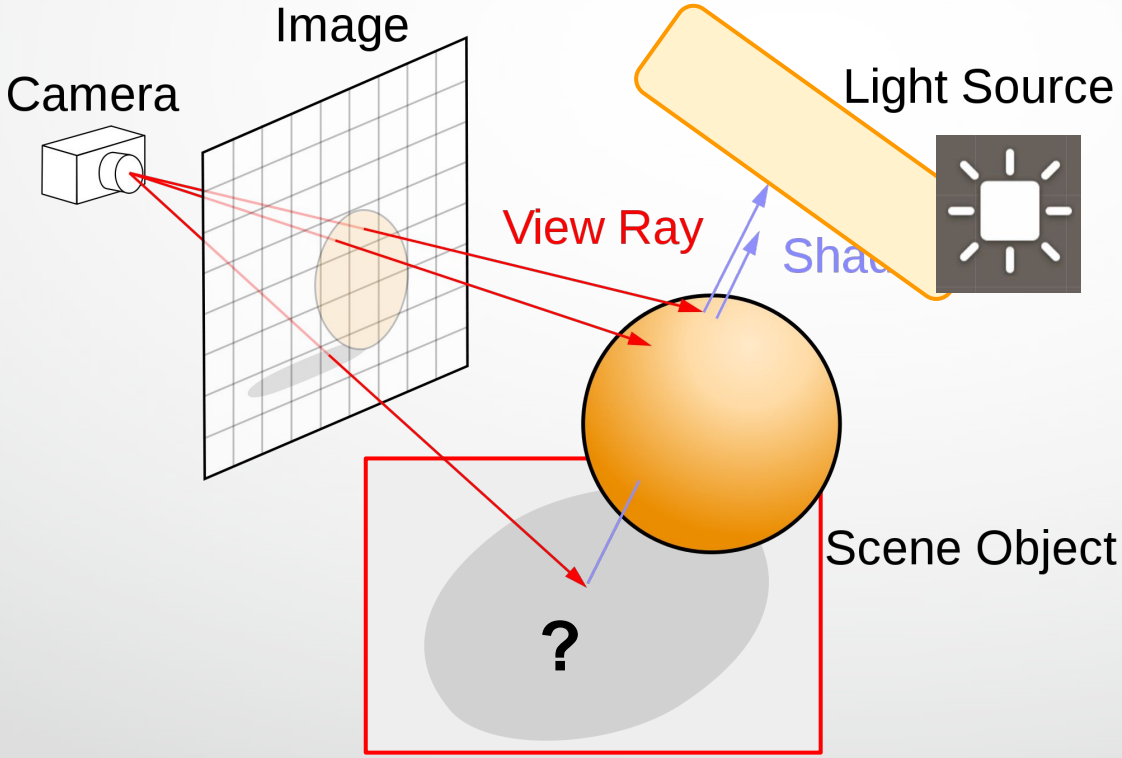
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

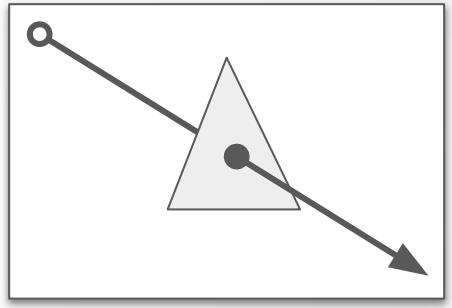


shadow rays

FeisStu

Shadow ?

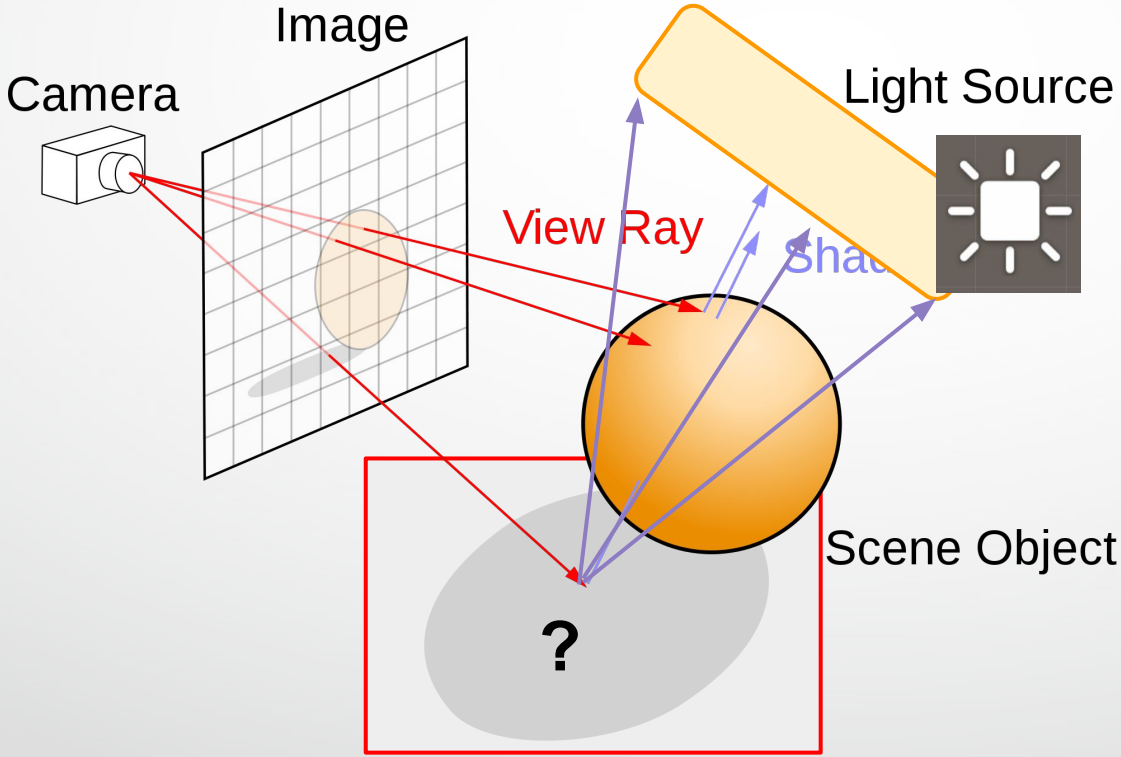




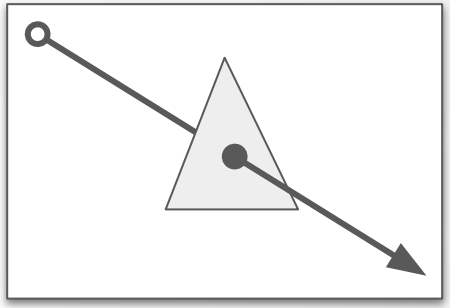
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Shadow ?

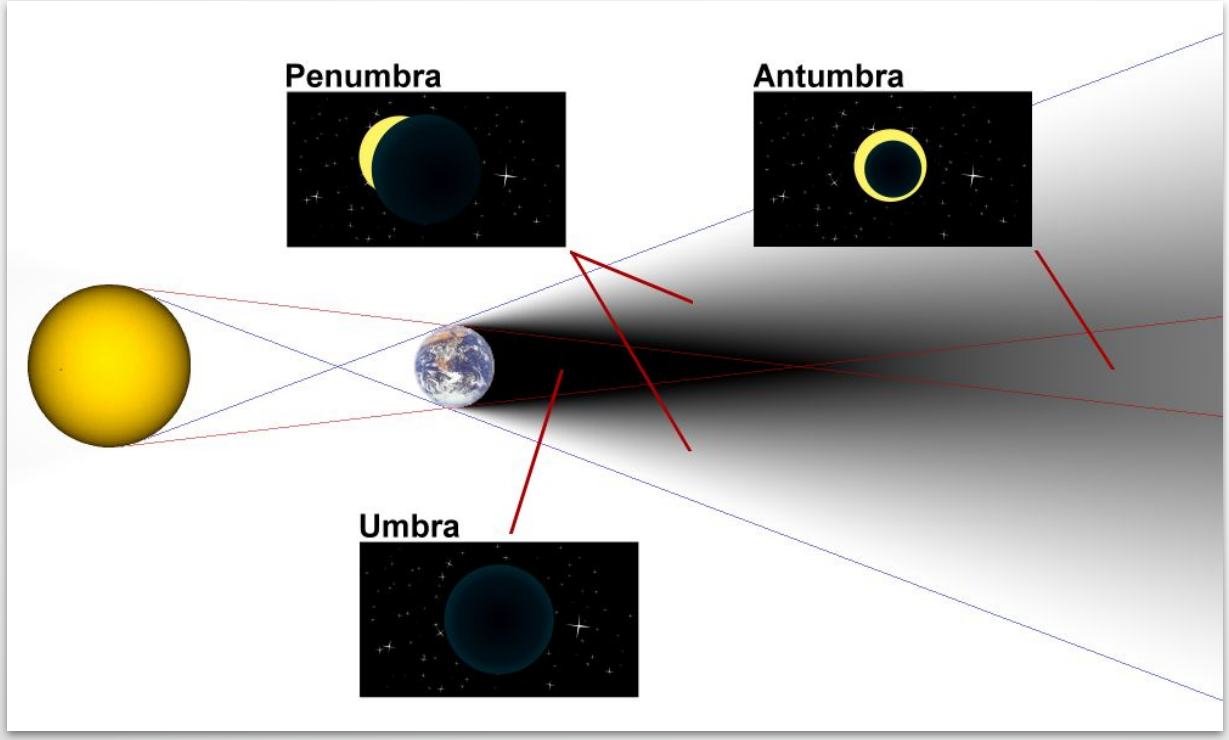


https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg



shadow rays ?

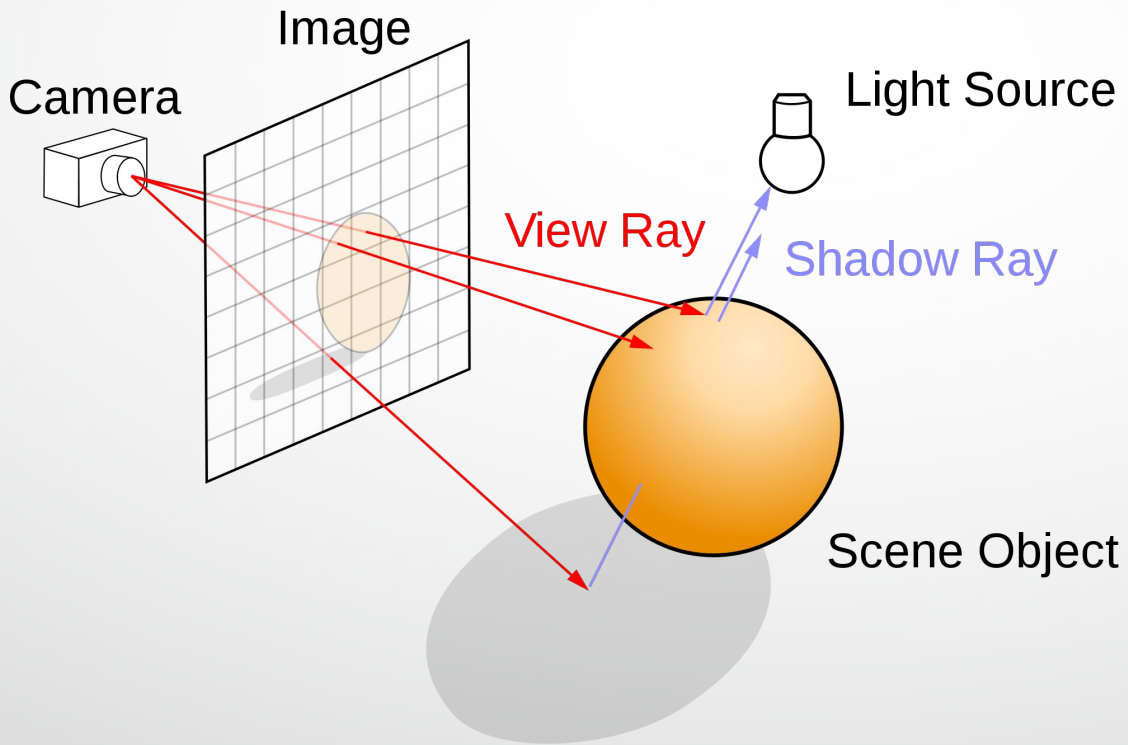
Soft shadows



https://en.wikipedia.org/wiki/Shadow

FeisStu

Direct illumination



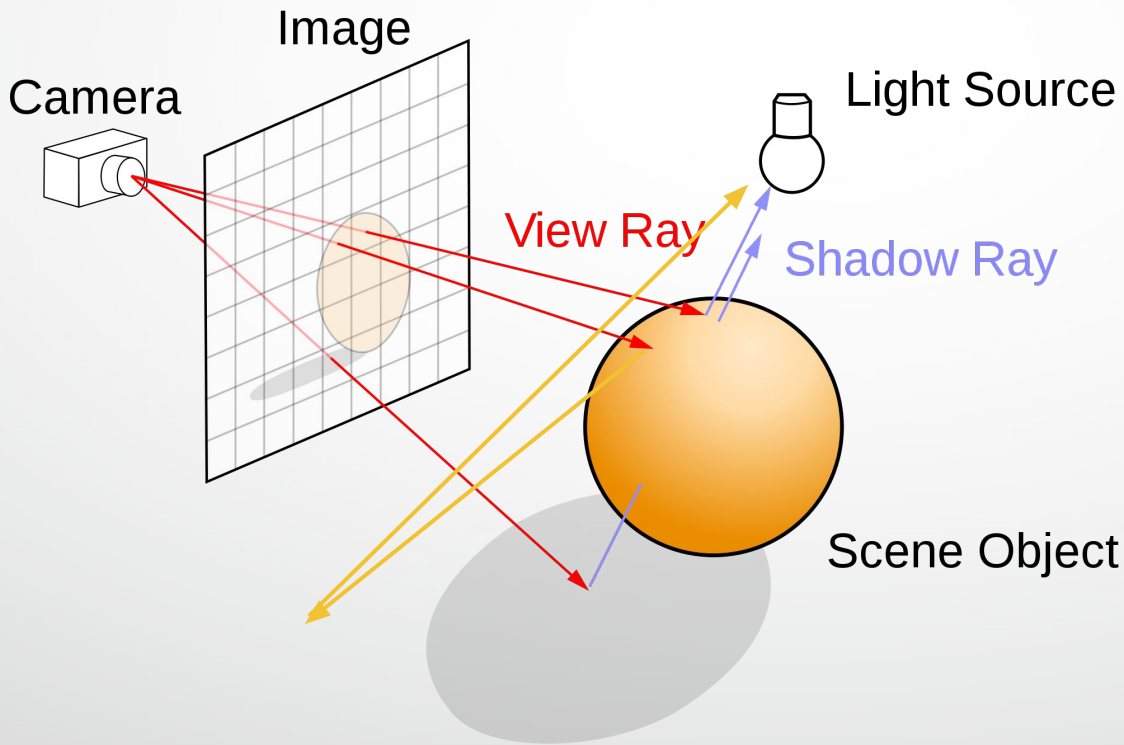
Number of lights

Hard / Soft shadows (Light type)

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Indirect illumination



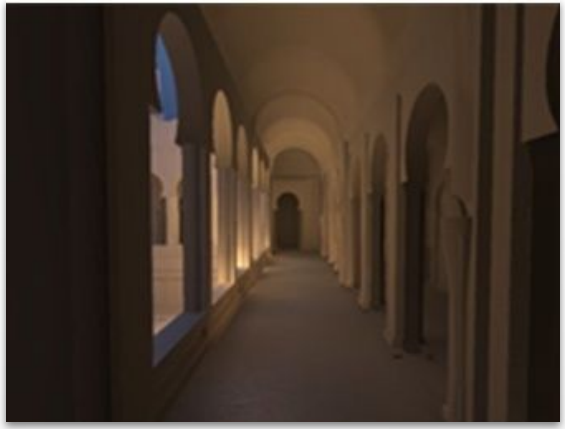
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Direct vs. indirect illumination

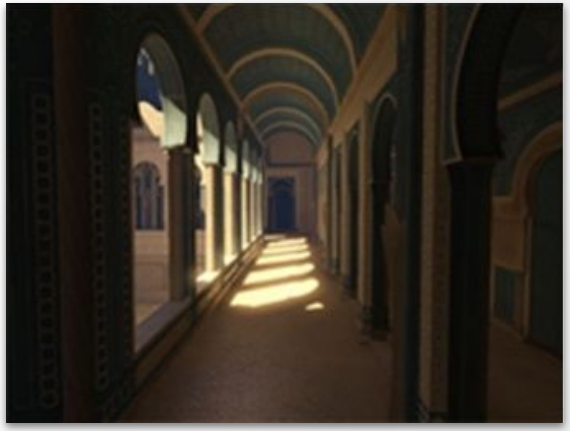
Direct

+



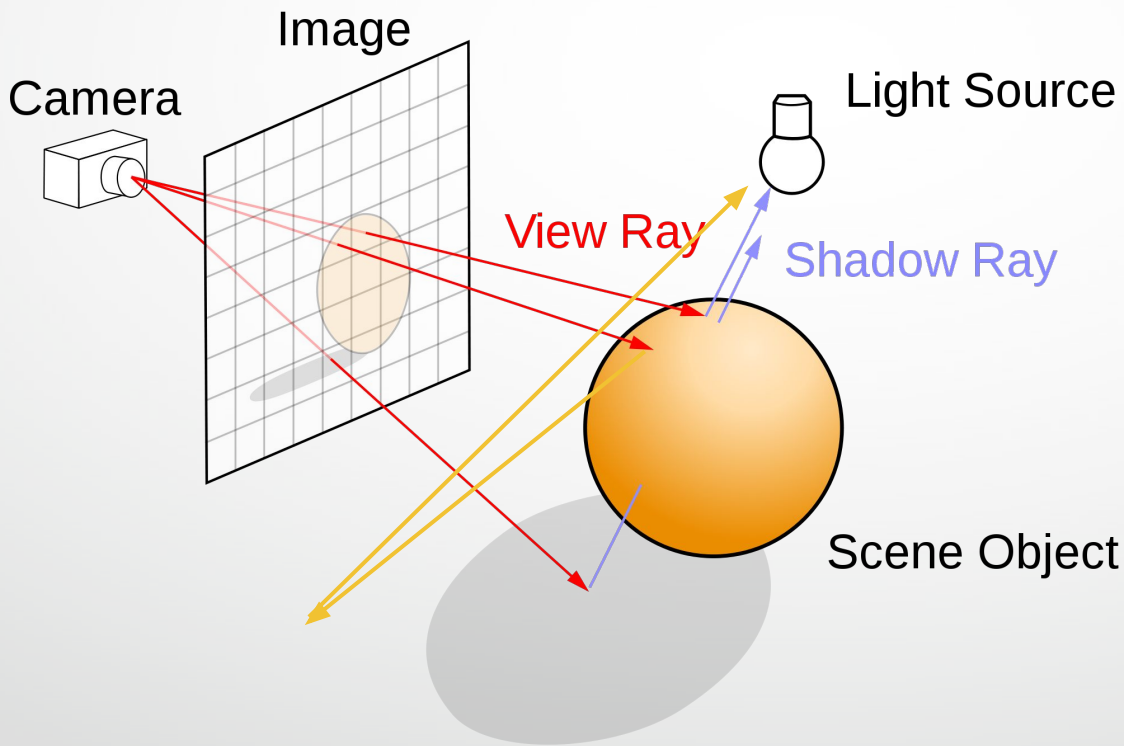
Indirect

=



Final

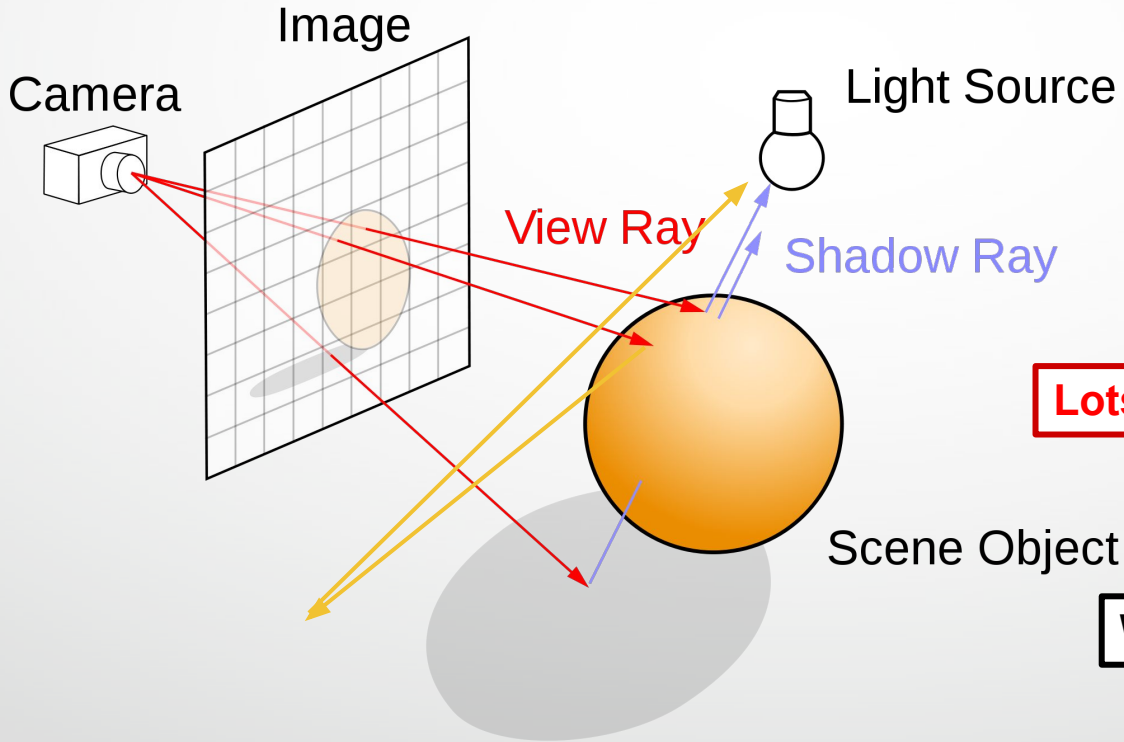
Global illumination = Indirect illumination

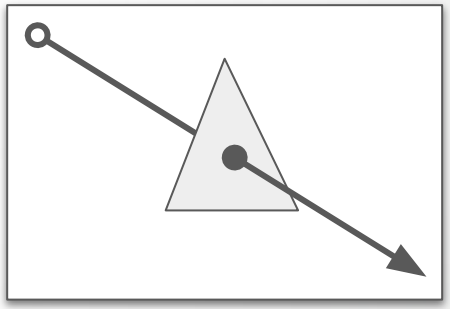


https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Global illumination = Indirect illumination





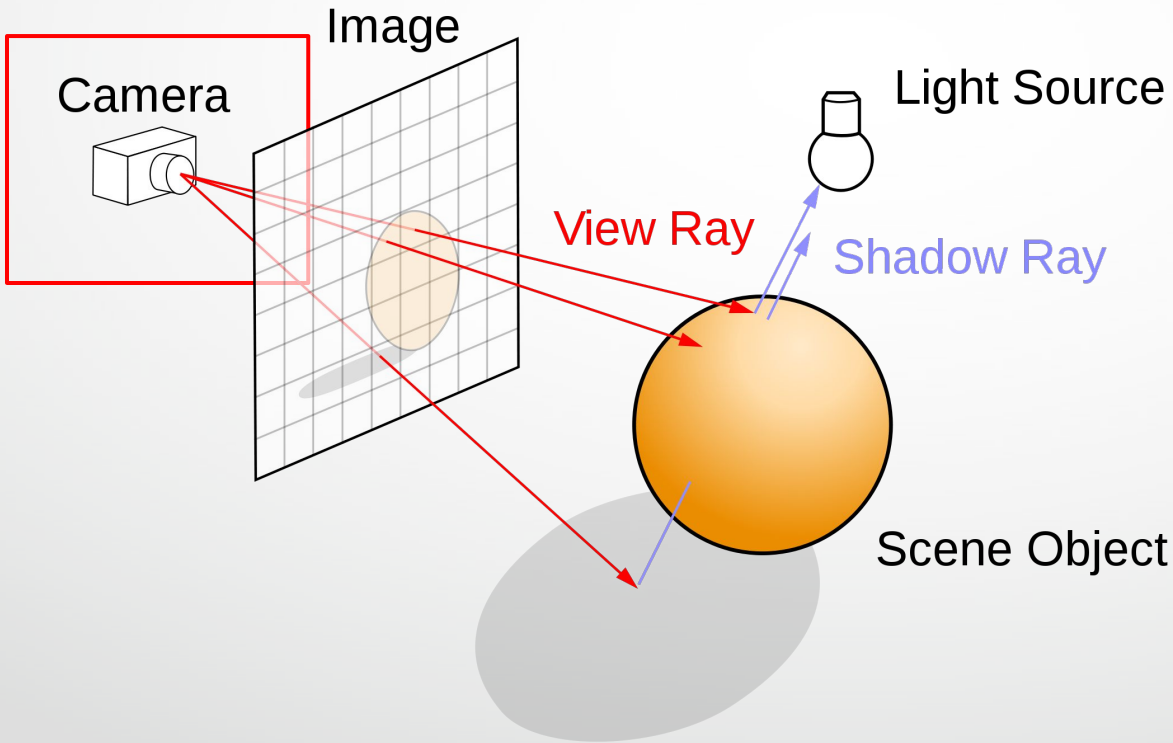
Lots of ray intersection tests

We'll talk about this later.

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Camera ?



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

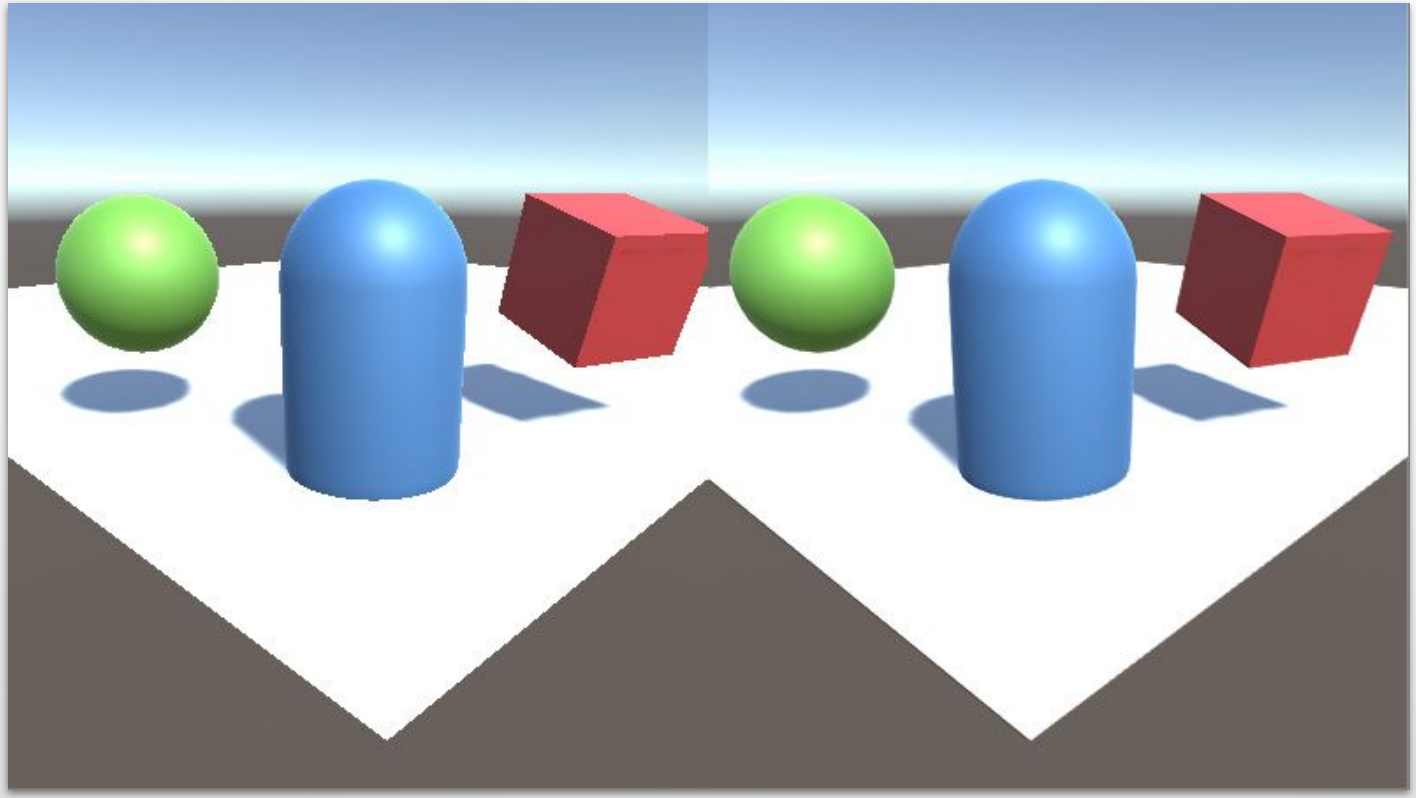
FeisStu

Camera

- Anti-aliasing
- Depth of field
- Auto exposure
- White balance
- Color adjustments
- Lens distortion
- Grain

- Motion blur
- HDR (High-dynamic-range)
  - Bloom
  - Tonemapping
- ...

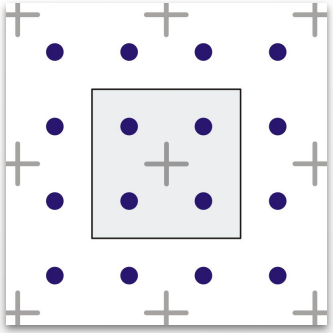
Anti-aliasing



Anti-aliasing (cont'd)

1) **Supersampling**: casting multiple rays/samples in a pixel

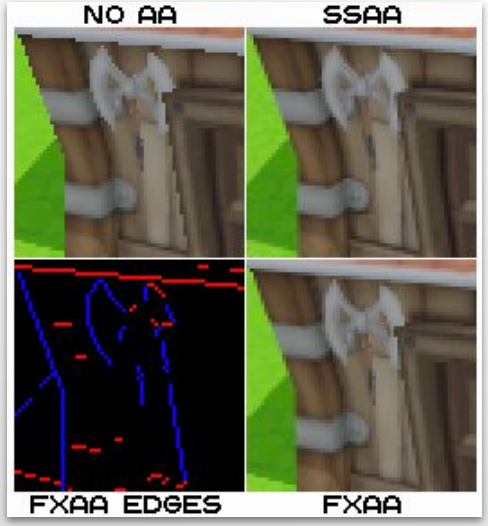
https://en.wikipedia.org/wiki/Fast_approximate_anti-aliasing



https://en.wikipedia.org/wiki/Supersampling

Anti-aliasing (cont'd)

1) **Supersampling**: casting multiple rays/samples in a pixel
2) **Approximation**: smoothing undesirable jagged edges

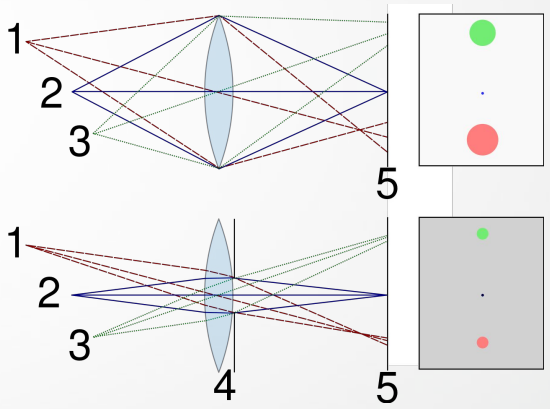


https://en.wikipedia.org/wiki/Fast_approximate_anti-aliasing

FeisStu

Depth of Field

https://en.wikipedia.org/wiki/File:Dof_blocks_f4_0.jpg

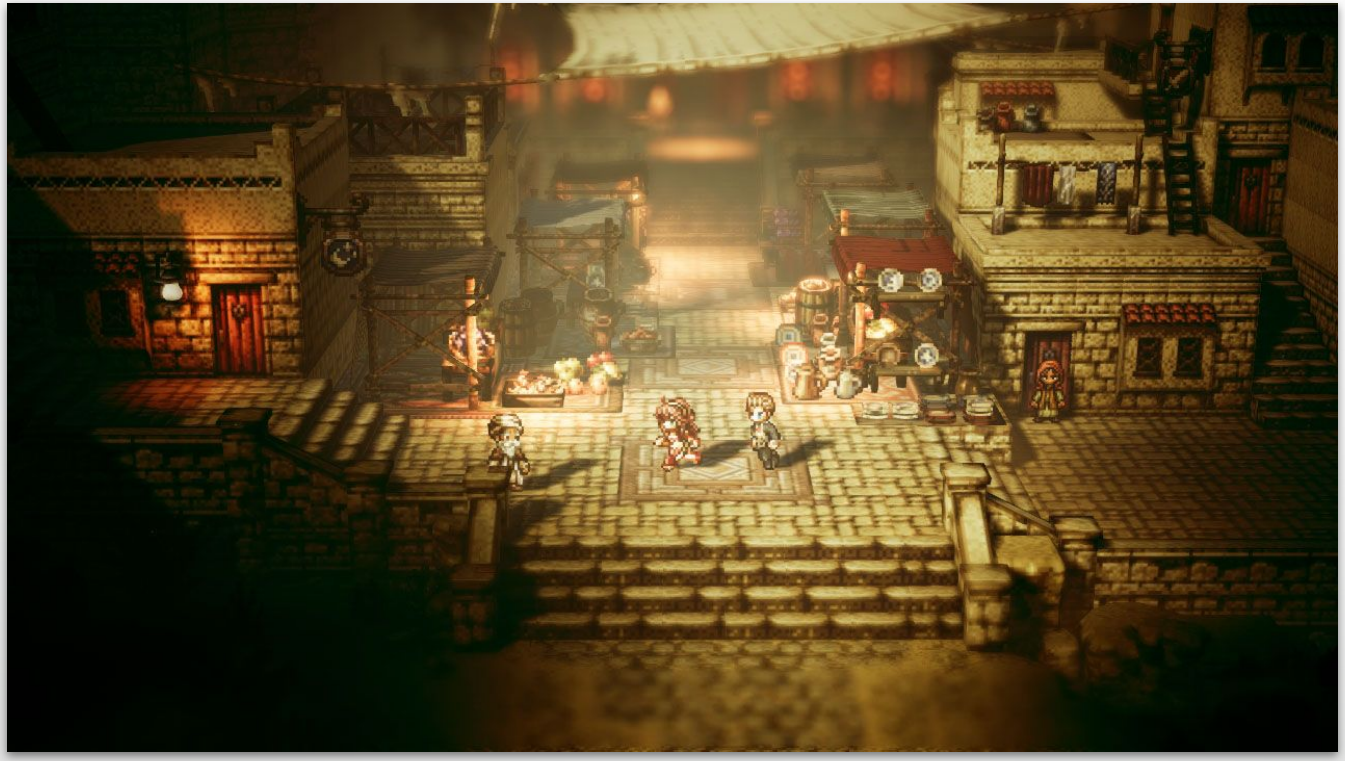


1) **Lots of rays/samples**

2) **Approximation**

FeisStu

Depth of Field (cont'd)



https://www.nintendo.com/games/detail/octopath-traveler-switch/

FeisStu

Bloom

https://en.wikipedia.org/wiki/Bloom_(shader_effect)

FeisStu

Post Processing (before)

https://docs.unity3d.com/Manual/PostProcessingOverview.html

FeisStu

Post Processing (after)

https://docs.unity3d.com/Manual/PostProcessingOverview.html

FeisStu

Post Processing (package for Built-in RP)

unity Manual Scripting API Changelog License Search docs.unity3d.com

Post Processing 2.3.0 v

Enter here to filter...

- Home
- Installation
- Quick-start
- Effects
  - Ambient Occlusion
  - Anti-aliasing
  - Auto Exposure
  - Bloom
  - Chromatic Aberration
  - Color Grading
  - Deferred Fog
  - Depth of Field
  - Grain
  - Lens Distortion
  - Motion Blur
  - Screen Space Reflections
  - Vignette
- Scripting

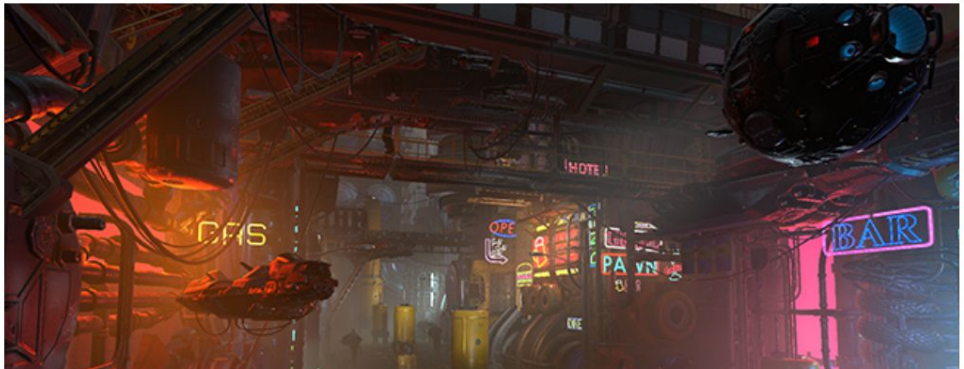
Manual / Home

Home

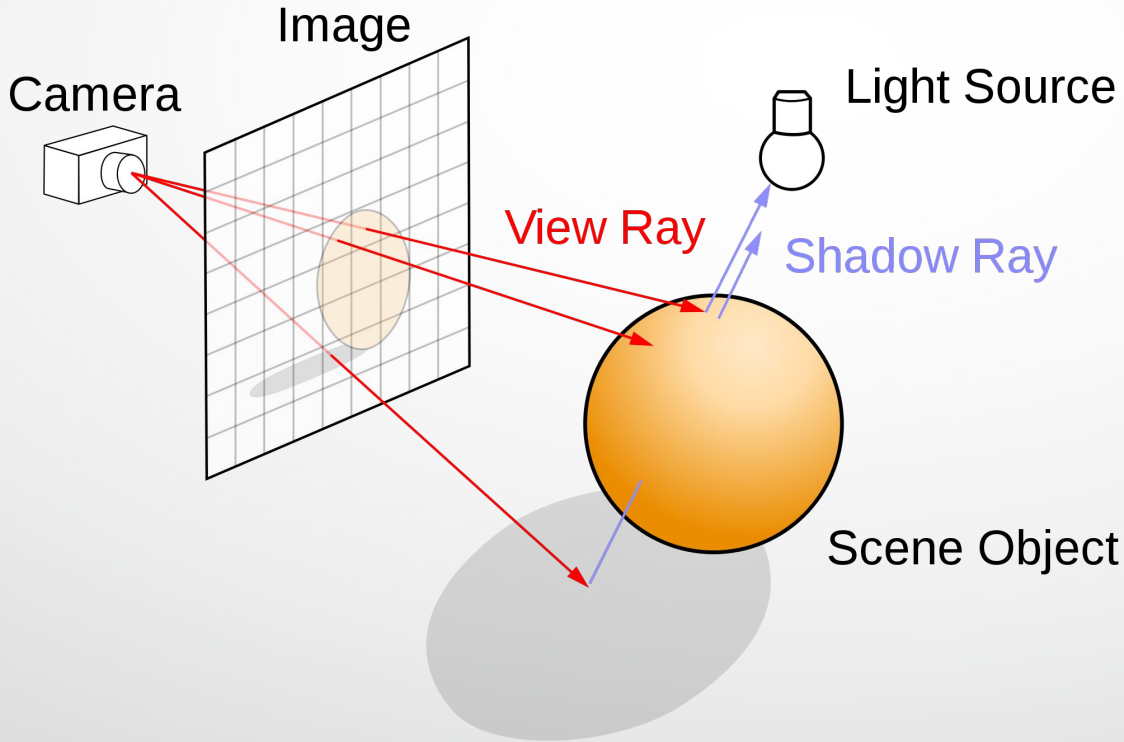
Post-processing is the process of applying full-screen filters and effects to a camera's image buffer before it is displayed to screen. It can drastically improve the visuals of your product with little setup time.

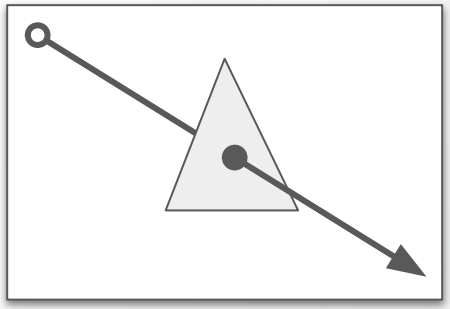
You can use post-processing effects to simulate physical camera and film properties.

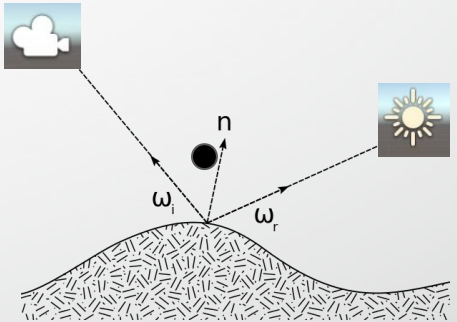
The images below demonstrate a scene with and without post-processing.



Ray tracing



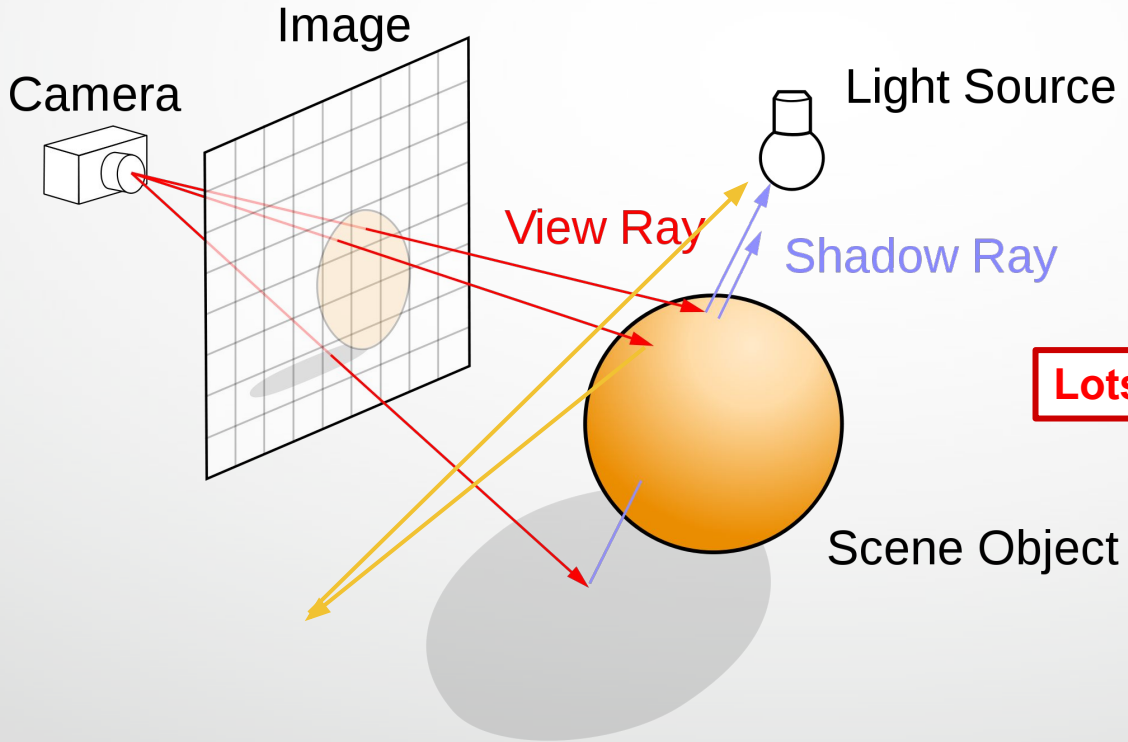


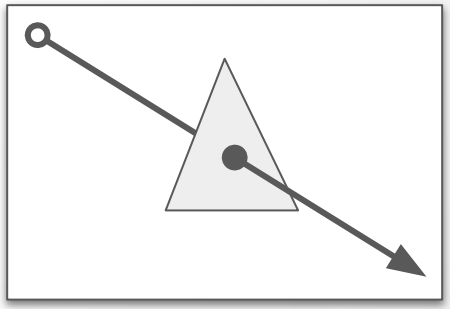


https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Global illumination = Indirect illumination



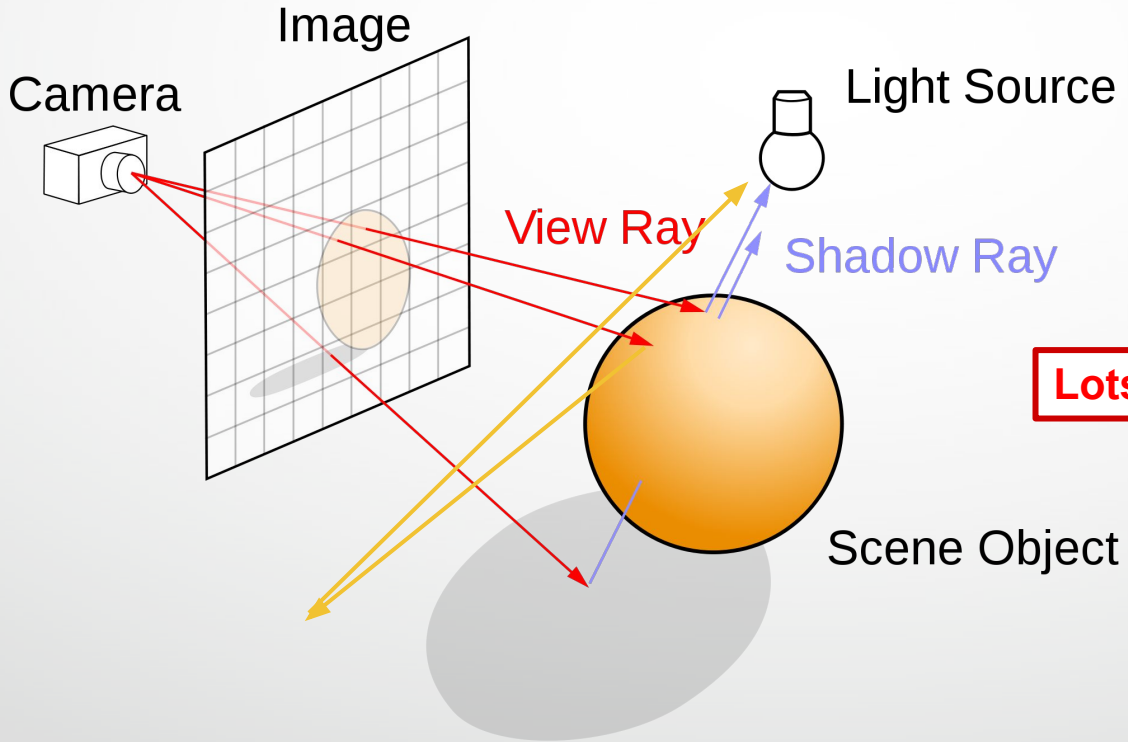


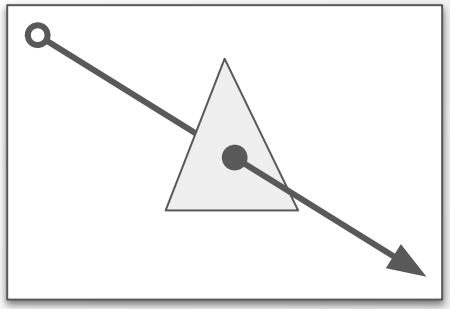
Lots of ray intersection tests

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Global illumination = Indirect illumination





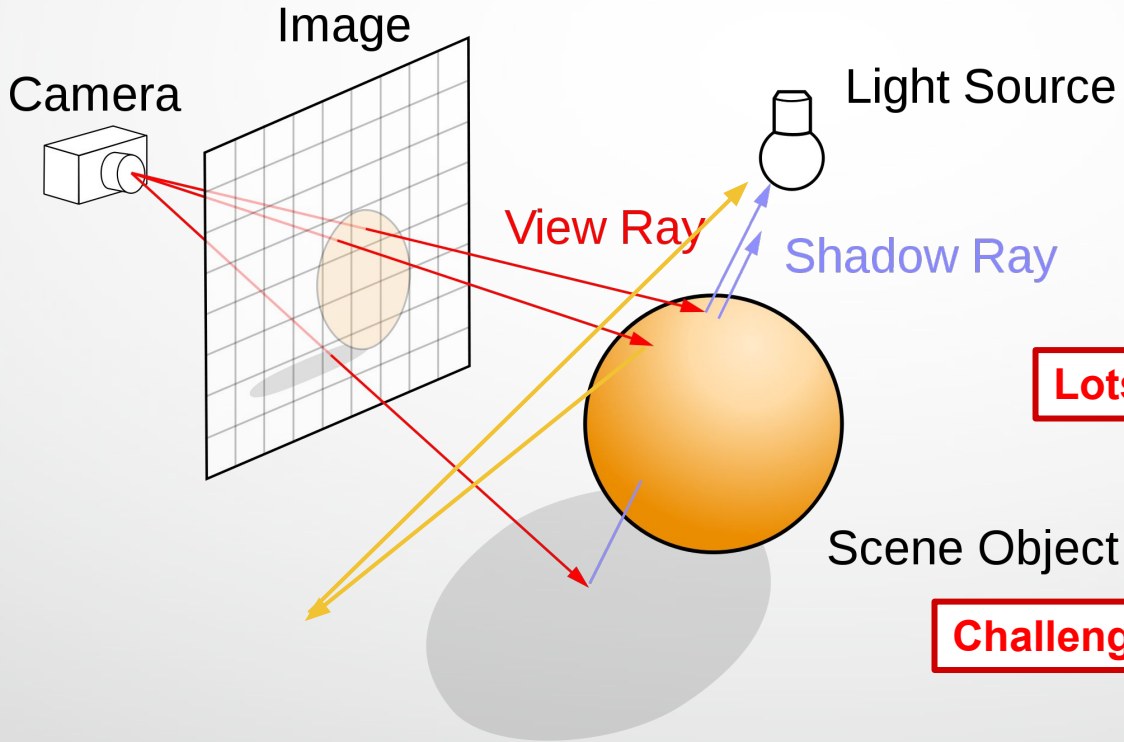
Lots of ray intersection tests

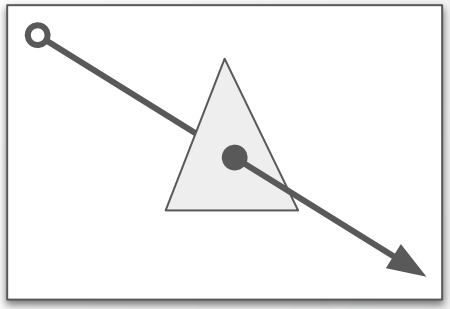
hardware acceleration ?

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Global illumination = Indirect illumination





Lots of ray intersection tests

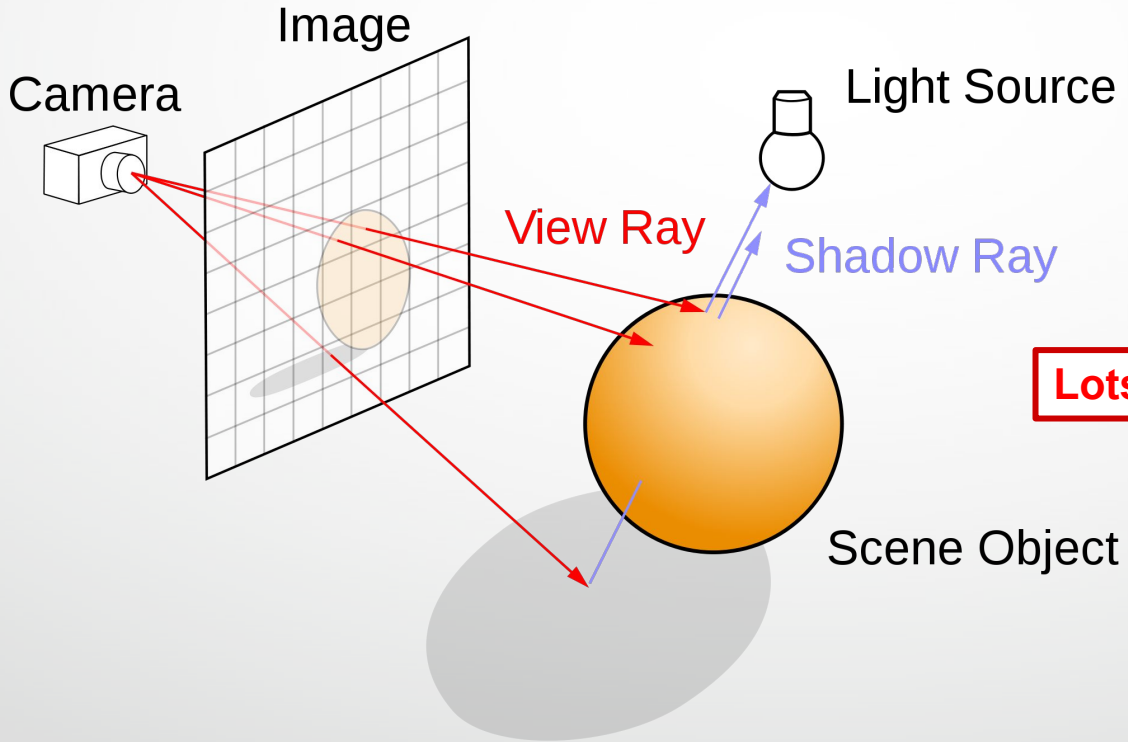
hardware acceleration ?

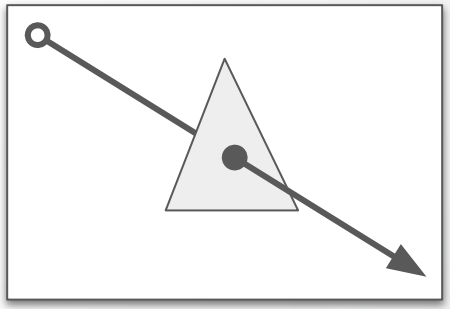
Challenge: divergence of ray paths

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Direct illumination





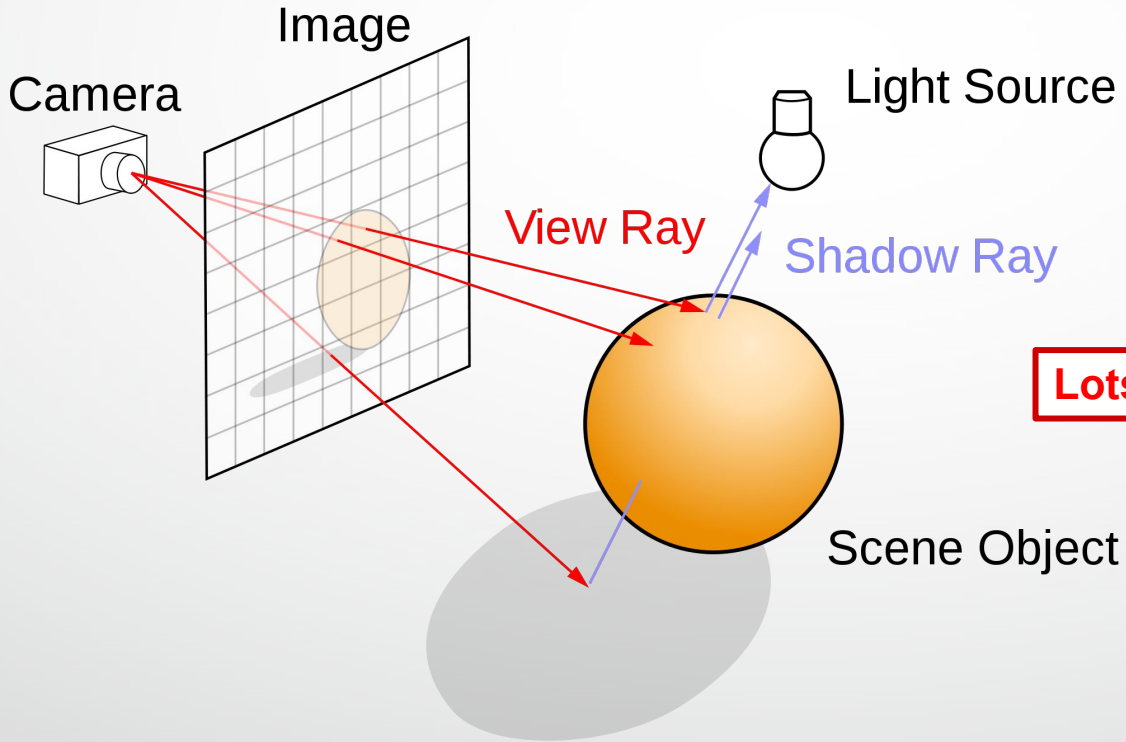
Lots of ray intersection tests

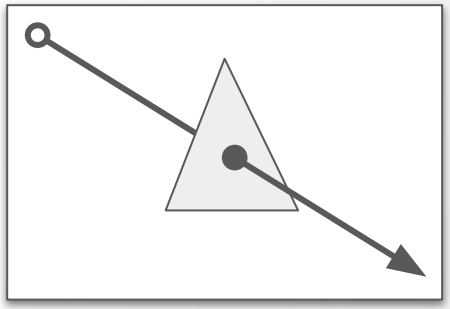
hardware acceleration ?

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

Direct illumination





Lots of ray intersection tests

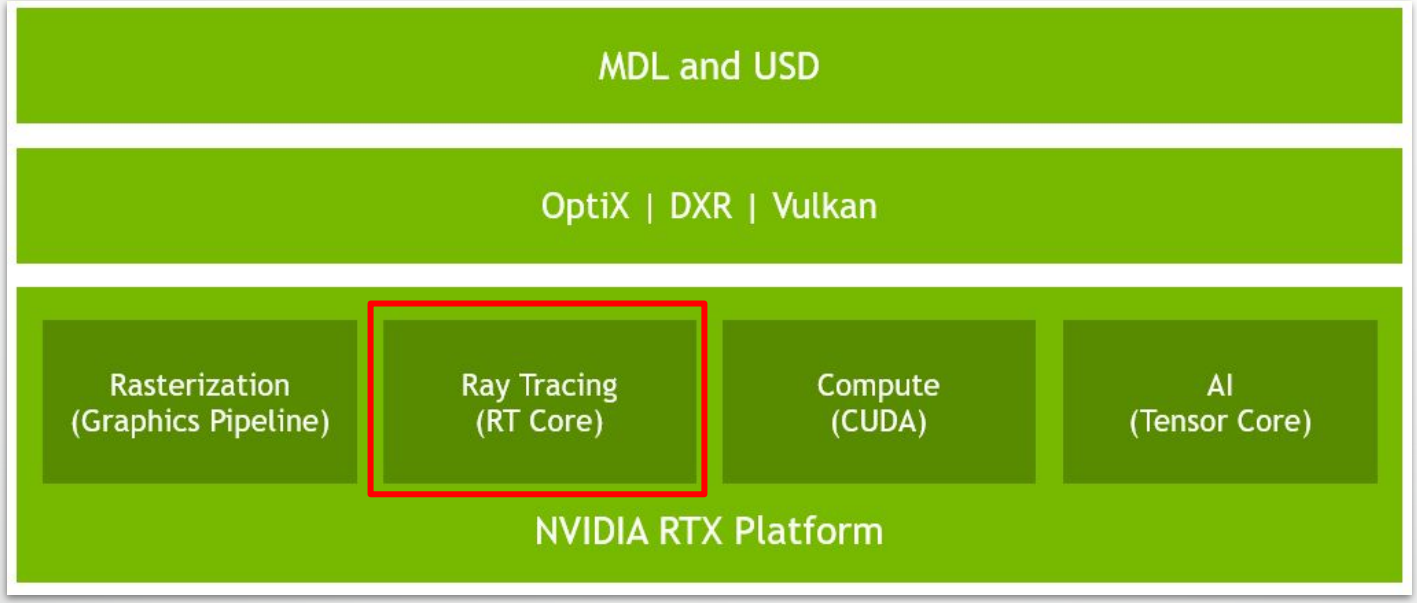
hardware acceleration ?

GPU & VRAM

https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

FeisStu

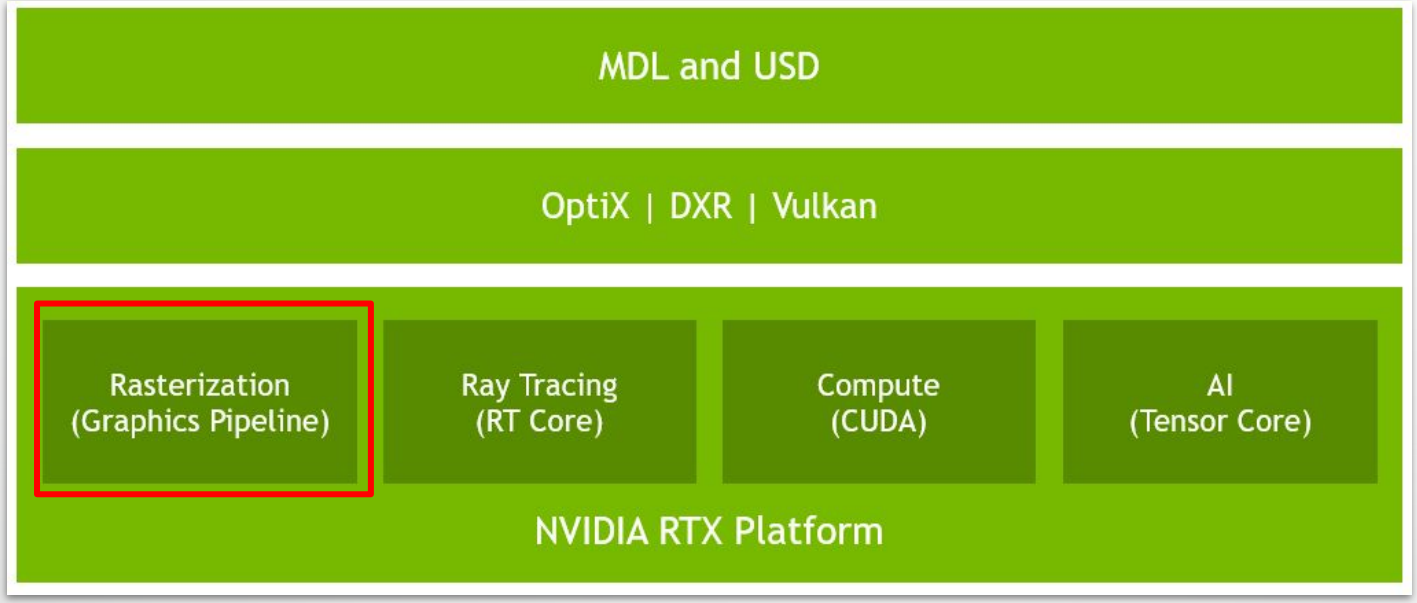
Case study: NVIDIA RTX™ Platform



https://developer.nvidia.com/rtx

FeisStu

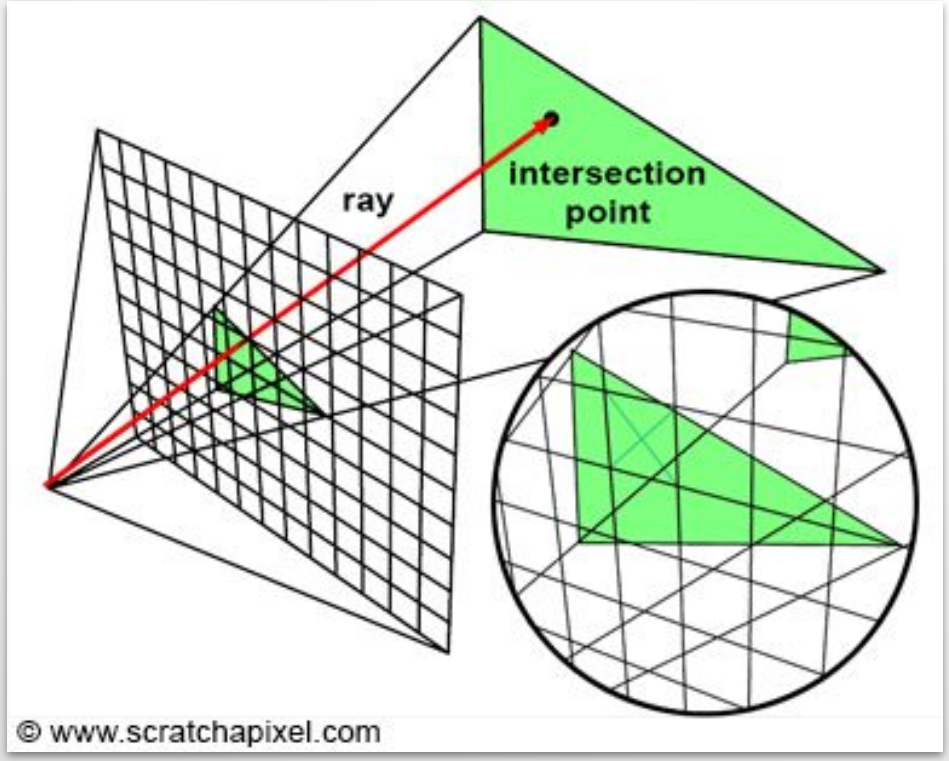
Case study: NVIDIA RTX™ Platform



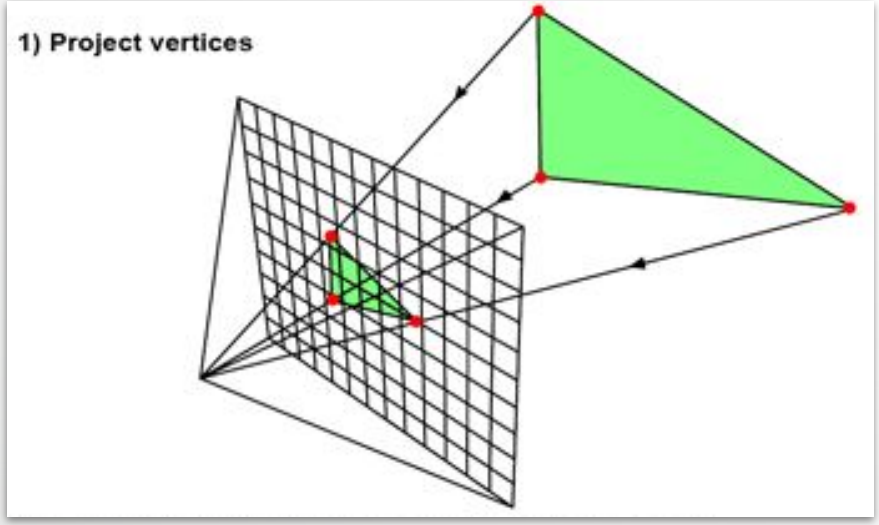
https://developer.nvidia.com/rtx

FeisStu

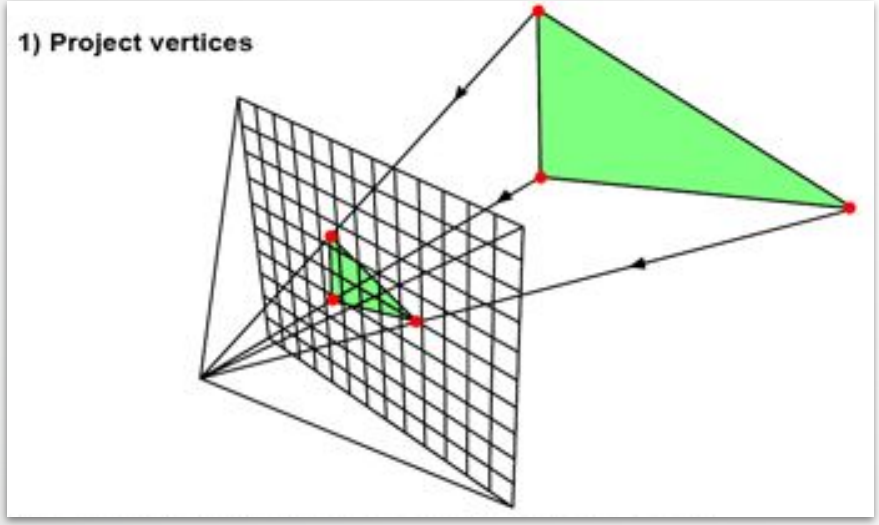
Rasterization

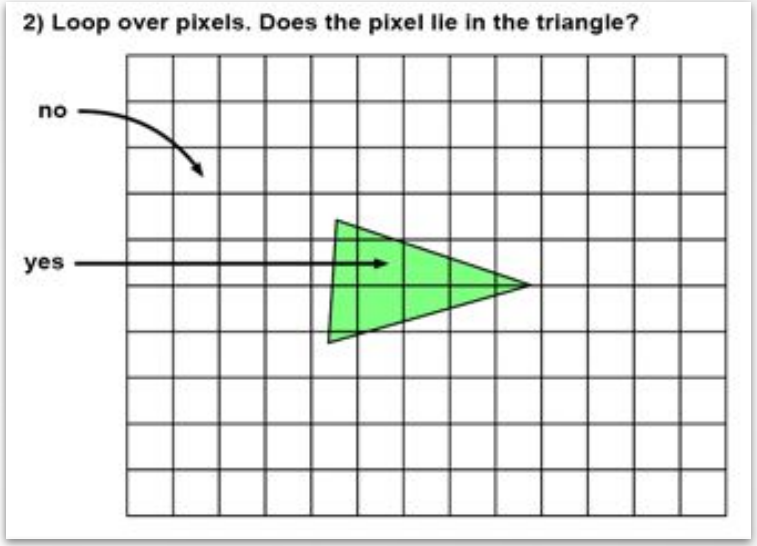


Rasterization (cont'd)

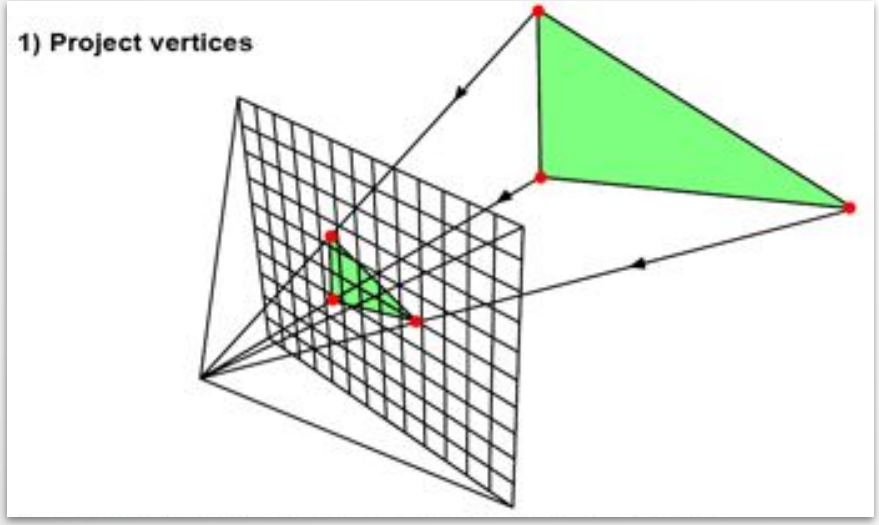


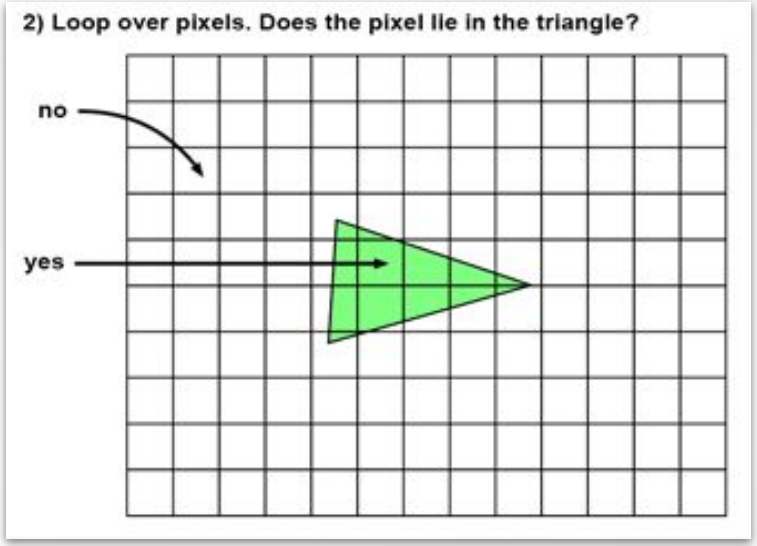
Rasterization (cont'd)

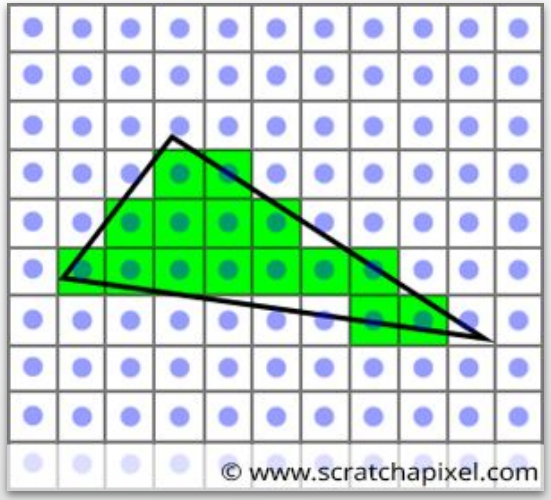




Rasterization (cont'd)

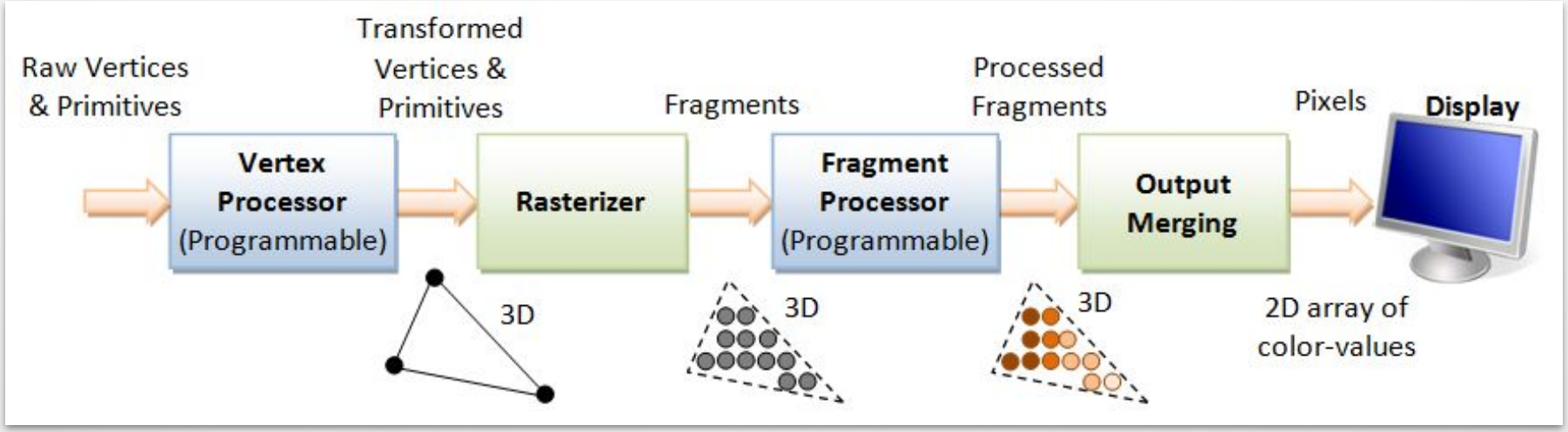


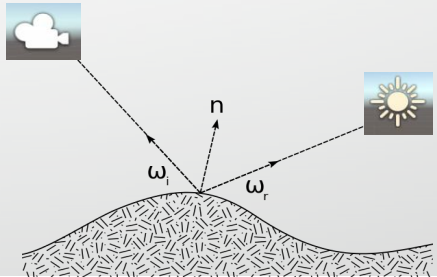




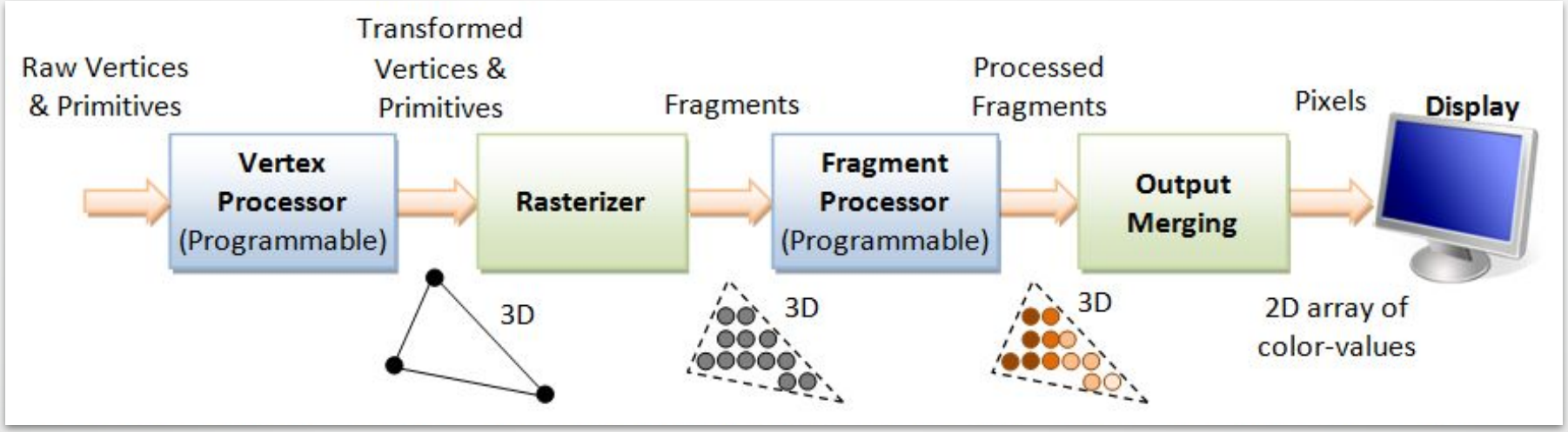
FeisStu

3D Graphics Rendering Pipeline

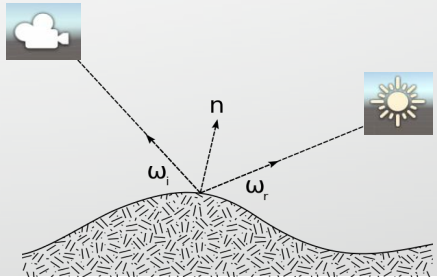




3D Graphics Rendering Pipeline

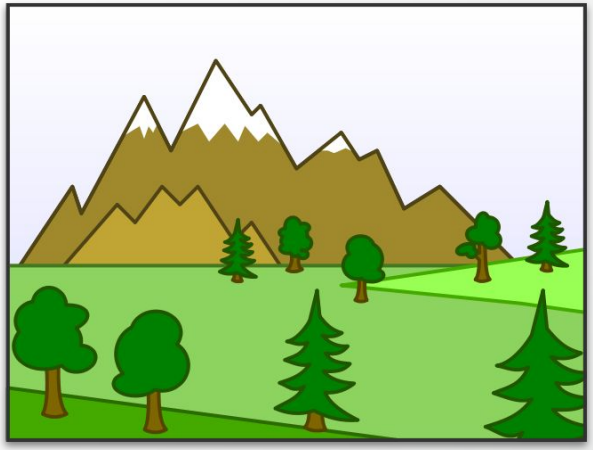


Which triangle/mesh needs to be drawn first ?



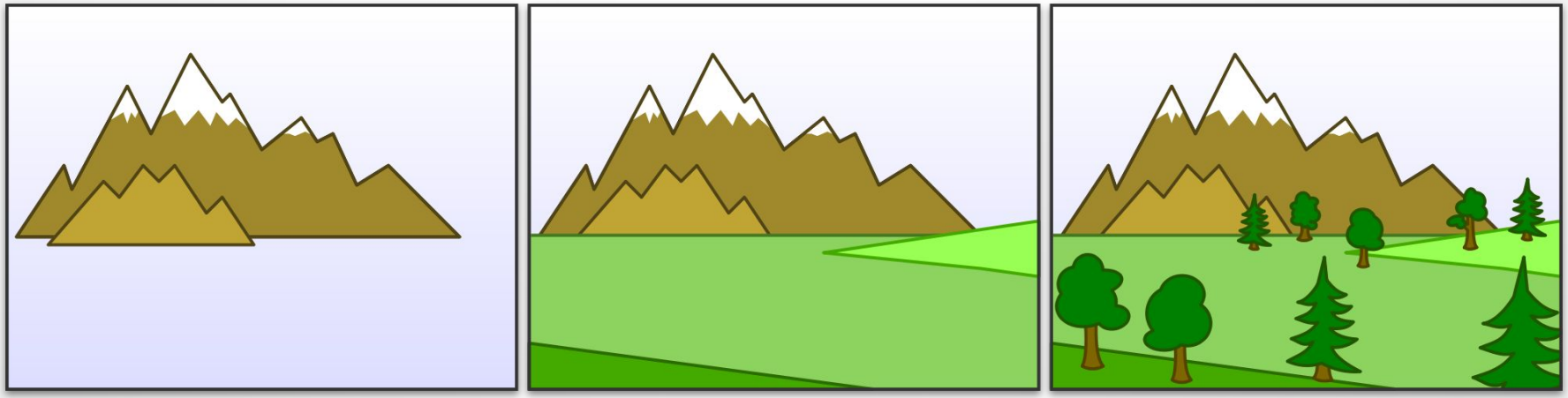
FeisStu

Painter's algorithm



https://en.wikipedia.org/wiki/Painter%27s_algorithm

Painter's algorithm



https://en.wikipedia.org/wiki/Painter%27s_algorithm

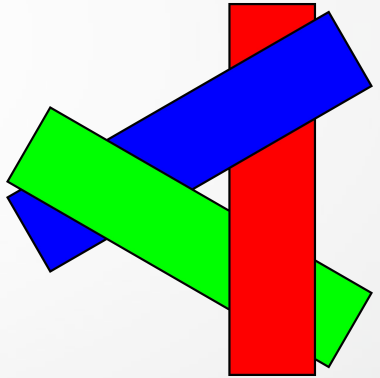
Far

Near

Painter's algorithm (cont'd)

?

?

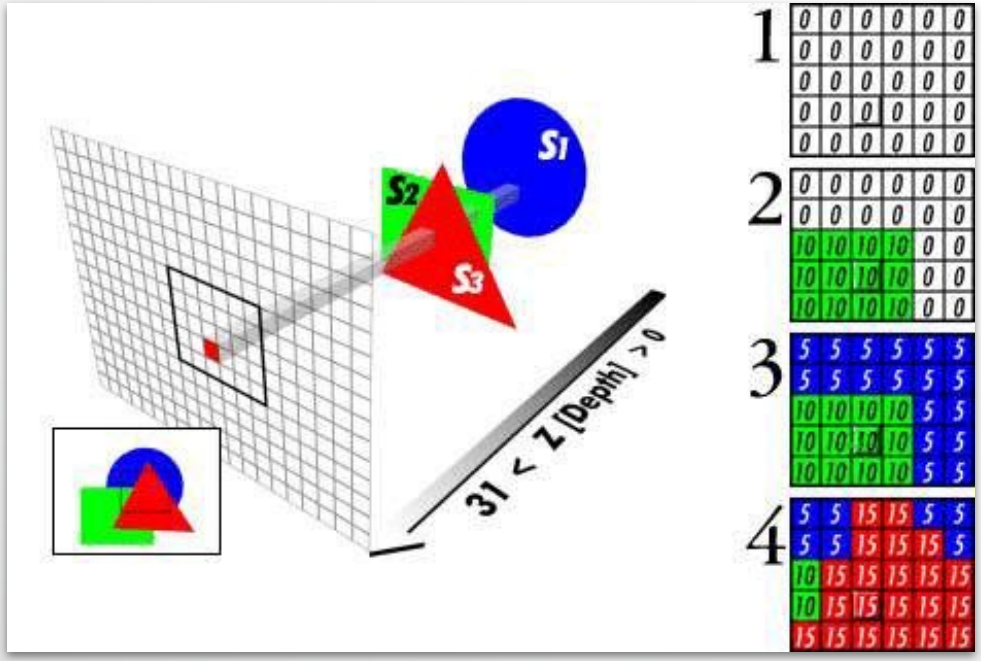


https://en.wikipedia.org/wiki/Painter%27s_algorithm

Far

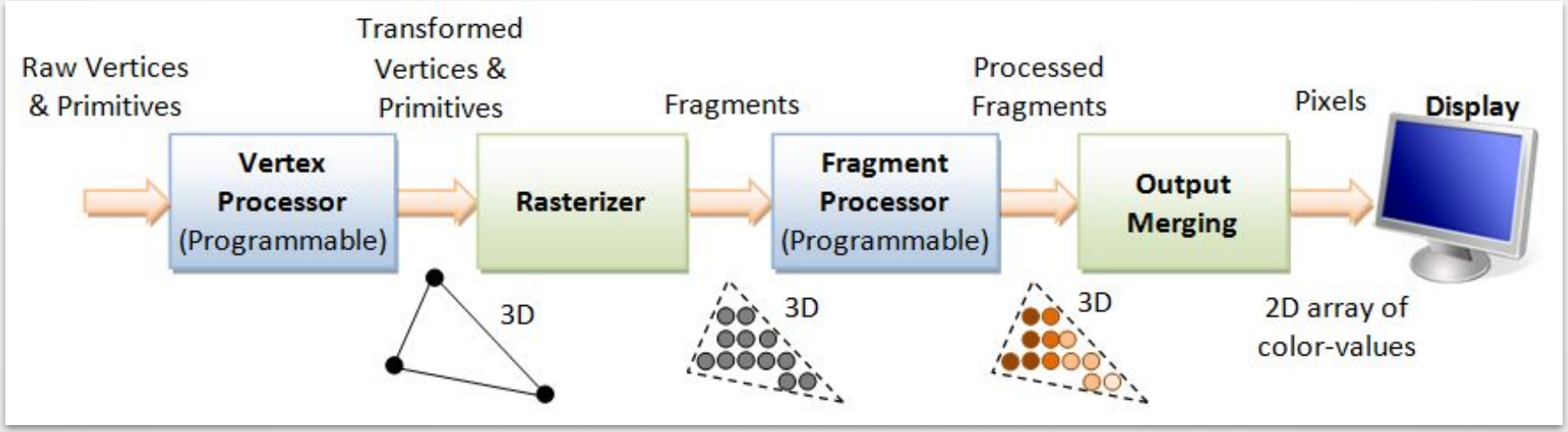
Near

Z (Depth)-buffering

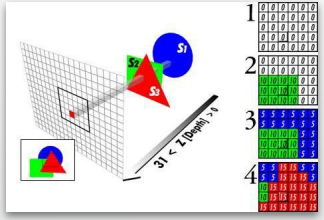


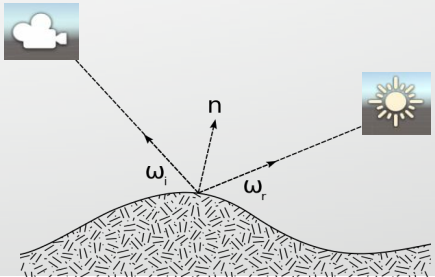
https://www.racketboy.com/retro/about-video-games-rasterization-and-z-buffer

3D Graphics Rendering Pipeline



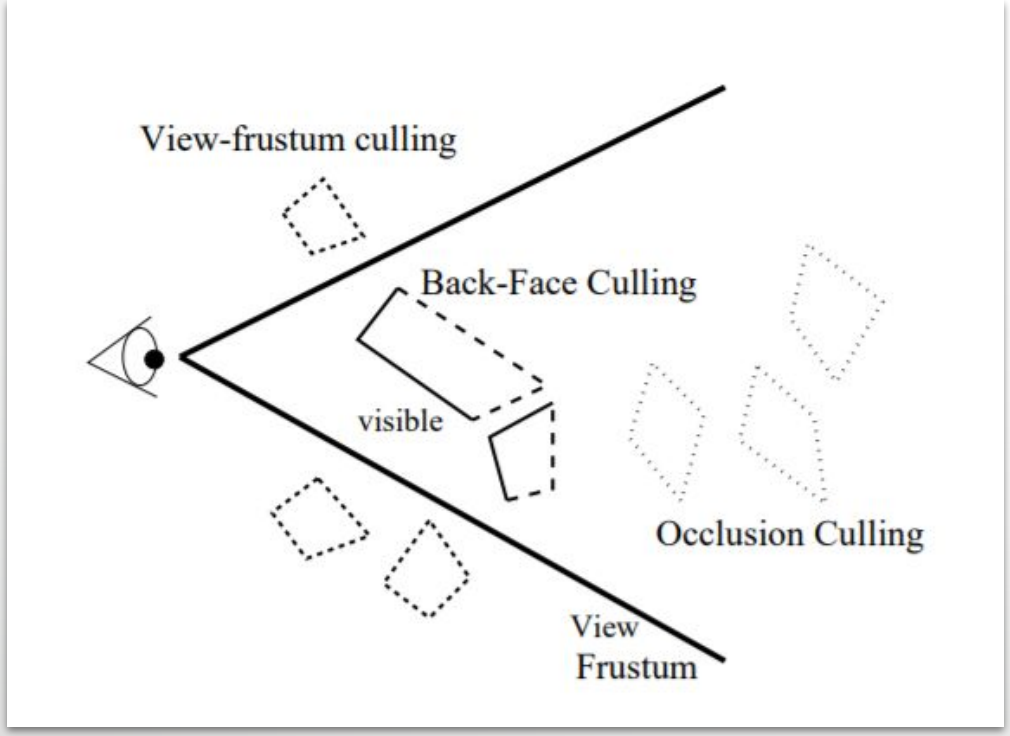
Are all triangles visible ?





FeisStu

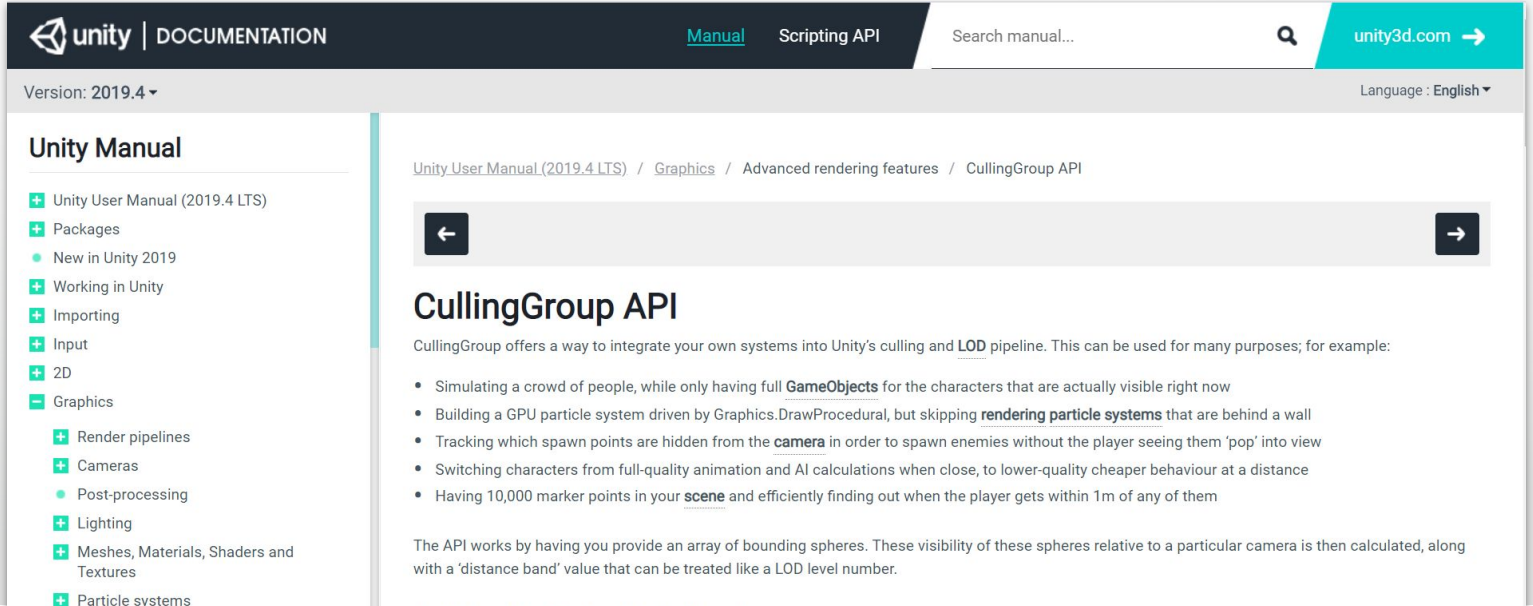
Culling



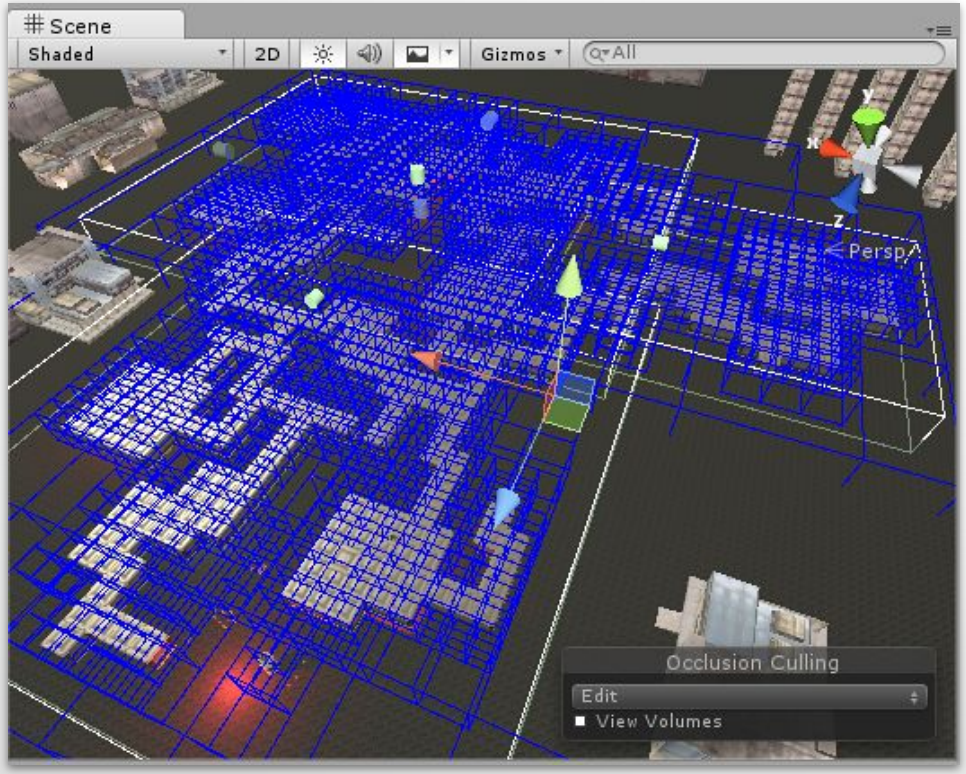
http://people.csail.mit.edu/fredo/PUBLI/surveyTVCG.pdf

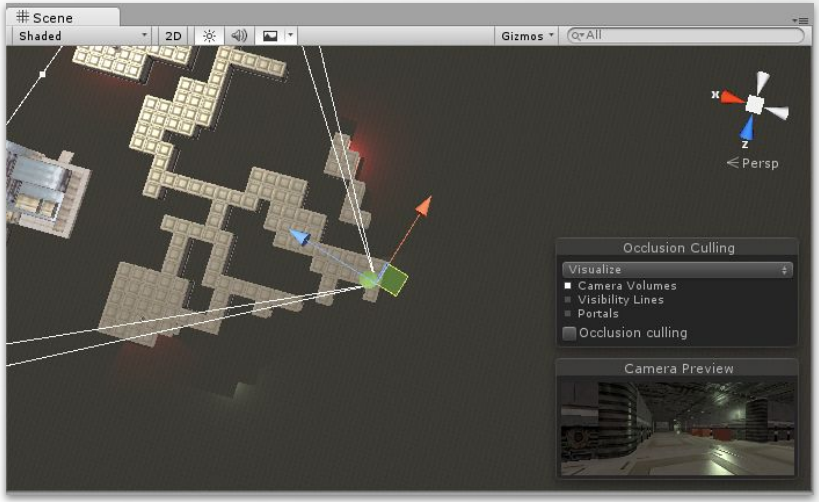
View-frustum culling

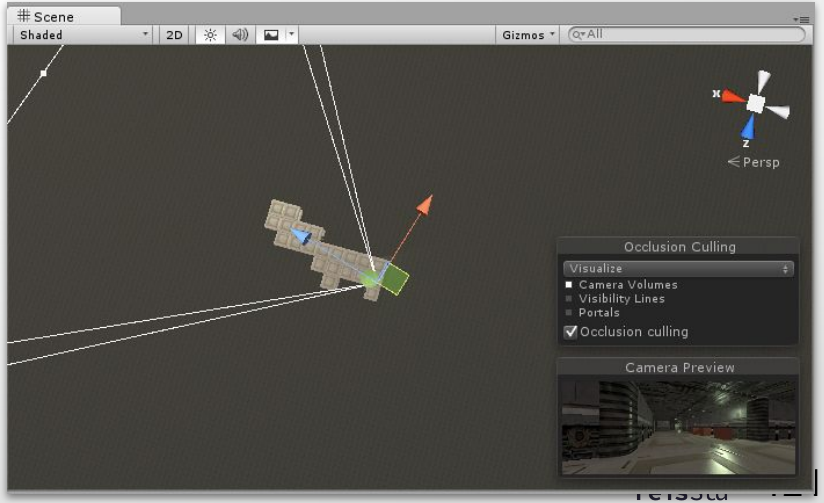
- Yes, Unity applies frustum culling to every renderer.
  - But, you can also use the CullingGroup API



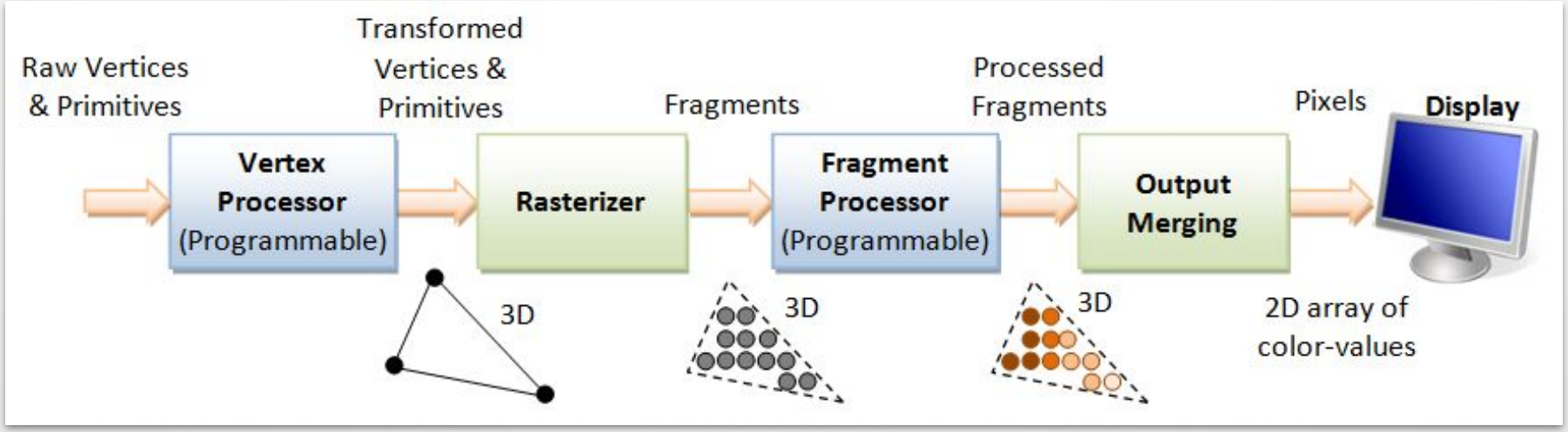
Occlusion culling



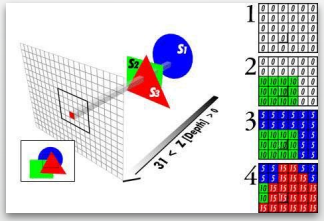


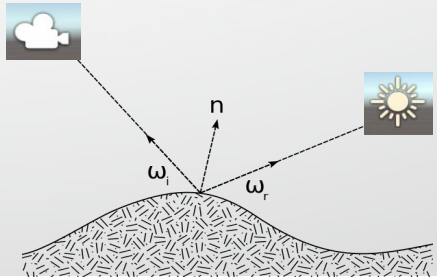


3D Graphics Rendering Pipeline



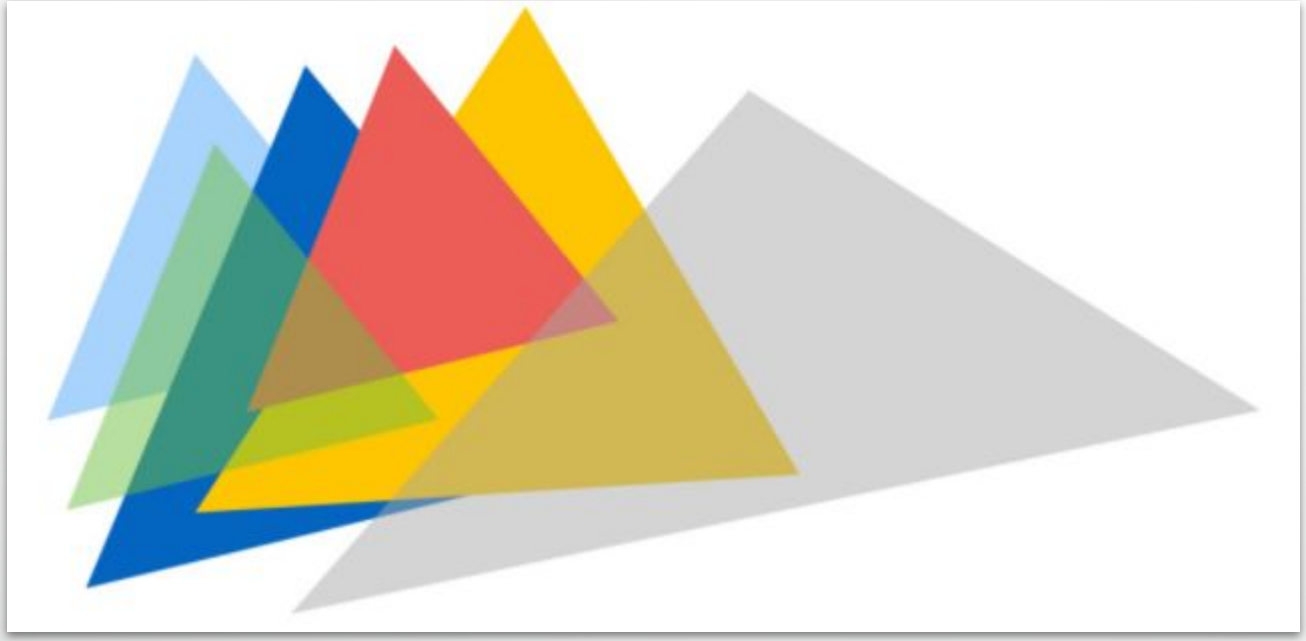
Are all triangles opaque ?





FeisStu

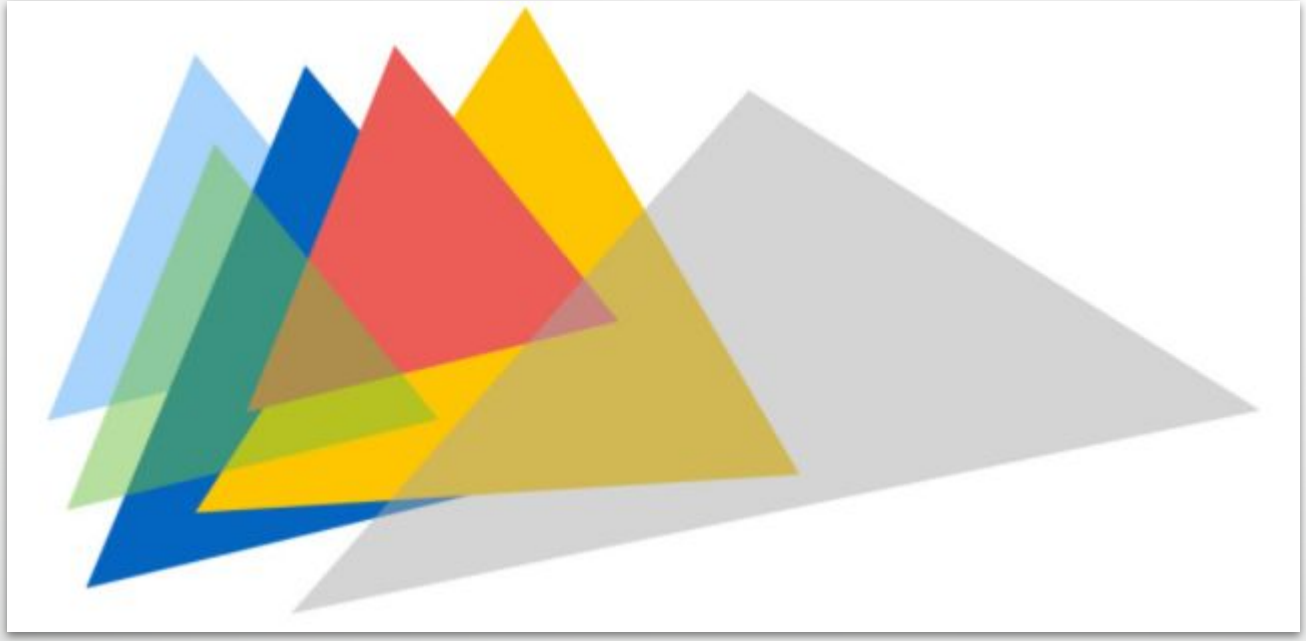
Transparency ?

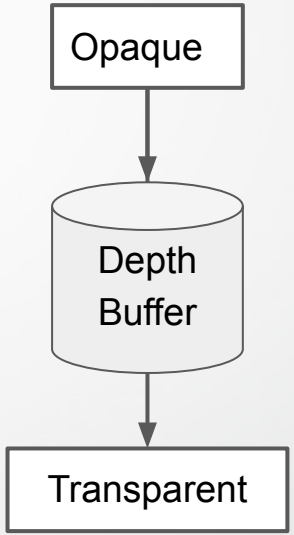


http://cs248.stanford.edu/winter19/lecture/pipeline/slide_055

- Fog
- Particles
- Visual effects
- ...

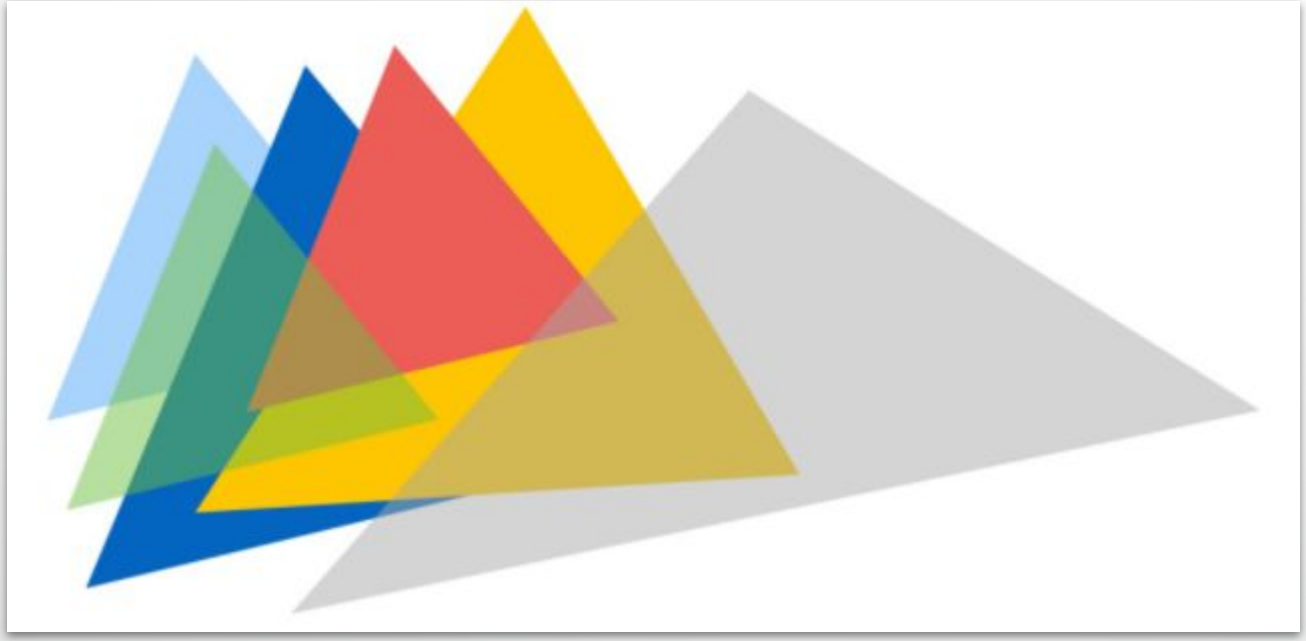
Transparency ?

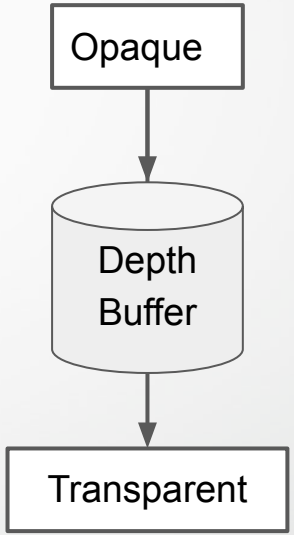




http://cs248.stanford.edu/winter19/lecture/pipeline/slide_055

Transparency ?

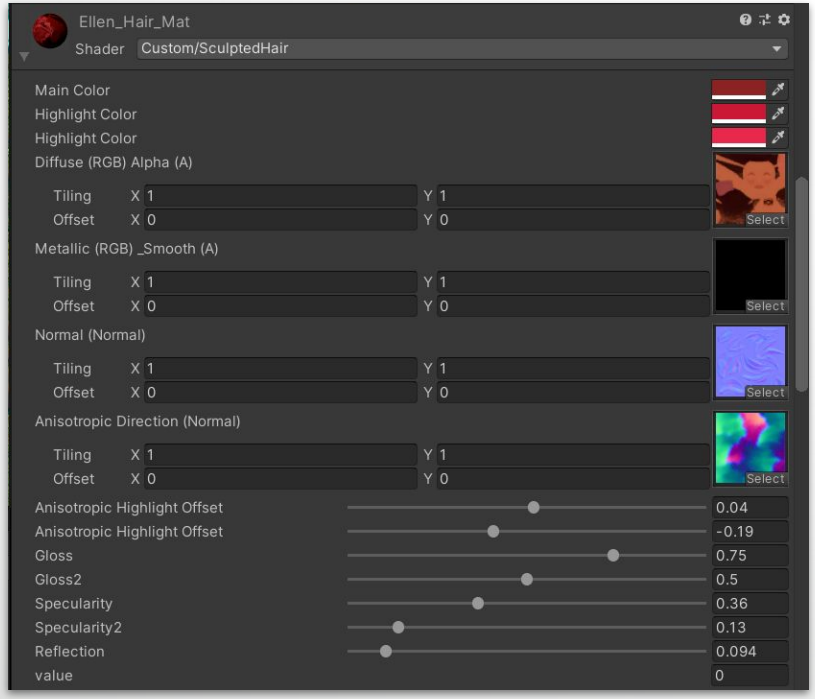


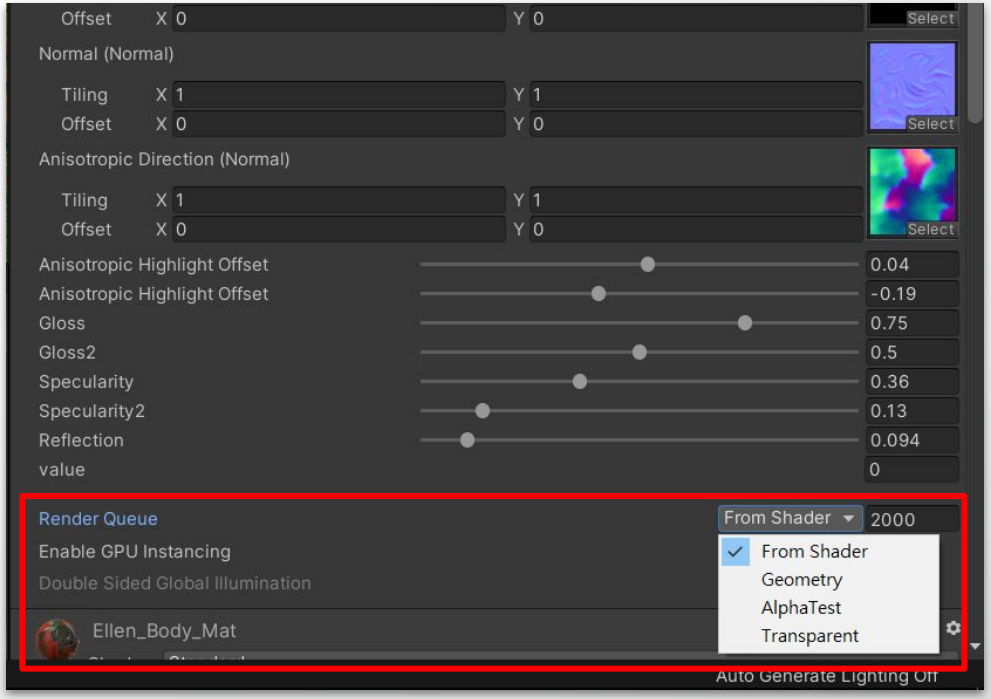


http://cs248.stanford.edu/winter19/lecture/pipeline/slide_055

Does the order of transparent objects matter ?

Render Queue



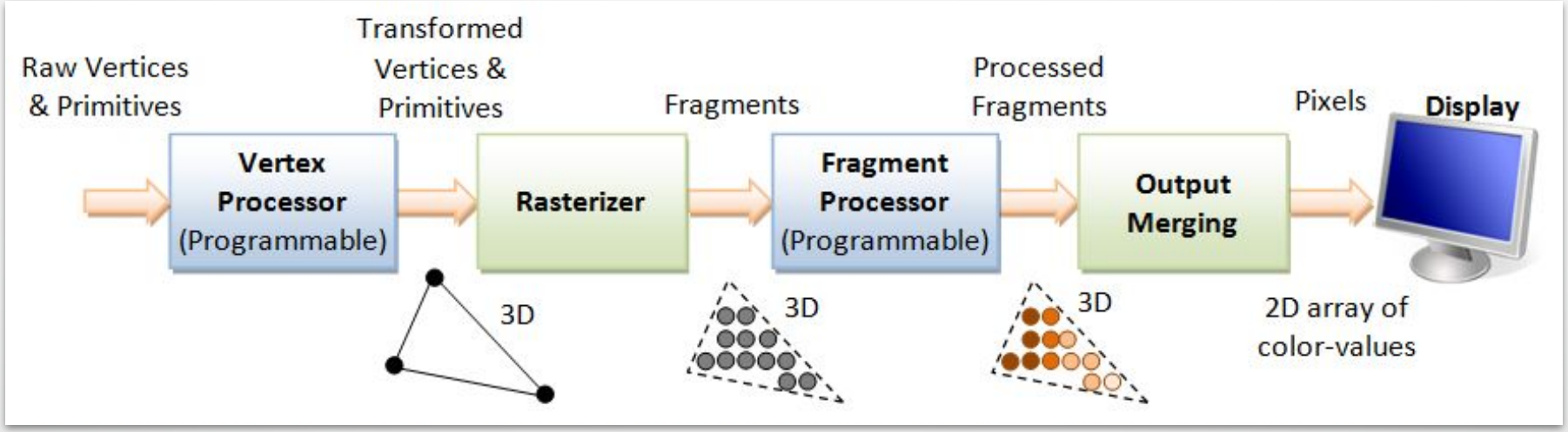


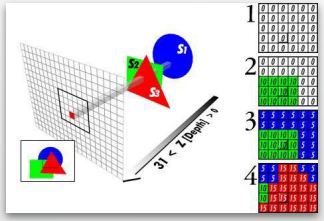
FeisStu

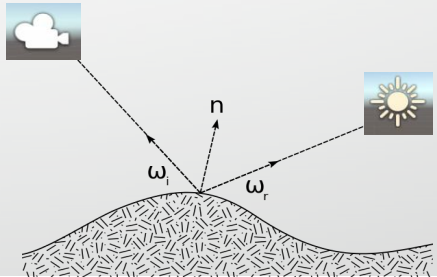

How many draw calls are requied ?
DEVOTION
https://shop.redcandlegames.com/#game

Draw call

Needs to have The same rendering order and material

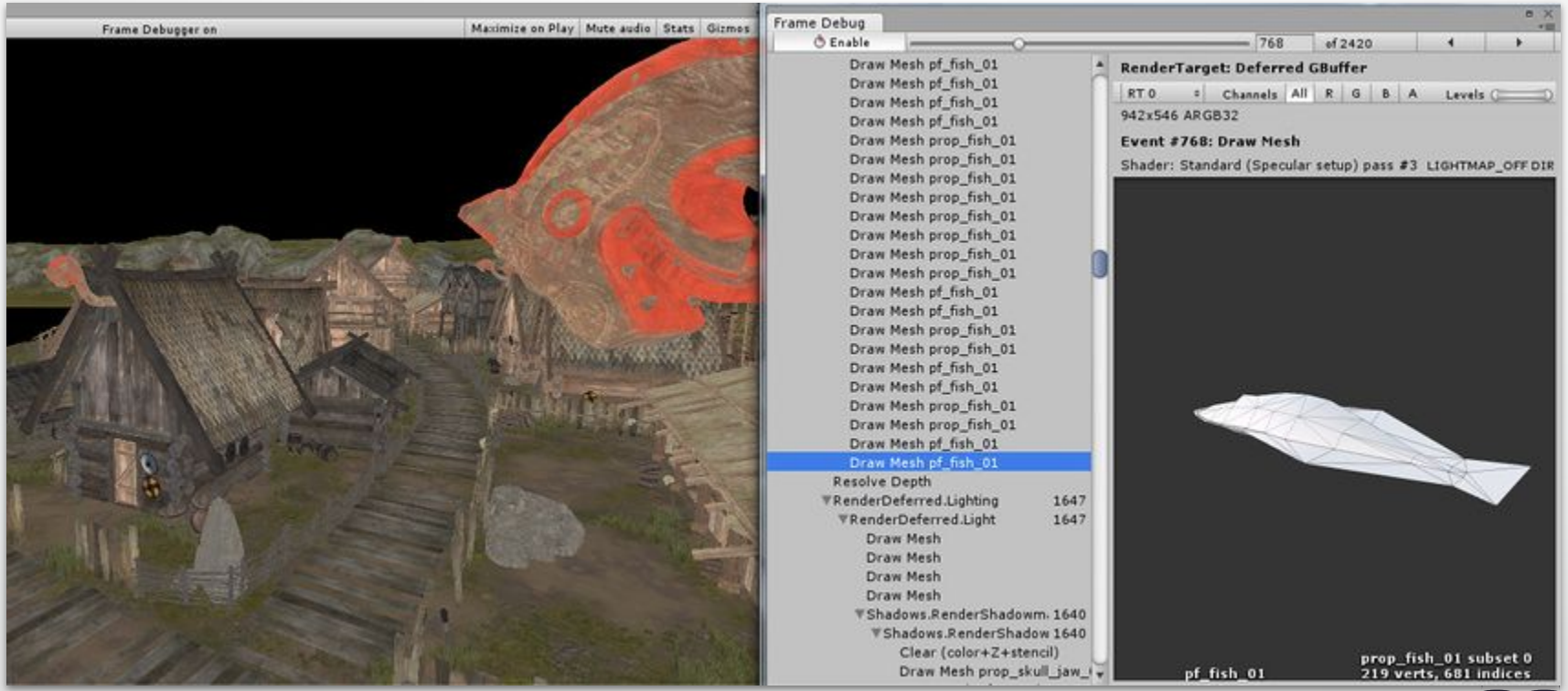






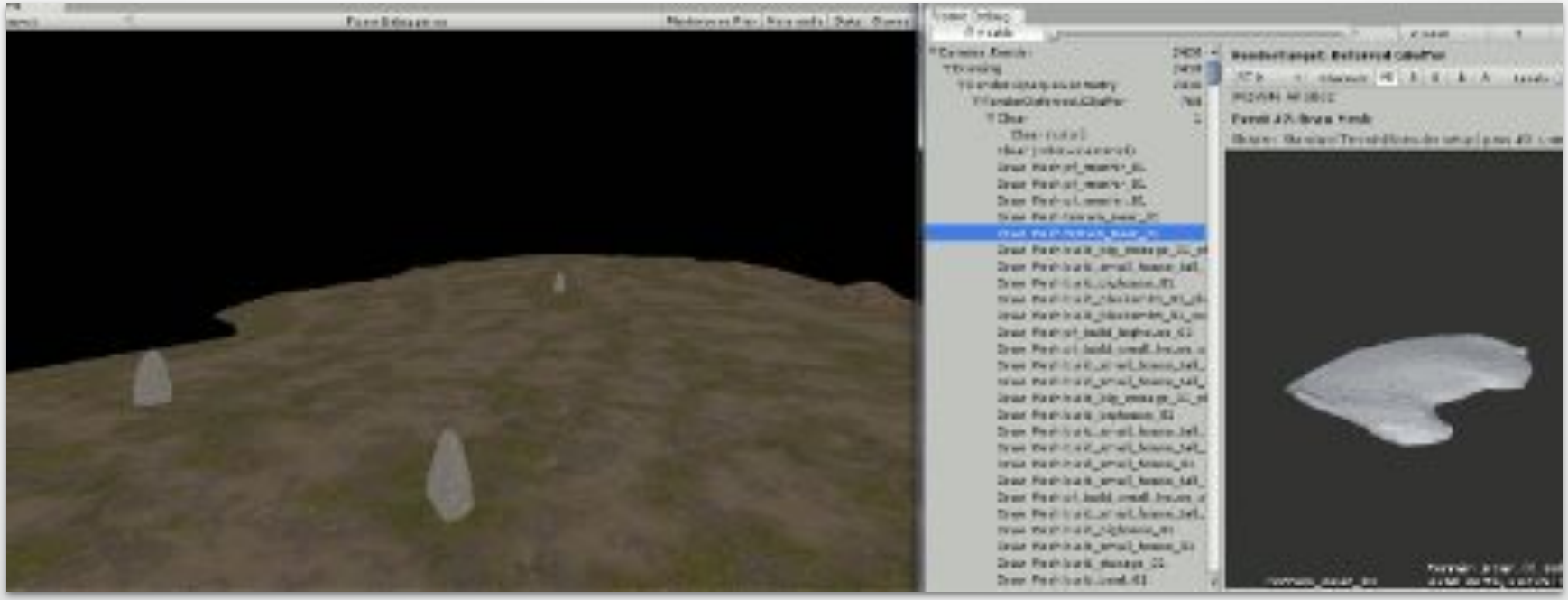
FeisStu

Frame debugger



https://docs.unity3d.com/Manual/FrameDebugger.html

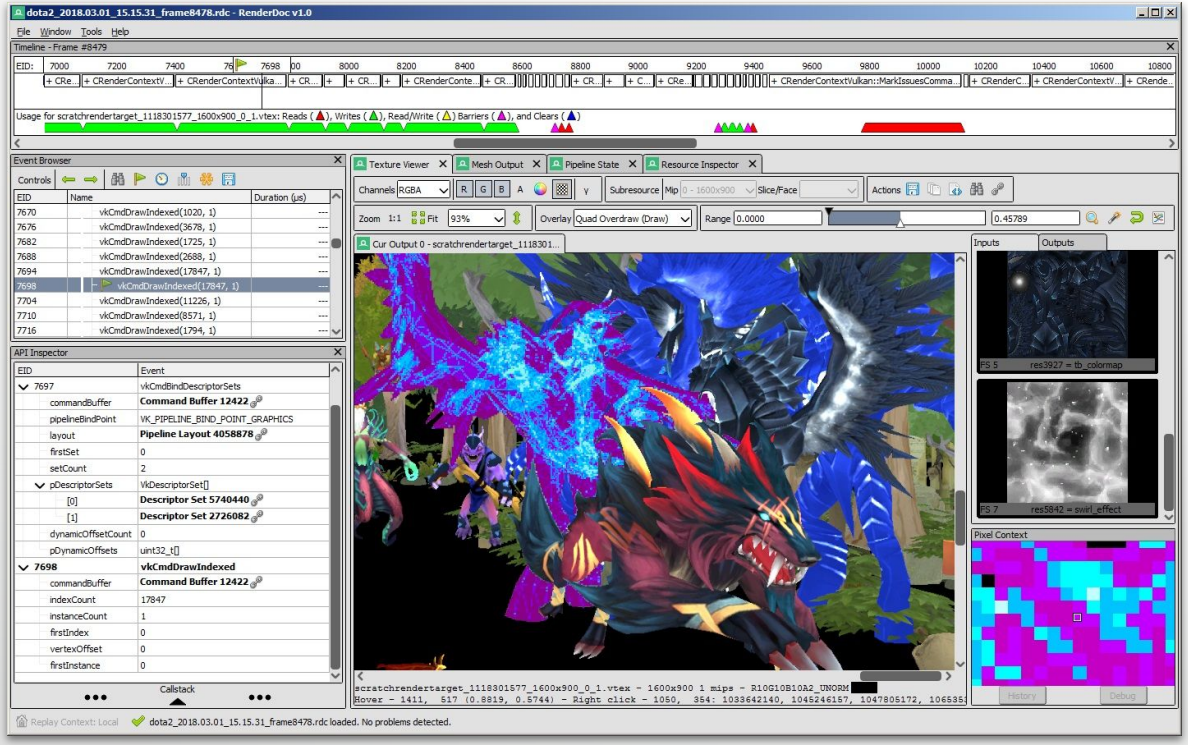
Frame debugger



https://docs.unity3d.com/Manual/FrameDebugger.html

FeisStu

Case study: RenderDoc

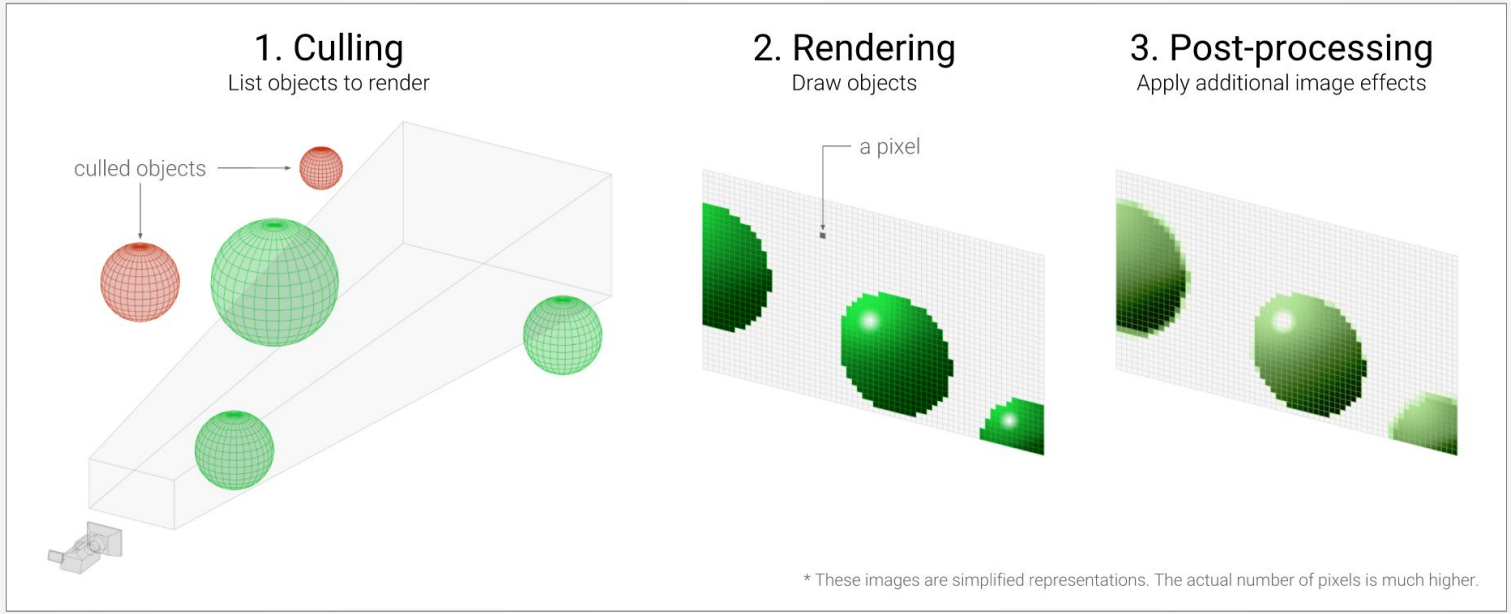


https://renderdoc.org/

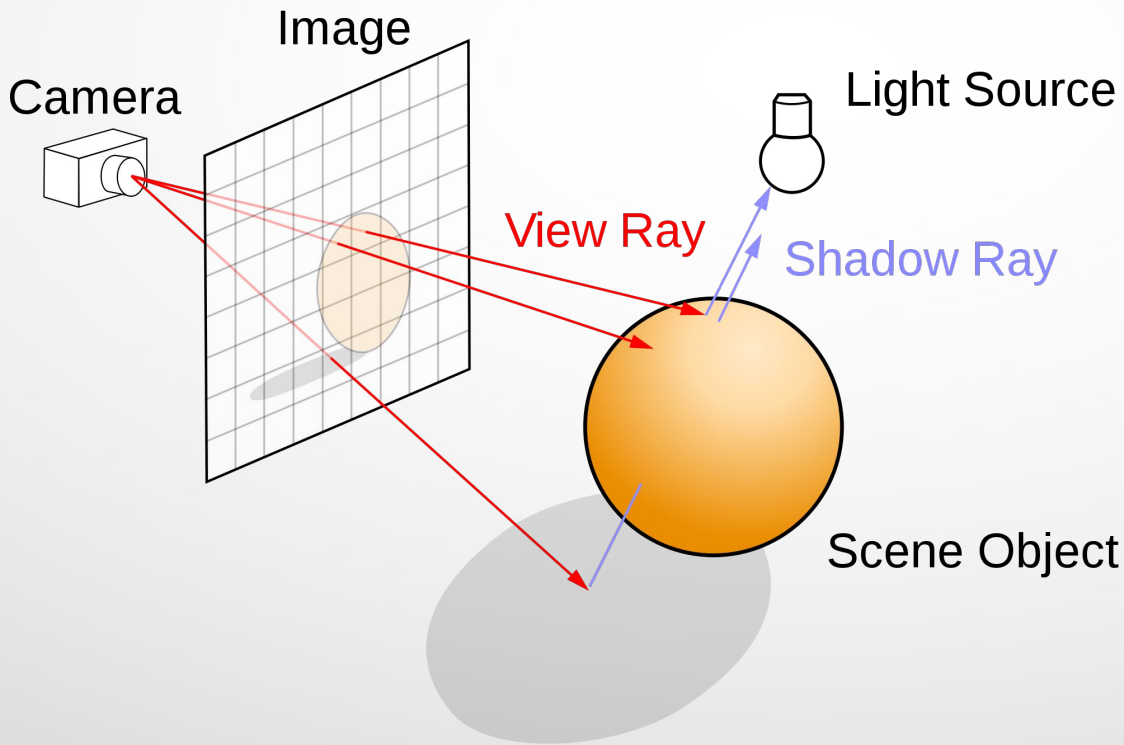
Unity render pipeline stages

Spotlight On

Render pipeline stages

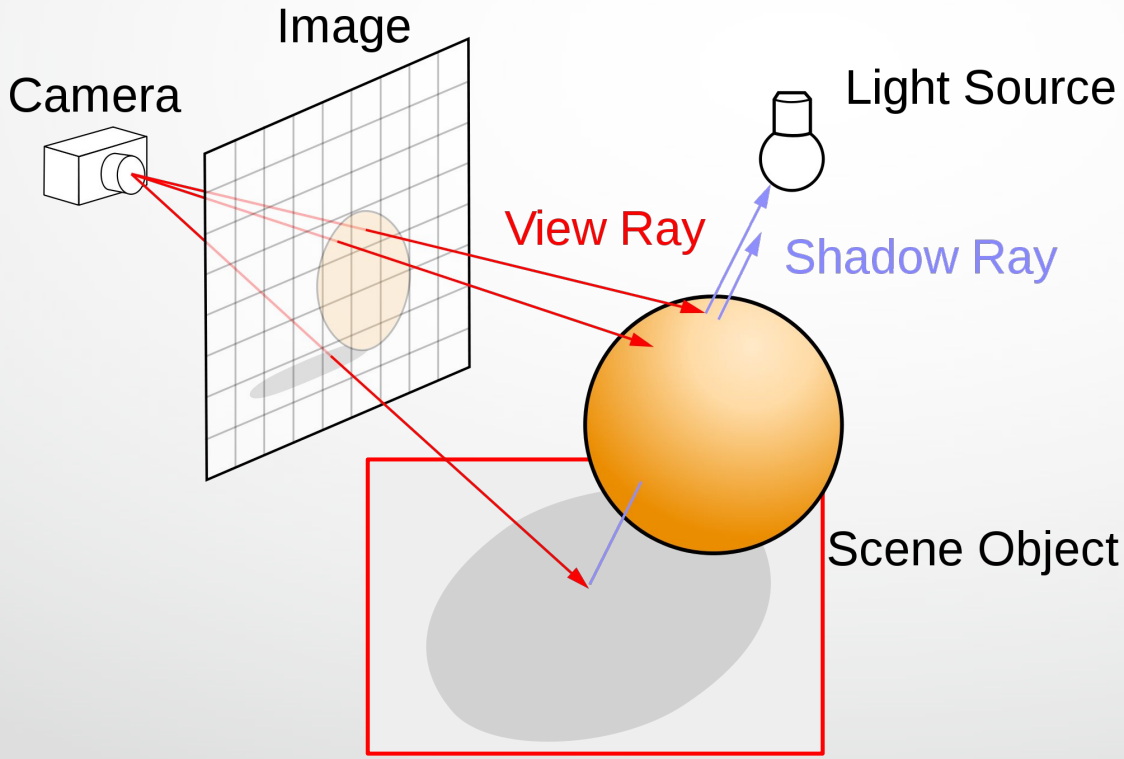


Rasterization vs. Ray tracing



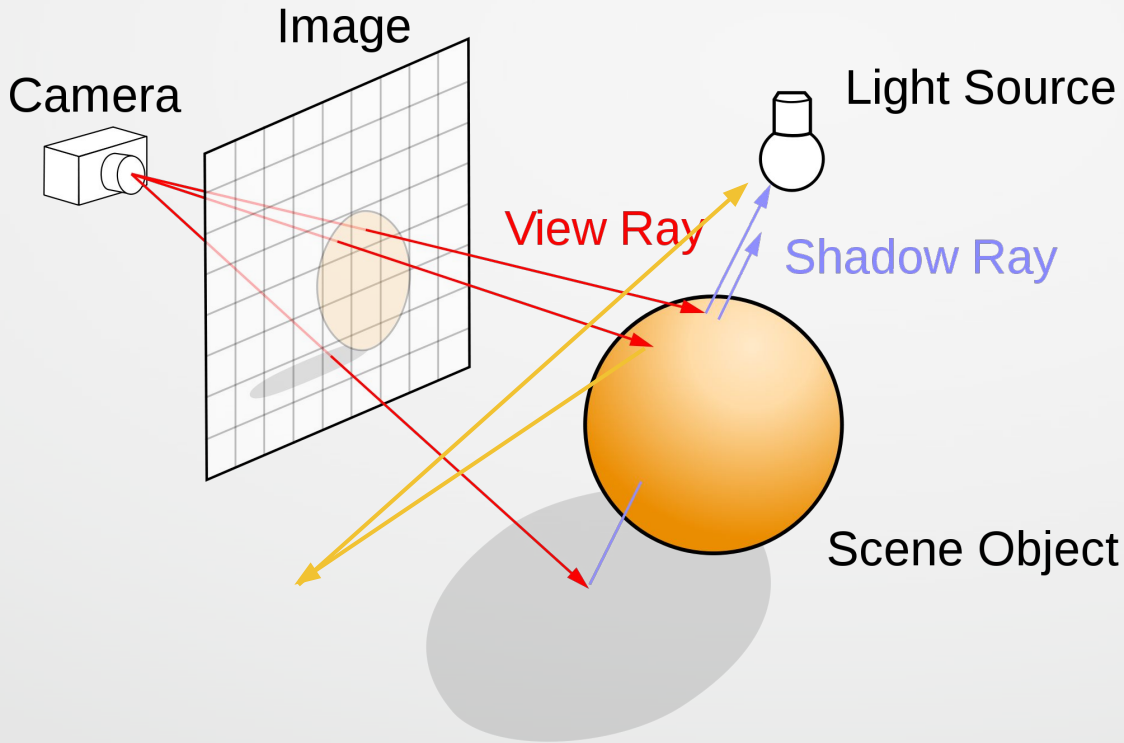
https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

Hard / soft shadows ?



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

Indirect illumination ?



https://en.wikipedia.org/wiki/Ray_tracing_(graphics)#/media/File:Ray_trace_diagram.svg

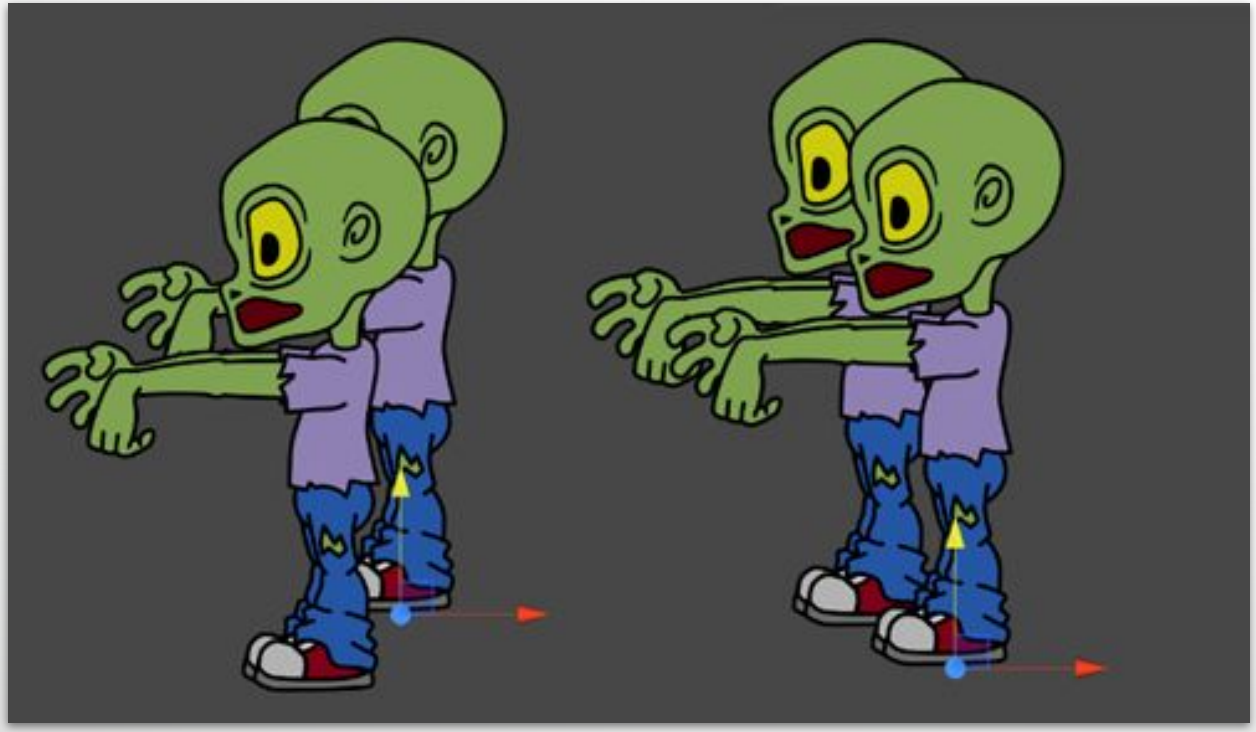
FeisStu

Shadow and global illumination

- Shadow
  - Shadow mapping, ambient occlusions (post-processing)
- Lightmap
  - Baked indirect lighting (static objects)
  - Baked ambient occlusions
- Light Probes
  - Baked indirect lighting (dynamic objects)
- Reflection Probes

FeisStu

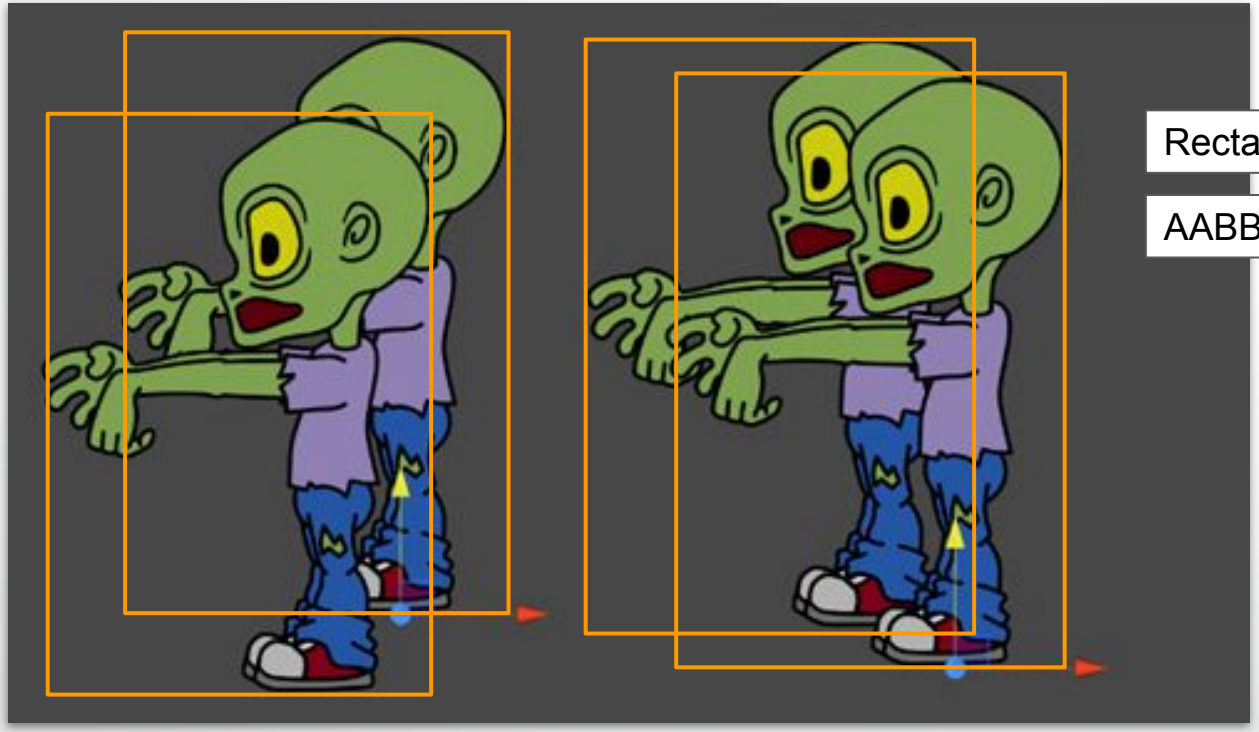
How about 2D / GUI ?



https://docs.unity3d.com/Manual/Sprites.html

FeisStu

How about 2D / GUI ?



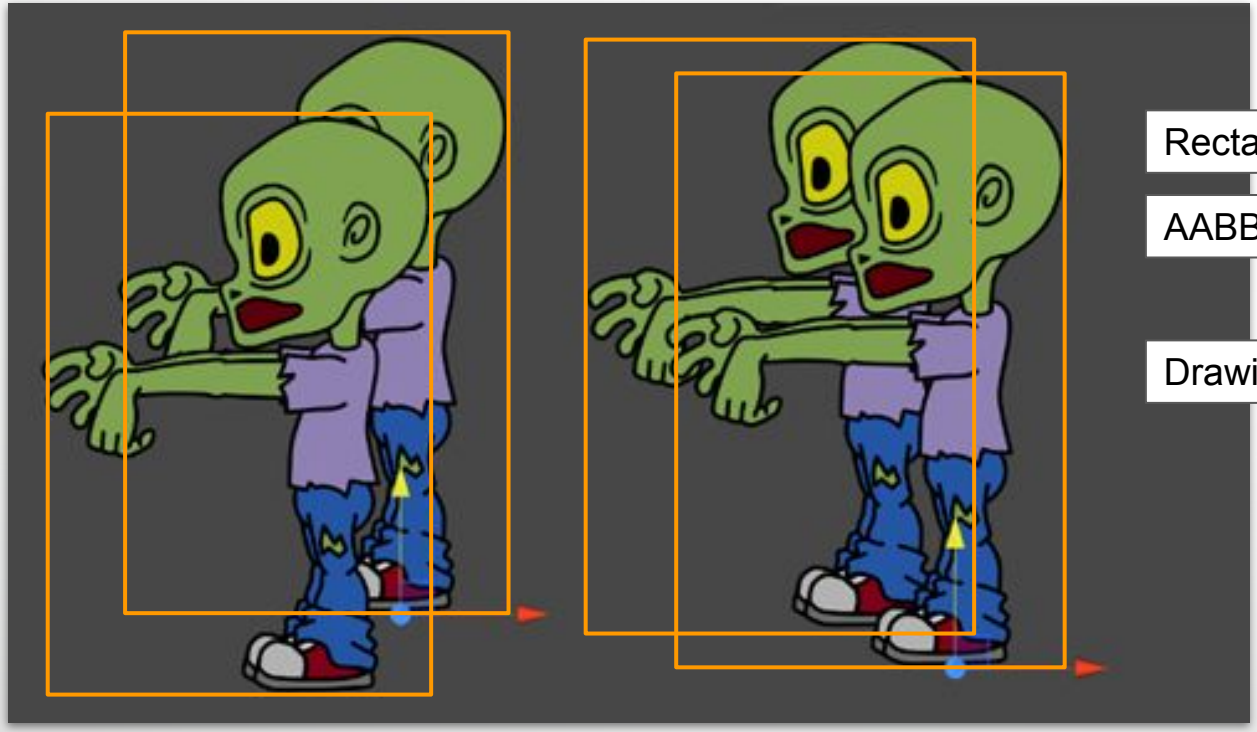
Rectangle ?

AABB (Axis-Aligned Bounding Box)

https://docs.unity3d.com/Manual/Sprites.html

FeisStu

How about 2D / GUI ?



Rectangle ?

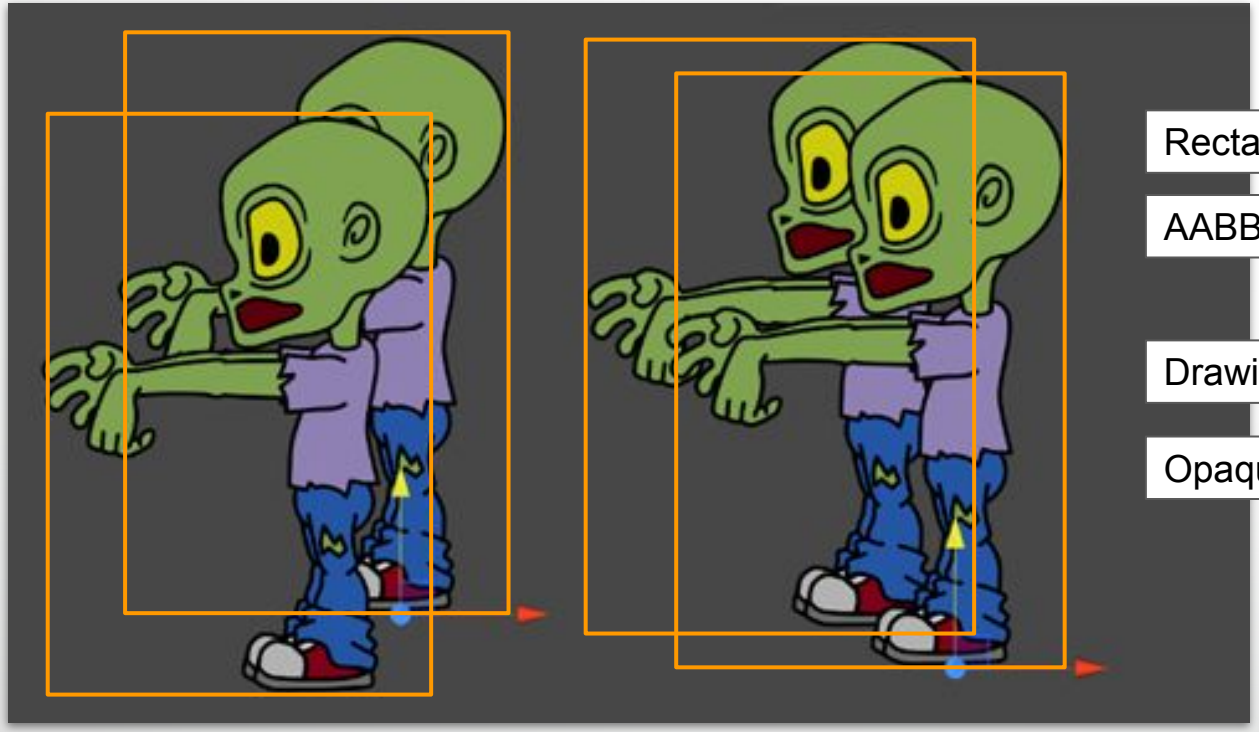
AABB (Axis-Aligned Bounding Box)

Drawing order: painter's algorithm ?

https://docs.unity3d.com/Manual/Sprites.html

FeisStu

How about 2D / GUI ?



Rectangle ?

AABB (Axis-Aligned Bounding Box)

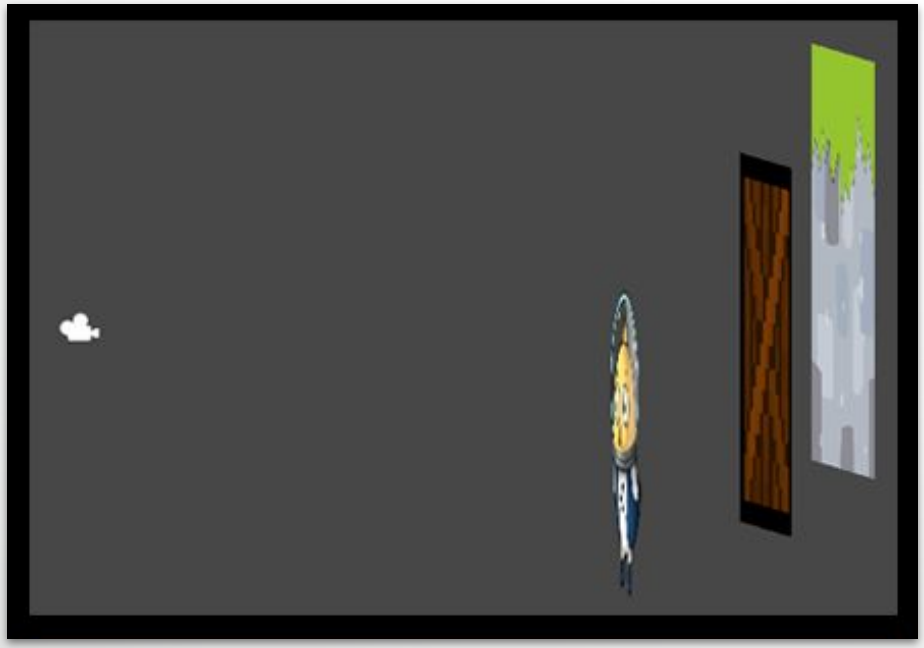
Drawing order: painter's algorithm ?

Opaque vs. Transparent

https://docs.unity3d.com/Manual/Sprites.html

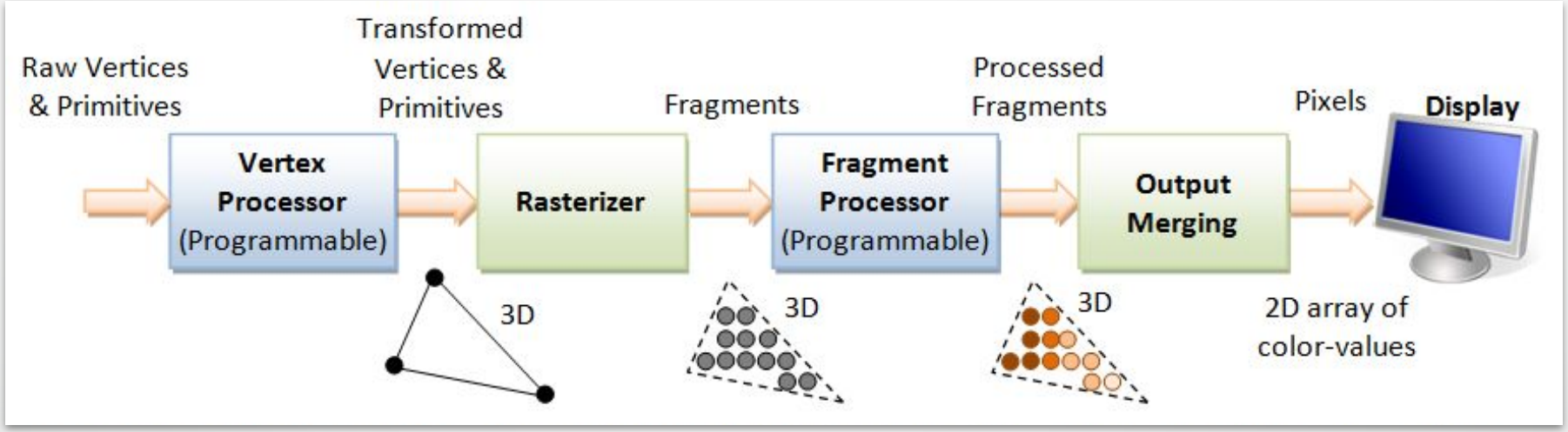
FeisStu

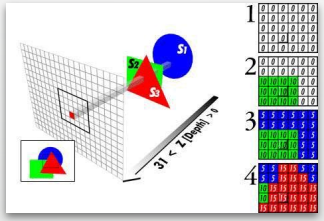
How about 2D / GUI ? (cont'd)

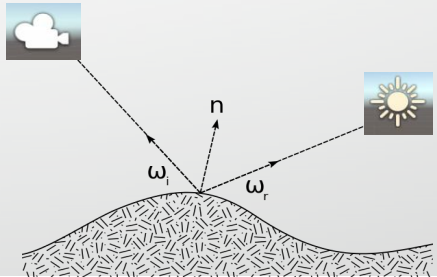


Draw call

Needs to have The same rendering order and material

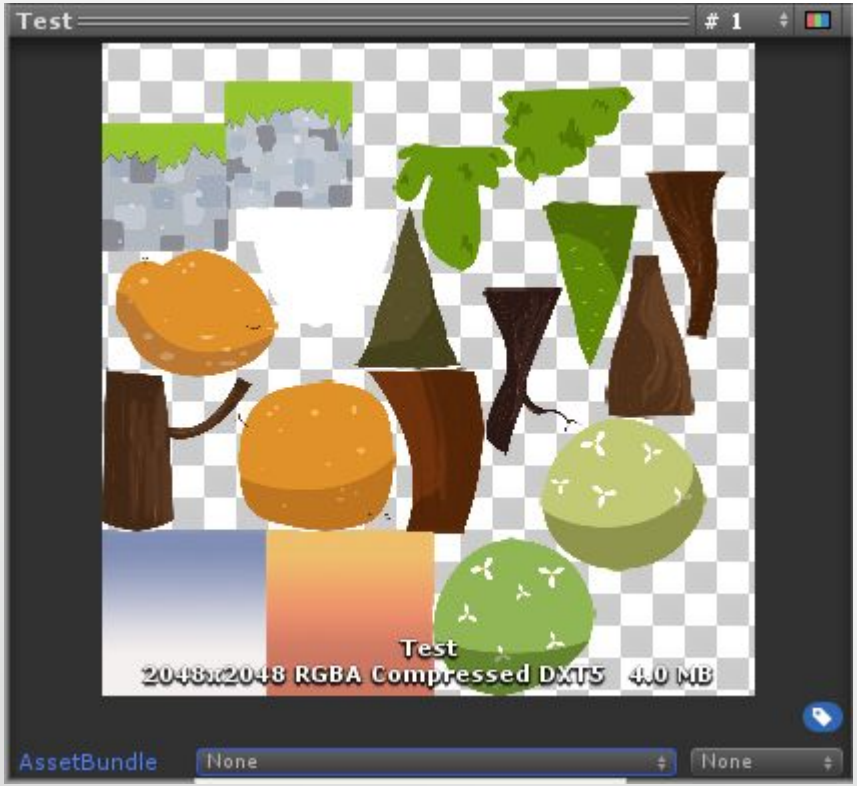






FeisStu

Sprite Atlas

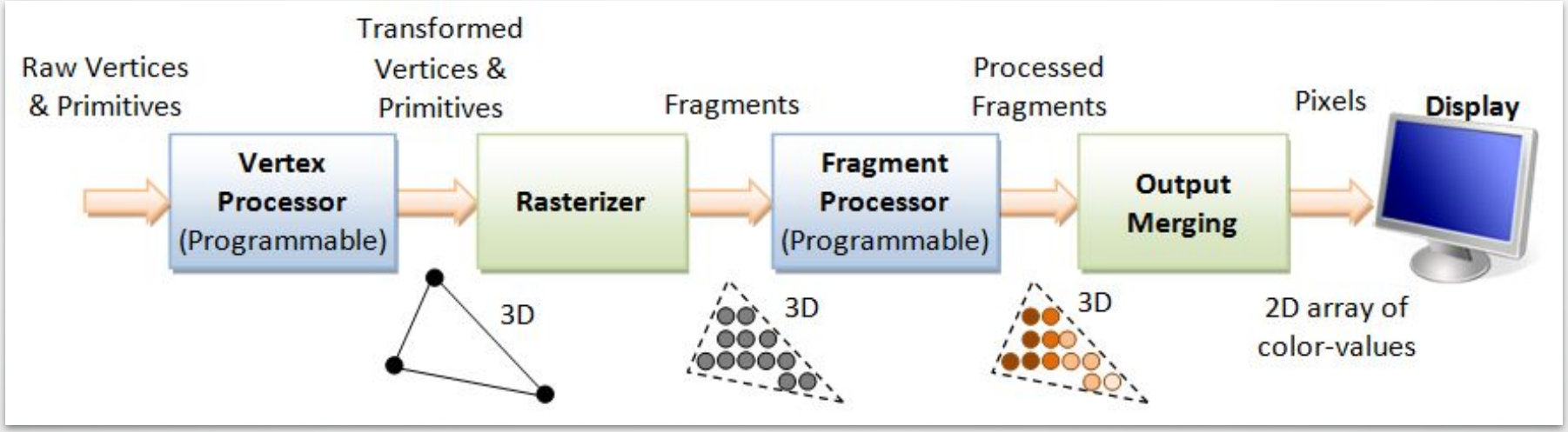


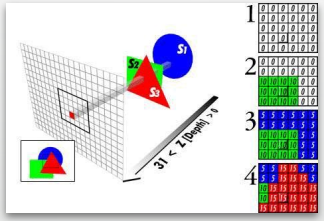
Pack as many textures as possible ?

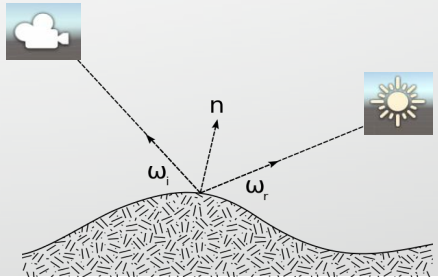
FeisStu

Q & A

Deferred shading vs. Forward shading







FeisStu