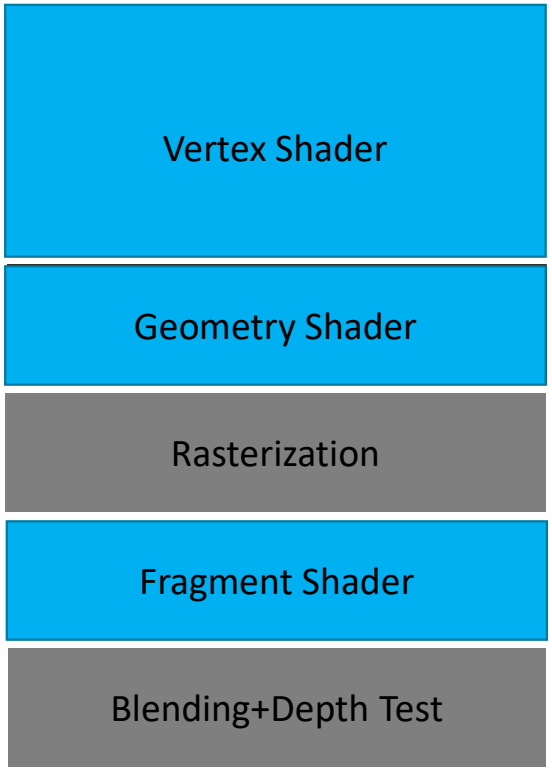
3D Computer Graphics and Animation

Advanced Texturing
Working on your image

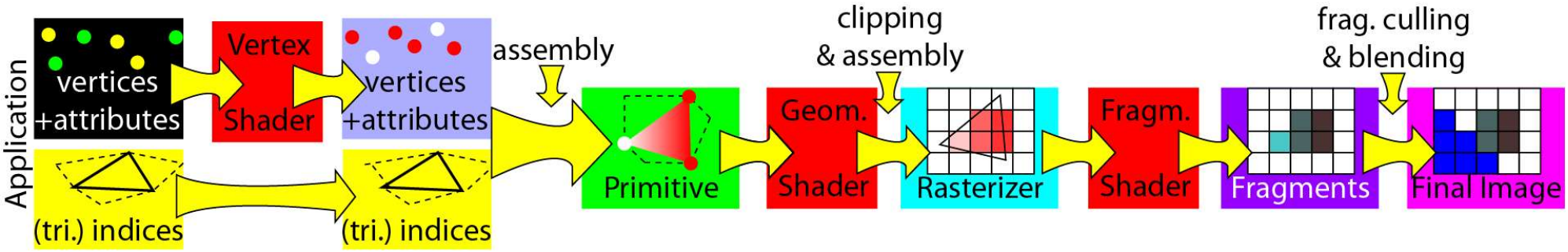
Elmar Eisemann

Delft University of Technology

Programmable Graphics Pipeline



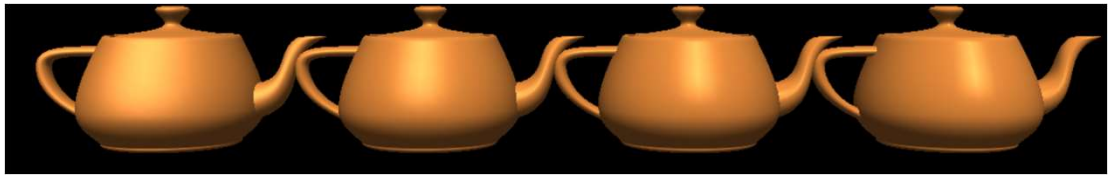
Programmable Graphics Pipeline



Shading

How to transform

in

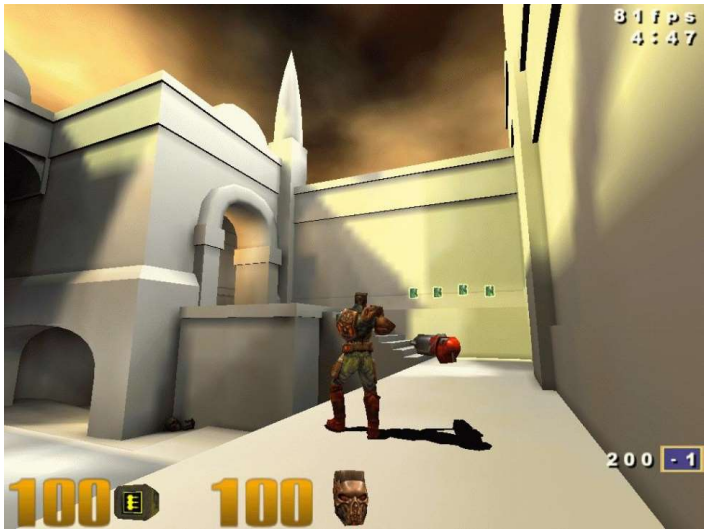


Advanced Material Models

https://blog.playcanvas.com/physically-based-rendering-comes-to-webgl/

Something is still missing...

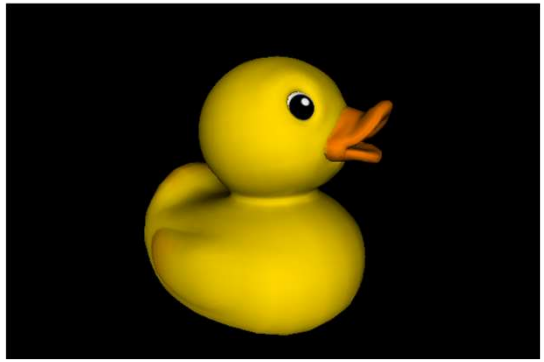
Misses quick material changes

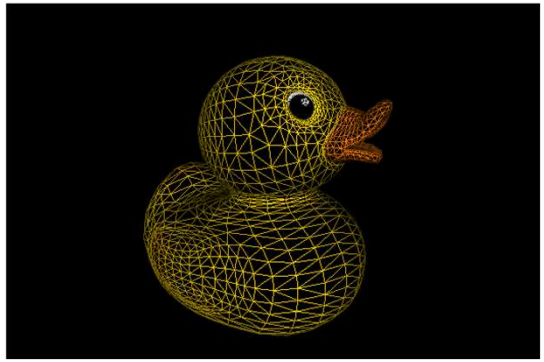


Today

Textures

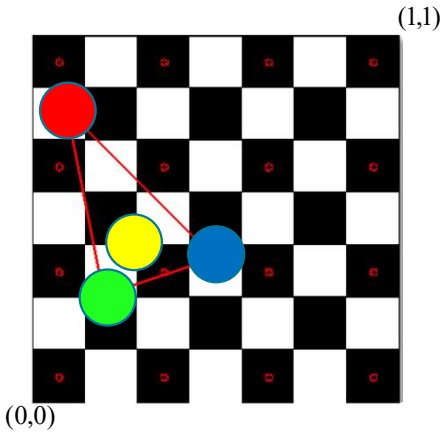
- Can be used to provide, e.g.,
  - parameters for ambient/diffuse material models





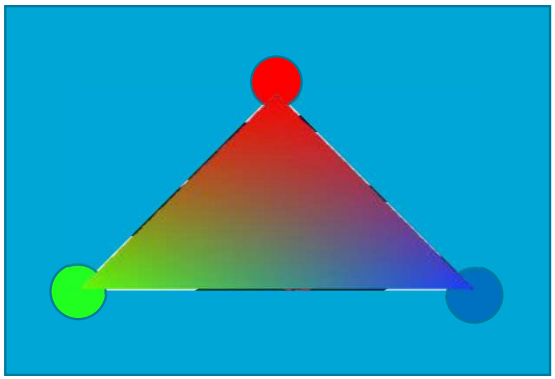
Textures

- Image mapped on the surface via texture coordinates
  - Specified at each vertex `glTexCoord{123}{fi}`
  - Interpolated over triangles



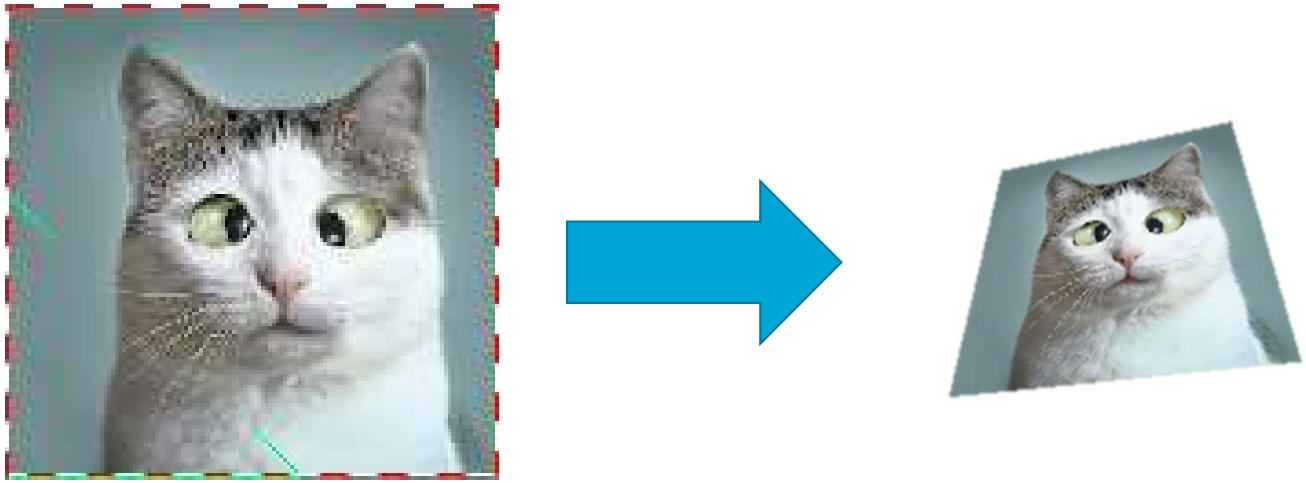
domain

```
TextureCoords(0.2f,0.3f);
Vertex(-1.0f,0.0f);

TextureCoords(0.5f,0.4f);
Vertex(+1.0f,0.0f);

TextureCoords(0.1f,0.8f);
Vertex( 0.0f,1.0f);
```



3D View

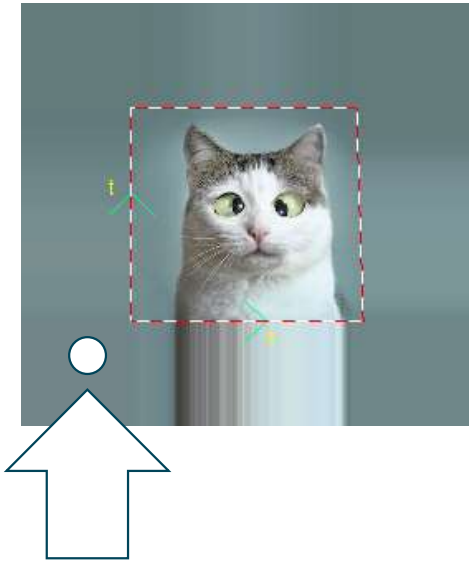
Example of a mapped texture



What happens outside the texture?

- Different modes can be defined **per axis:**

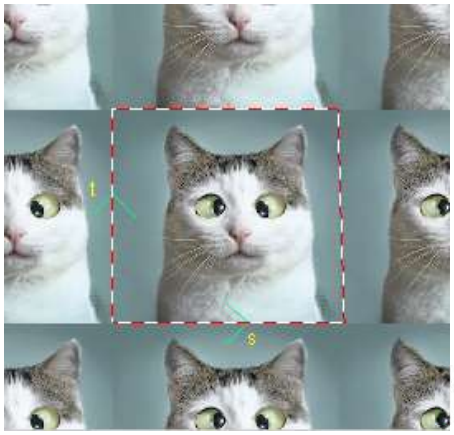
Border = constant color



Texture coordinate?
(-0.25, -.0.25)

Clamp = keep texel value on the border

Repeat = repeat at borders

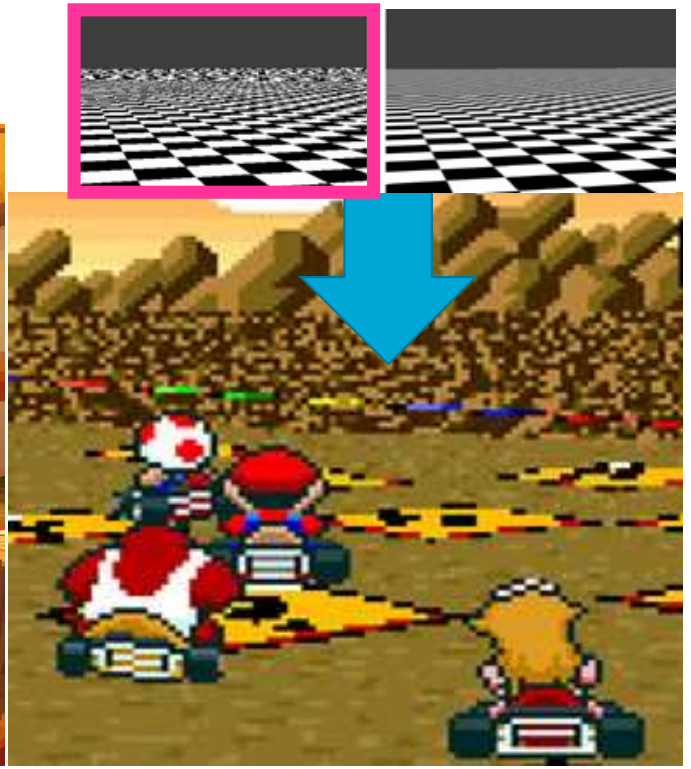


Texture Mapping: Special Software

Unwrapping a Mesh

Texture issues: Aliasing

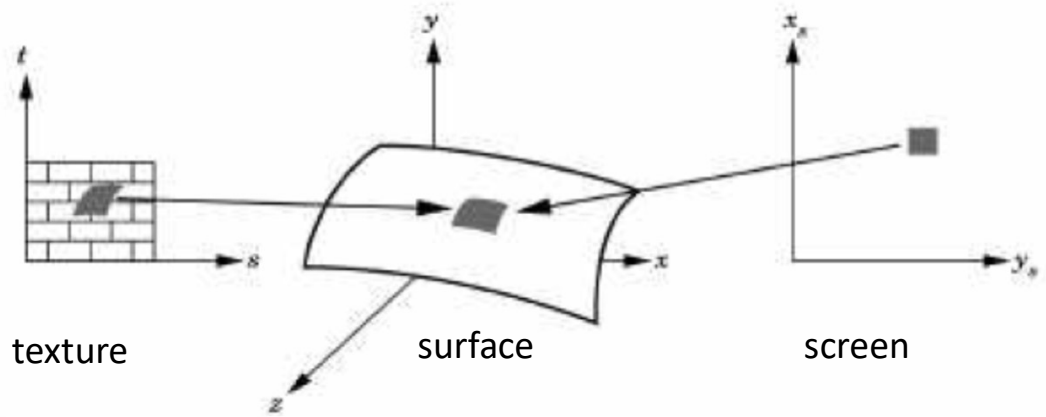
- Not always completely beautiful…



Aliasing

- From screen to texture:

  Not a one-to-one pixel mapping!



1. Oversampling

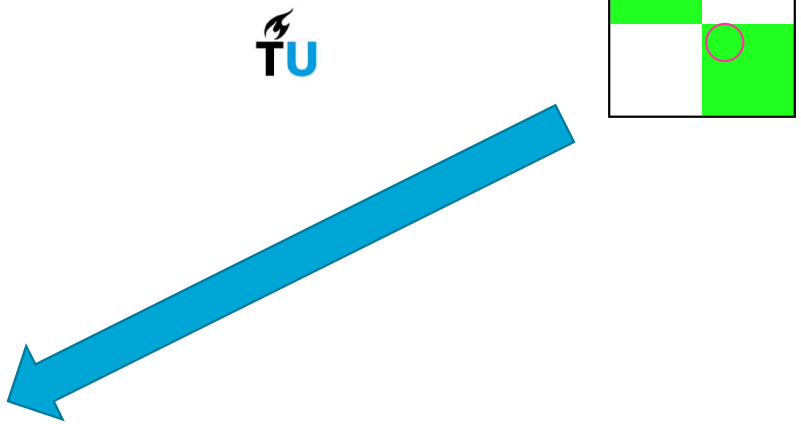
- Due to limited texture resolution

Minecraft

Resident Evil 4

1. Oversampling

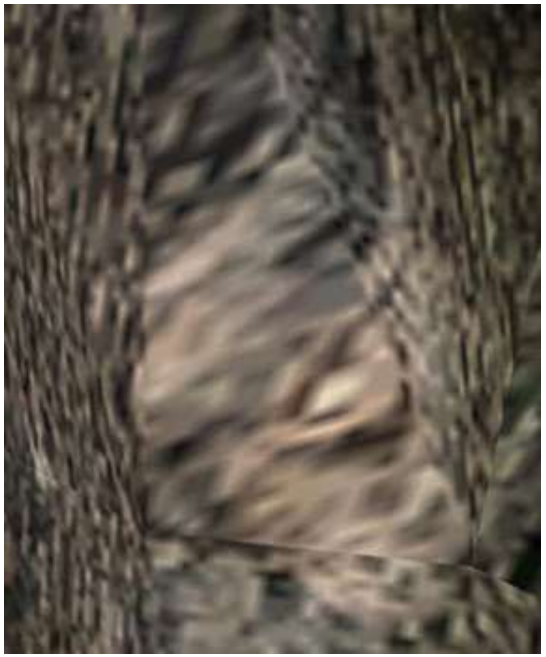
- Pixel smaller than texel



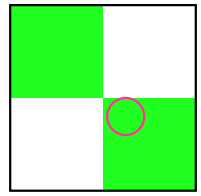
Nearest Neighbor

1. Oversampling

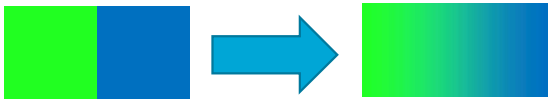
- In practice:
  Often less blocky but washed out (most applications use texel interpolation)



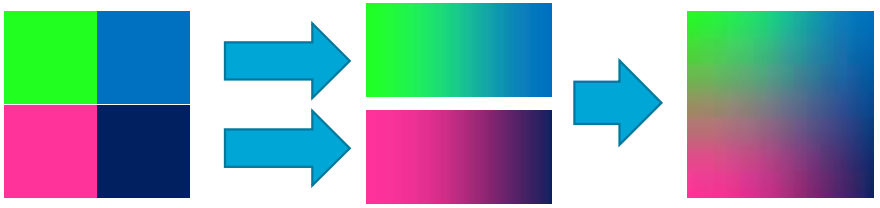
1. Oversampling



- Bilinear interpolation
  - Linear interpolation: (1-alpha) * col1 + alpha * col2

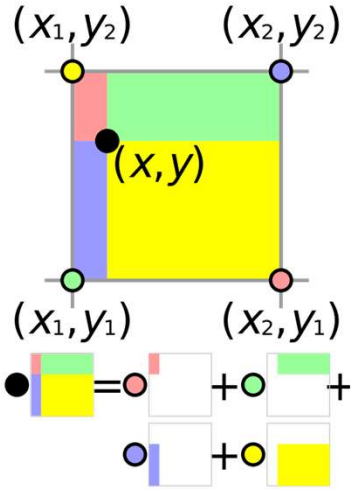


  - Bilinear interpolation:



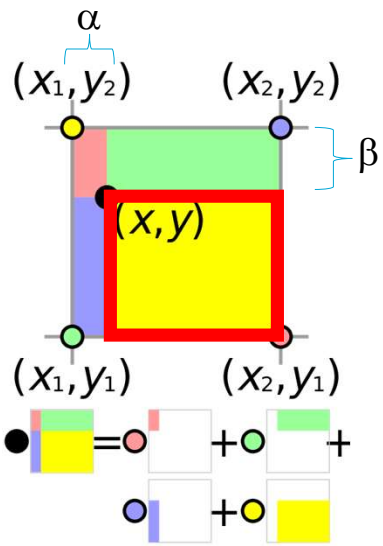
Linear in x direction

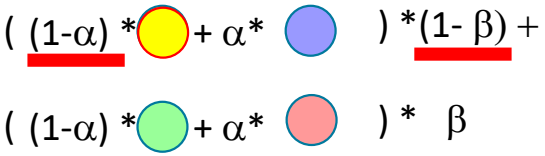
Linear in y direction



1. Oversampling

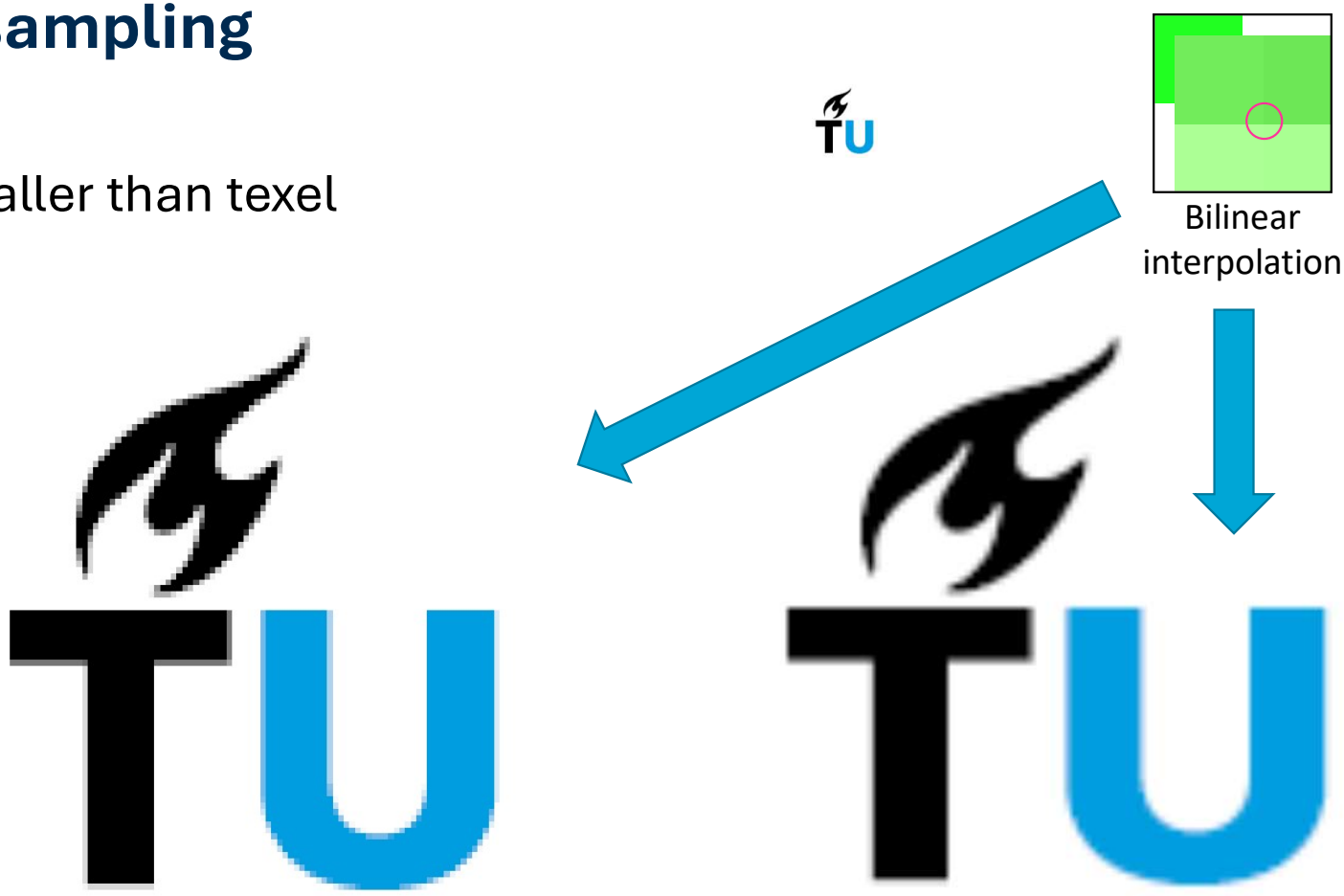
- Why does this work?





1. Oversampling

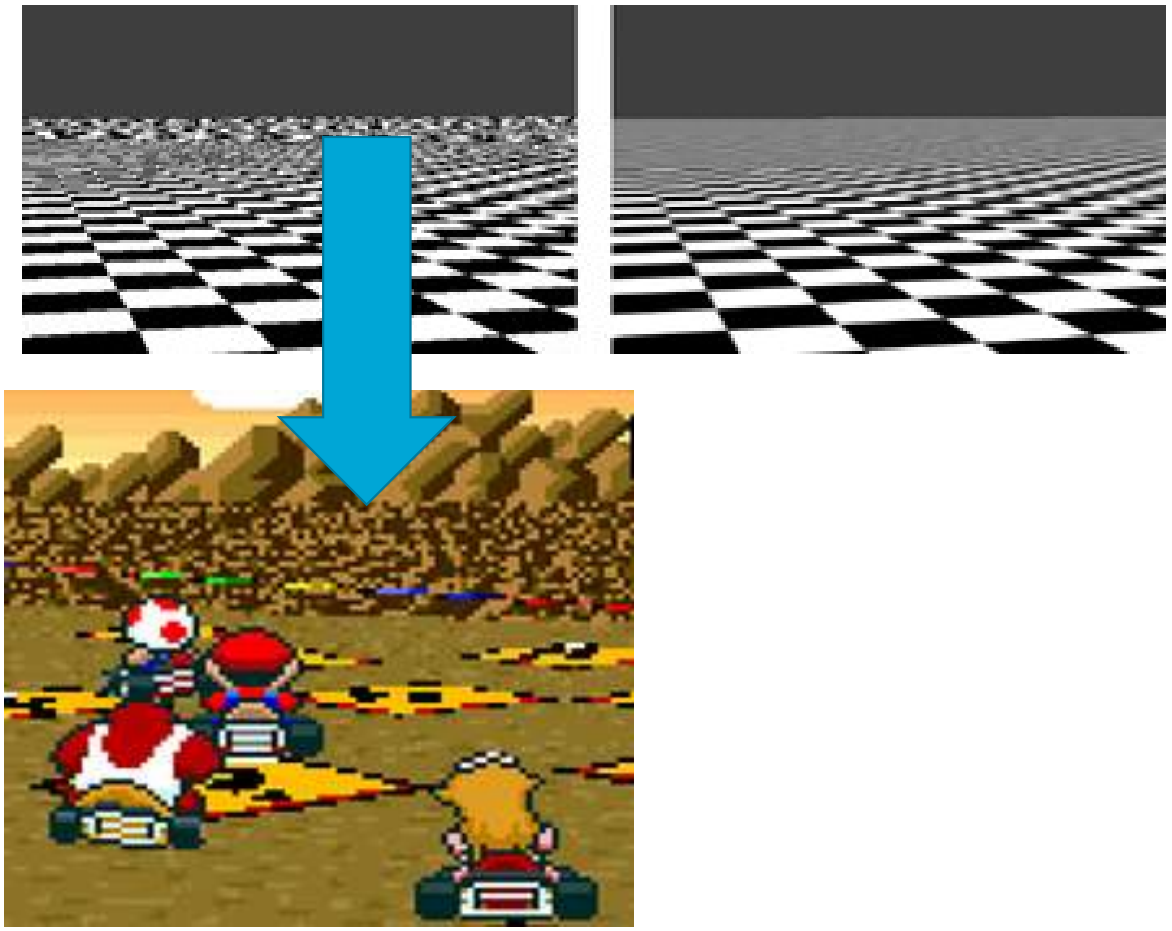
- Pixel smaller than texel



Nearest Neighbor

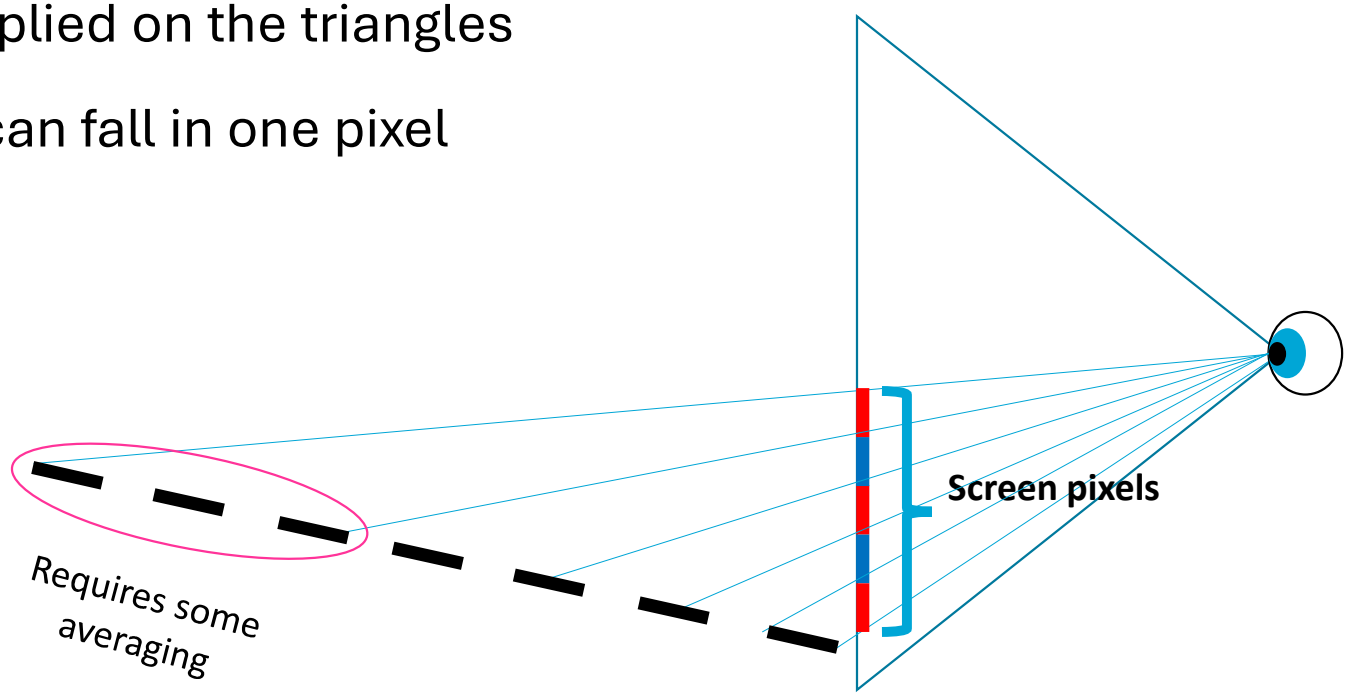
Bilinear Interpolation

2. Undersampling



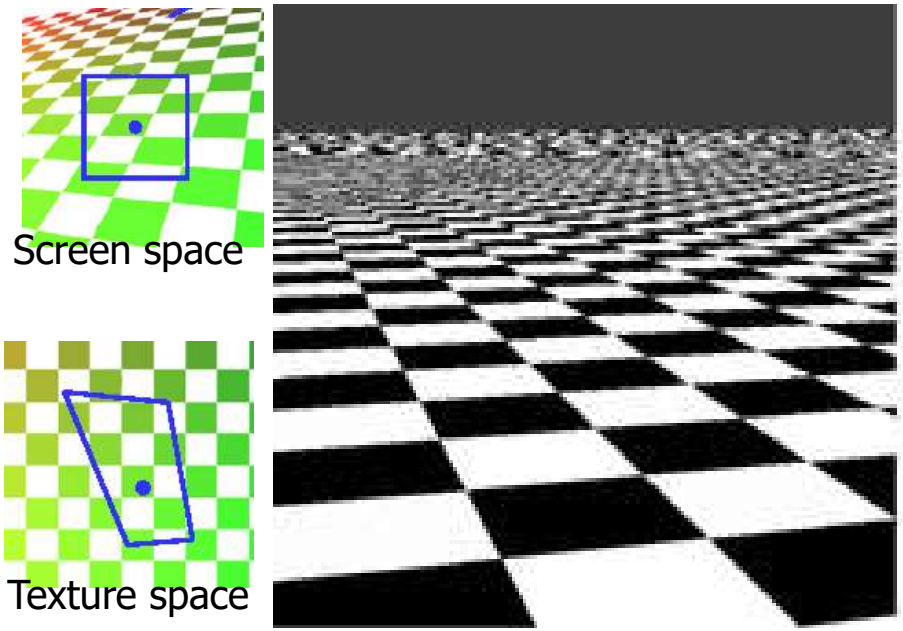
2. Undersampling

- Textures are applied on the triangles
- Several texels can fall in one pixel



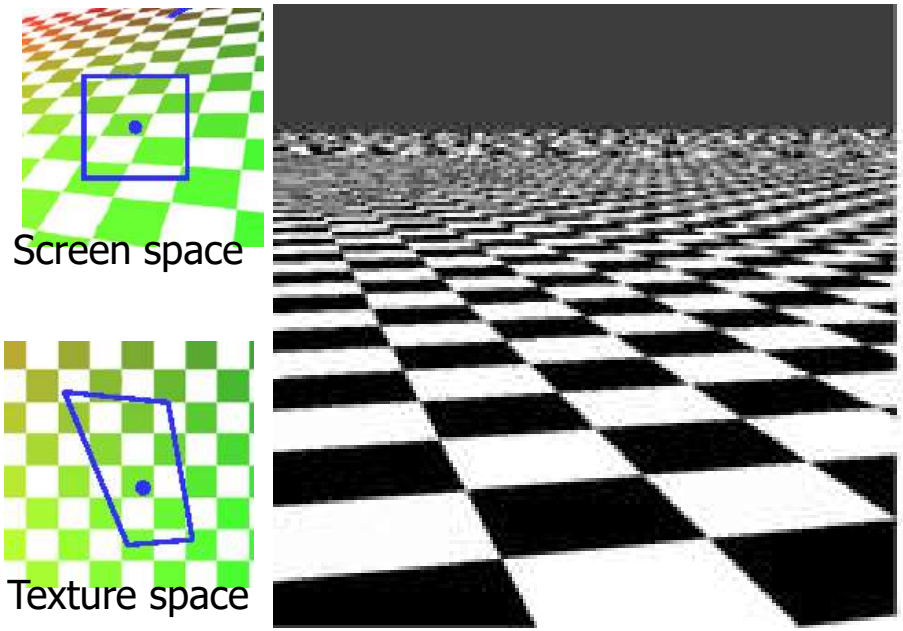
2. Undersampling

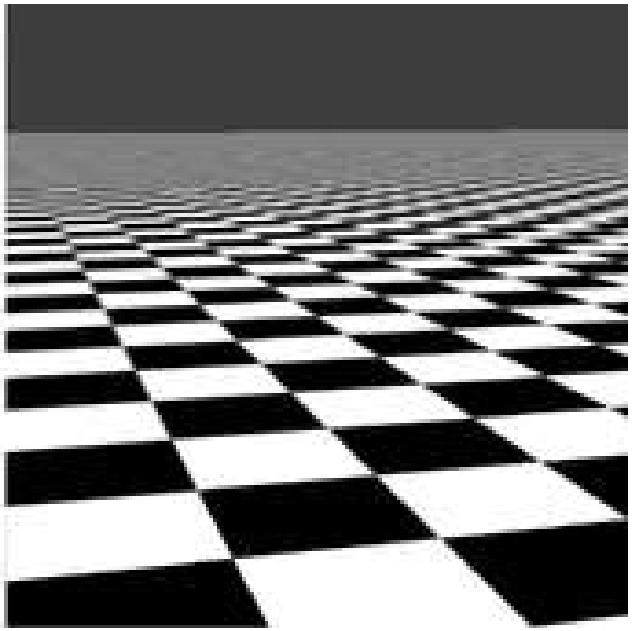
- One screen pixel does not necessarily correspond to one texel



2. Undersampling

- One screen pixel does not necessarily correspond to one texel





2. Undersampling

- Naïve solution:
  Render at high resolution
  then average the result

- Very costly….

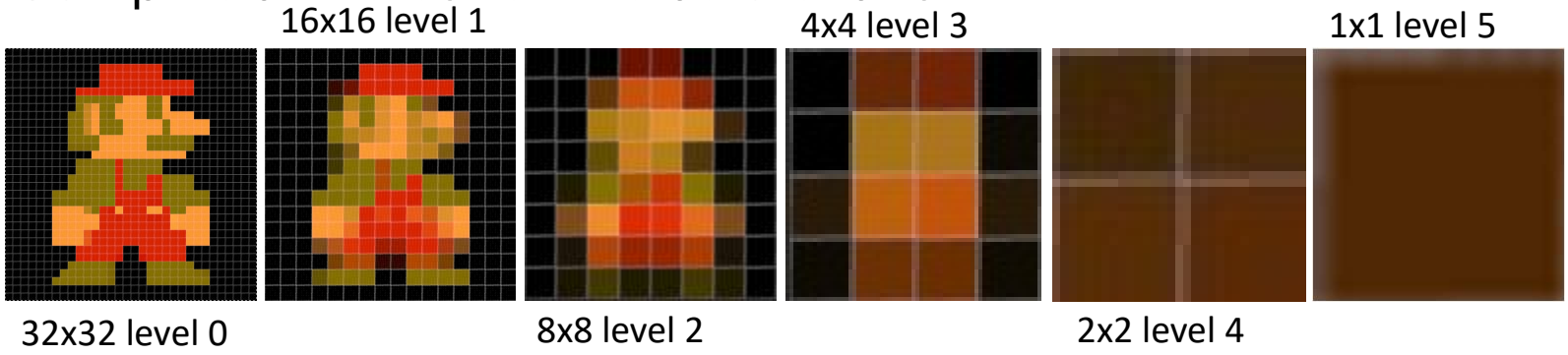
MipMapping: Approximate Filtering

- ***Mip*** **=** ***m****ultum* ***i****n* ***p****arvo*

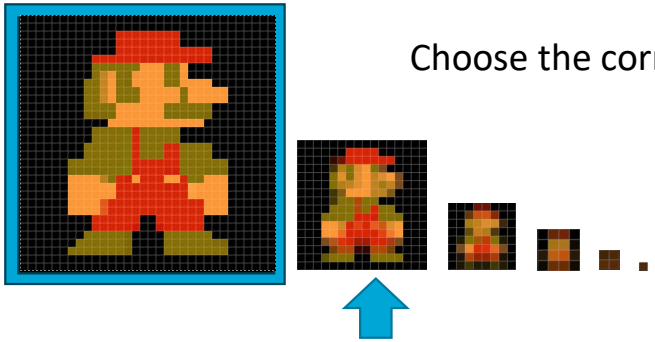
- meaning "much in little"

MipMapping: Approximate Filtering

- Idea: Compute a filtered texture beforehand



Imagine
A 32x32
screen

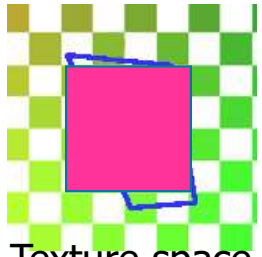


Choose the correct level depending on pixel-to-texel matching

e.g., 4 texels per pixel

MipMapping: Approximate Filtering

- Precompute *Mipmap pyramid:*
  - *Hierarchical texture*
  - *Reduce resolution by 2x2 on each level*



Texture space

Per pixel, approximate region with a square of size 2^k k is then the mipmap level

For example:
4x4 pixels covered

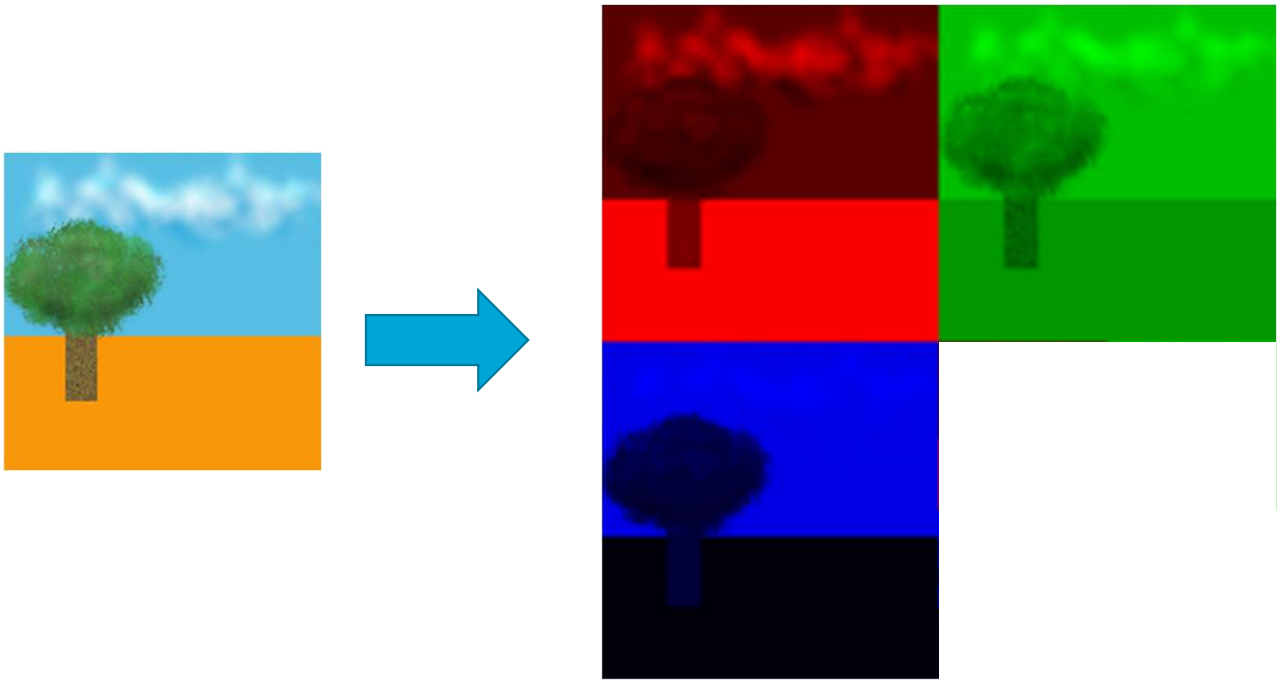
Then 16 lookups needed to average

Here: One pixel on LOD2 is a 4x4 pixel area in the original texture

A single texture lookup is enough in LOD2.

MipMapping: Memory Requirements

- Visual Proof that it is <1/3 extra memory



MipMapping: Memory Requirements

- We assume memory of the original texture is 1
- Texture resolution 2^n x 2^n
- Then the memory cost for all mipmap levels is:

$$\sum_{i=0}^{n} \left(\frac{1}{4}\right)^i$$

$$= \frac{1 - 1/4^{n+1}}{1 - 1/4}$$

$$= \frac{4}{3}(1 - 1/4^{n+1})$$

In the limit the cost is 1/3 higher.

MipMapping: OFF

- Just Nearest Neighbor looks noisy

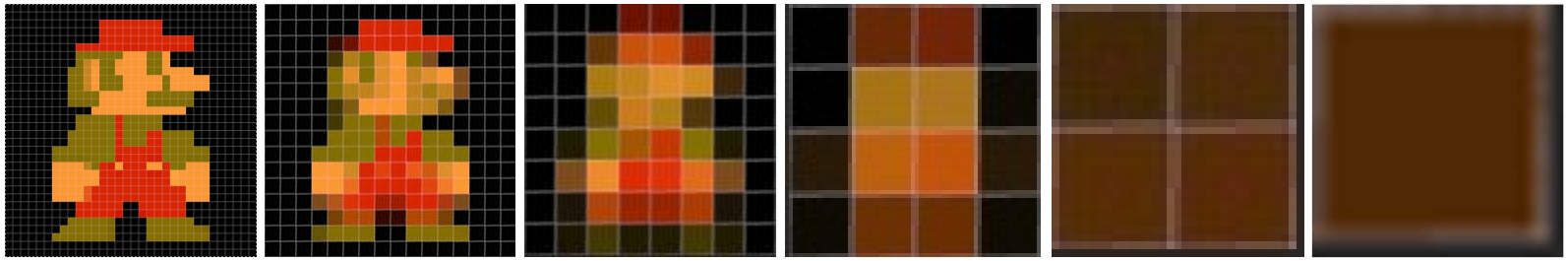
MipMapping: Off + Linear Filtering

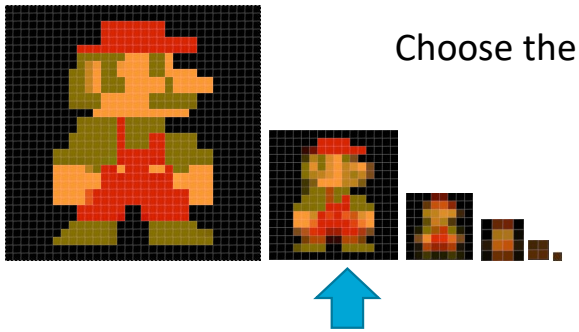
Similarly bad!

Better!

MipMapping: Approximate Filtering

- Idea: Compute a filtered texture beforehand





Choose the correct level depending on pixel-to-texel matching

e.g., 4 texels per pixel

MipMapping On (use nearest level)

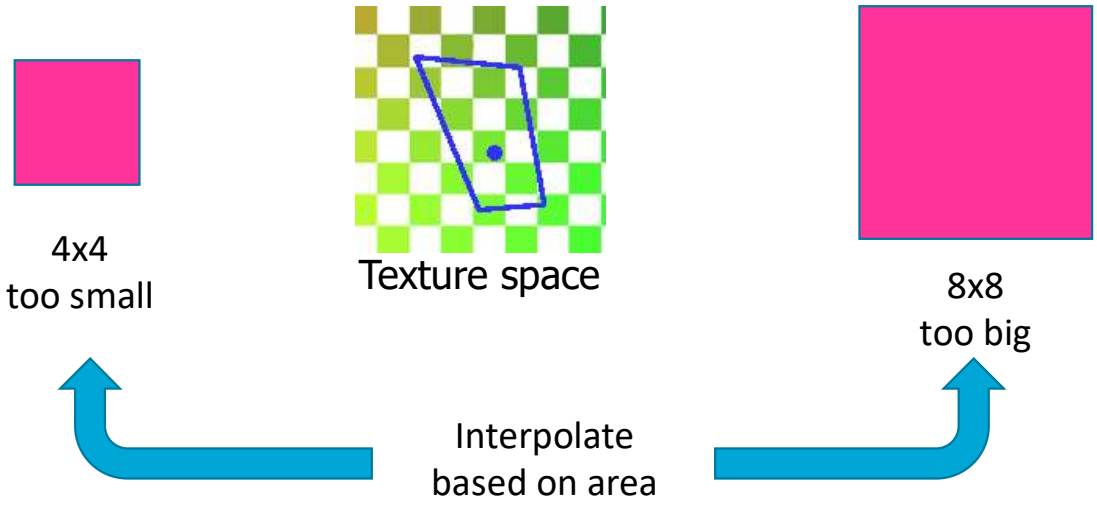
- Discontinuities at jumps from one level to next

Much nicer!

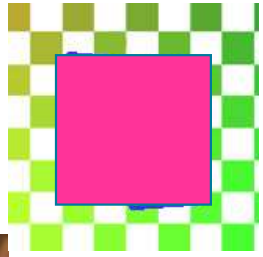
Bad: Different levels visible!

MipMapping

- Discontinuities come from changing region size



MipMapping: TriLinear Filtering

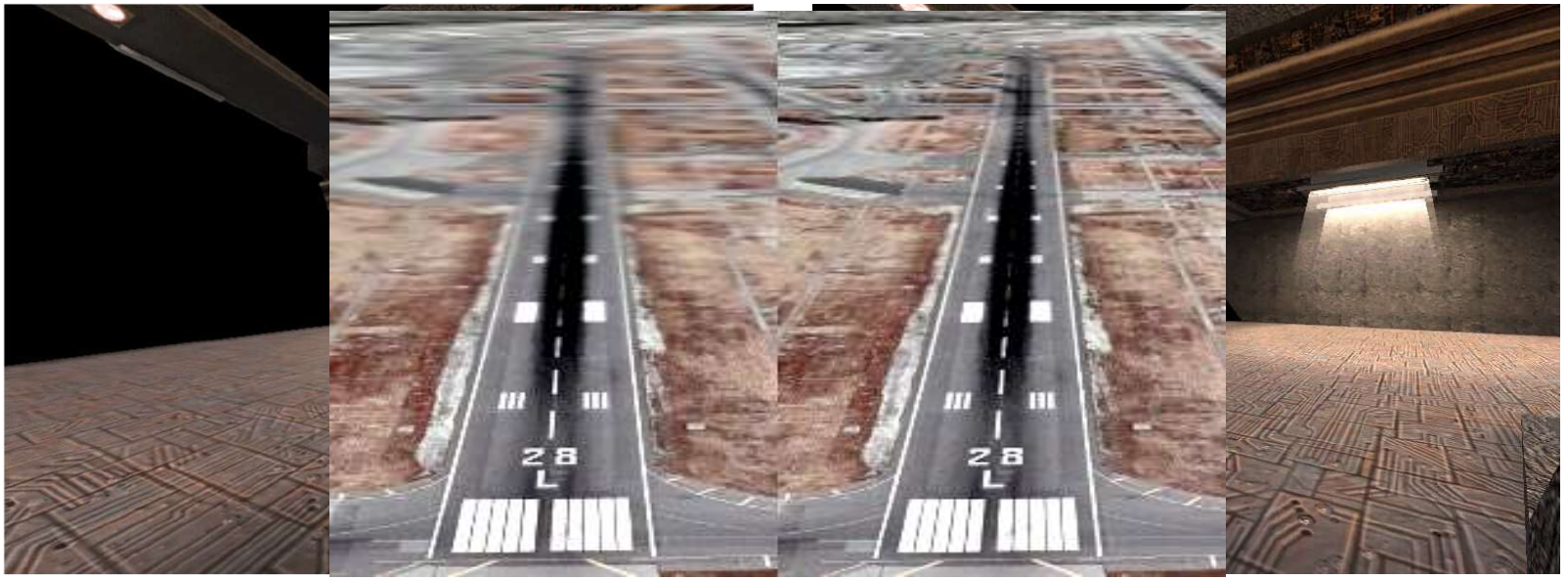


- Blend (mix) between different levels

Anisotropic: Comparison

MipMapping

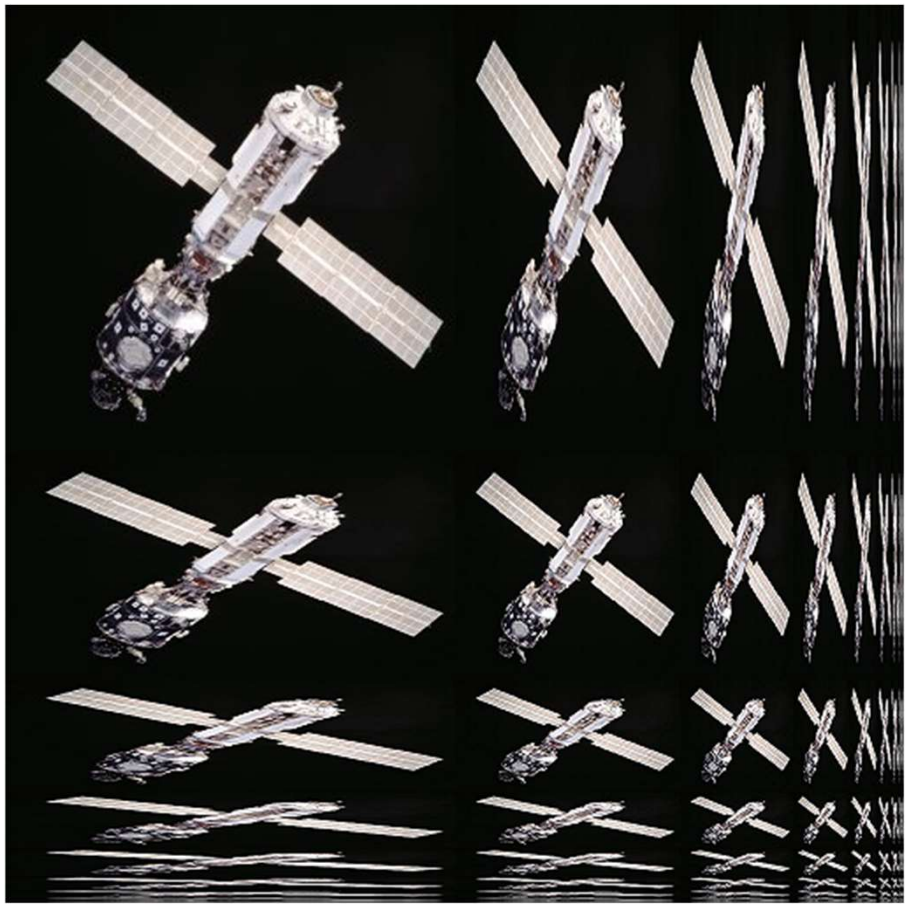
Anisotropic



- Better approximates pixel projection in texture

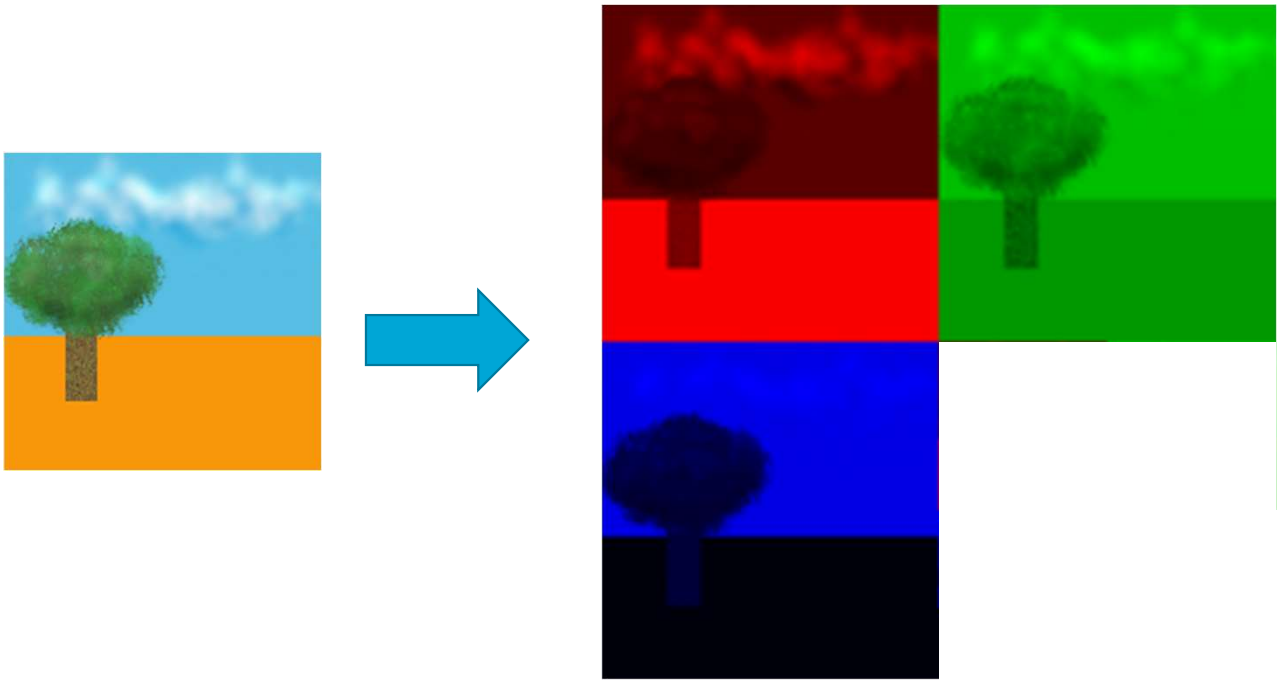
Anisotropic MipMaps

- Storage:
  - Power of two scaling on each axis
  - Fits into 4 times original memory

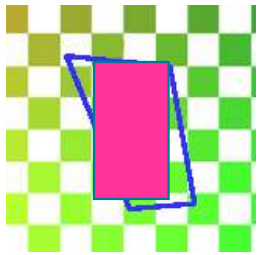


MipMapping: Memory Requirements

- Visual Proof that it is <1/3 extra memory

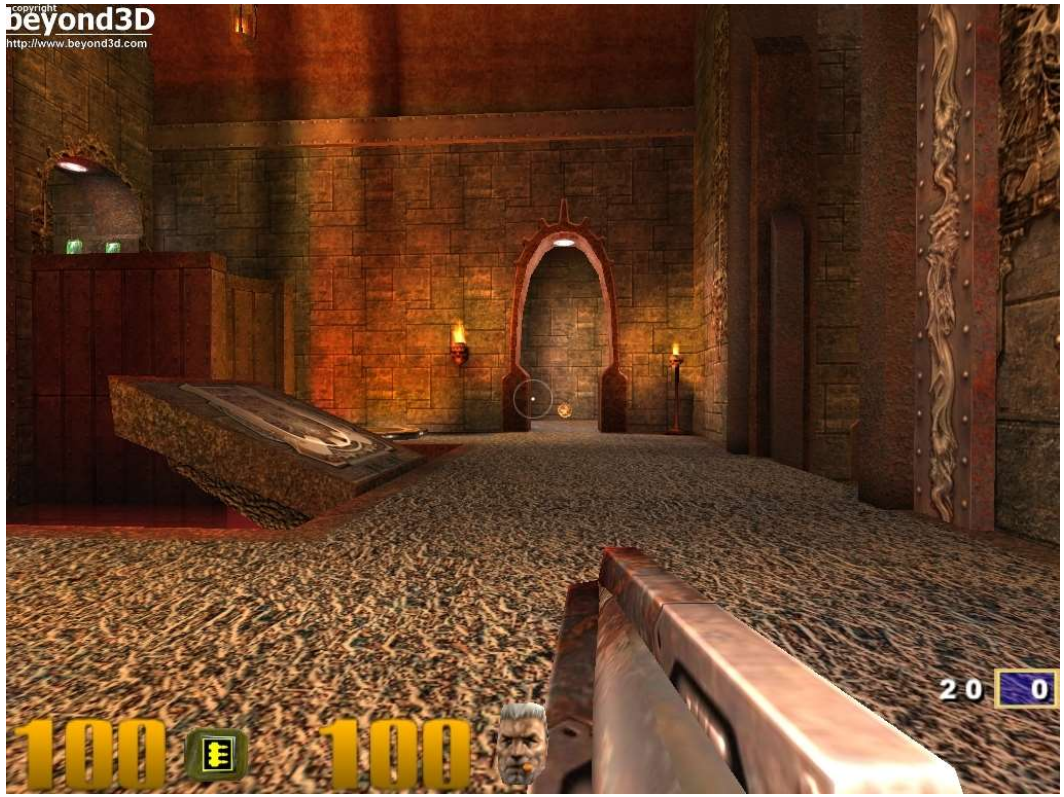


Anisotropic Filtering



- Better approximate real region

Beautiful!

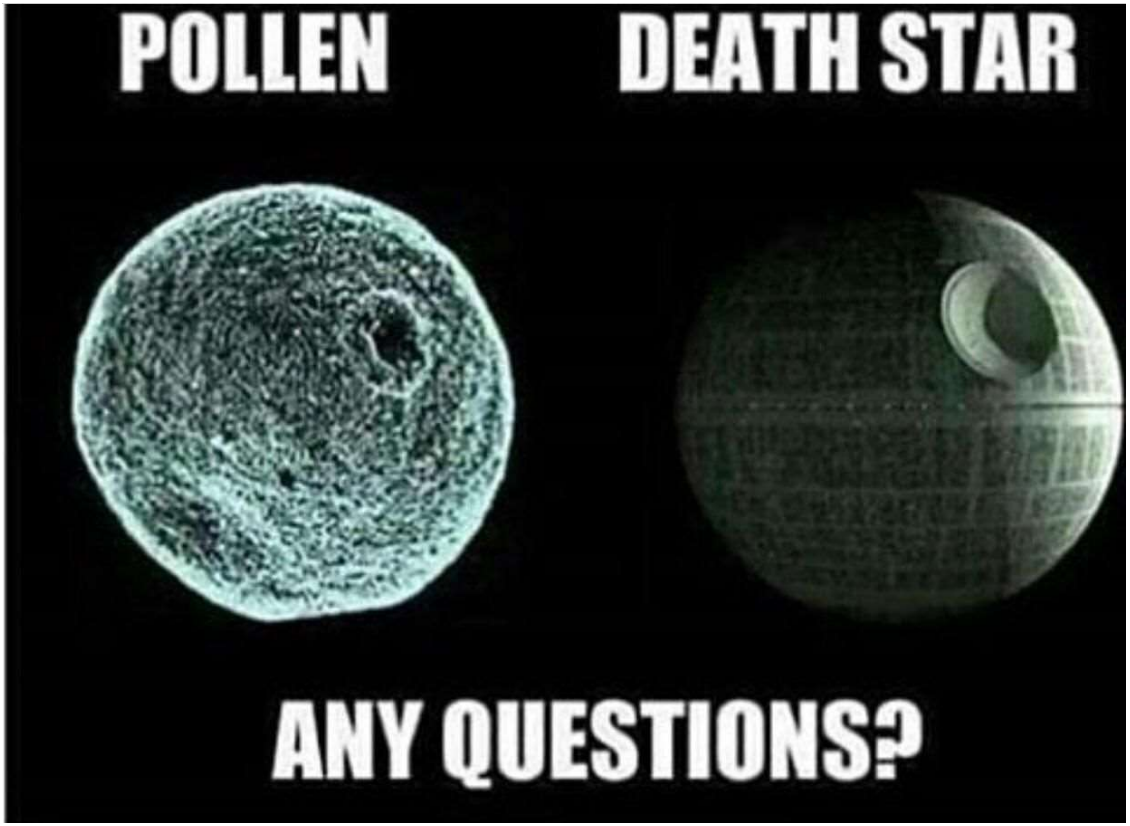


Textures

- Image content mapped on surfaces
- Increase detail level without geometric cost

- Many applications for color textures

Questions?



Advanced Texture Representations

- Texture Data can represent more than just colors of an image

- Modern definition:
  A texture stores values associated to a domain

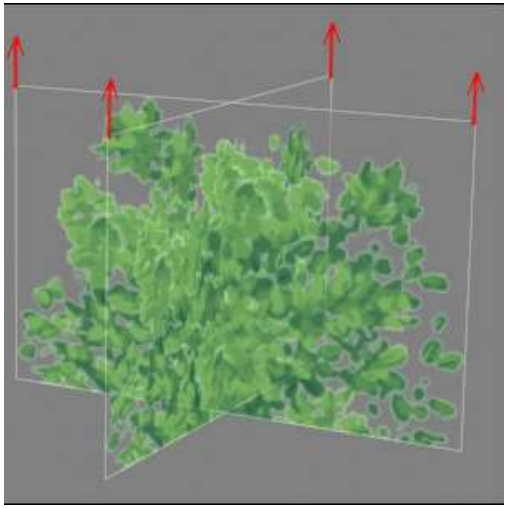
Advanced Texture Representations

- Texture Data can represent more than just colors of an image

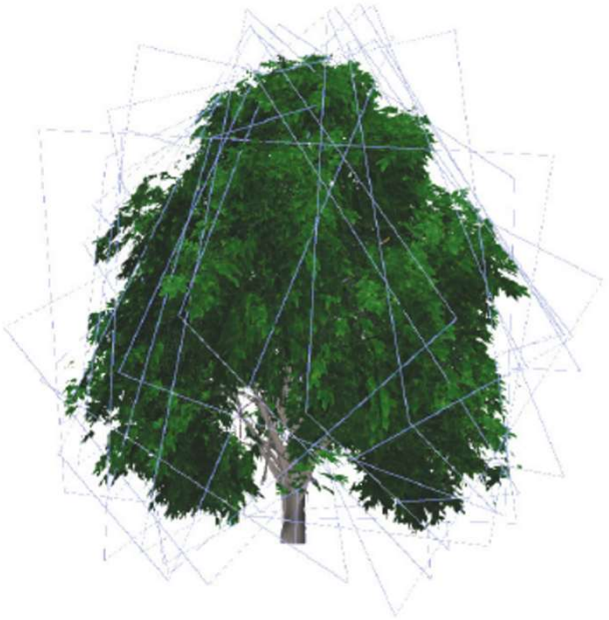
- Modern definition:
  A texture stores **values** associated to a domain

Alpha Blending

- Used to create transparency effects
- Textures can have (R,G,B,A) values:
  - If A=0 : texel is considered transparent – not drawn
  - If A=1 : texel is drawn

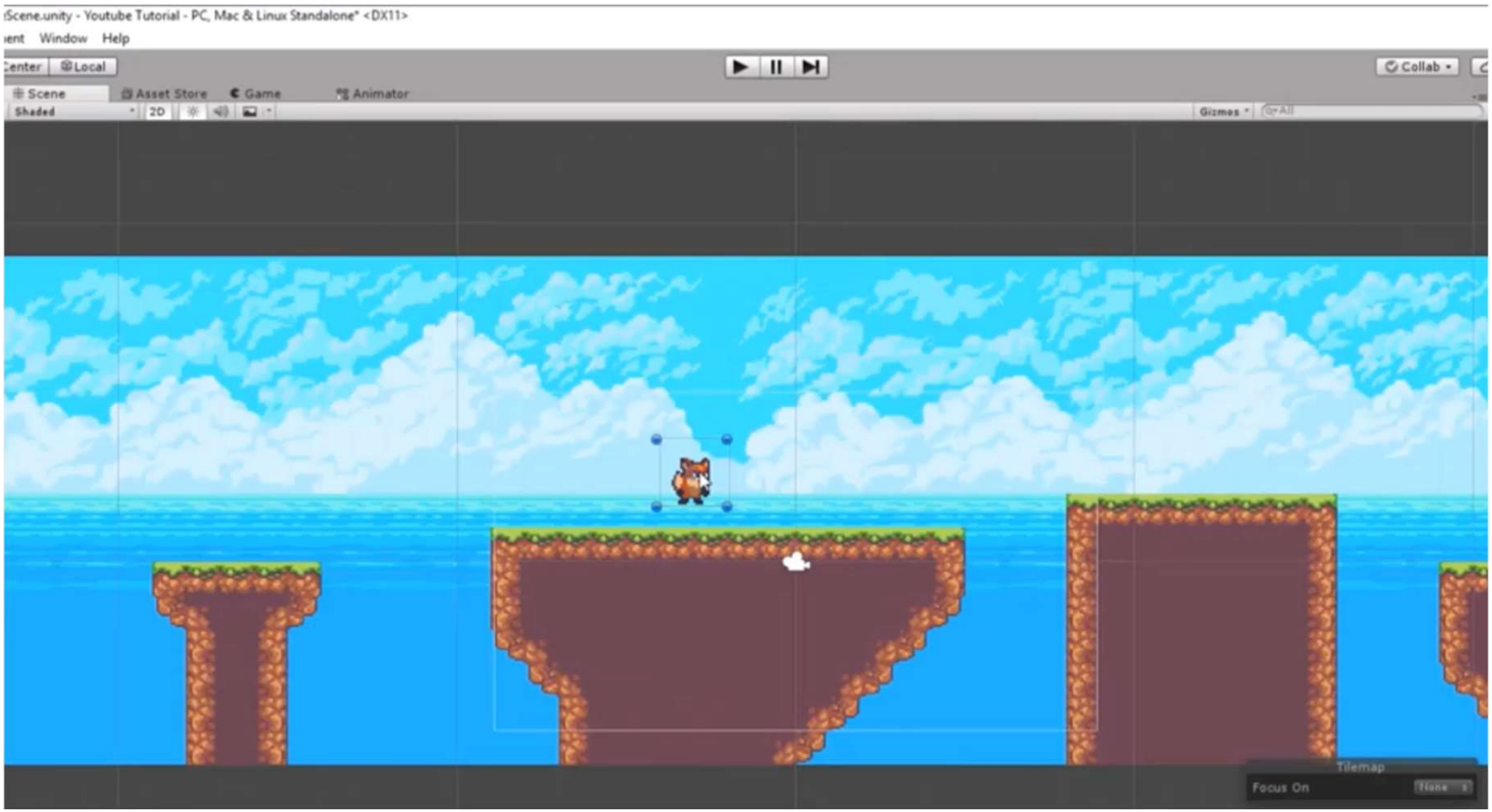


Alpha Blending - Stacked Polygons



- Source: irrlicht engine, artlplants.blogspot.com

Alpha Blending - Sprites (here Unity)



Use pixel discard based on alpha to make the object background transparent

Alpha Blending - Water Particles

- Textured quads used as water particles

Advanced Texture Representations

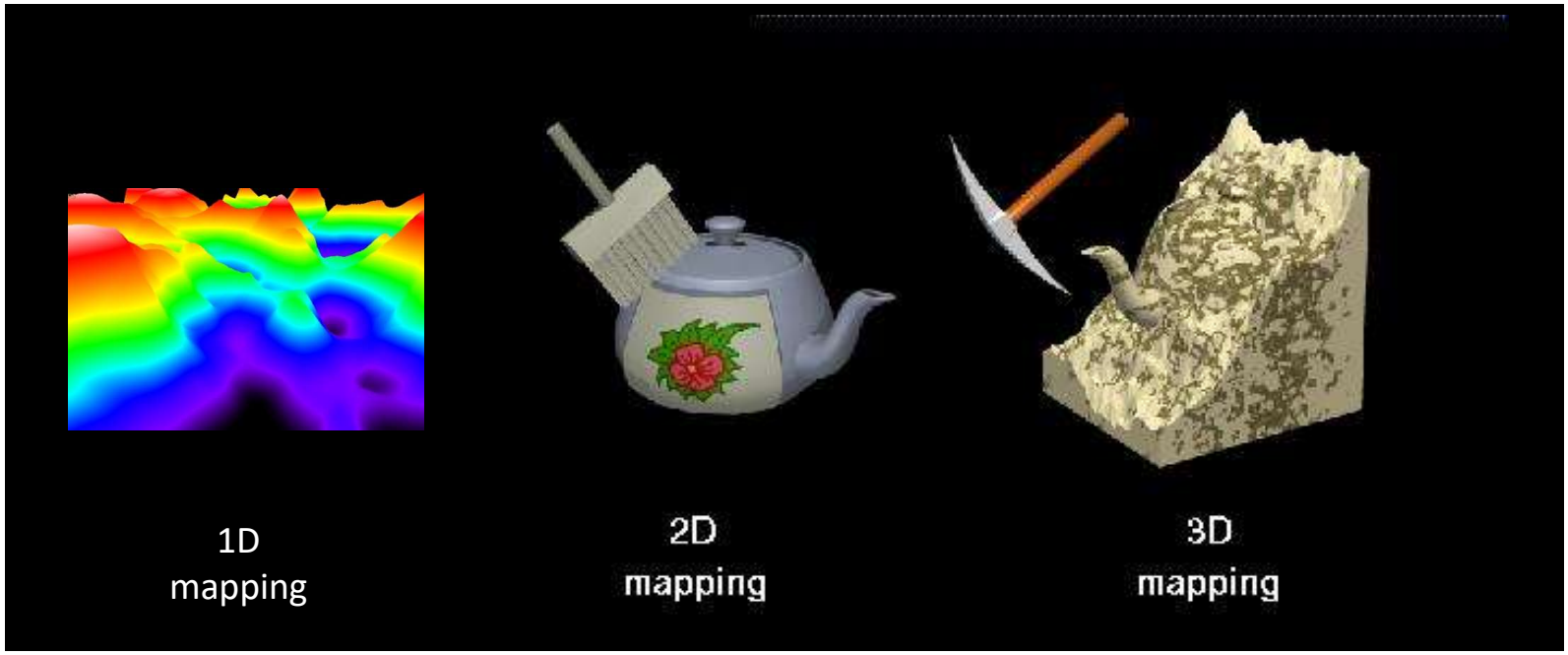
- Texture Data can represent more than just colors of an image

- Modern definition:
  A texture stores values associated to a **domain**

domain is accessed via the texture coordinates

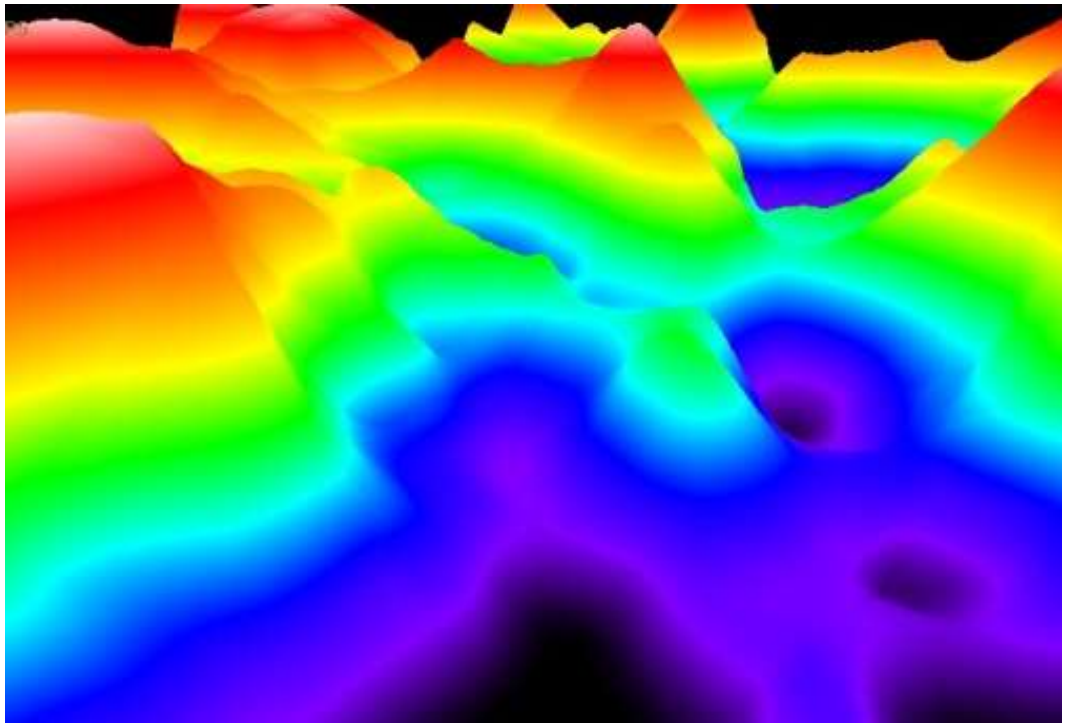
Advanced Texture Representations

- Different domain dimensions
- Texture coordinates can be 1D, 2D, 3D…
  GPUs provide native support up to 3D



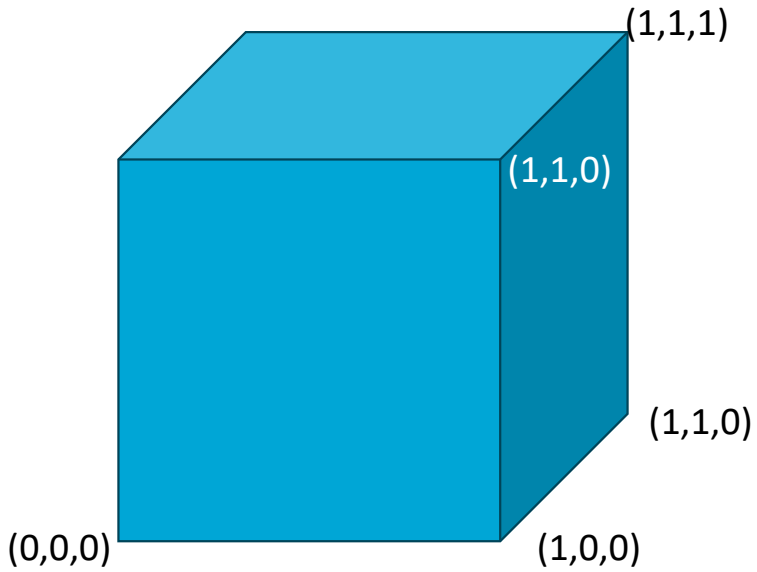
1D Textures

- We can theoretically use a 1-pixel wide 2D Texture via glTexCoord2f(0,z).
- A 1D Texture always has 1-pixel width. Access with glTexCoord1f(z).



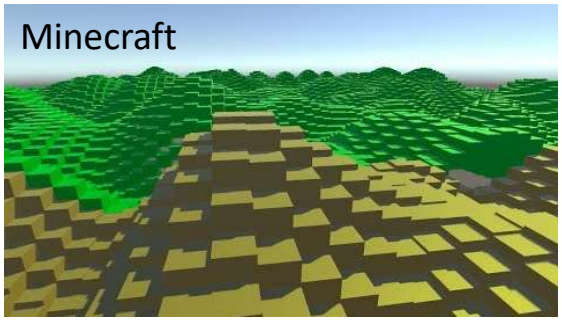
What are 3D Textures?

- They are accessed with 3 texture coordinates (u,v,w)
- Bilinear interpolation in 2D textures becomes trilinear with 3D textures



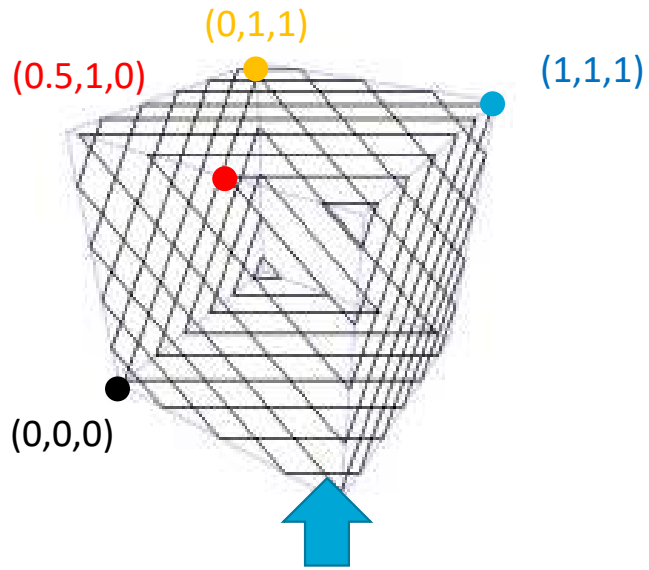
3D Textures are Volumes

- The equivalent of a texels is a small cubic volume (**vo**lume **el**ement=**voxel**)



Volume Rendering

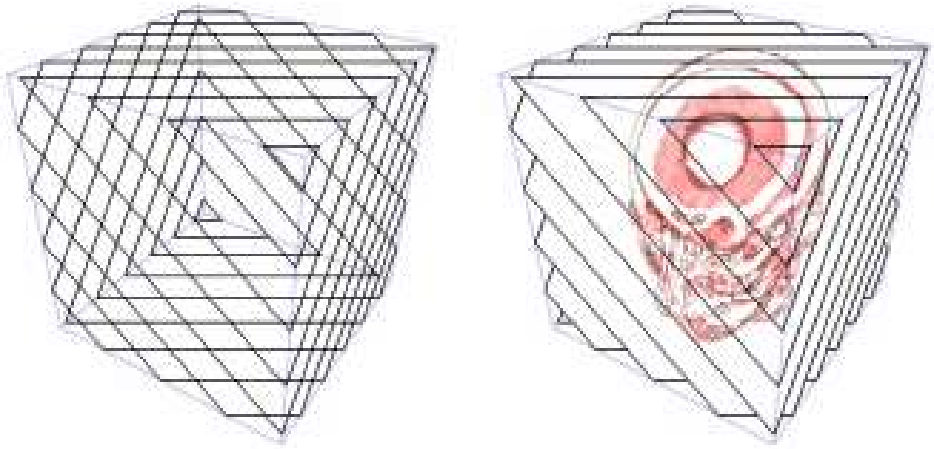
- How to render volumes?
  - One often-employed method is slicing



Draw slices with 3D texture coordinates corresponding to the position of the vertices in the volume

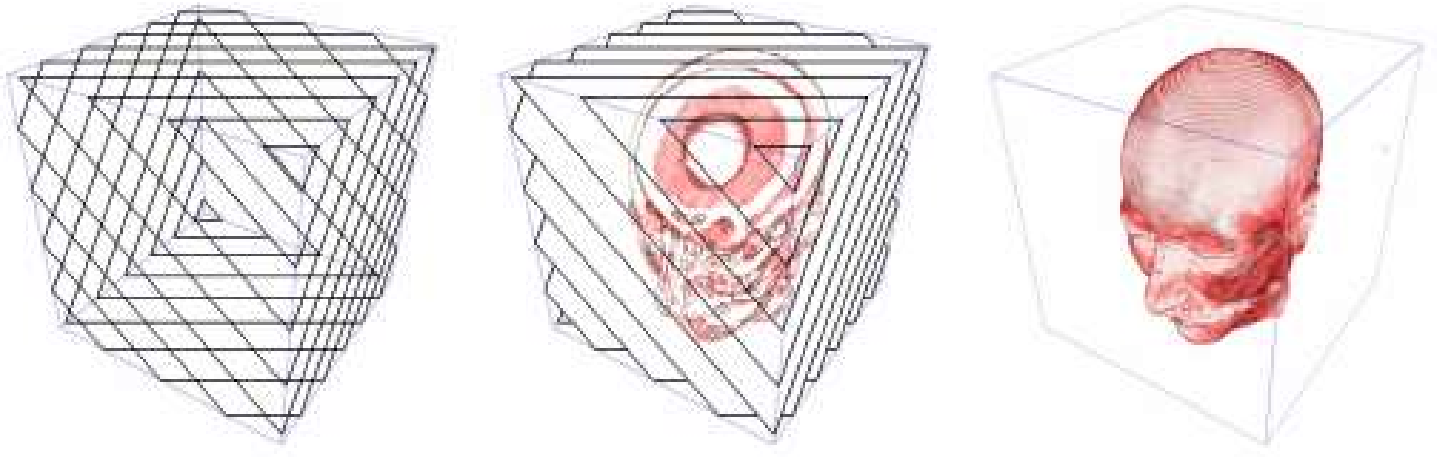
Volume Rendering

- How to render volumes?
  - One often-employed method is slicing



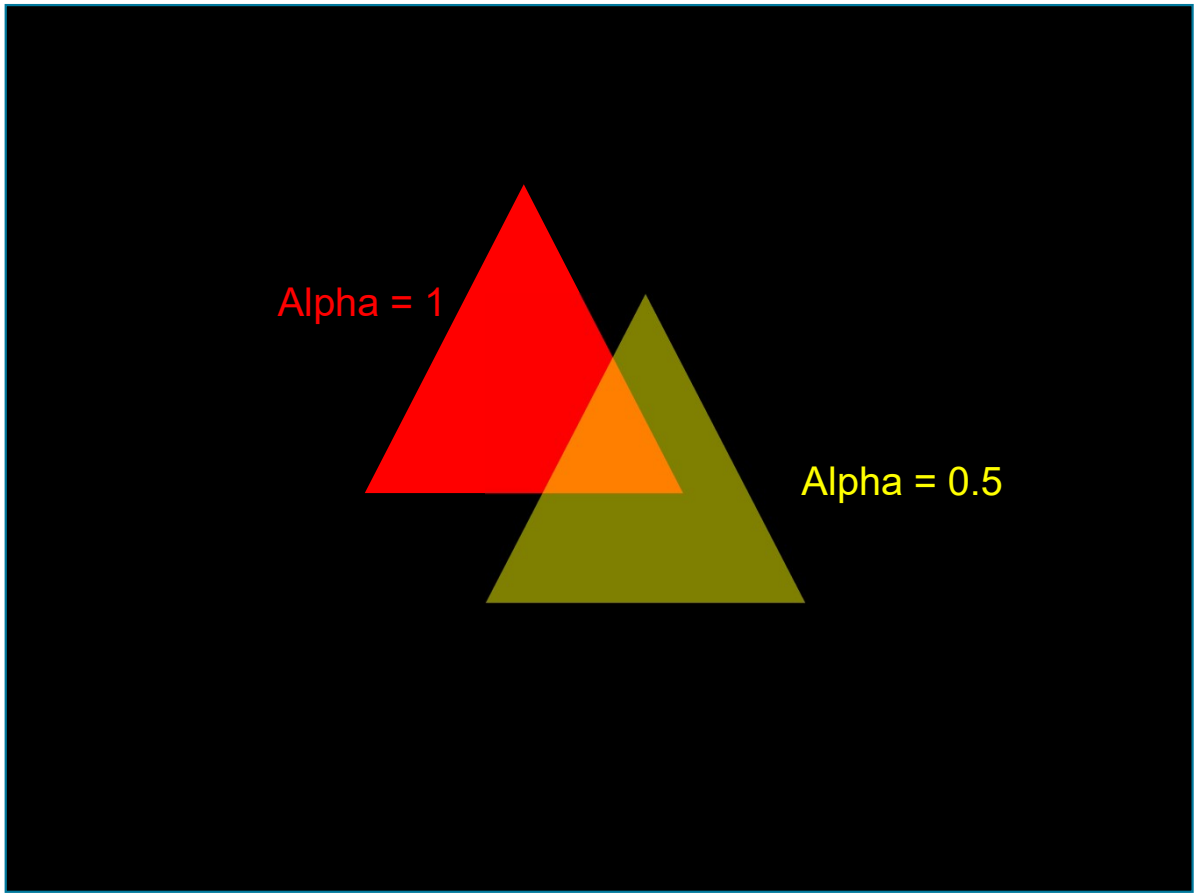
Volume Rendering

- How to render volumes?
  - One often-employed method is slicing



But wait? Why are there parts that are transparent?

Alpha Blending

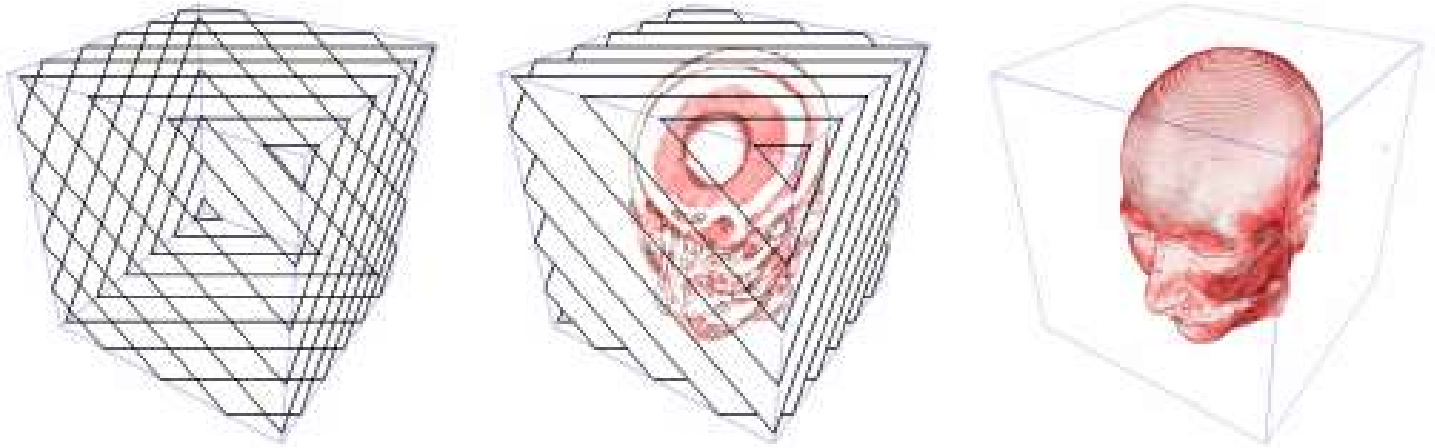


Standard Alpha Blending

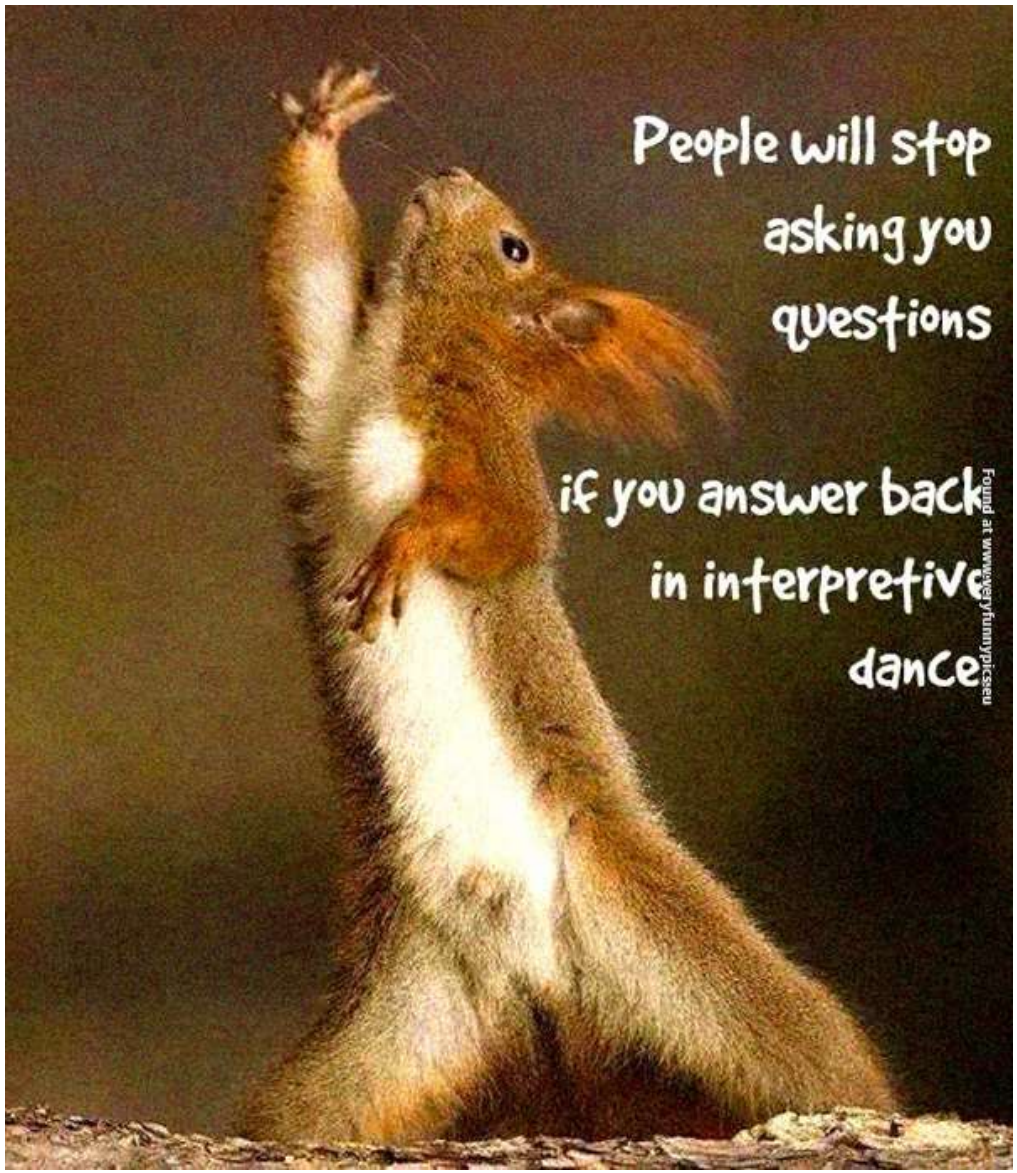
- Color with four components: RGB and an alpha value
- Alpha describes the “opacity”
  - 0 : object is completely transparent
  - 1 : object is completely opaque
  - 0.5: 50% of the stored color and 50% background
- In general, drawn (R,G,B,A) and in a pixel with color ($R_B G_B B_B$) results in: A * RGB+ (1-A) * $R_B G_B B_B$

Volume Rendering

- One often-employed method is slicing



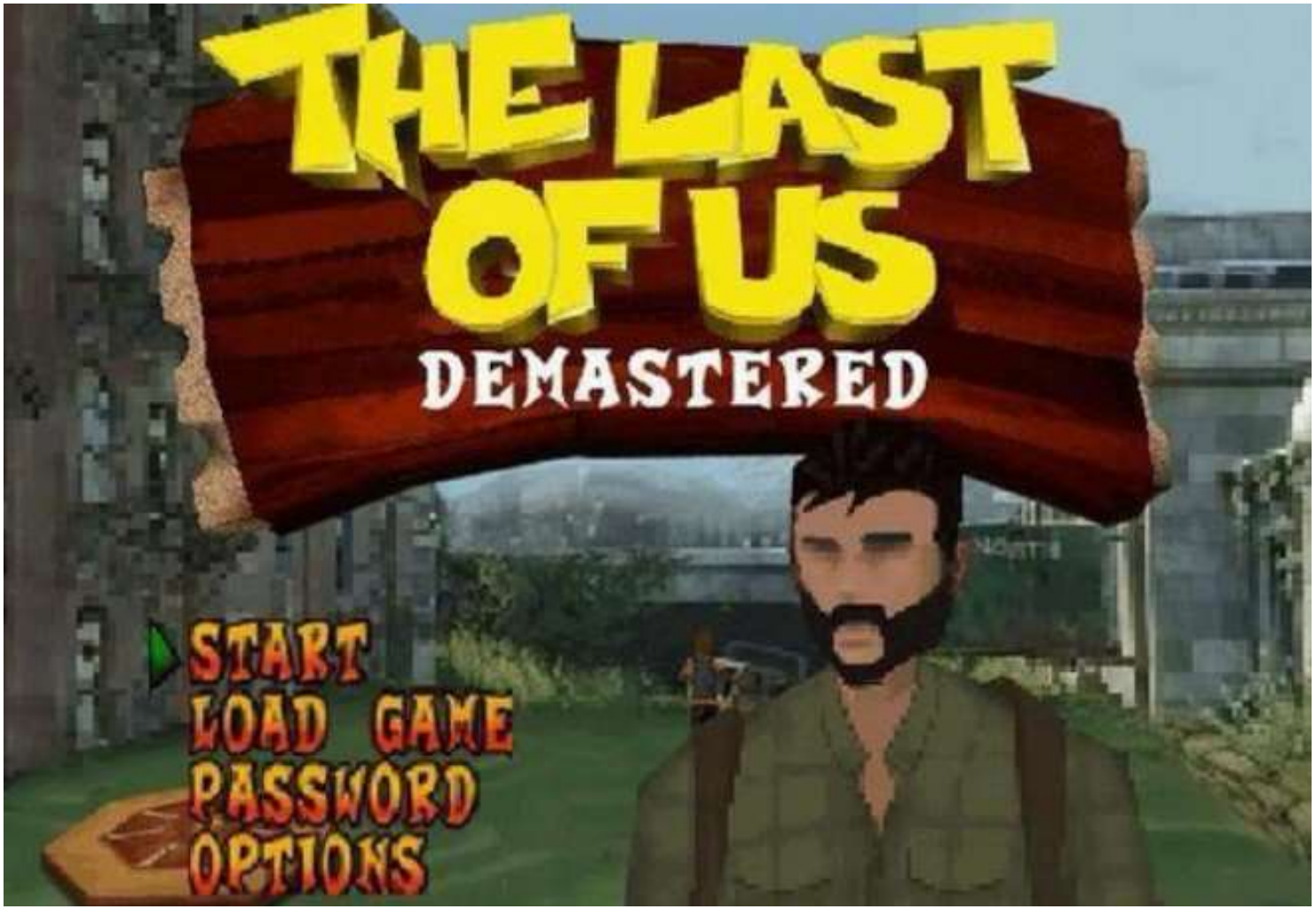
Questions?


People will stop
asking you
questions
if you answer back
in interpretive
dance
Found at www.veryfunnypicseu

Advanced Texture Use-Cases

- Encode material properties
- Approximate the environment
- Represent Geometry
- Encode complex functions

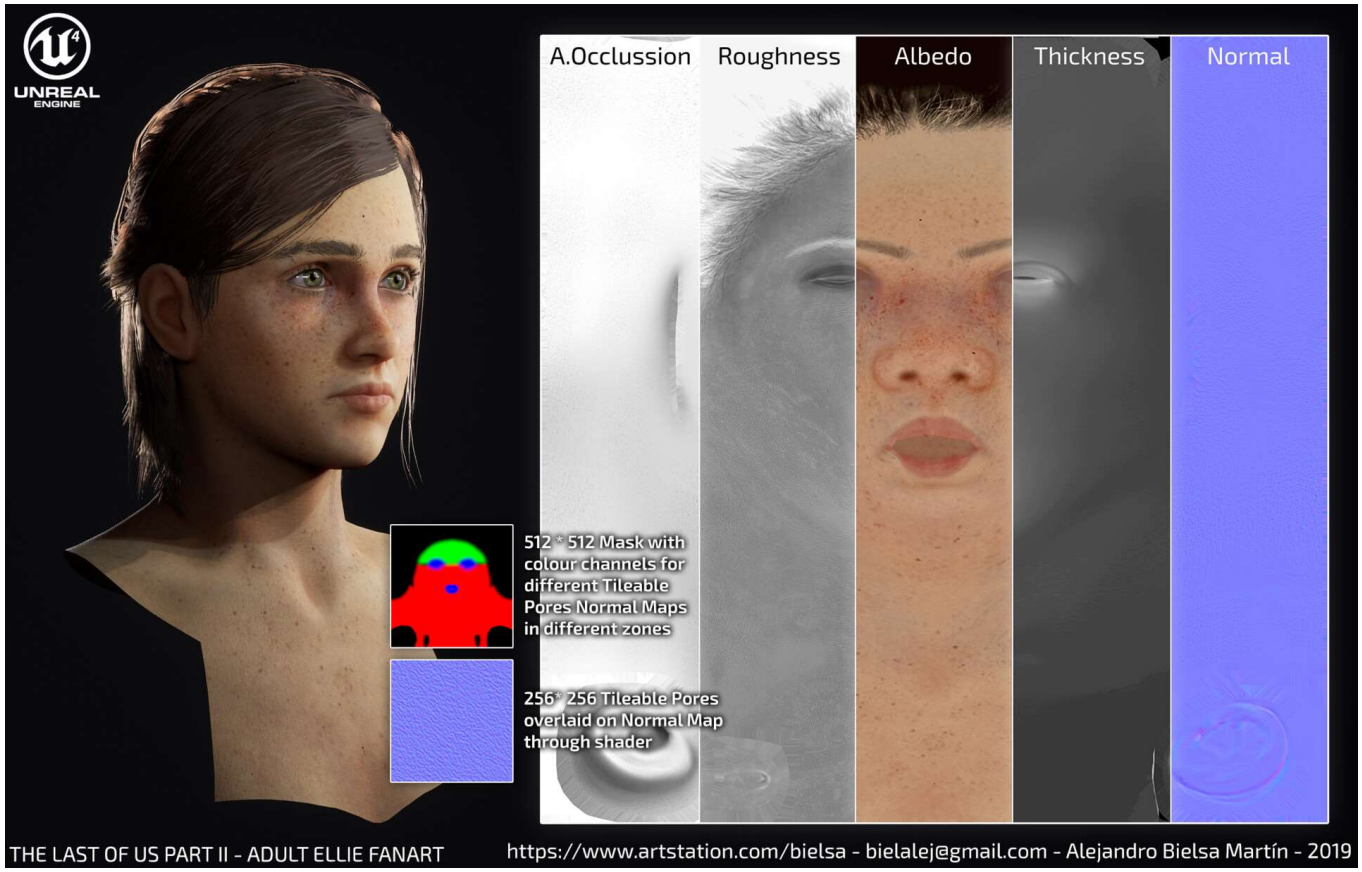
Last of Us on PS1



Last of Us on PS1

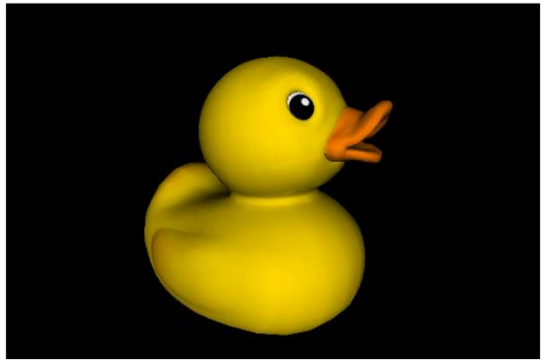
Last of Us 2

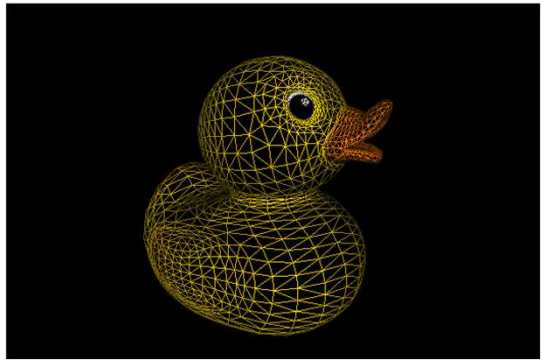
Professional Example



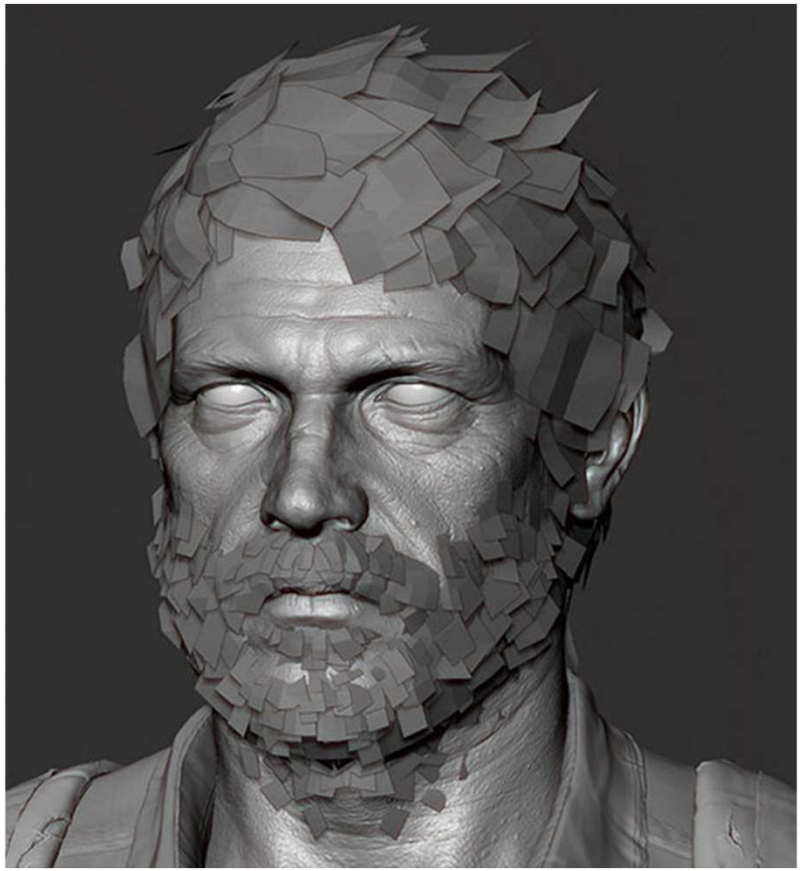
Textures: Materials

- Can be used to provide, e.g.,
  - parameters for material models (ambient, diffuse…)





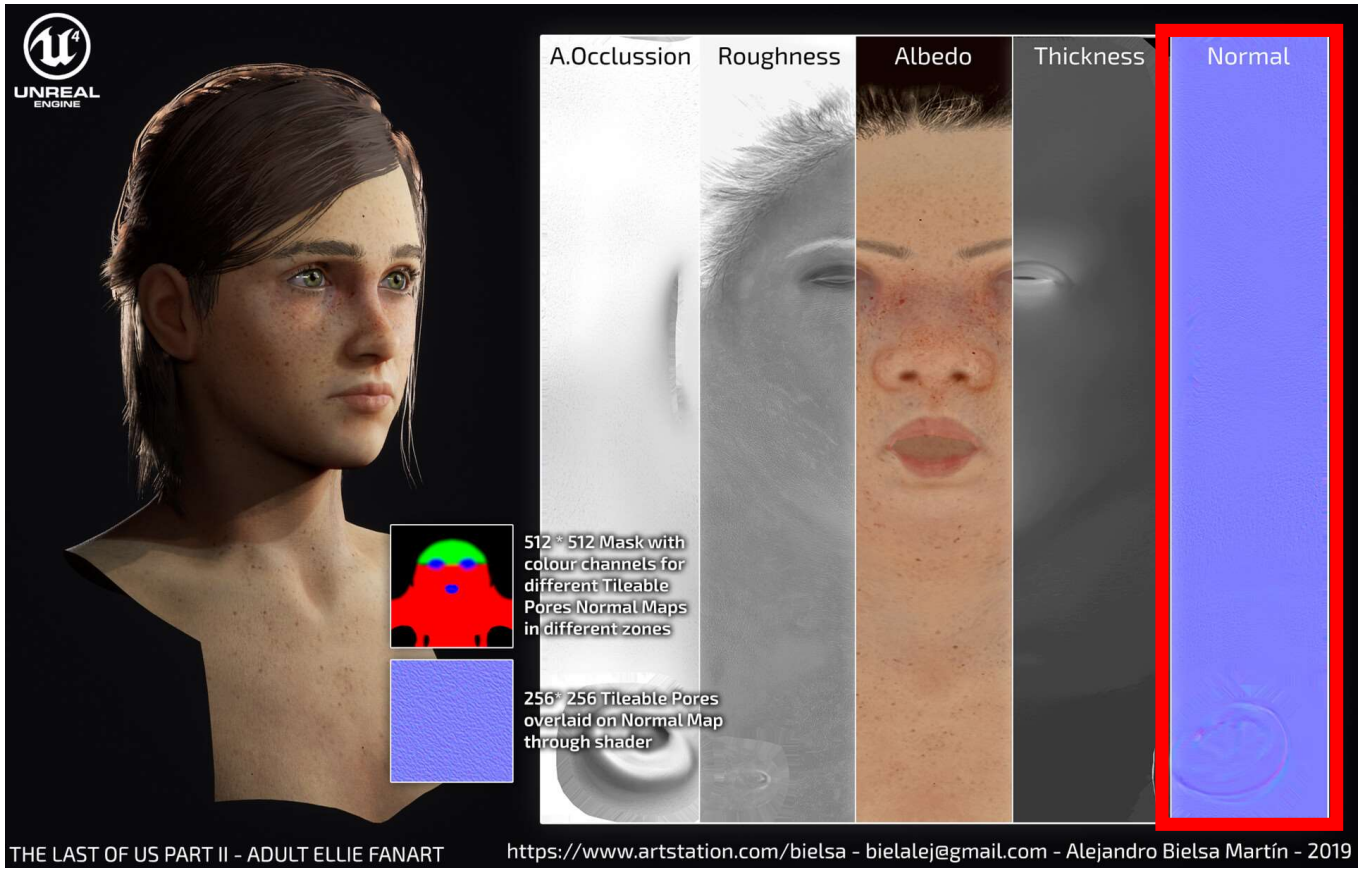
Alpha Blending - Hair



Alpha Blending - Hair

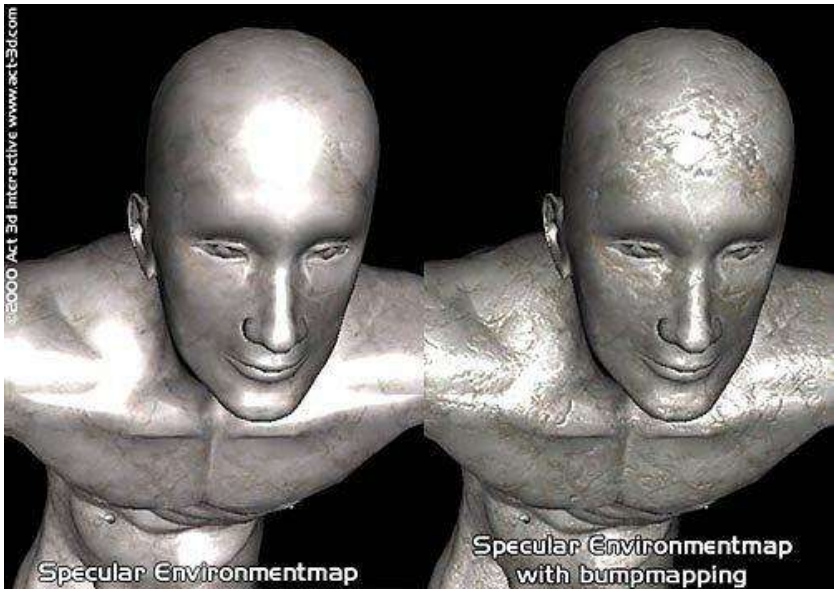
Last of us

Professional Example



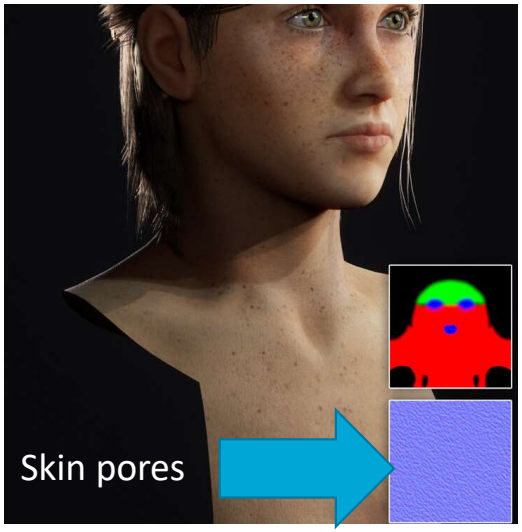
Textures

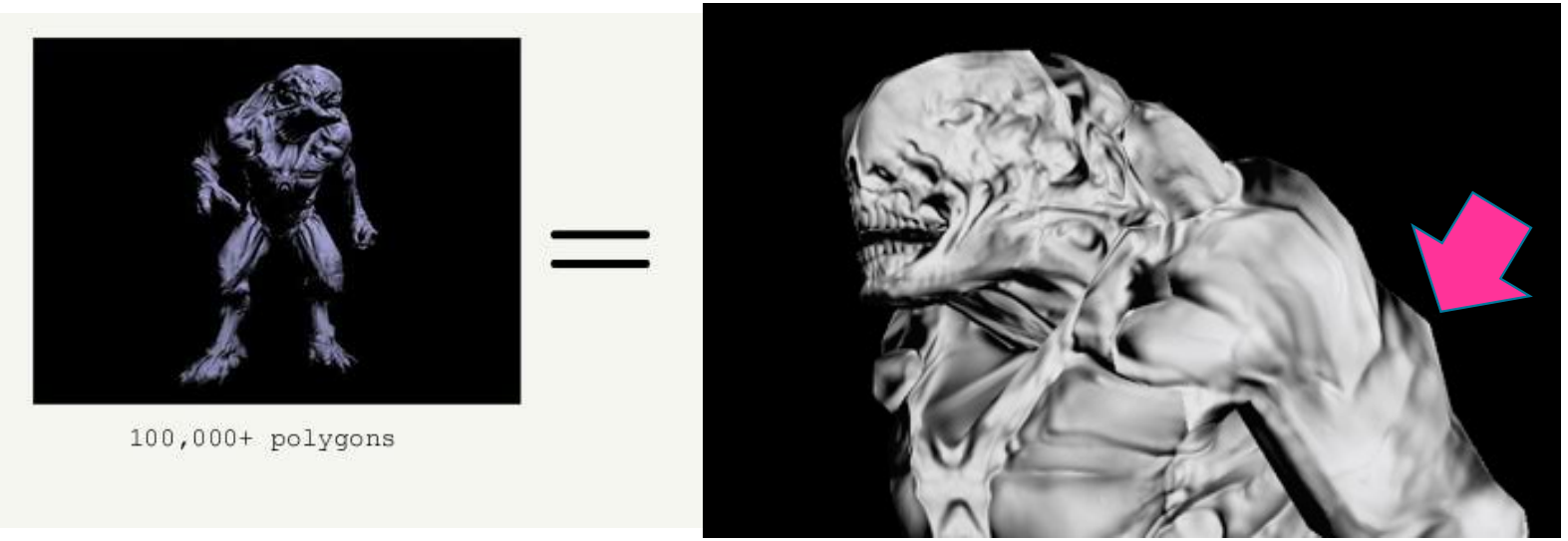
- Can be used to provide, e.g.,
  - Colors for ambient/diffuse
  - Normals (*bump mapping*)



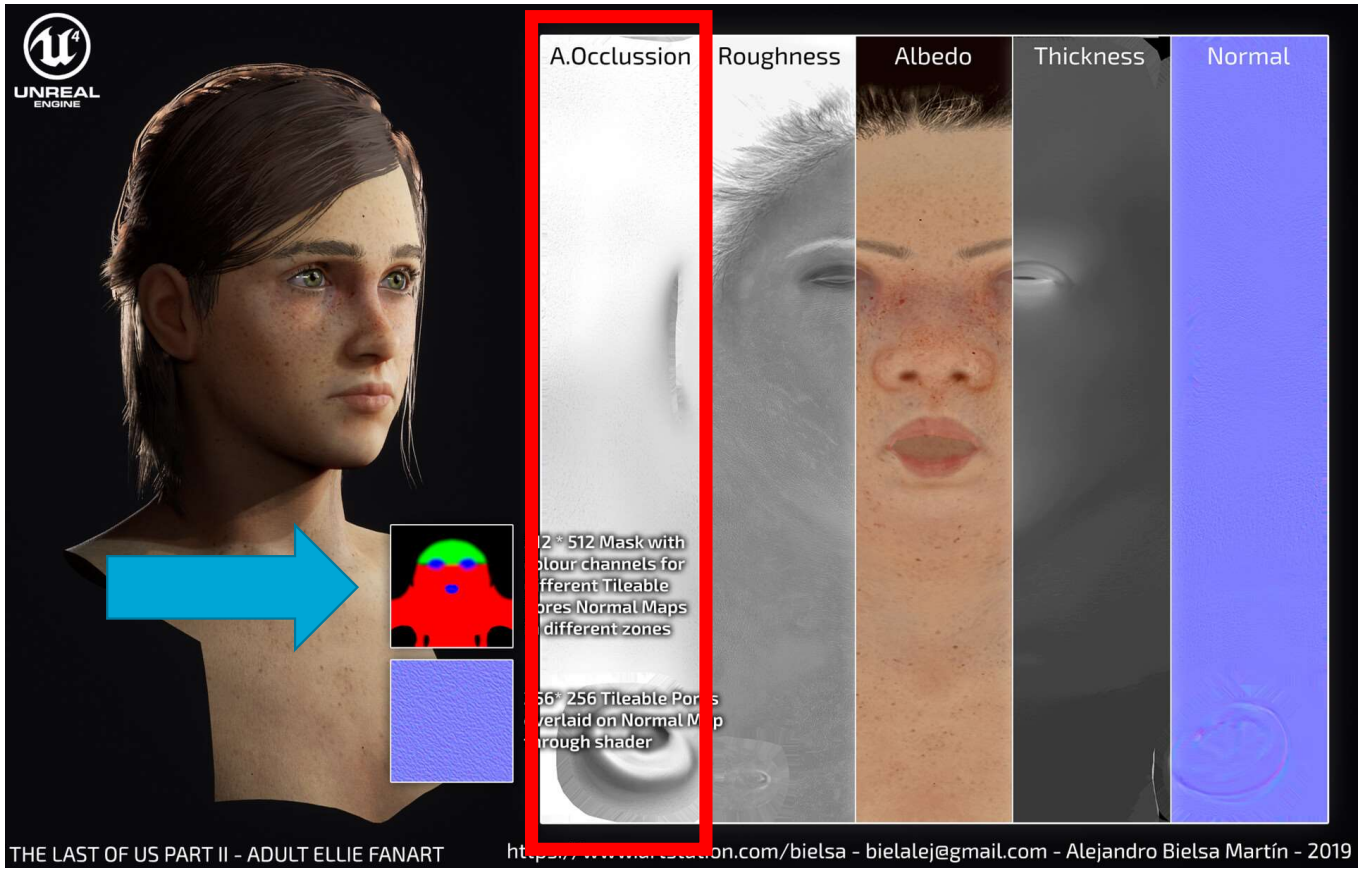
Textures

- Normal Mapping





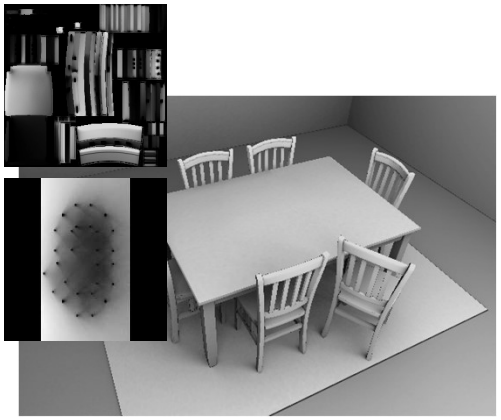
Professional Example



Textures

- Can be used to provide, e.g.,
  - Colors for ambient/diffuse
  - Normals (*bump mapping*)
  - Precomputed illumination (*light map*)

Diffuse mapped on scene



Lightmap mapped on scene

Combined

Textures

- Can be used to provide, e.g.,
  - Colors for ambient/diffuse
  - Normals (*bump mapping*)
  - Precomputed illumination (*light map*)

Quake – id Software

Materials

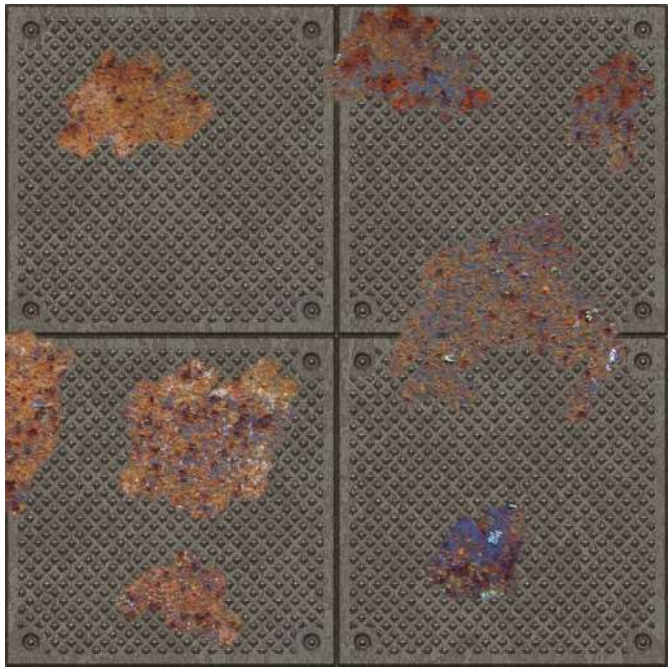
*

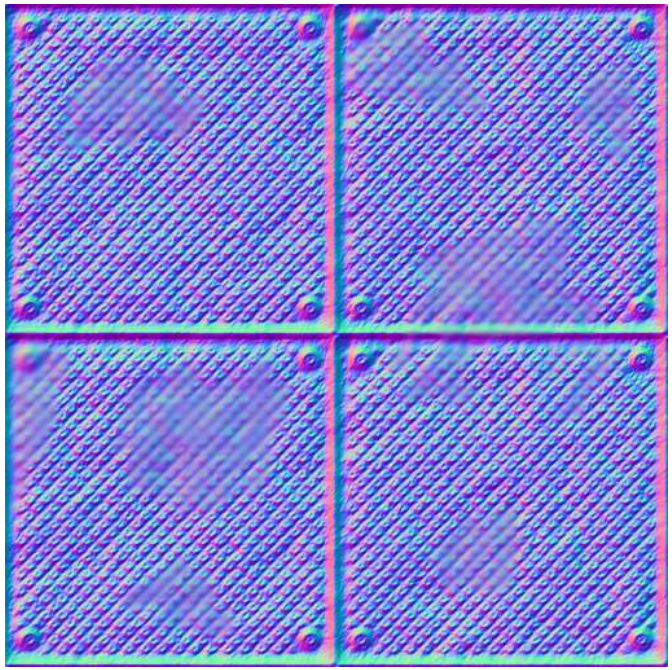
Light Map

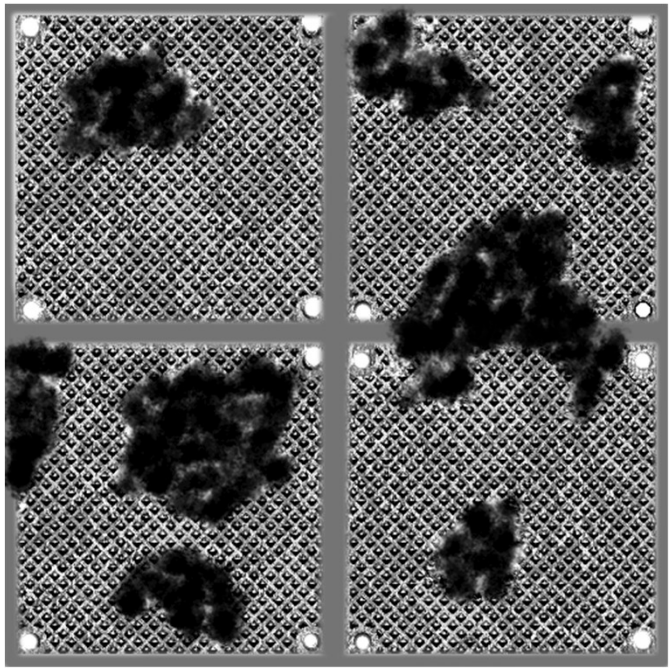
=

Final result

Demo - Building a Corroded Metal Sheet







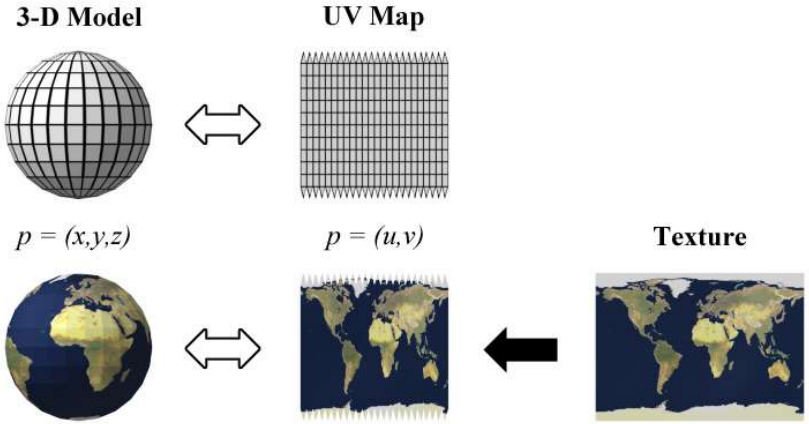
Advanced Texture Use-Cases

- Encode material properties
- **Approximate the environment**
- Represent Geometry
- Encode complex functions

Environment Mapping

- Textures can encode an environment
  An environment map (approximation of scene)

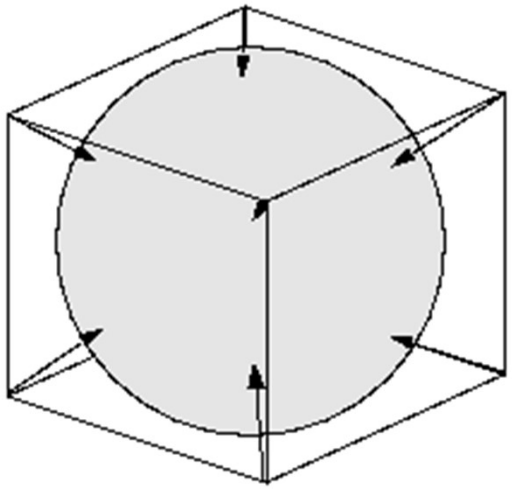
Spherical mapping



Environment Mapping

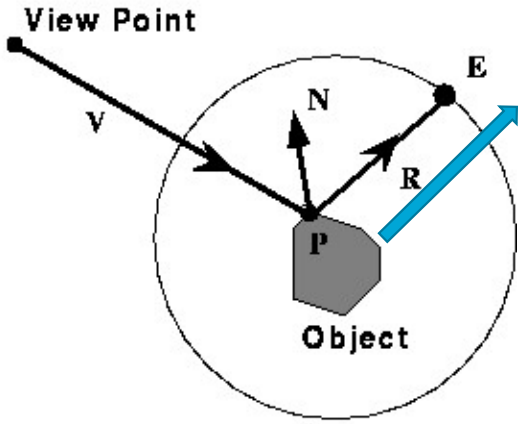
- Textures can encode an environment
  An environment map (approximation of scene)

Cube mapping



Environment Mapping

- Textures can encode an environment
  An environment map (approximation of scene)
  is useful for, e.g., reflections (texcoords = ray)

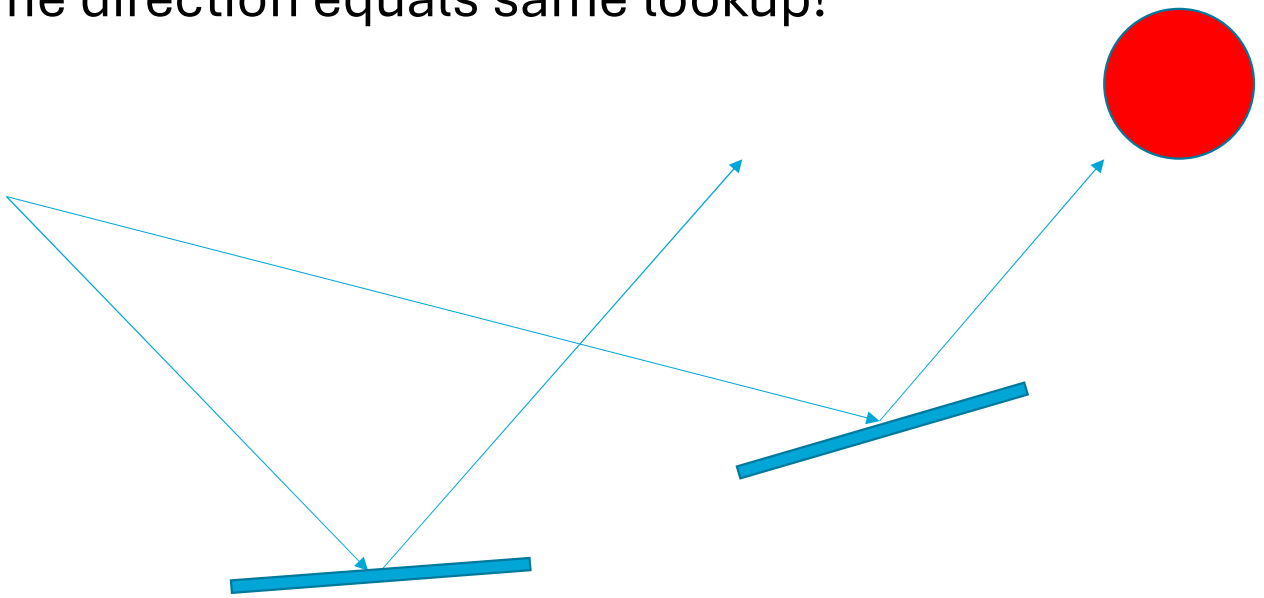


Ray direction determines lookup position

- This *environment mapping* is an approximation!

Environment Mapping

- Typical approximation:
  Assume environment is infinitely far away.
  Same direction equals same lookup!



- Red ball reflection seen by both or none.

Environment Mapping

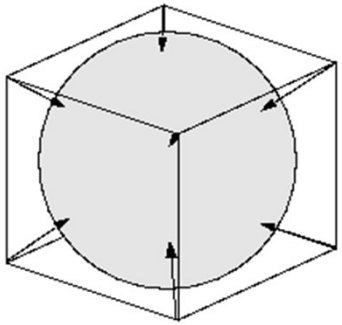
Not blown away yet?

- It is getting better!

- Textures can be dynamically created!

Dynamic Textures

Dynamic Cube map



[Hollaender et al. EGSR 2011]

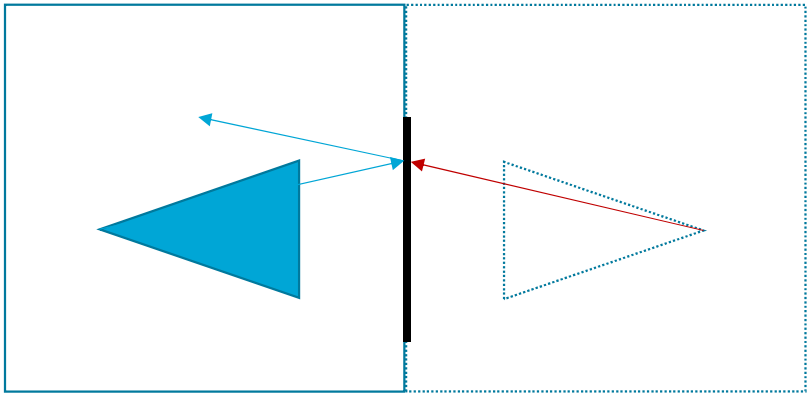
Dynamic Textures

- Flat mirrors can be well simulated with textures

- Duke Nukem3D

Environment Mapping

- Flat mirrors can be well simulated with textures

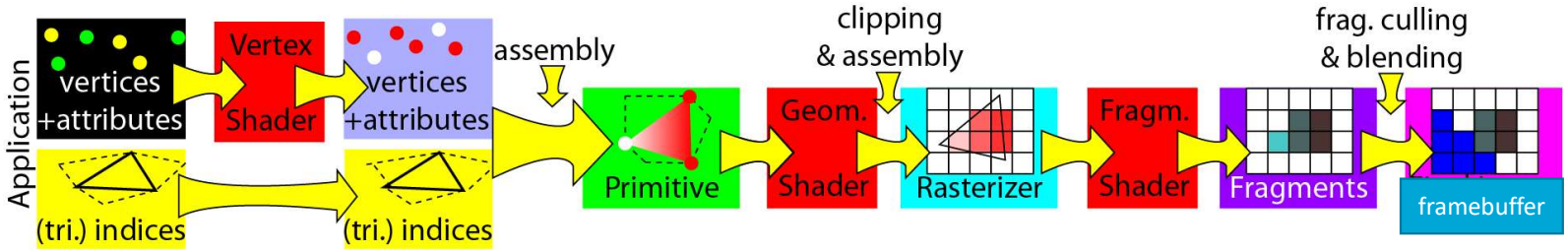


Fact Check:
DukeNukem marked the mirror pixels, flipped the scene and rendered again…

- Virtually flip camera center along mirror.
- Take an image from this virtual camera
- Apply image as a texture onto the mirror.

Render to a texture?

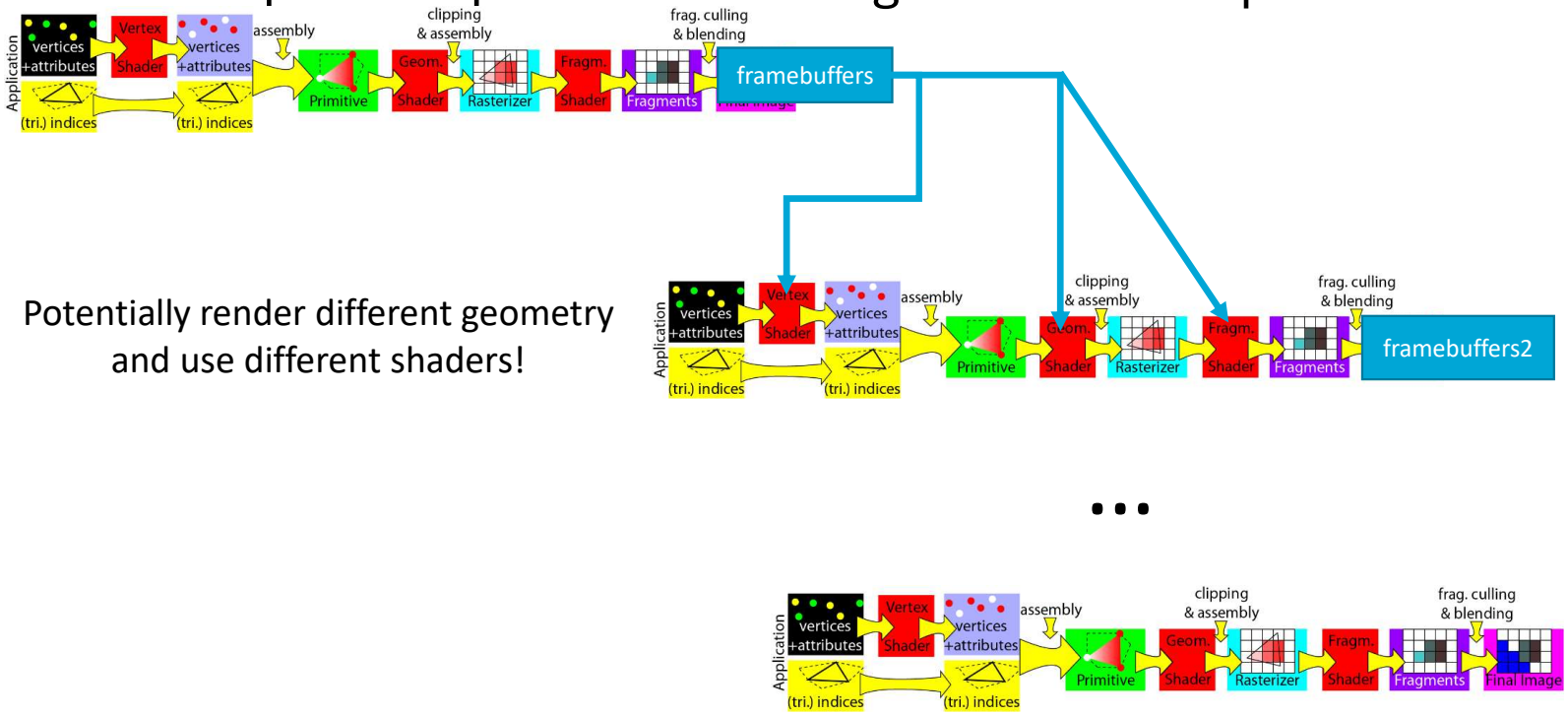
- Rendering to texture is done by attaching a “framebuffer object” to your pipeline.
- Output is written to this framebuffer



- You can attach several buffers/textures and write to the **same** pixel location (MRT – multiple render targets) – i.e., one fragment is drawn into several images!

Iterative Rendering

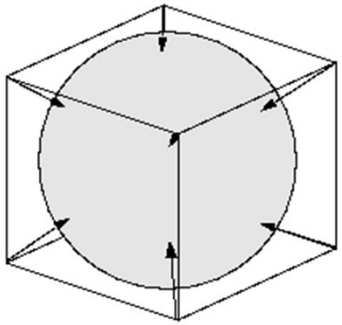
- Take texture output as input texture during a new render pass



Potentially render different geometry and use different shaders!

Dynamic Textures

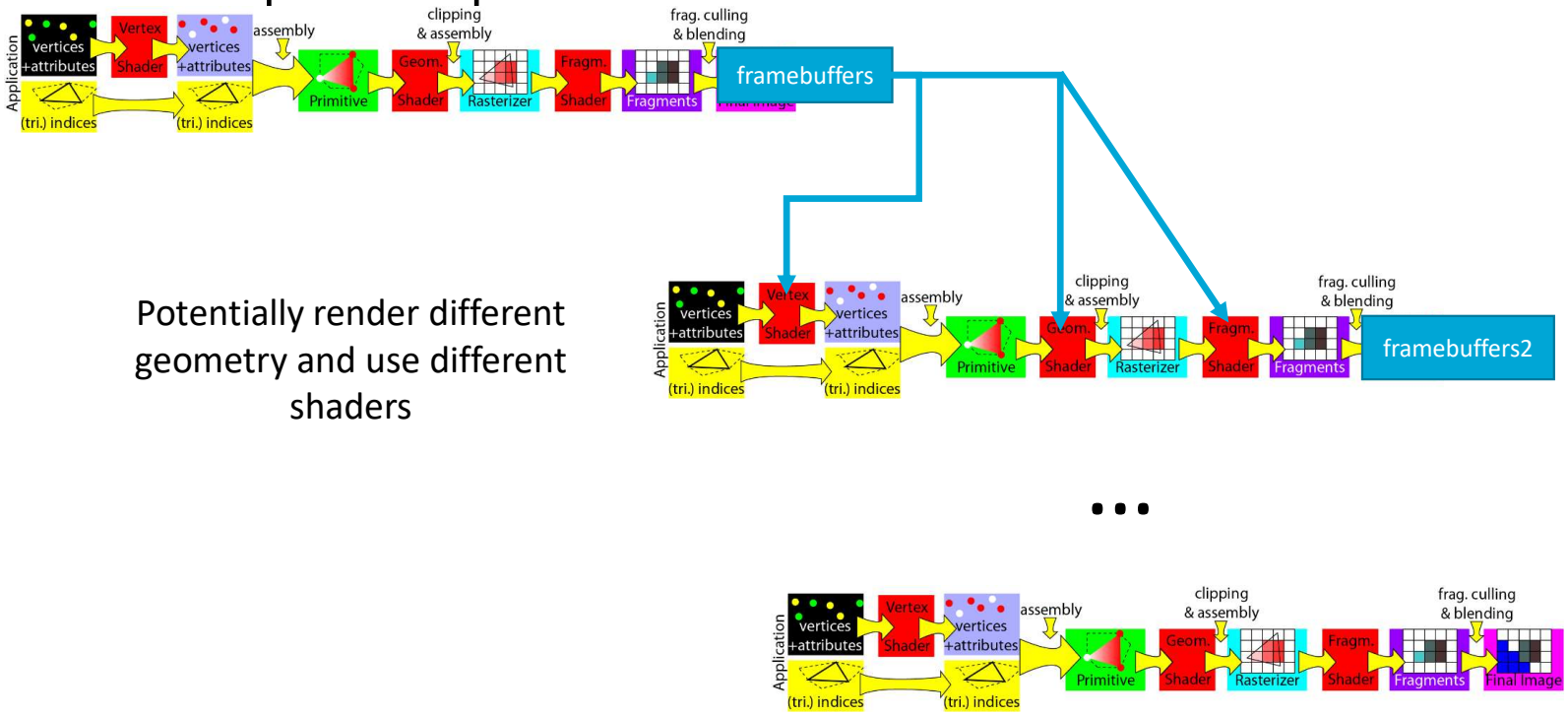
Dynamic Cube map



[Hollaender et al. EGSR 2011]

Iterative Rendering

- Take texture output as input to a new render



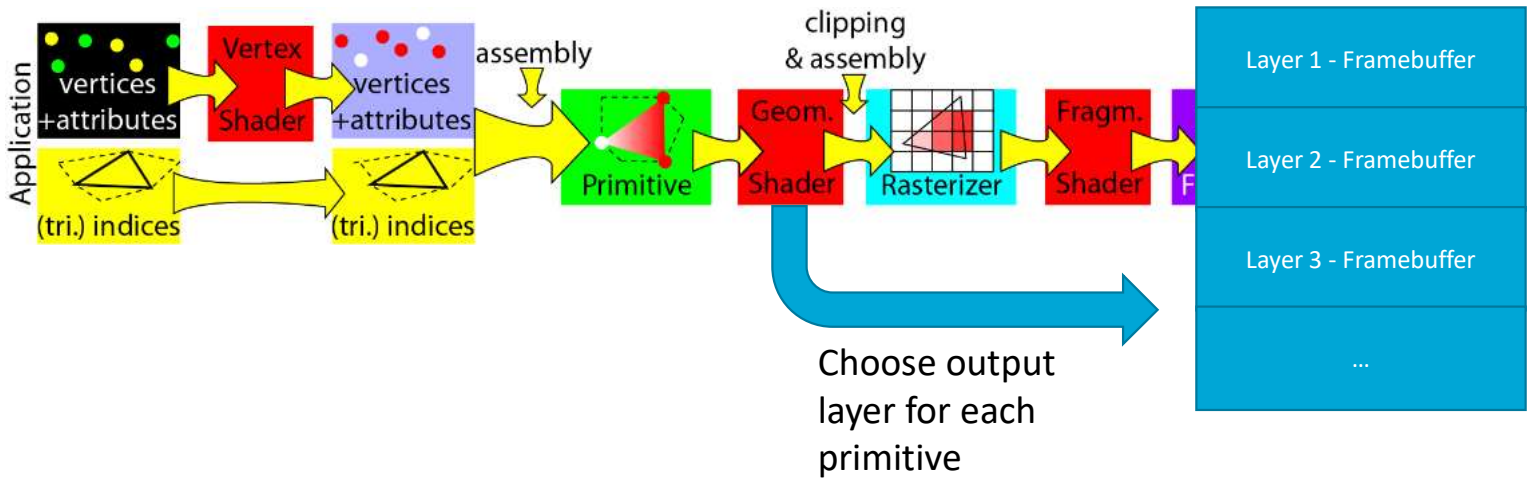
Potentially render different geometry and use different shaders

• • •

You can attach several buffers/textures and write to the **same** pixel location
(MRT – multiple render targets)

Multi-Output Rendering

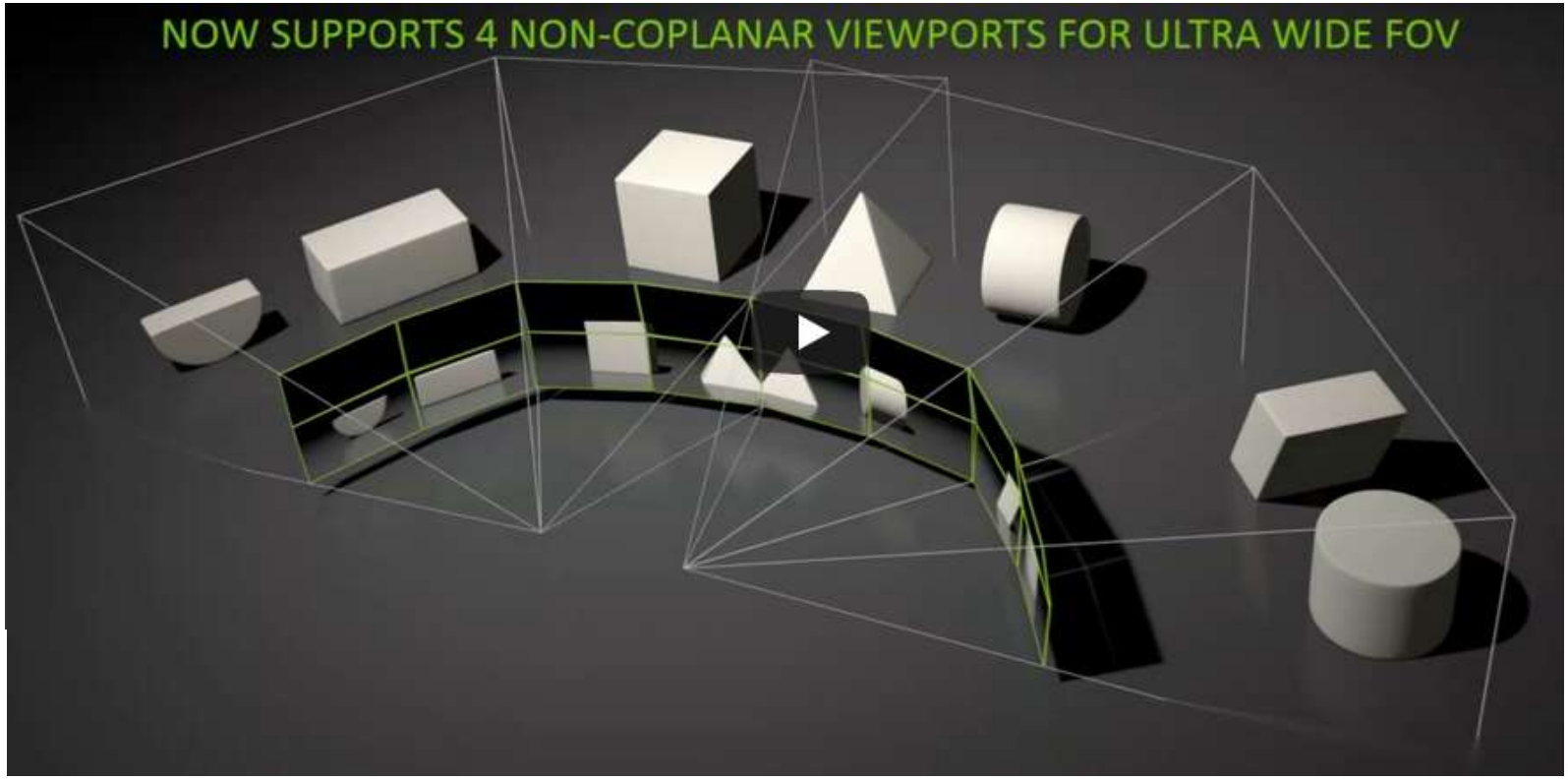
- Output into a specific texture: Layered Rendering



- The operation redirects geometry to only ONE layer of a layered buffer (contrary to using MRT, where it appears in all buffers)

Multi-Output Rendering

- Simultaneous Multi-Projection (SMP)



Later today:

- We will use such iterative rendering solutions to address shadows

https://www.pinterest.com/pin/482307441323449946/

Advanced Texture Use-Cases

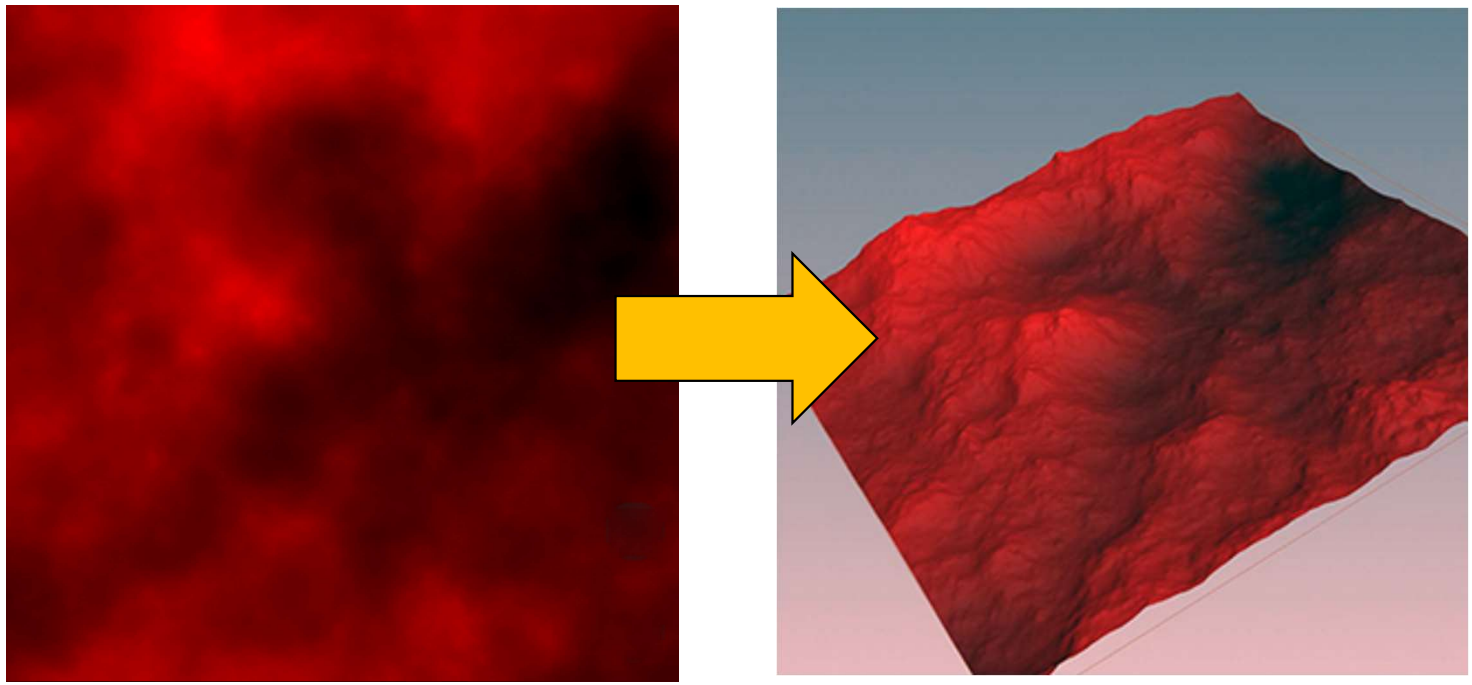
- Encode material properties
- Approximate the environment
- **Represent Geometry**
- Encode complex functions

Terrain definition

- Source: DocMojo

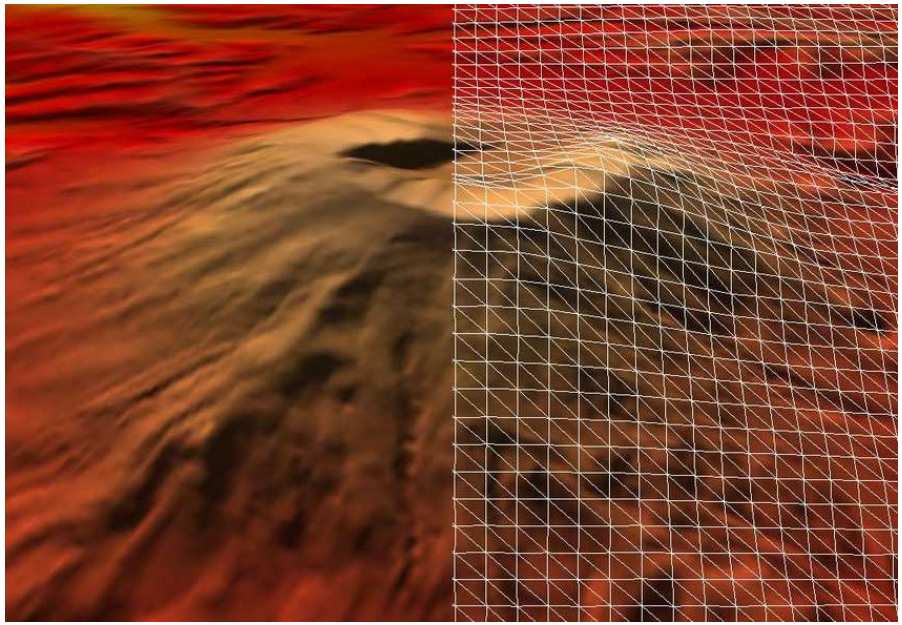
Terrain definition

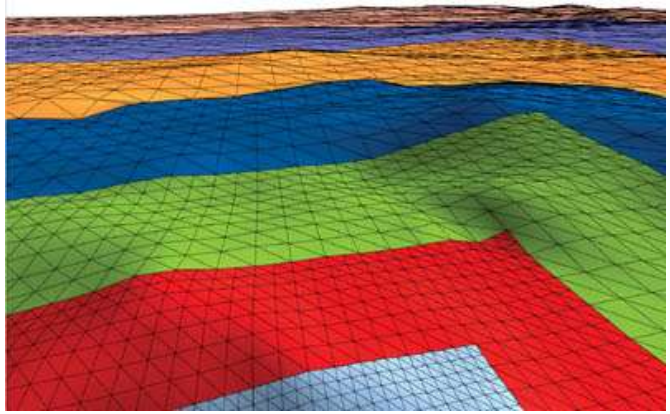
- Source: https://bubblepins.com/blog/output-terrain-texture-maps-houdini-height-fields



Height Map Representations

- Use triangle grid and move vertices according to texture data





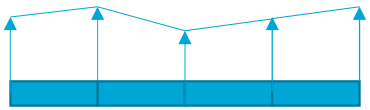
Possible to make grid subdivision adaptive, based on distance to the observer
e.g., Clipmaps [GPU Gems 2]

Alternatively, move a pre-subdivided grid along with the observer that has higher resolution nearby

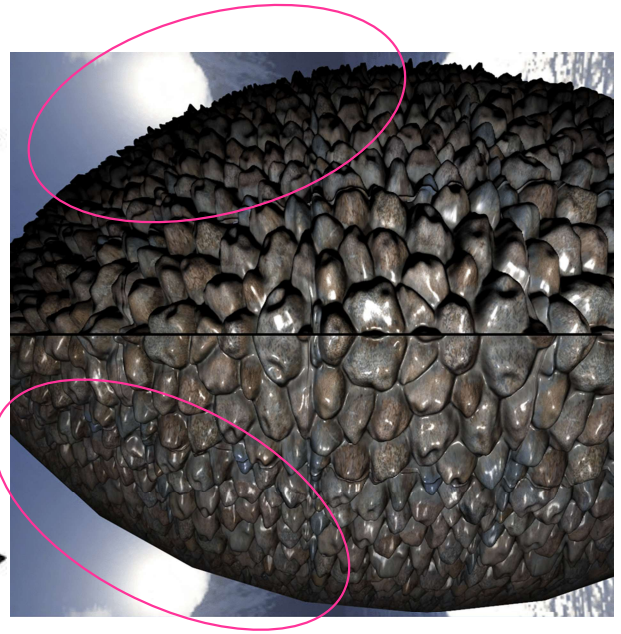
Displacement Mapping

- Store not only normals, but also an offset

Displacement mapping

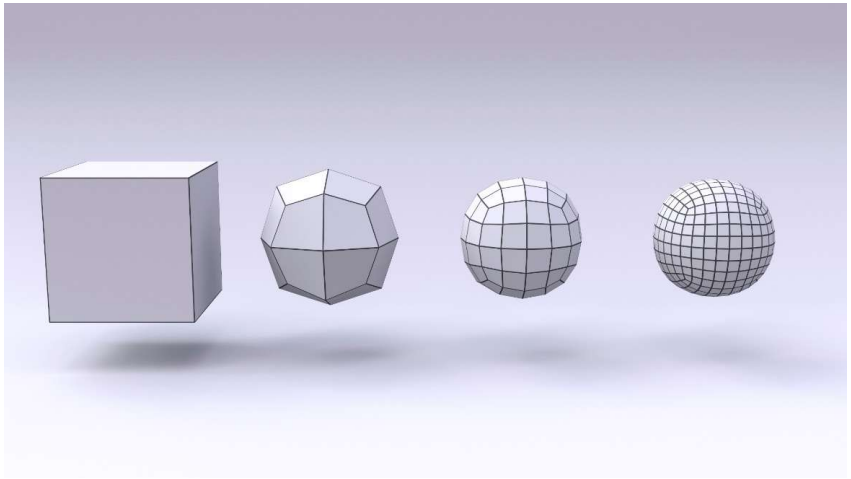


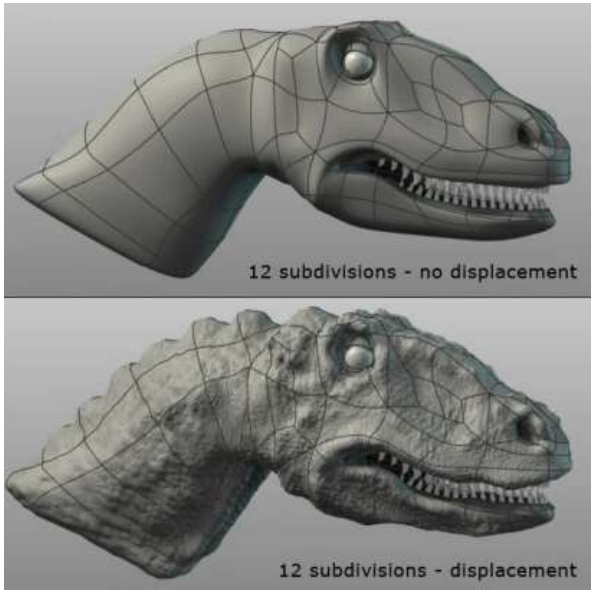
Normal mapping



Displacement Mapping

- Store not only normals, but also an offset
- Often used together with subdivision surfaces

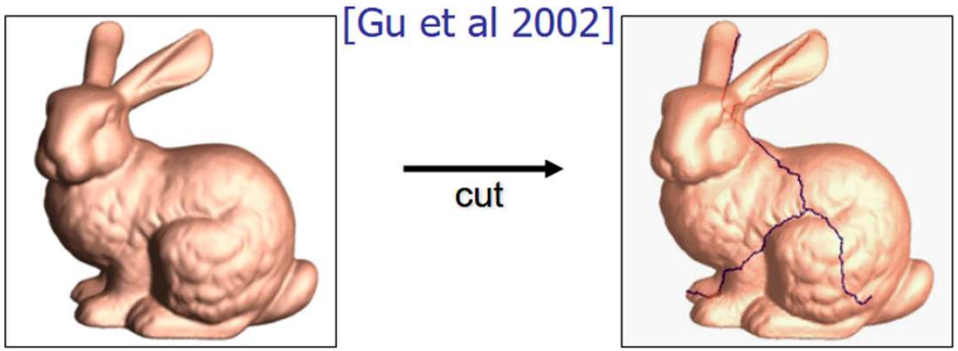




- Leads to a large amount of geometry - mostly used in offline rendering

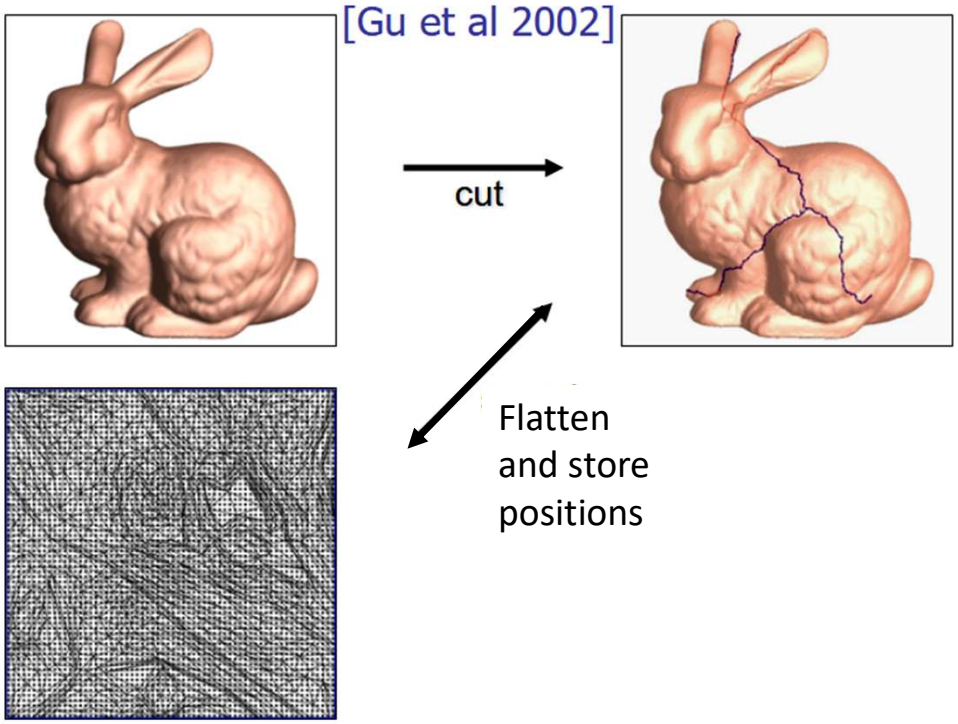
Geometry Images

- Textures can encode a general object



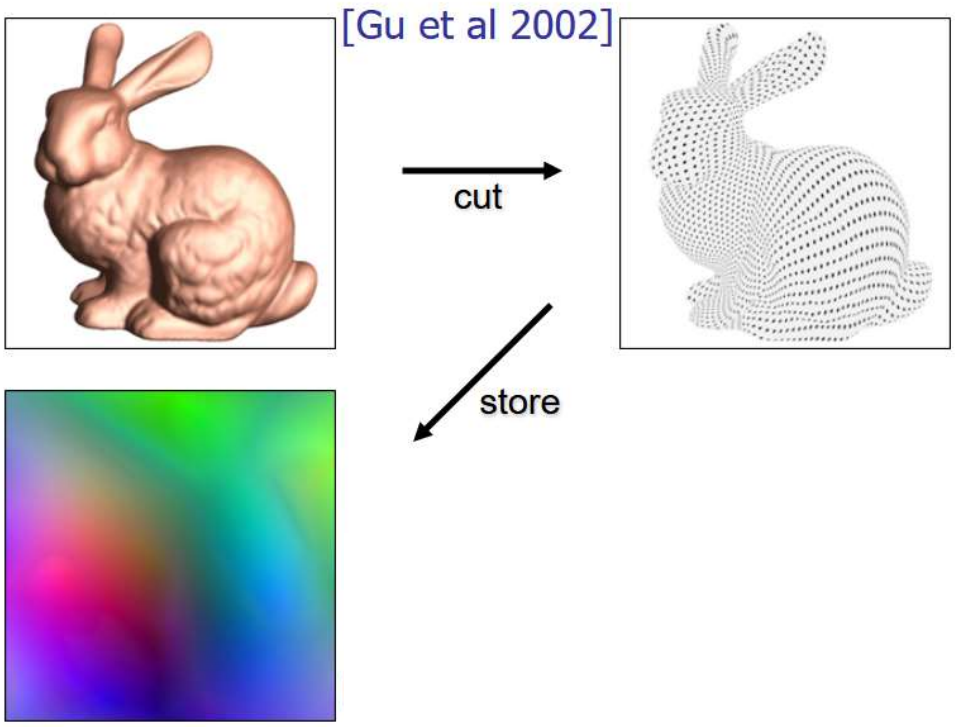
Geometry Images

- Textures can encode a general object



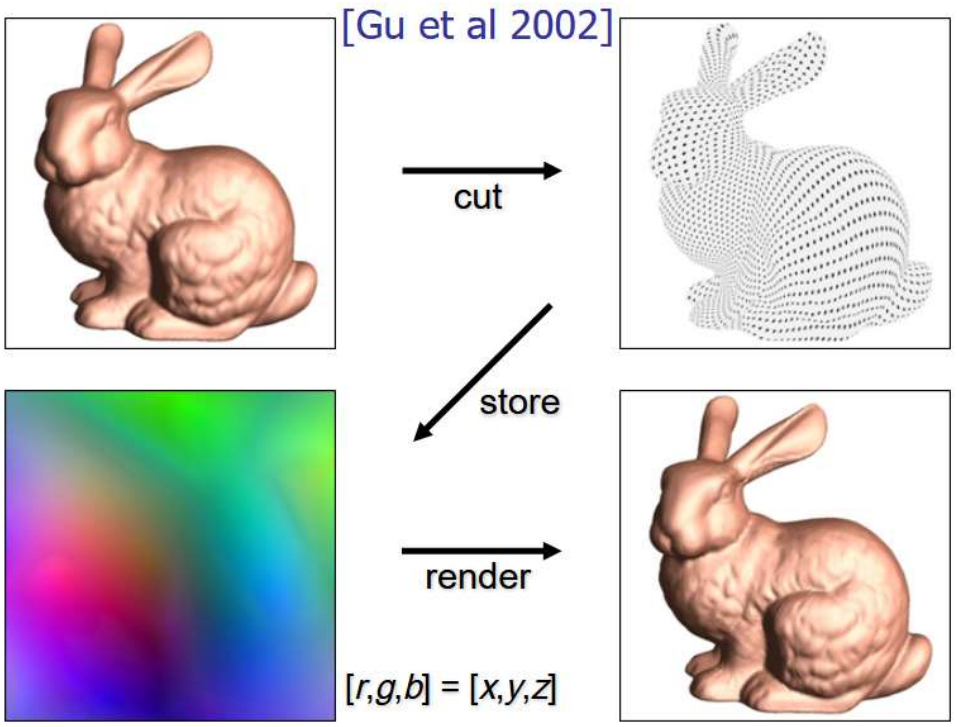
Geometry Images

- Textures can encode a general object



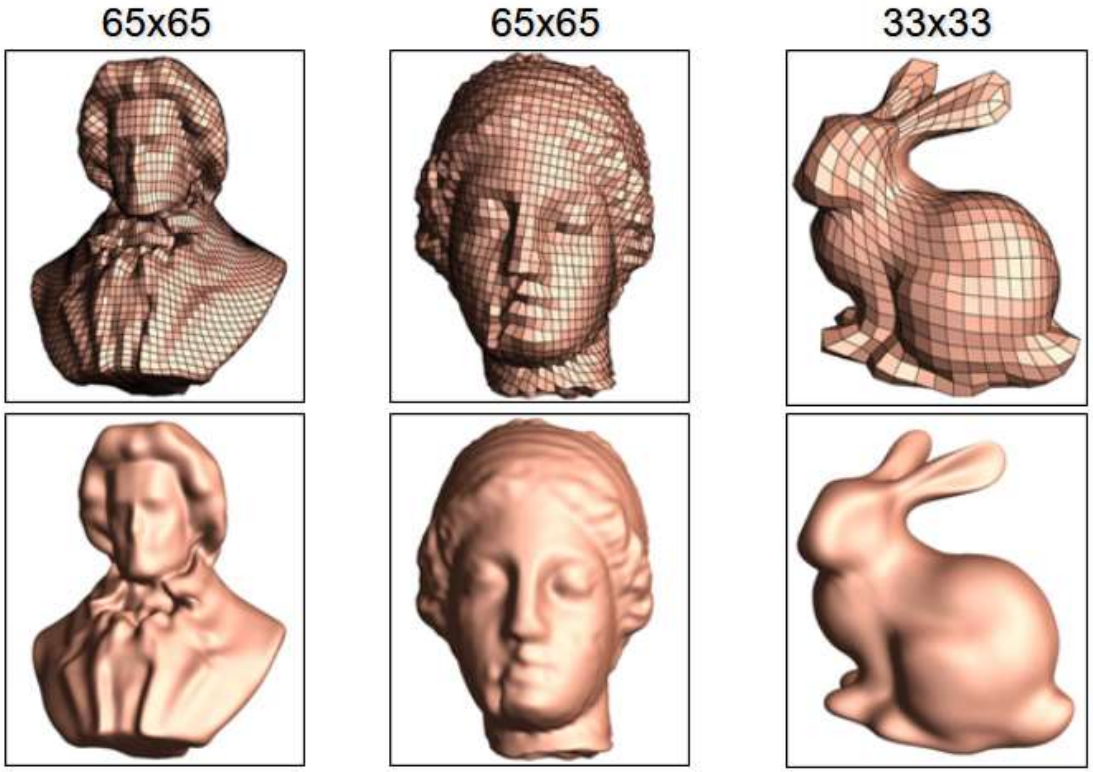
Geometry Images

- Textures can encode a general object



Geometry Images

- Low-Resolution Examples

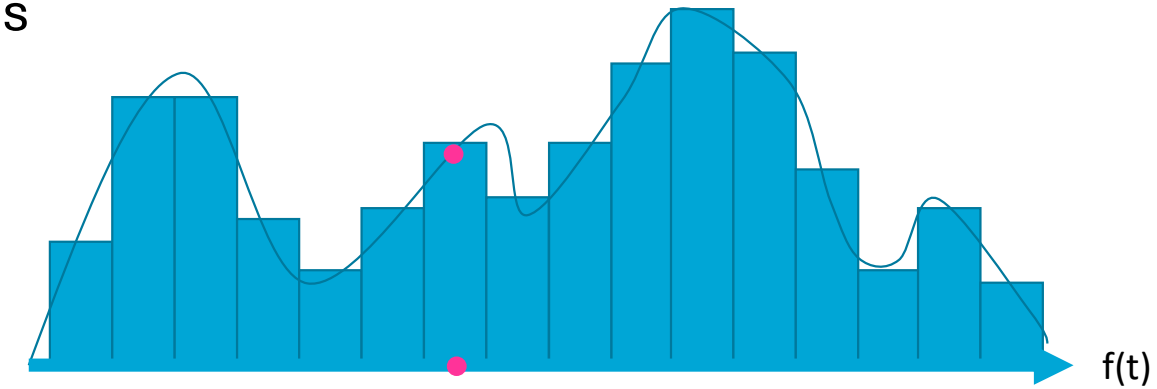


Advanced Texture Use-Cases

- Encode material properties
- Approximate the environment
- Represent Geometry
- **Encode complex functions**

Textures as Lookup Table

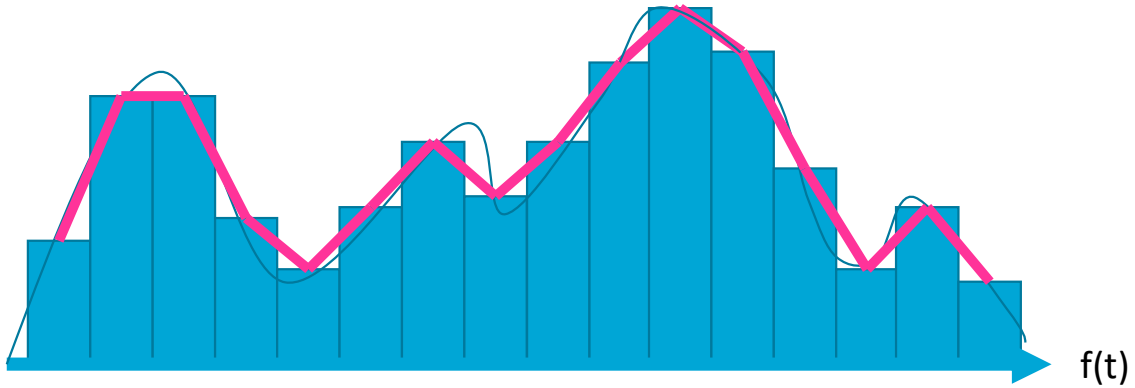
- Approximate complex computations via texture lookups



Replace computation by a lookup in a texture

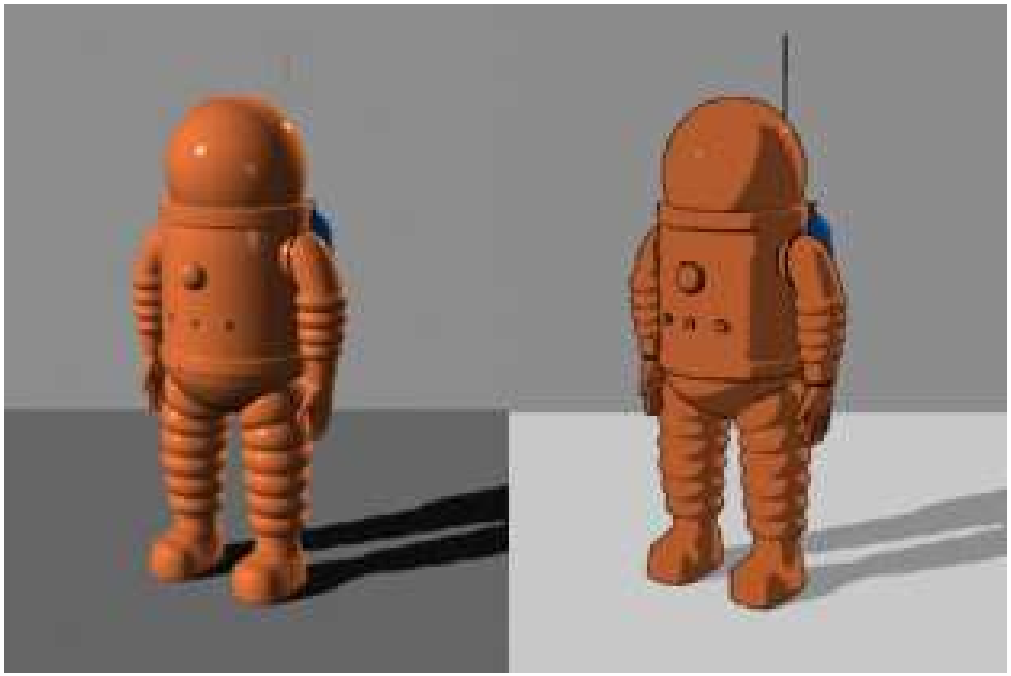
Textures as Lookup Table

- Activating linear texture interpolation leads to a piecewise linear approximation



Simple Example: Lookup Table Use

- Non-photorealistic materials



- Cel-shading: threshold the diffuse shading

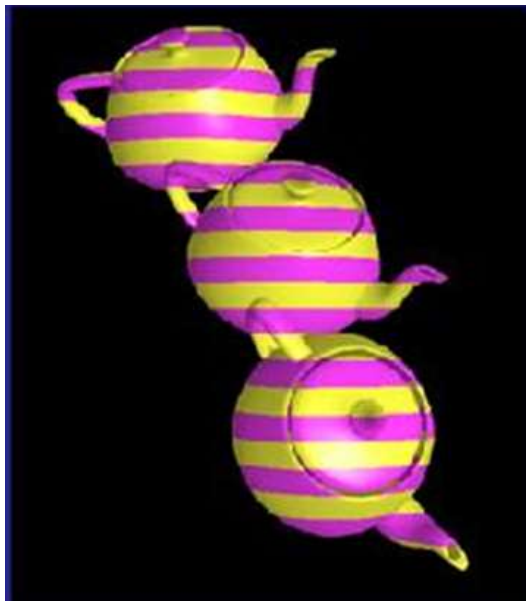
X-Toon (Barla et al. 06) – see Exercises

- Instead of calculating, implement it via a 1D texture lookup

Second parameter can be distance to camera, or curvature, or material coefficient, or…

Playing with Tex Coords: Creating Paper Structure

- Example application:
  Add paper grain over your rendered image



CGArchitect Forums

How would you define the texture mapping T to obtain screen-space texture coordinates?
Hint: Use the projected vertex position!

Advanced Texture Use-Cases

- Encode material properties
- Approximate the environment
- Represent Geometry
- Encode complex functions

Advanced Texture Use-Cases

- Encode material properties
- Approximate the environment
- Represent Geometry
- **Encode complex functions**

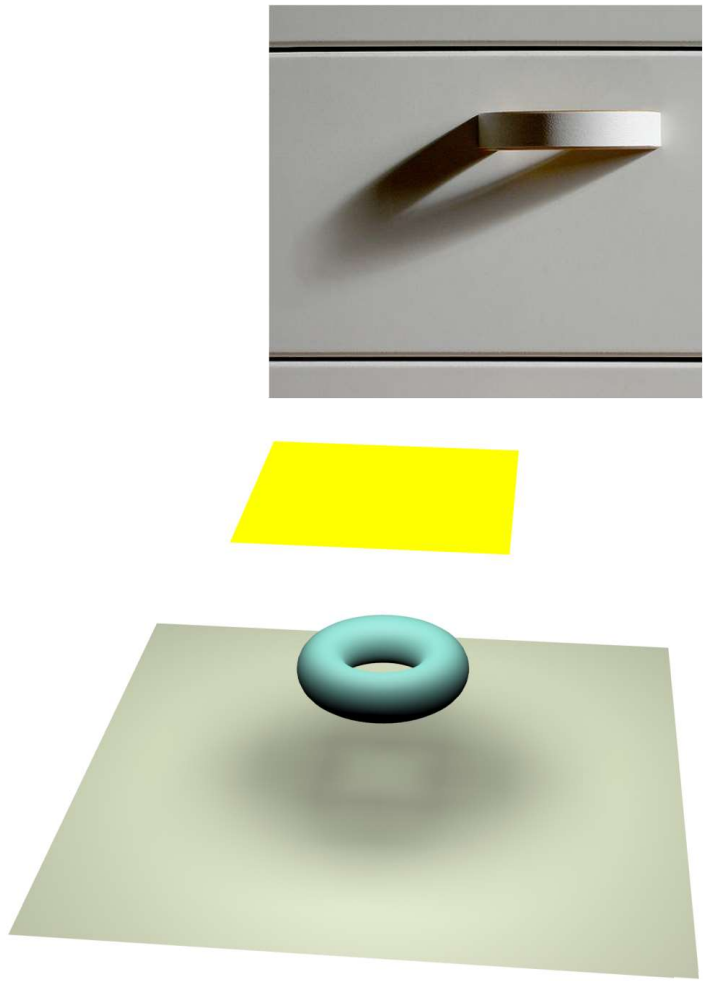
What is a Shadow?

- WordNet:

  Shade within clear boundaries

  or

  An unilluminated area.

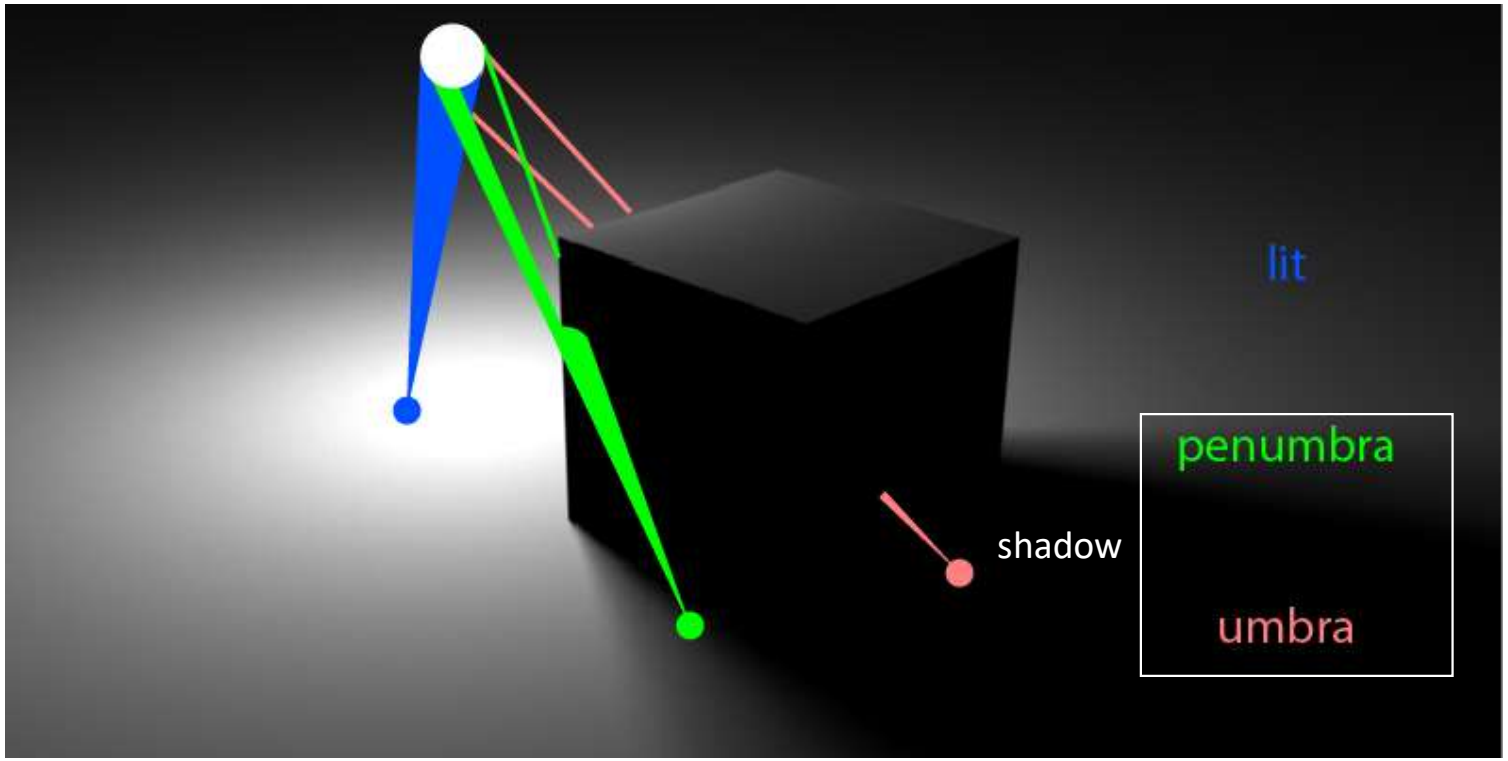


What is a Shadow?

- Hasenfratz et al. [2003]:

Shadow [is] the region of space
for which at least one point of the light source is occluded.

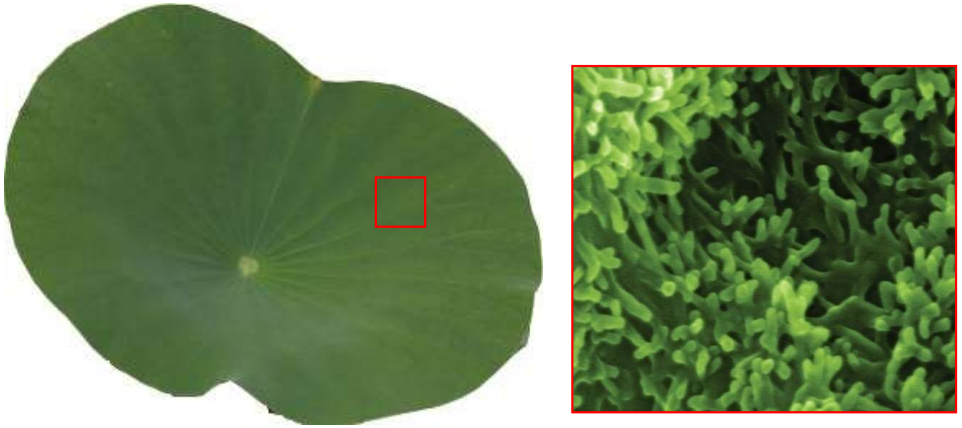
What is a Shadow?



What is a Shadow?

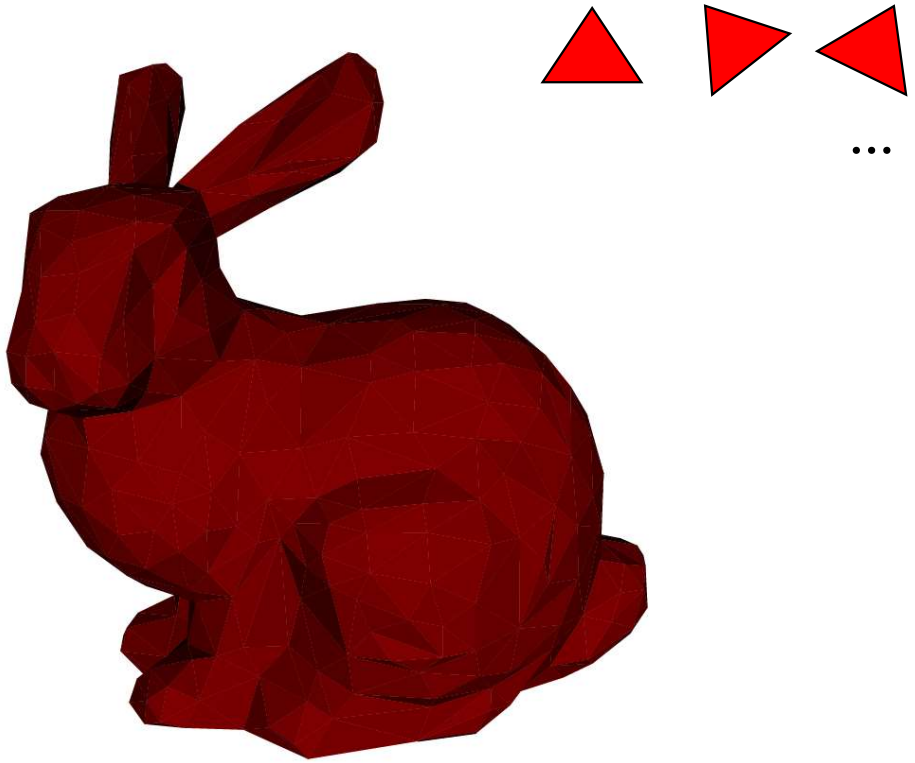
- Hasenfratz et al. [2003]:

Shadow [is] the region of space
for which at least one point of the light source is occluded.



Our models are simple…

- Typically a list of triangles



What is a Shadow?

- Hasenfratz et al. [2003]:

Shadow [is] the region of space
for which at least one point of the light source is occluded.

How to draw shadows?

- Artists know how to draw shadows!

  Or not?

Fra Carnevale
(1467)

How to draw shadows?

- Artists know how to draw shadows!

  Or not?

Signorelli (1488)

How to draw shadows?

- Artists know how to draw shadows!

  Or not?

Moore (1893)

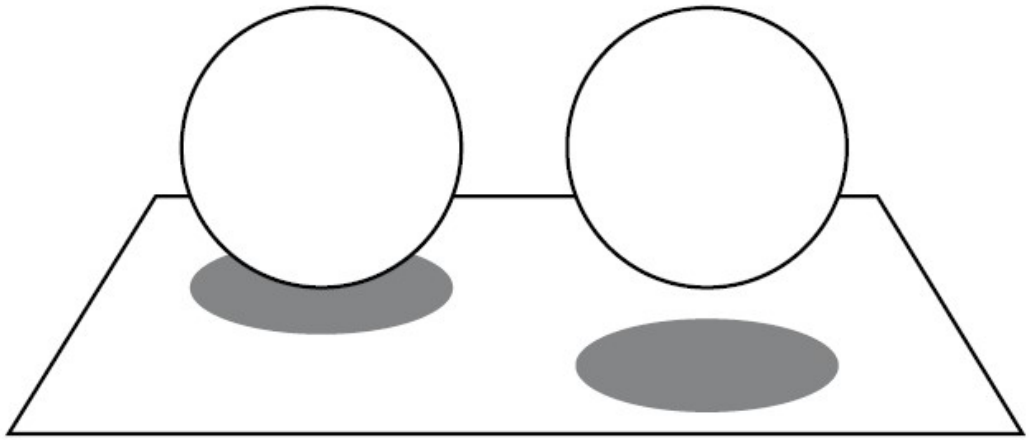
How to draw shadows?

- Drawing shadows is apparently difficult…

So why not just ignore shadows?

- Shadow of the Colossus, Sony

So why not just ignore shadows?

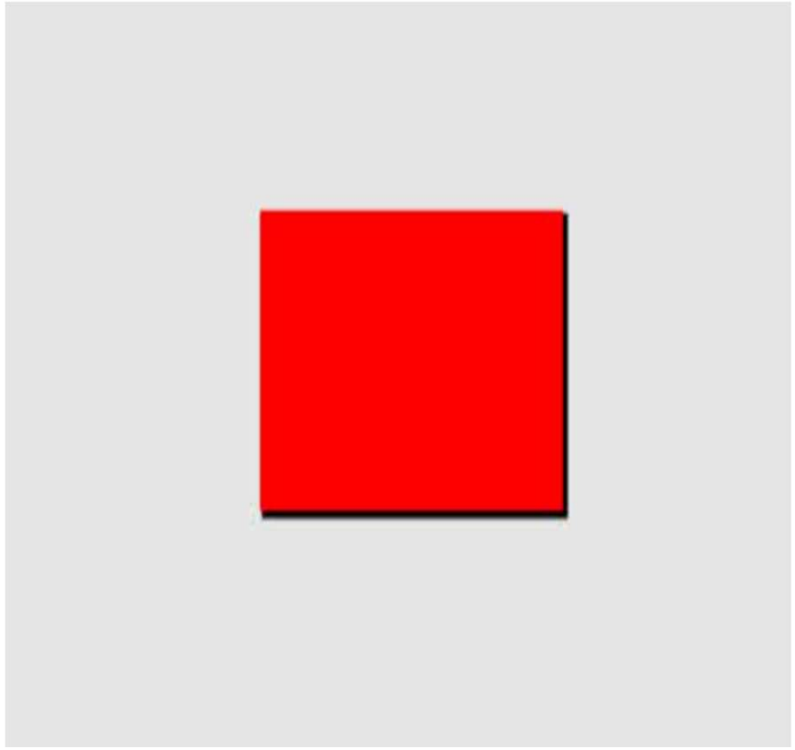


So why not just ignore shadows?

- Shadow of the Colossus, Sony

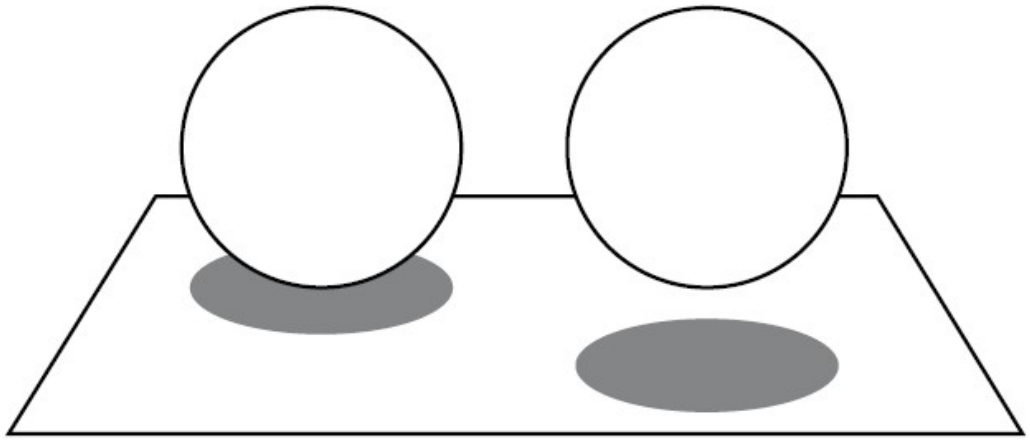
Psychophysical-Experiments

[Kersten et al. 96]

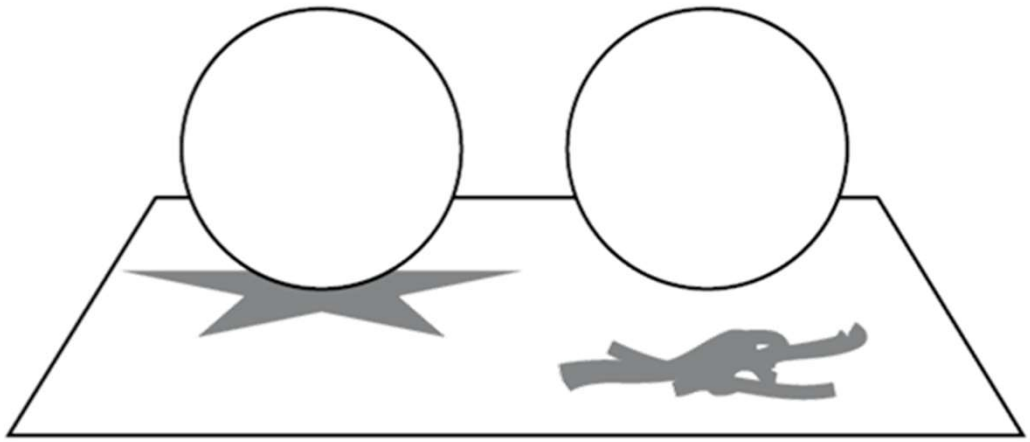


So why not just ignore shadows?

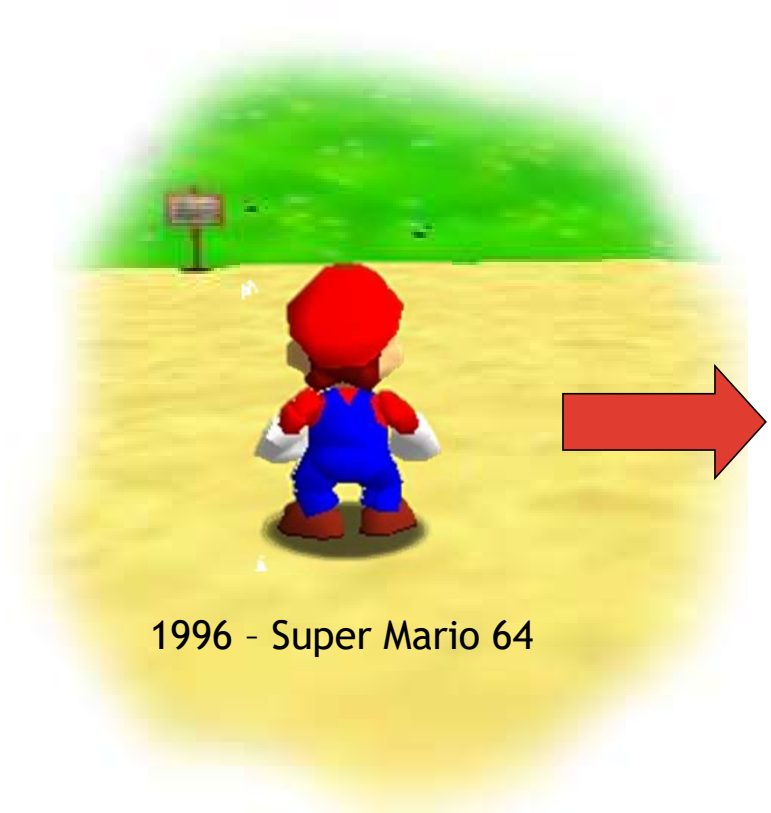
- But this is not a good argument for realistic shadows…



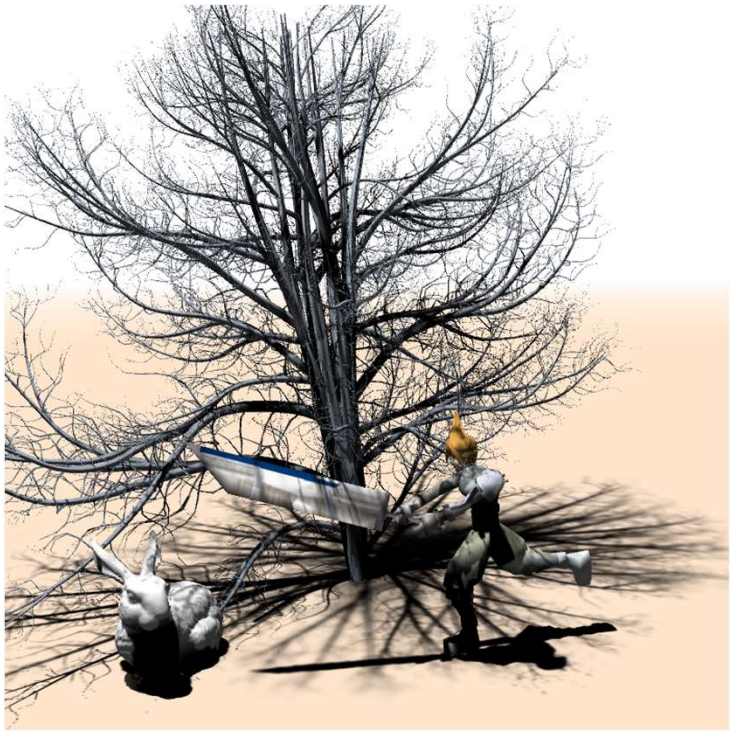
So why not just ignore shadows?



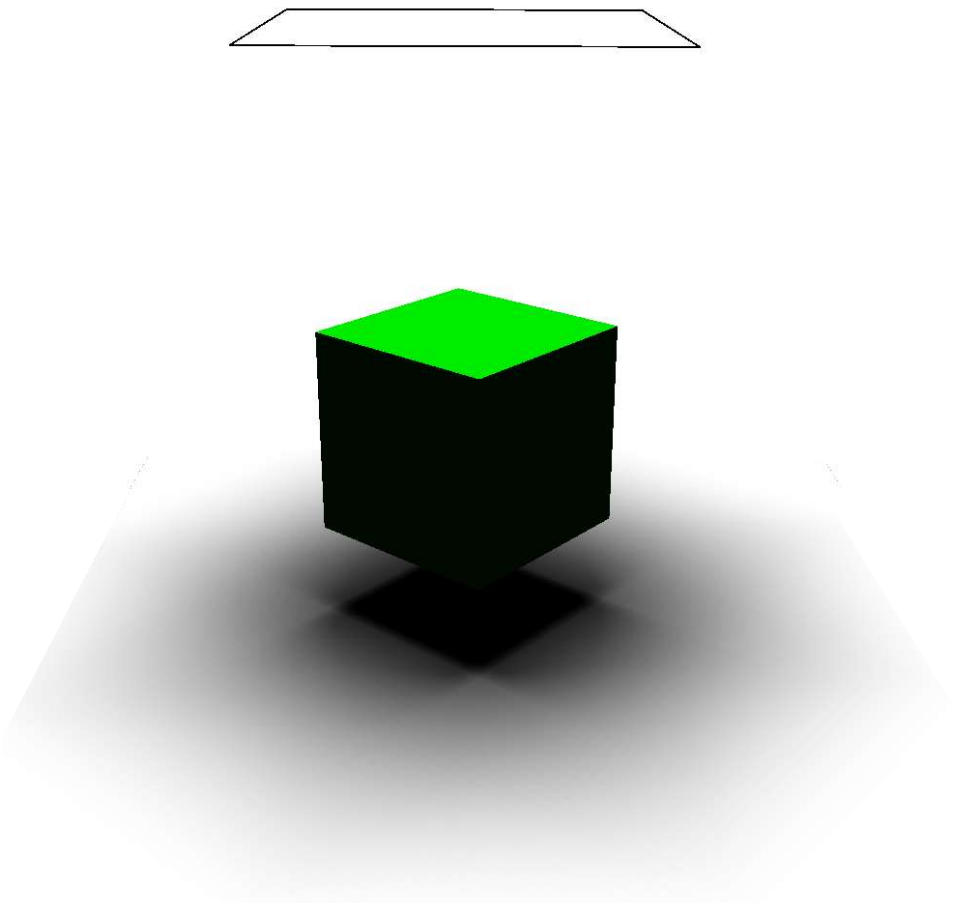
Simple shadows can be sufficient



1996 – Super Mario 64

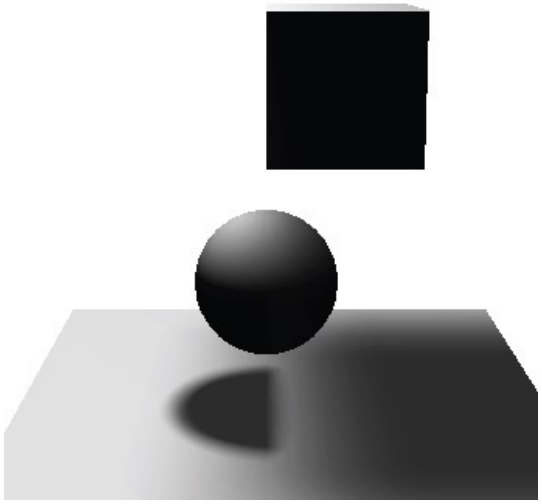


Plausible shadows



Plausible shadows

- Attention: “plausible” can often fail

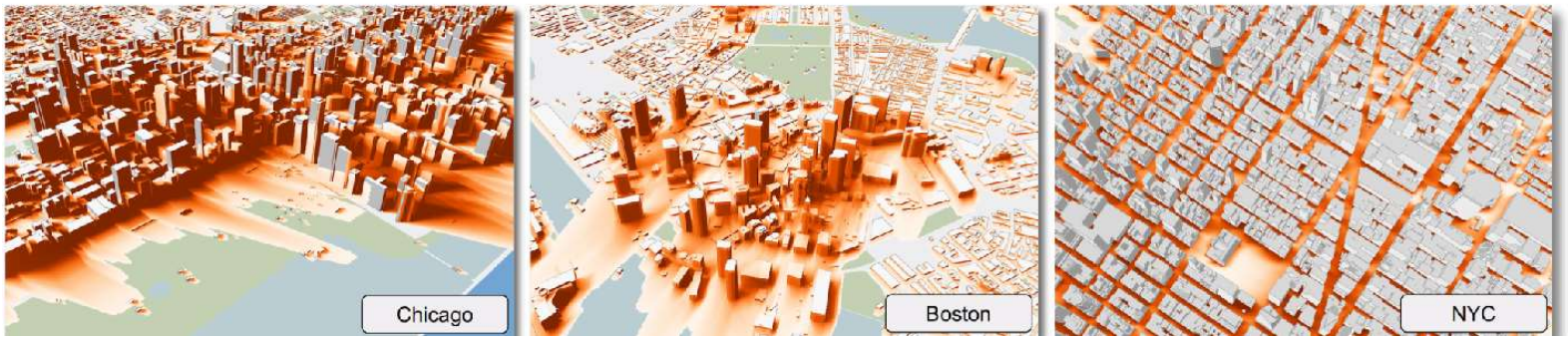


Realistic shadows are important

Realistic shadows are important

- Many contexts in which accuracy is needed:
  - Architecture
  - Simulation
  - Movies

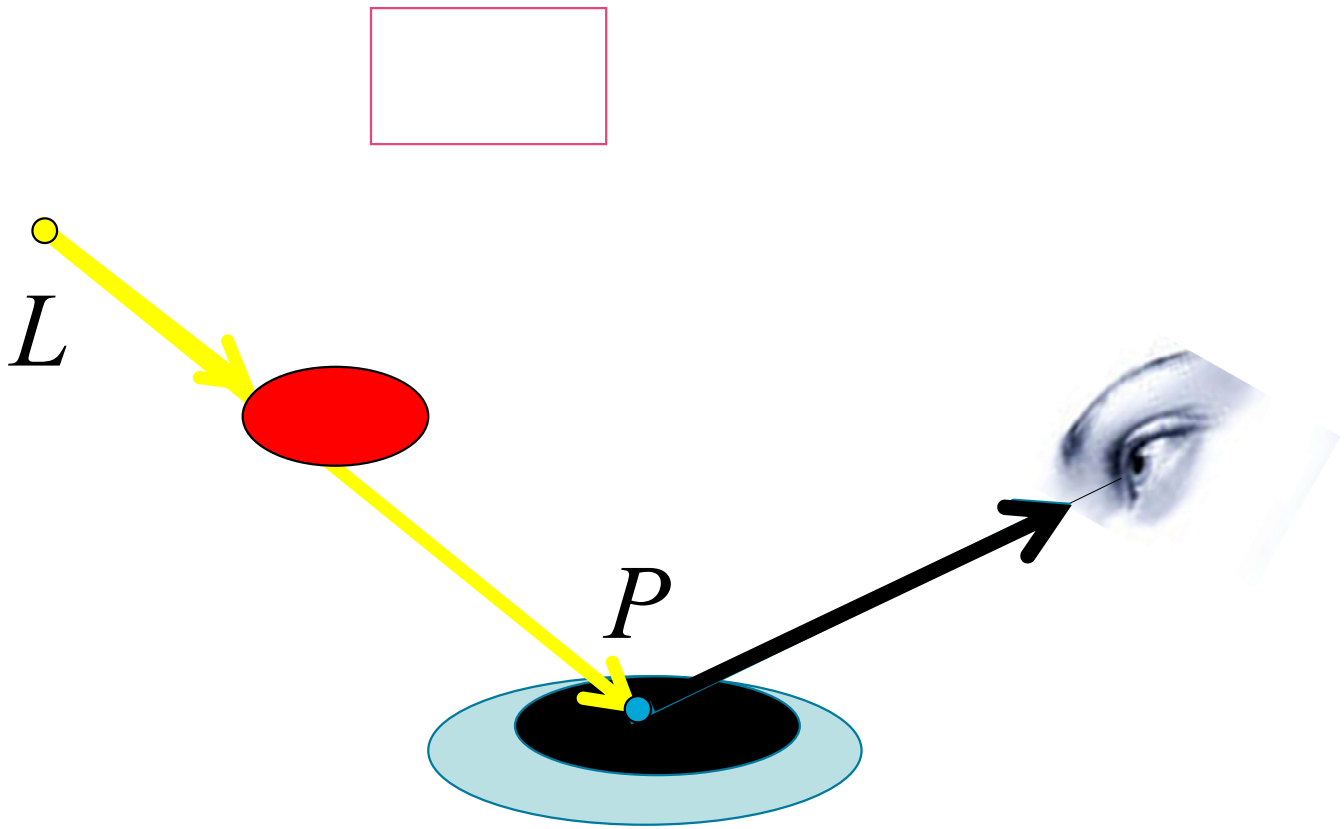
  …



Miranda et al. TVCG2019

Desert villa by Studio Aiko

How to compute shadows?



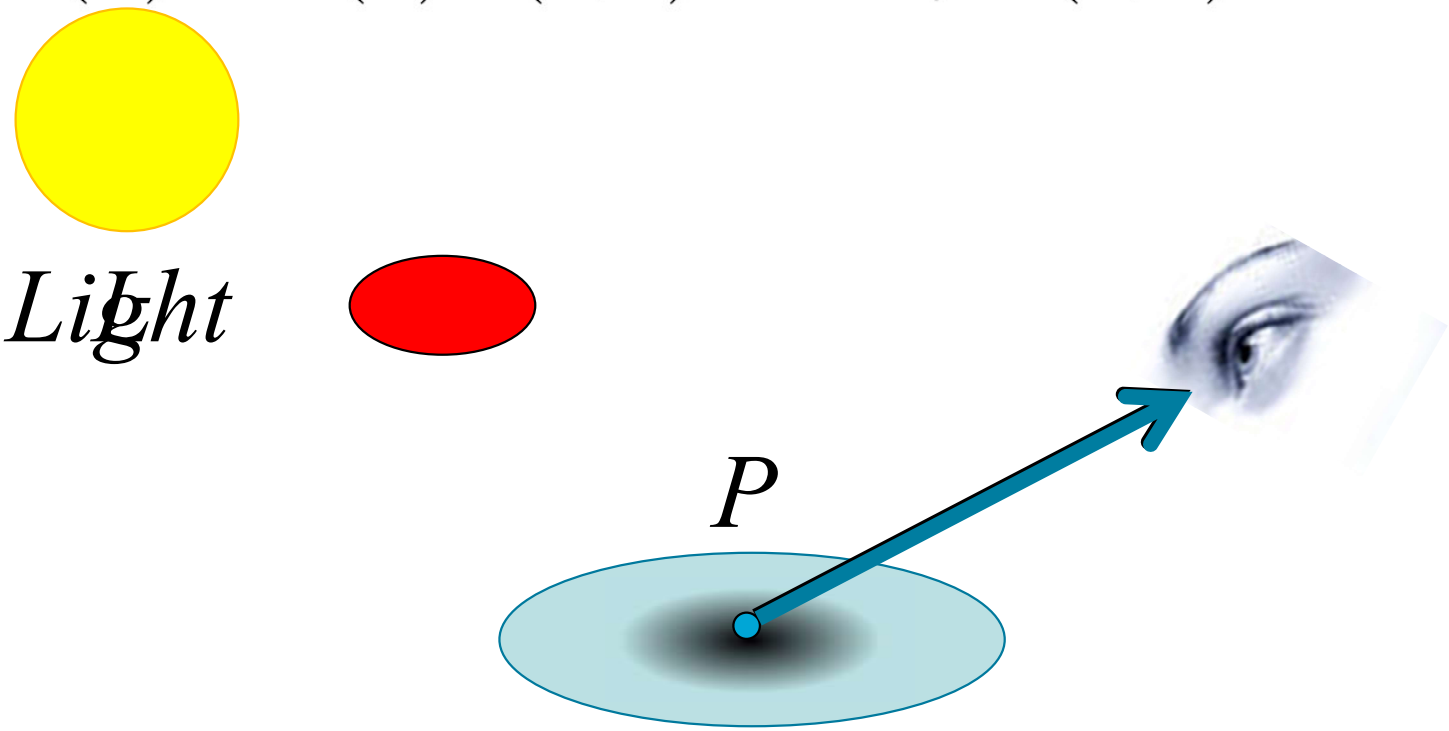
Hard Shadows

Point lights do not create penumbrae

Soft Shadows

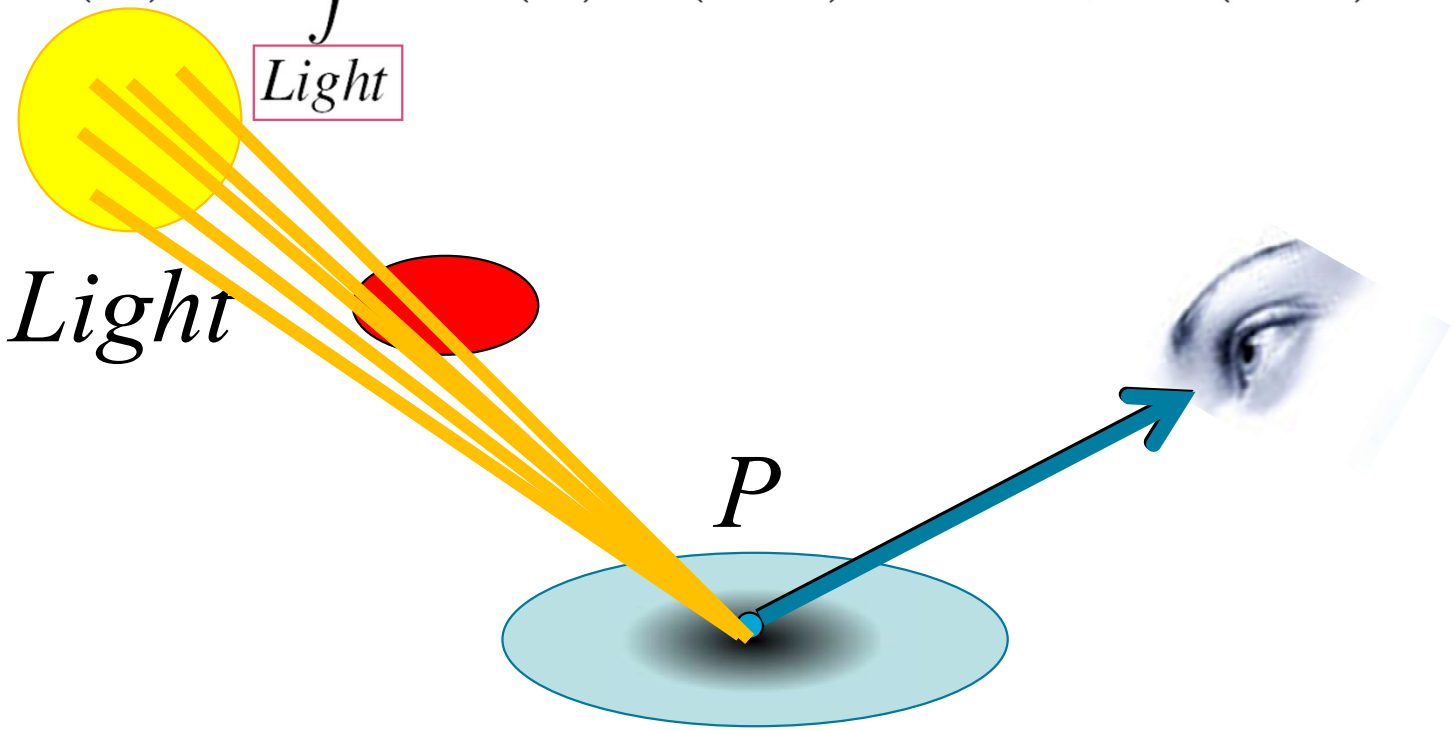
What is a Shadow ? – Part II

$$B(P) = E(L)\ v(P,L)\ Transfer(P,L)$$



What is a Shadow ? – Part II

$$B(P) = \int_{Light} E(L)\ v(P,L)\ Transfer(P,L)\ dL$$



Bad news

- Prior to 2008:

  All real-time shadow algorithms

  attempted to only approximate :

$$\int_{Light} v(P,L)\ dL$$

and all failed…

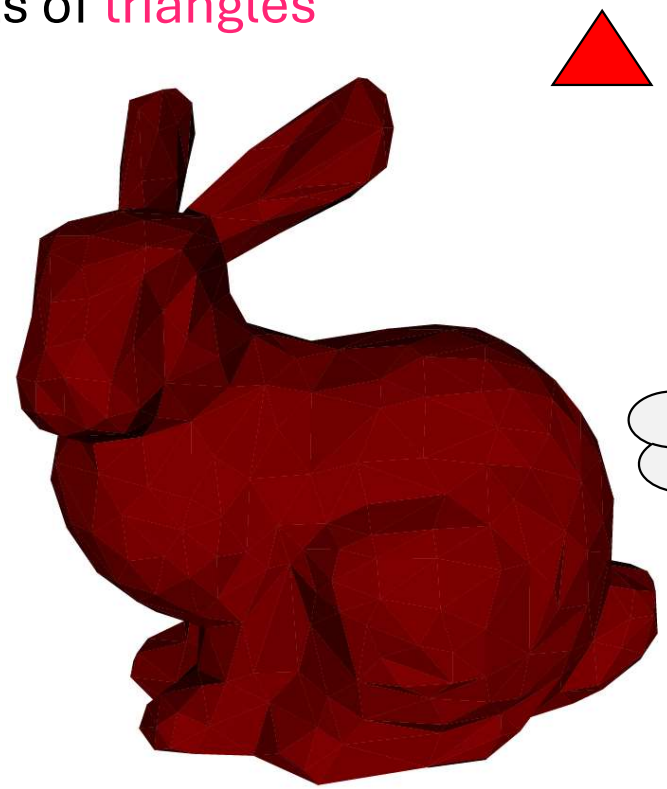
Today: Hard Shadows within the Rendering Pipeline

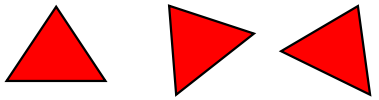
- Meaning: light source L is a point

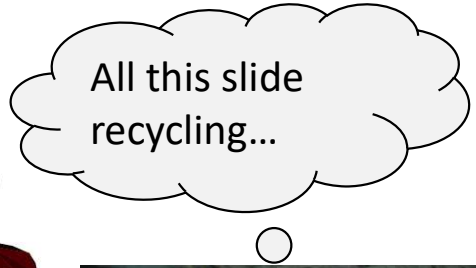
How to accelerate the process?

Simplified Graphics Pipeline

- Models are typically lists of triangles

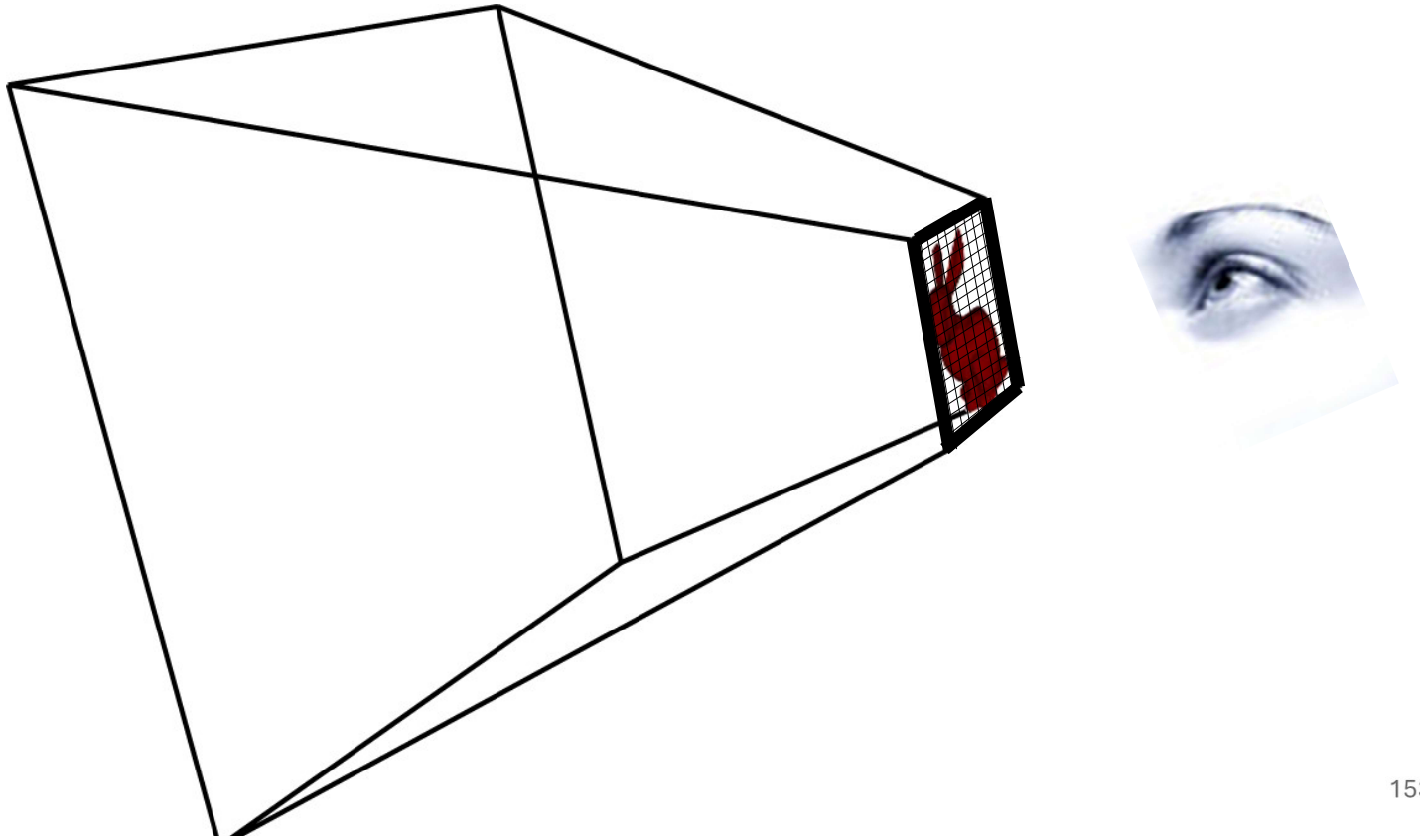






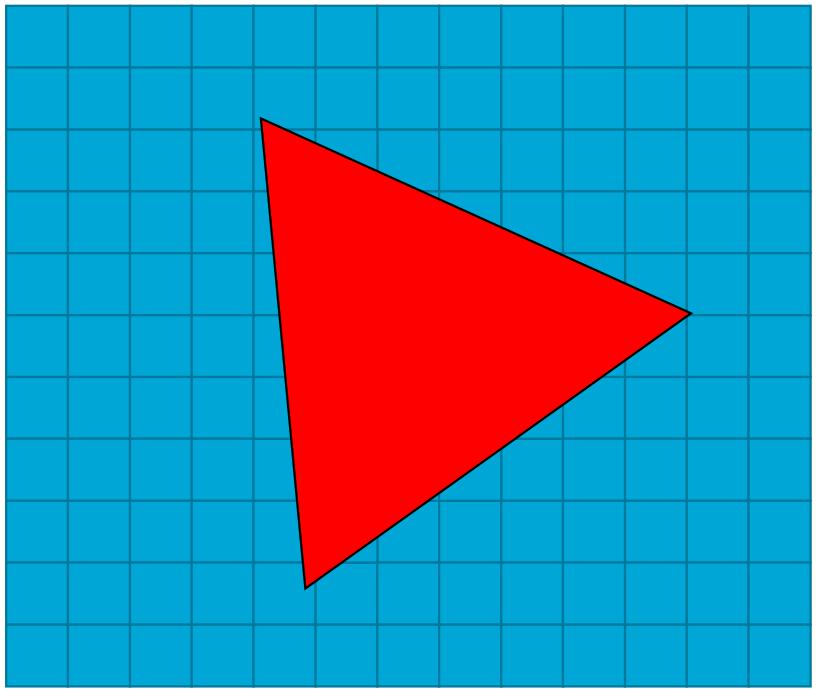
Simplified Graphics Pipeline

- Projection: Transform coordinates to screen



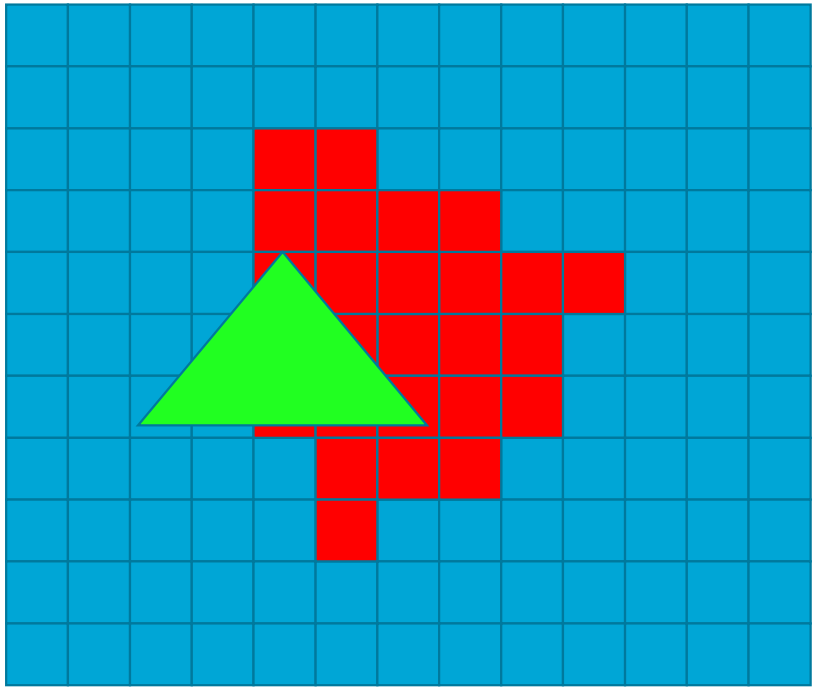
Simplified Graphics Pipeline

- Rasterization: Fill screen pixels



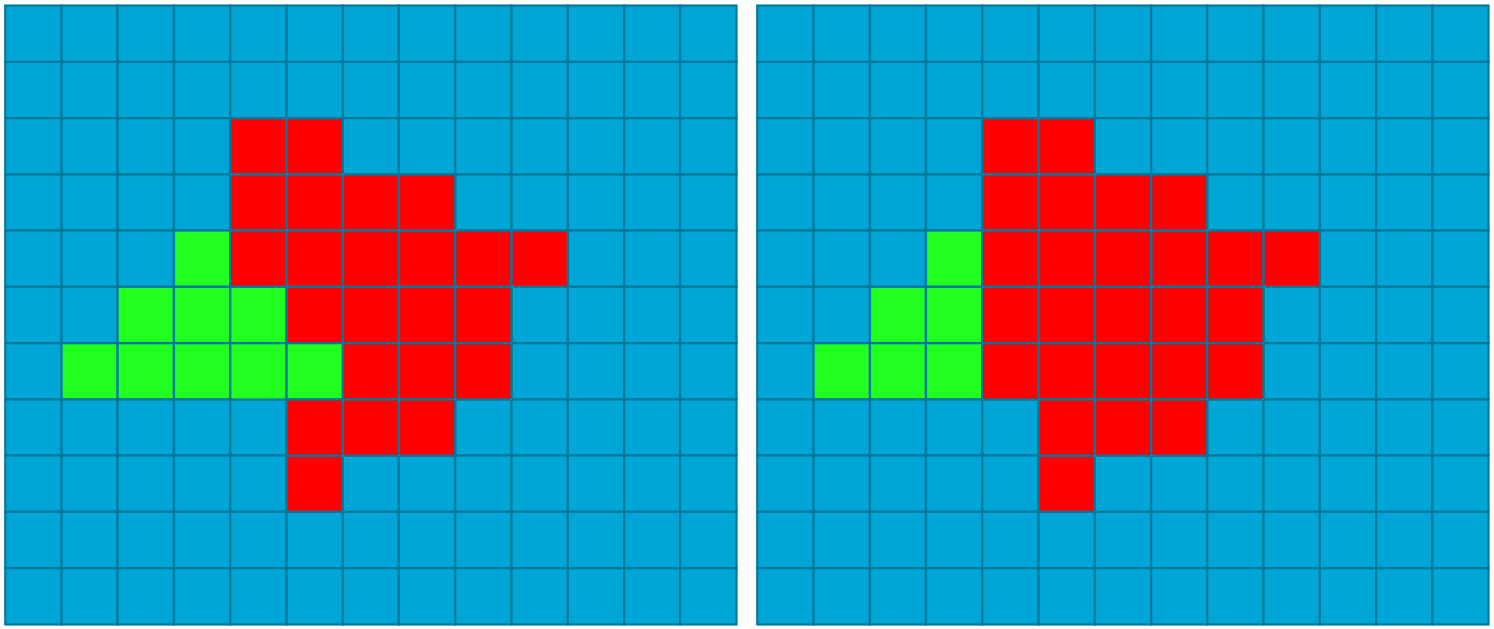
Simplified Graphics Pipeline

- Catch: Let's look at a second triangle…



Simplified Graphics Pipeline

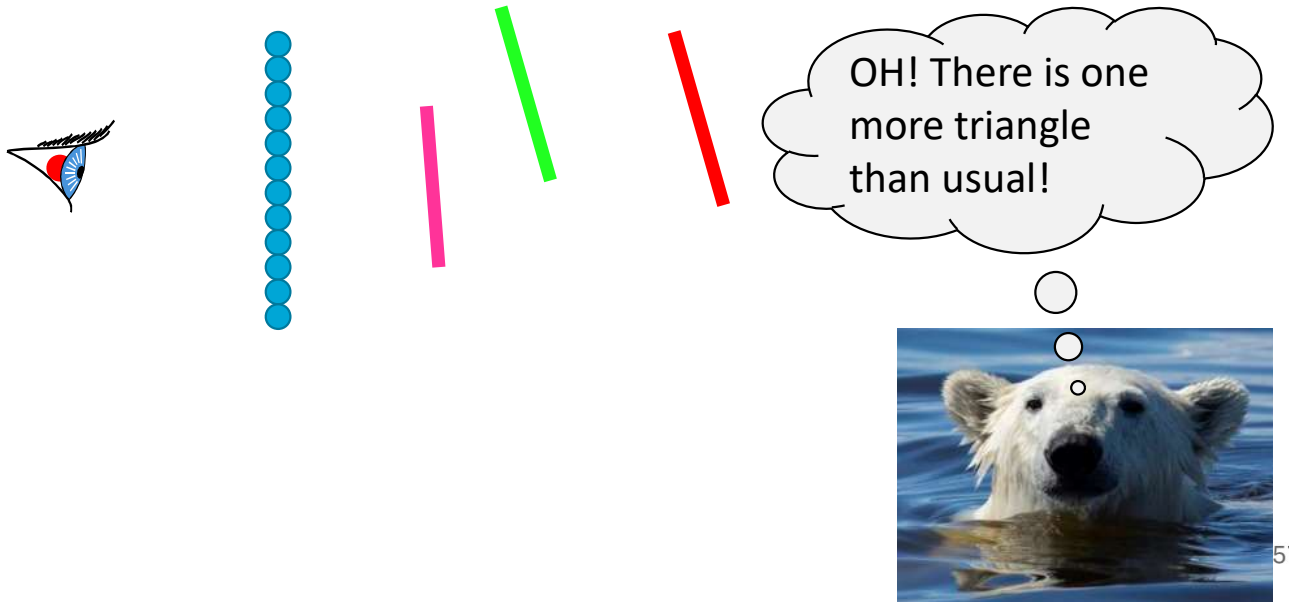
- Catch: Drawing order changes result



Need to keep nearest pixels

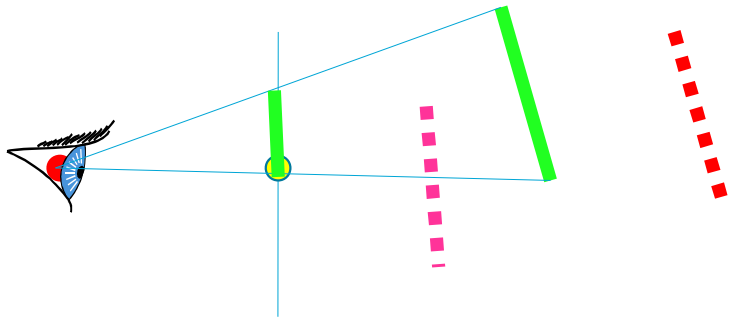
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



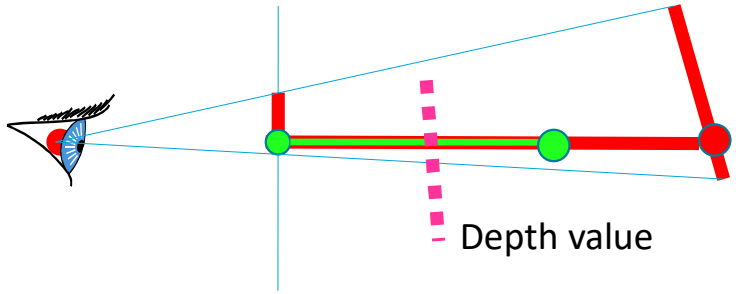
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



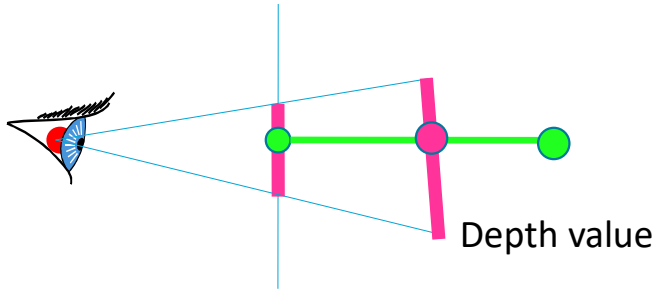
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



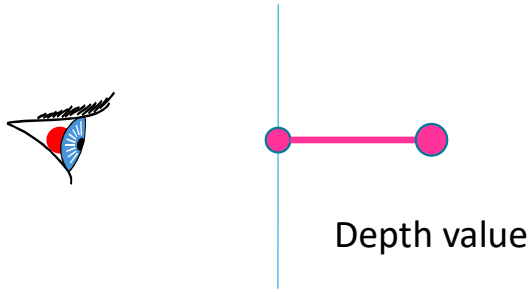
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



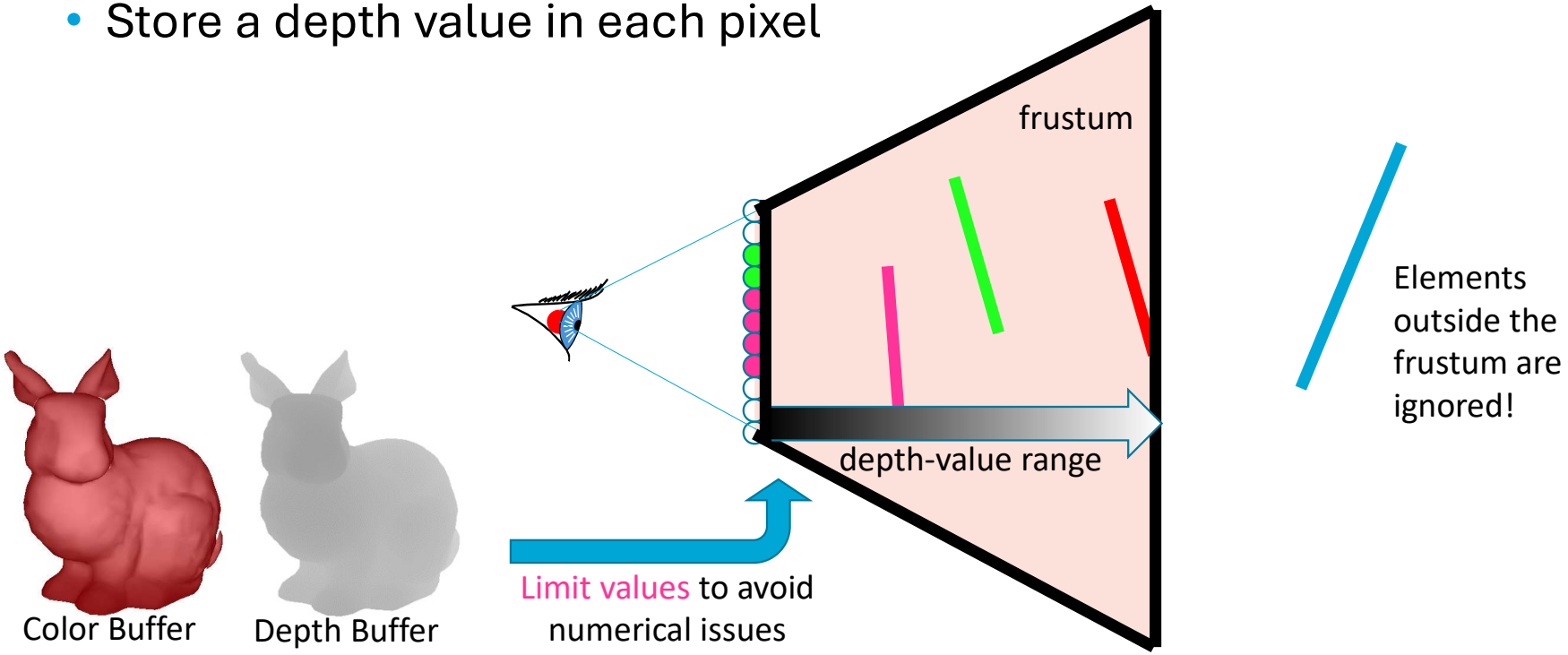
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a color and depth in each pixel



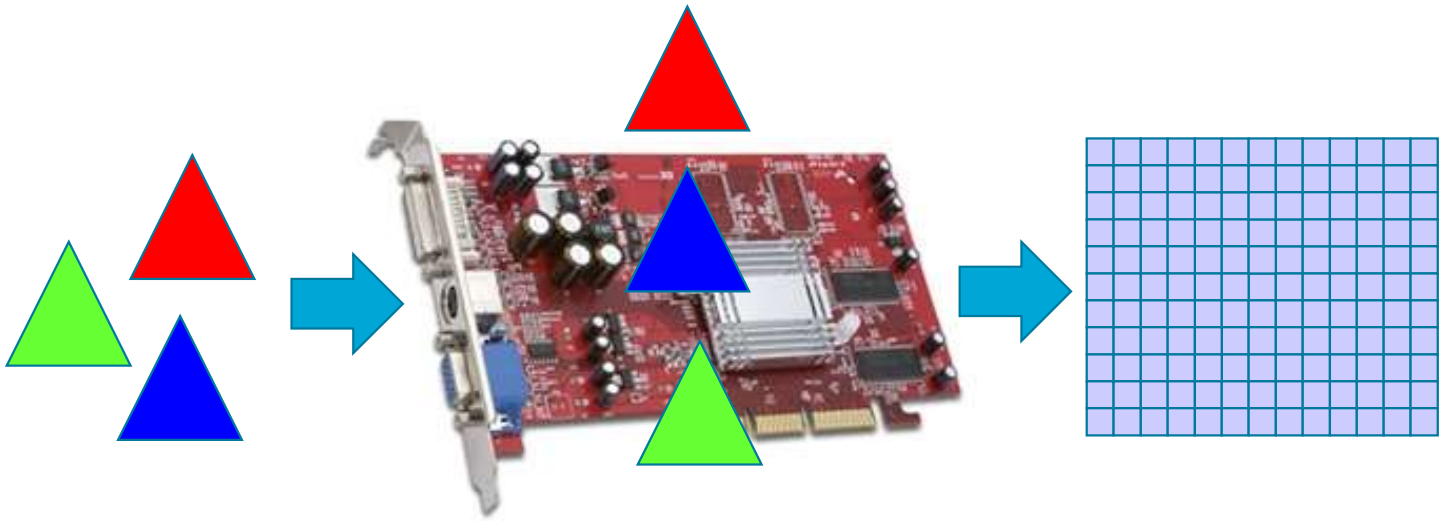
Simplified Graphics Pipeline [Catmull74] , [Strasser74]

- Depth Buffer: Avoid sorting triangles!
- Store a depth value in each pixel



Simplified Graphics Pipeline

- Highly parallelizable
  - ➡ Graphics Processing Units (GPUs)

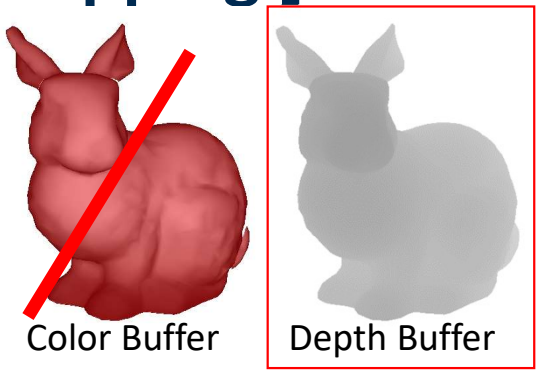


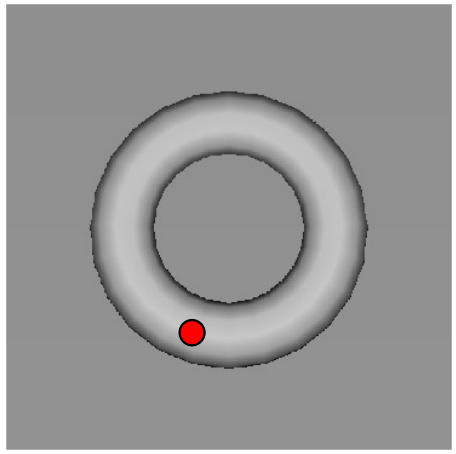
Simplified Graphics Pipeline

- Local computations:
- Processor only knows its current triangle

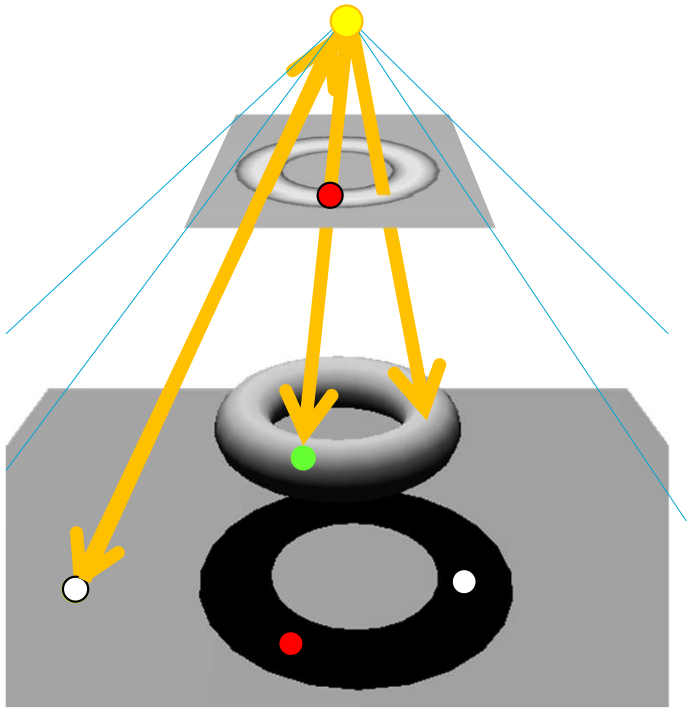
  this is NOT enough for shadows

Shadow Mapping [Williams78]





1. render view from light
use depth buffer

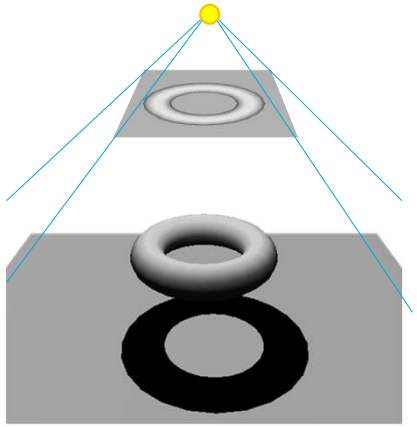


2. render from viewpoint

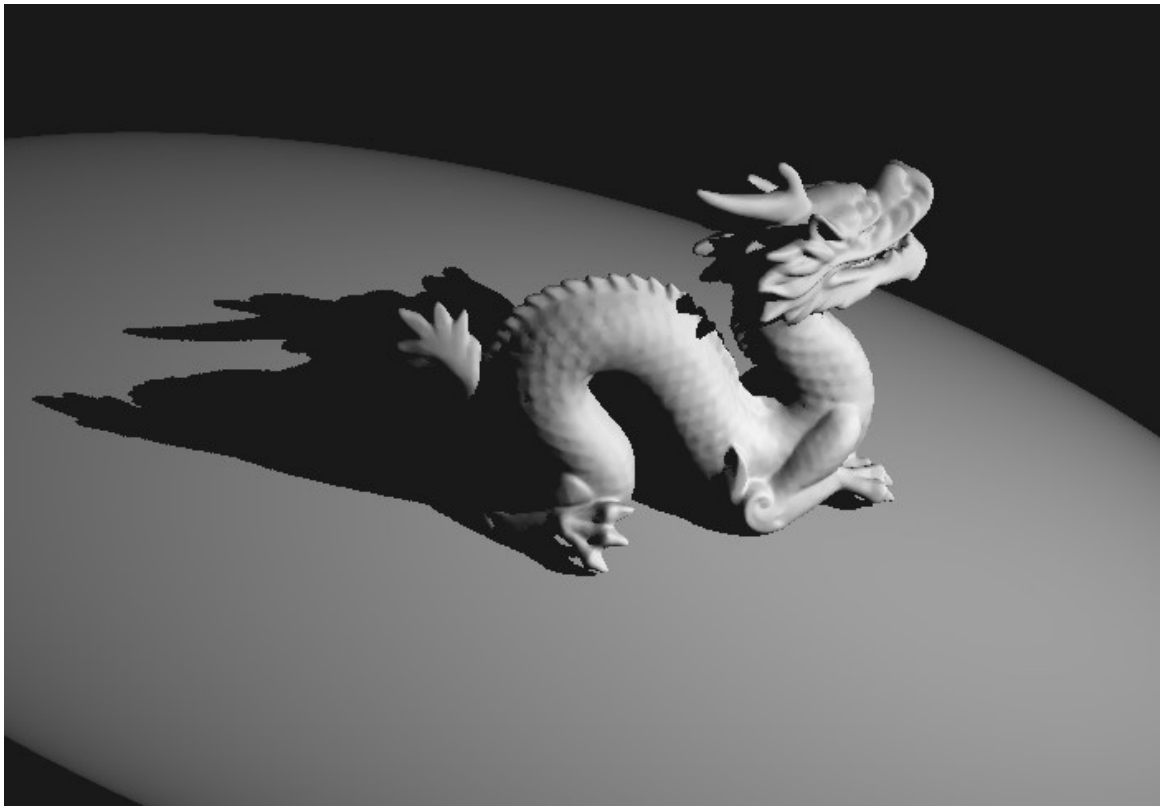
Shadow Mapping [Williams78]

1. Render depth buffer from light (**Shadow Map**)
   - Store result in a texture

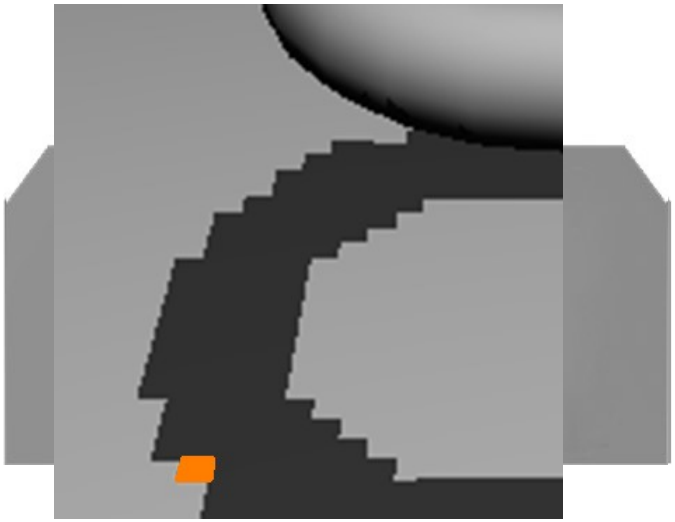
2. Render from viewpoint
   - For each drawn pixel:
     Compare its depth in the light's view
     to the stored depth at this texel location
     in the Shadow Map
     - Equal: pixel is lit
     - Farther: pixel is in shadow



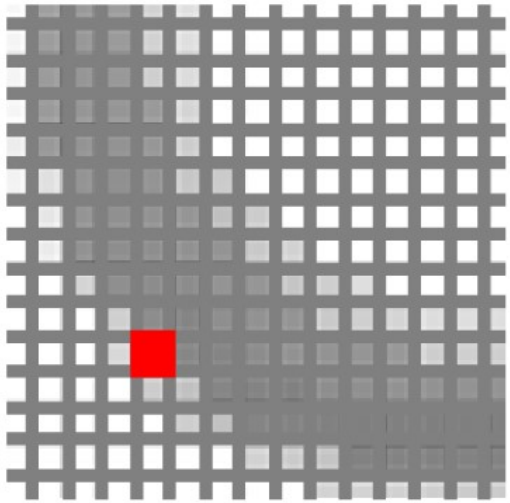
Demo



Problem 1: Discretization



from viewpoint

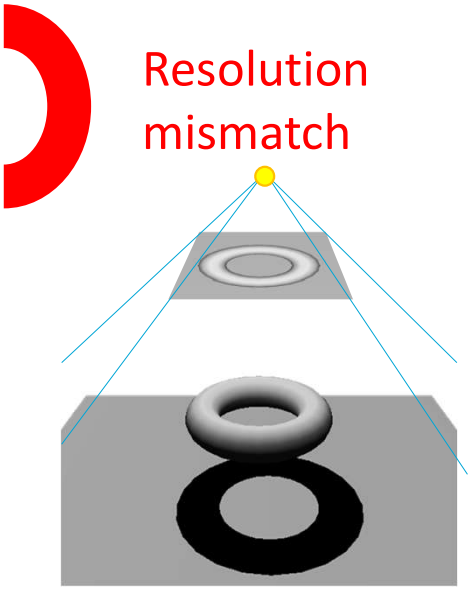


from light

Shadow Mapping [Williams78]

1. Render depth buffer from light (**Shadow Map**)
   - Store result in a texture

2. Render from viewpoint
   - For each drawn pixel:
     Compare its depth in the light view
     to the stored depth at the texel location
     in the Shadow Map
     - Equal: pixel is lit
     - Farther: pixel is in shadow

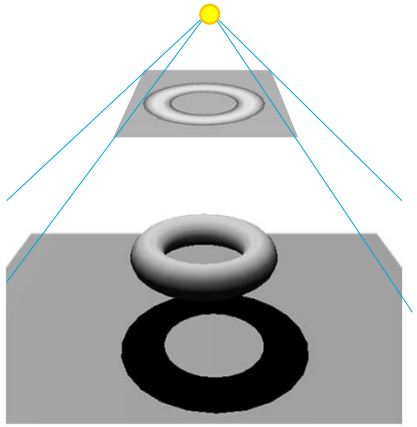


Example: Hellsweeper (2023)

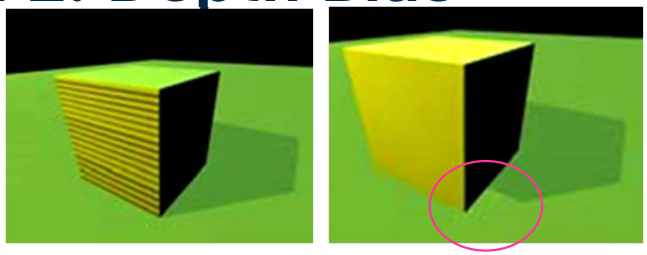
Shadow Mapping [Williams78]

1. Render depth buffer from light (**Shadow Map**)
   - Store result in a texture

2. Render from viewpoint
   - For each drawn pixel:
     Compare its depth in the light view
     to the stored depth at the texel location
     in the Shadow Map
     - Equal: pixel is lit
     - Farther: pixel is in shadow

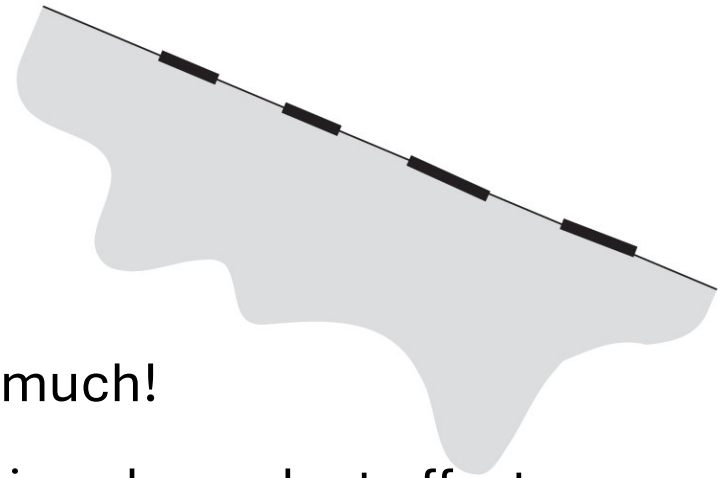


Problem 2: Depth Bias



Too small Too big

- Self-shadowing
  - Discretisation
  - Limited precision



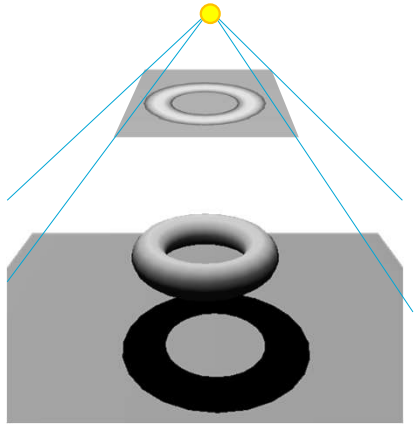
- Solution:
  add an offset, but not too much!
- OpenGL supports orientation-dependent offsets

Problem 2: Depth Bias

- “Real-world” example in Crysis by Crytek

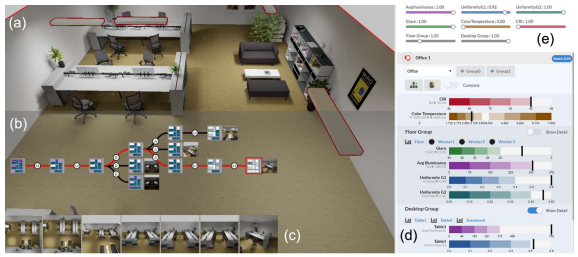
Conclusion: Shadow Mapping [Williams78]

- Simple, efficient, and easy to implement
- Compatible to most object representations
- Additional hardware support
- Variants are common
  - (games, movies…)



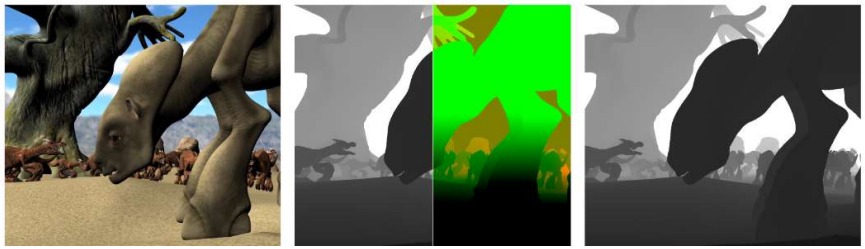
Shadow Maps have a widespread use!

Light Design



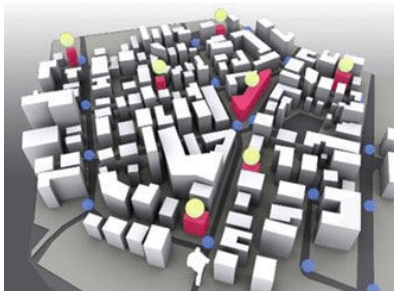
Schwaerzler et al. VIS2020

Real-time Visibility Testing of Objects



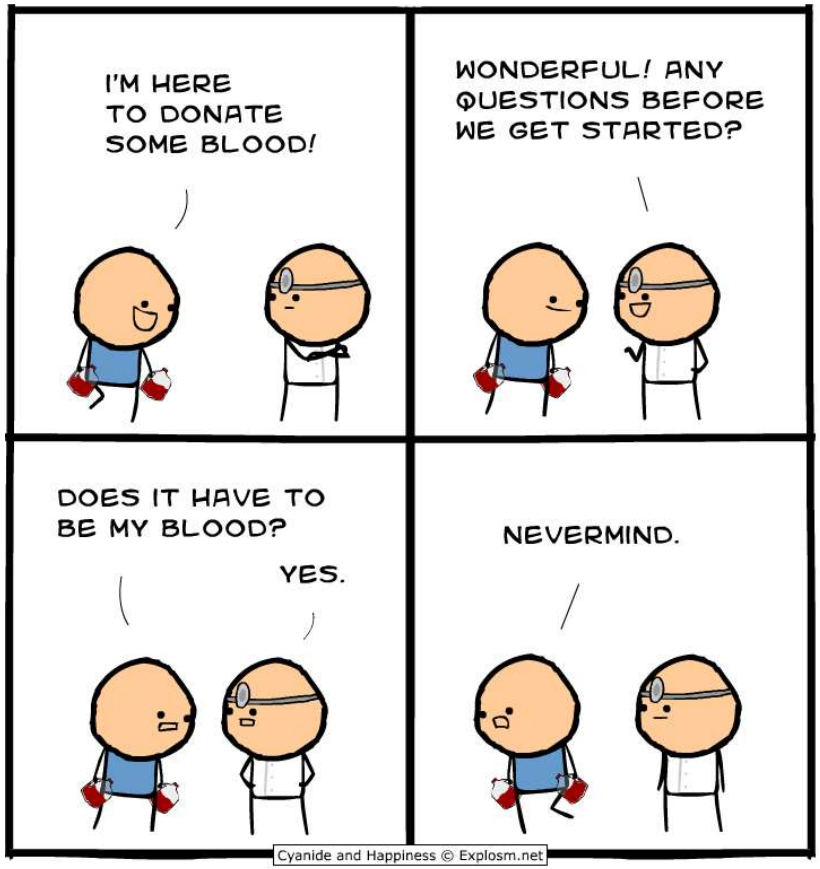
Lee et al. ToG2019

Urban Design

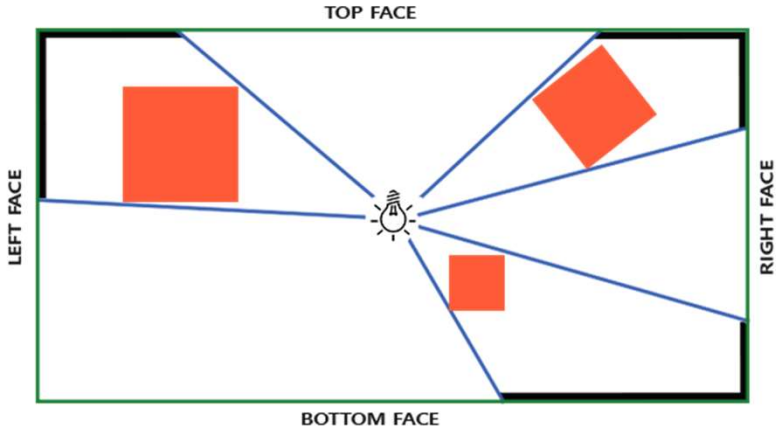


many many more...

Questions?


I'M HERE TO DONATE SOME BLOOD!
WONDERFUL! ANY QUESTIONS BEFORE WE GET STARTED?
DOES IT HAVE TO BE MY BLOOD?
YES.
NEVERMIND.
Cyanide and Happiness © Explosm.net

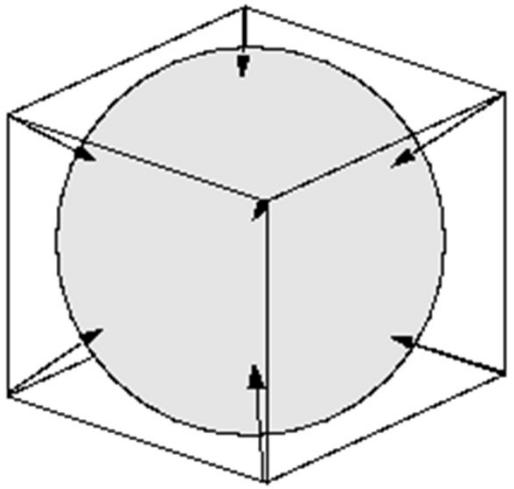
Omnidirectional Shadow Mapping?



Environment Mapping

- Render a shadow map for each view.
- Could use Geometry Shader to avoid 6 loops

Cube mapping



Thank you very much for your attention!