nvidia

Visual Computing

2024/2025

Class 8

Textures

deti departamento de electrónica, telecomunicações e informática

universidade de aveiro

Agenda

- Motivation
- Textures
- Texture Mapping
- Texture Features

Geometric Modeling – Limits

- Graphics cards can render **millions of triangles per second**
- **BUT**, that might not be sufficient…
- Skin / Terrain / Grass / Clouds / …

Shading Limits

Applying Phong's reflection model considering a single color over the object's surface:

Booooriiiing!

Very limited!

Setting a color for each vertex.... **How many vertices, then?**

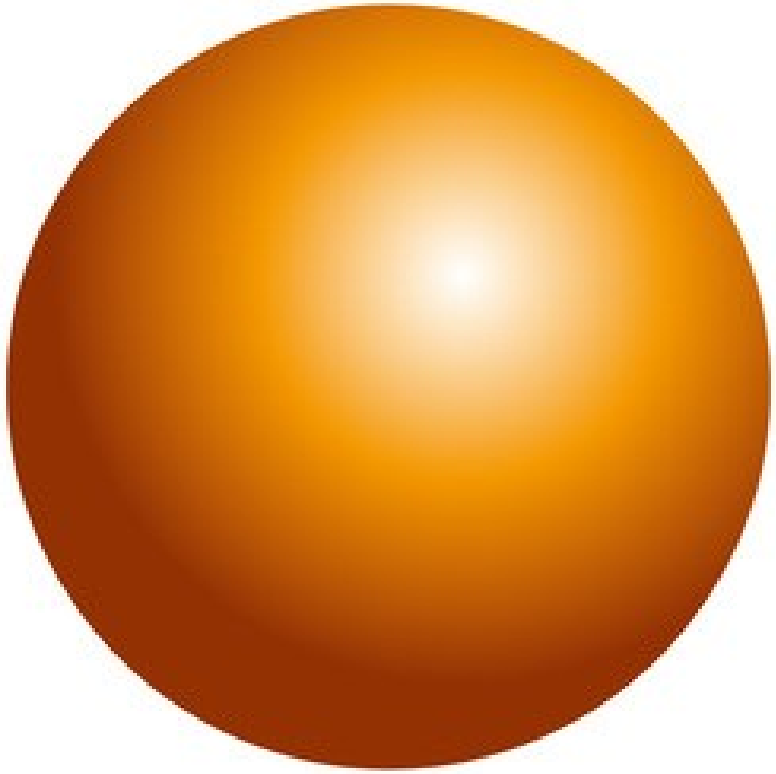
If you wish to model an orange from scratch...

... you must first create the universe

How to model / render an orange ?

An orange-colored **sphere** ?

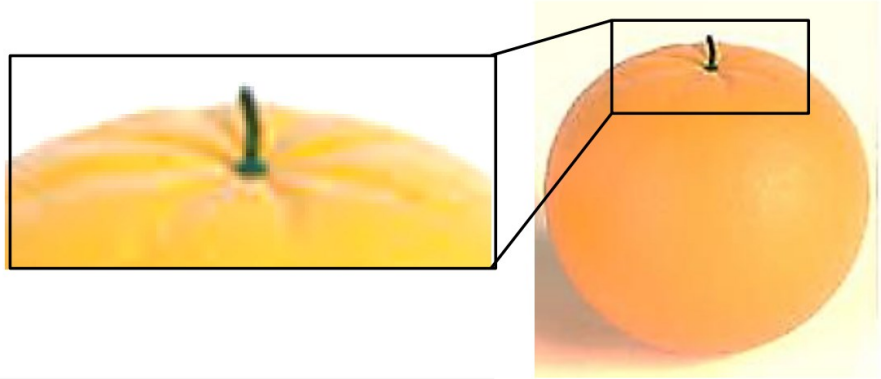
- Too simple !

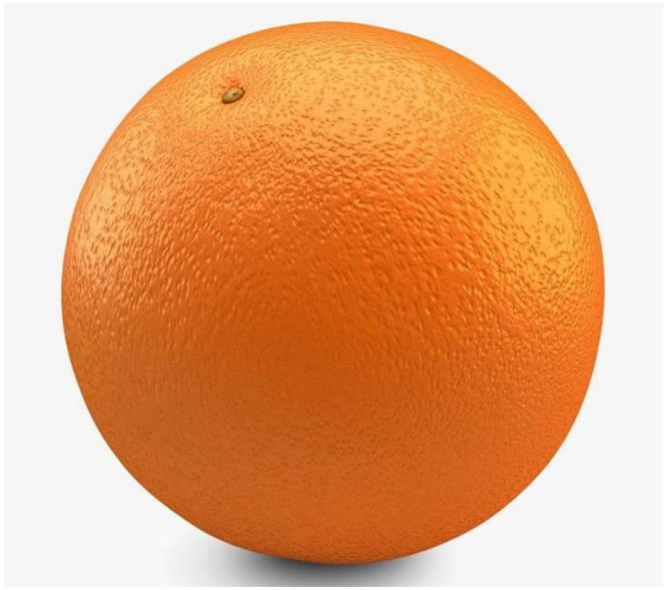


How to model / render an orange ?

A more **complex shape** to convey **details** ?

- How to represent surface features ?
- Takes **too many triangles** to model all the dimples…





How to model / render an orange ?

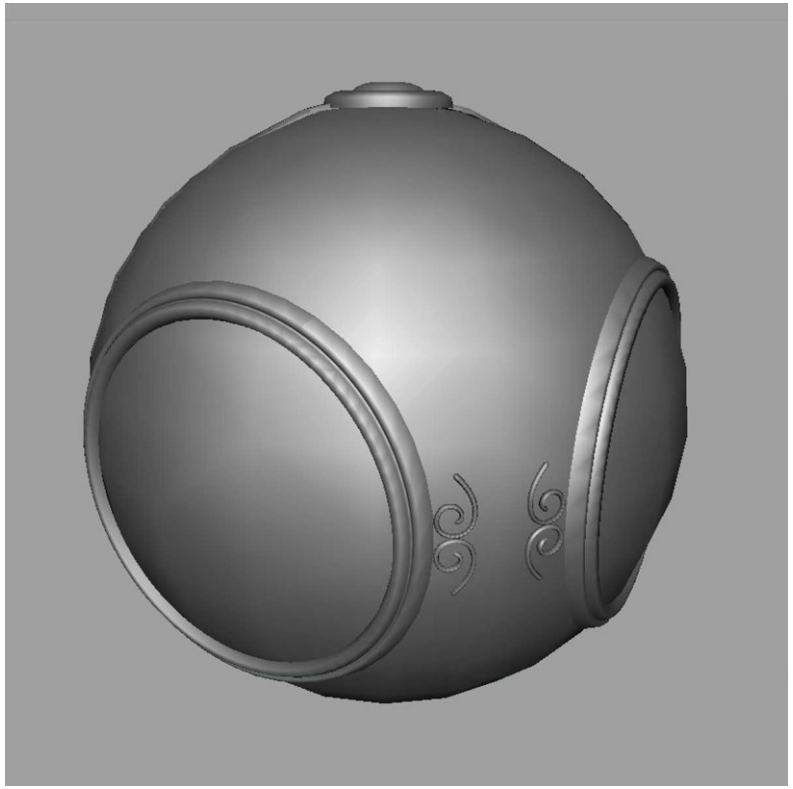
Simple **geometric model** + **Texture**

- Take a picture of a real orange
- Scan and “paste” it onto model

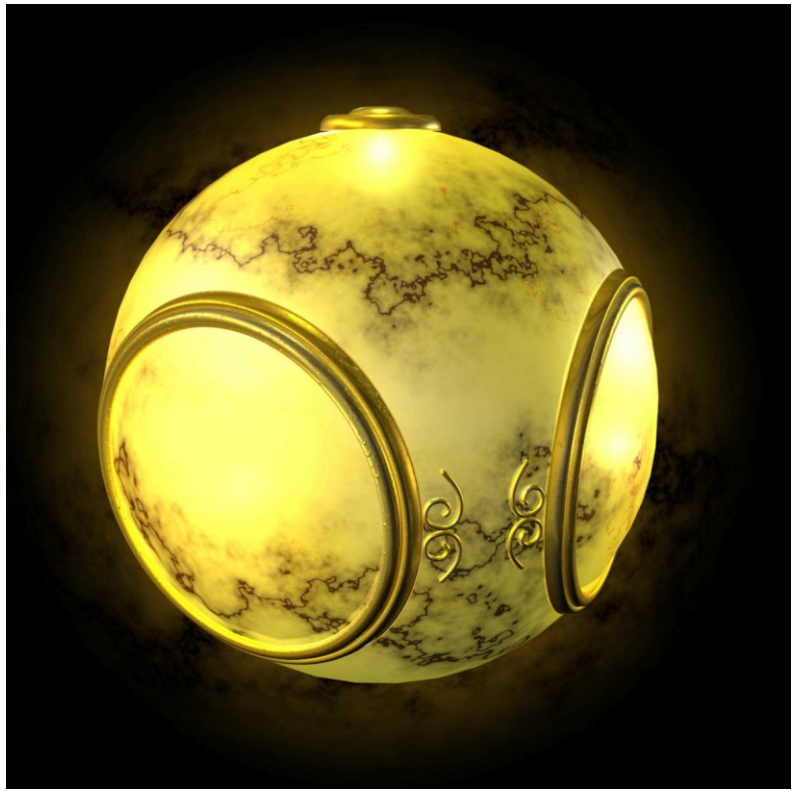
This is called Texture mapping

Textures

Texture Mapping



geometric model

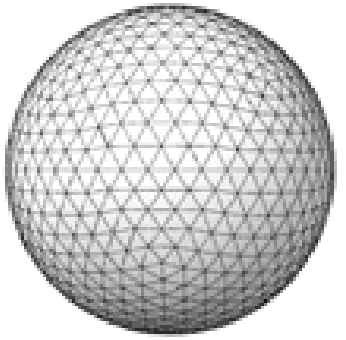


texture mapped

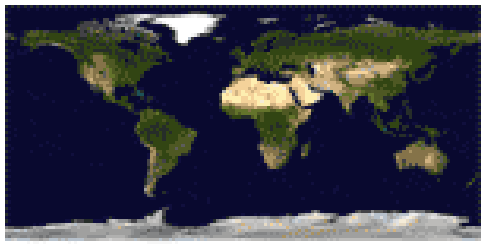
[Ed Angel]

Texture Mapping

- Paste the texture on a surface to **add detail** without adding more triangles
  - Get surface color or alter computed surface color
- Simplest surface detail hack
- Implemented in hardware on every **GPU**



Sphere with no texture



Texture image

Sphere with texture

Basic Concept

Glue an image to a model

Map 2D image to a 3D surface

Texture coordinates

- (s,t), represent a location on the image

Assign texture coordinates to vertices

- what part of the texture goes in what part of the model

Basic Concept

Once we know which is the **value of the texture** for a vertex, use that as the **color of the vertex**.

This is called parametric texture mapping

Texture Image

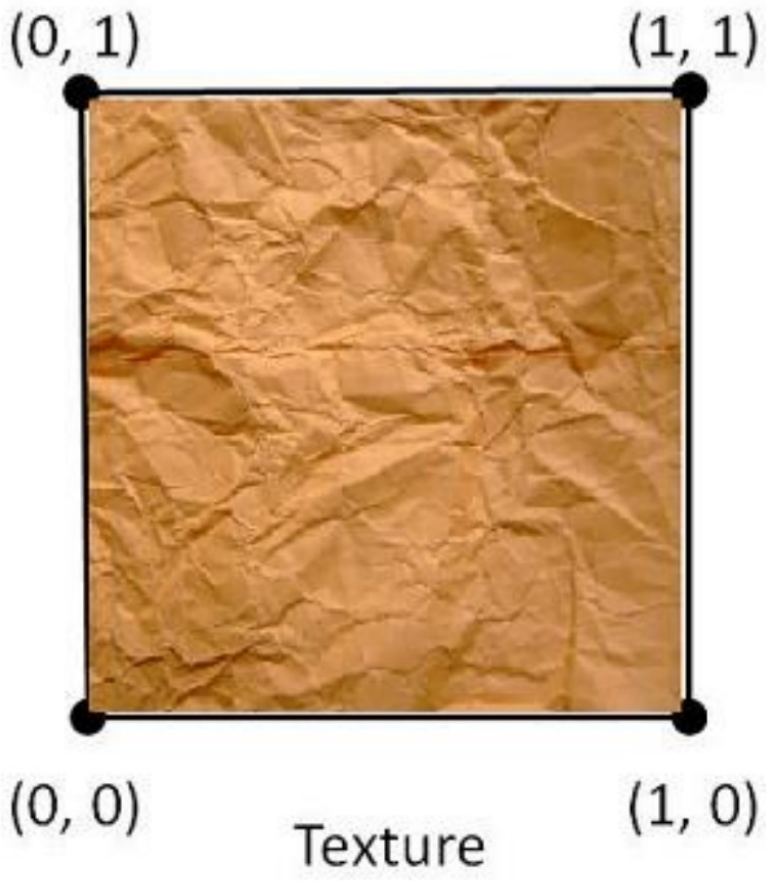
2D Image (height x width)

Elements are called **texels** (texture elements

Each texel has coordinate (s,t)

s and t normalized to [0,1] range

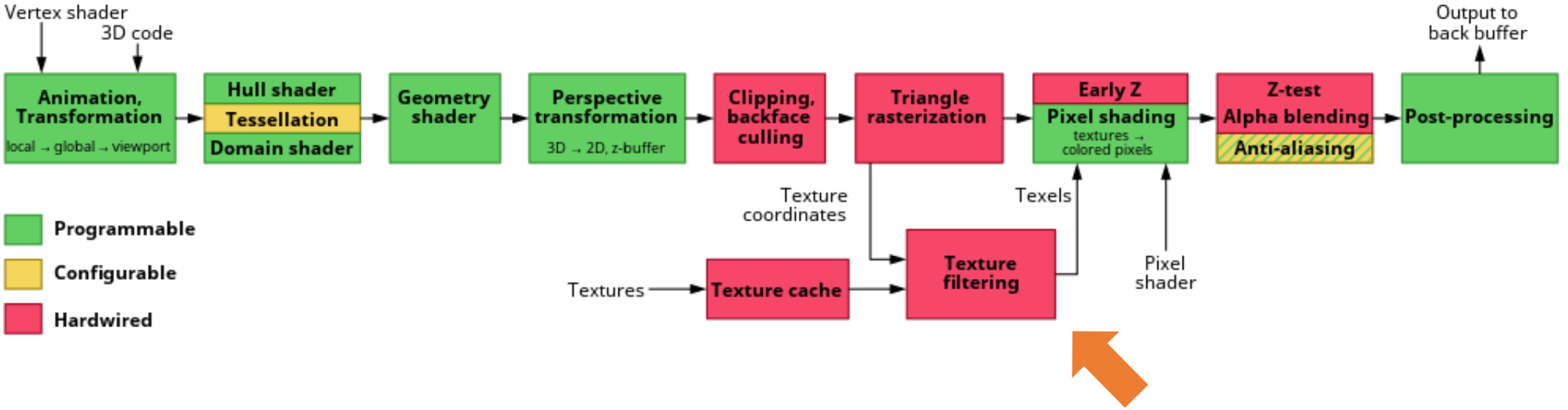
s and t have **nothing to do with pixels**



Texture mapping

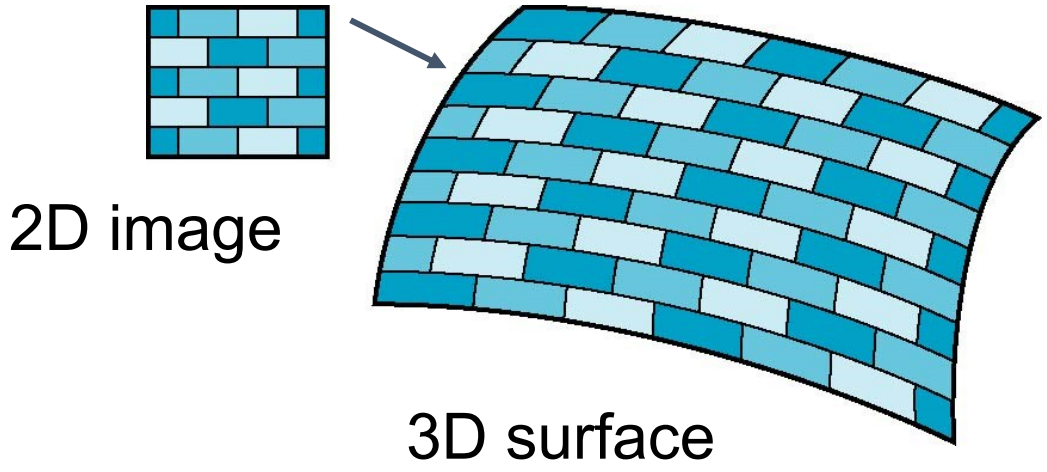
Where does it take place ?

- Mapping techniques are implemented at the **end** of the rendering **pipeline**
  - Very efficient because **few polygons** make it past the clipper. **Why?**



Mapping – Is it simple ?

Although the idea is simple – map an image to a surface – there are 3 or 4 coordinate systems involved

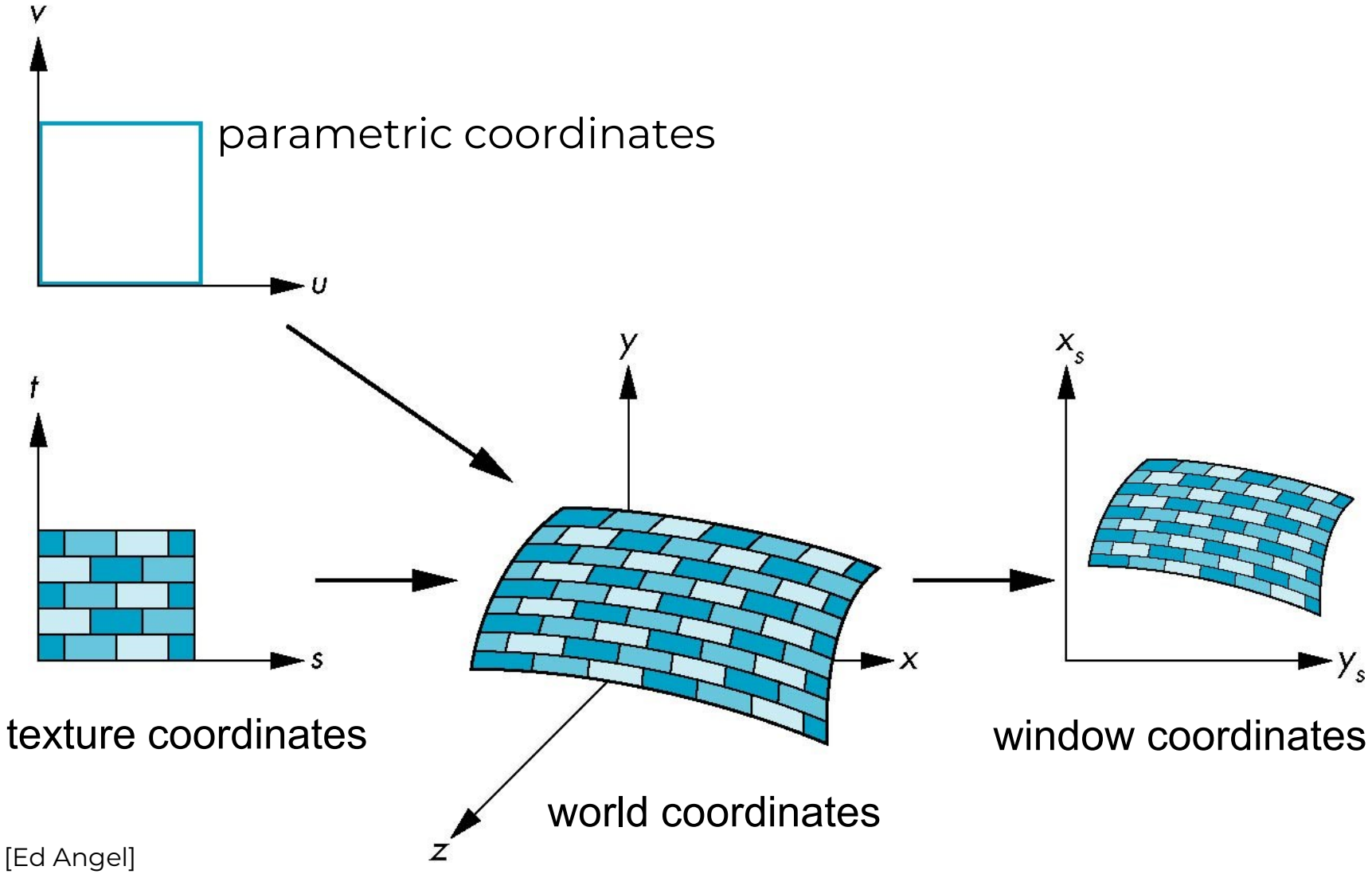


[Ed Angel]

Coordinate Systems

- Parametric coordinates
  - May be used to model **surfaces**
- Texture coordinates
  - Used to identify points in the **image** to be mapped
- Object or World Coordinates
  - Conceptually, where the mapping takes place
- Window Coordinates
  - Where the final image is really produced

Texture Mapping



[Ed Angel]

Mapping Functions

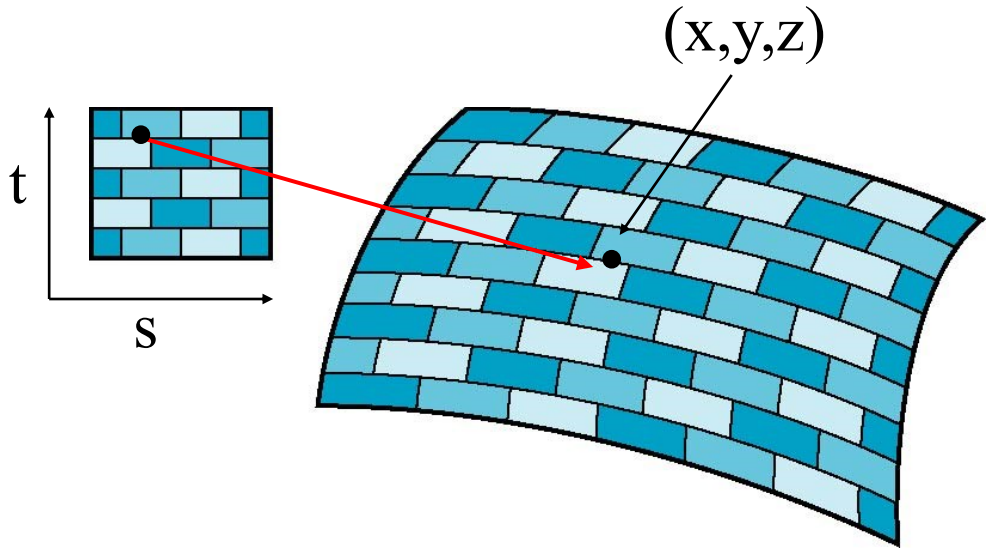
Mapping from texture coordinates to a point on a surface

Appear to need three functions

$$x = x(s,t)$$
$$y = y(s,t)$$
$$z = z(s,t)$$

But we really want to **go the other way**

Why?



Backward Mapping

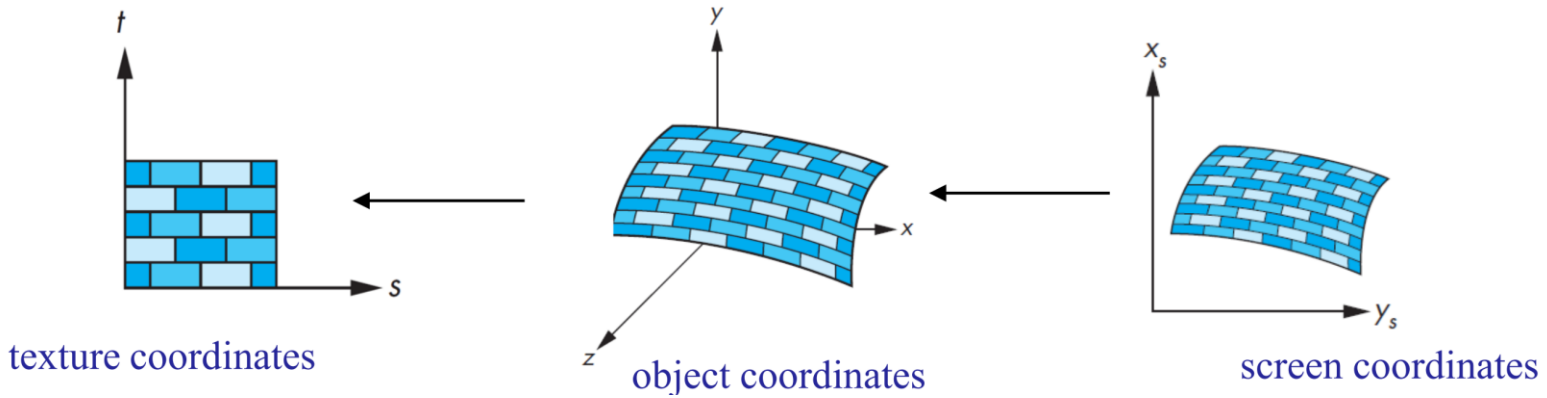
Given a **pixel**, we want to know to which **point on an object** it corresponds

Given a **point on an object**, we want to know to which **point in the texture** it corresponds

Need a map of the form

$$s = s(x,y,z)$$
$$t = t(x,y,z)$$

Such functions are difficult to find in general

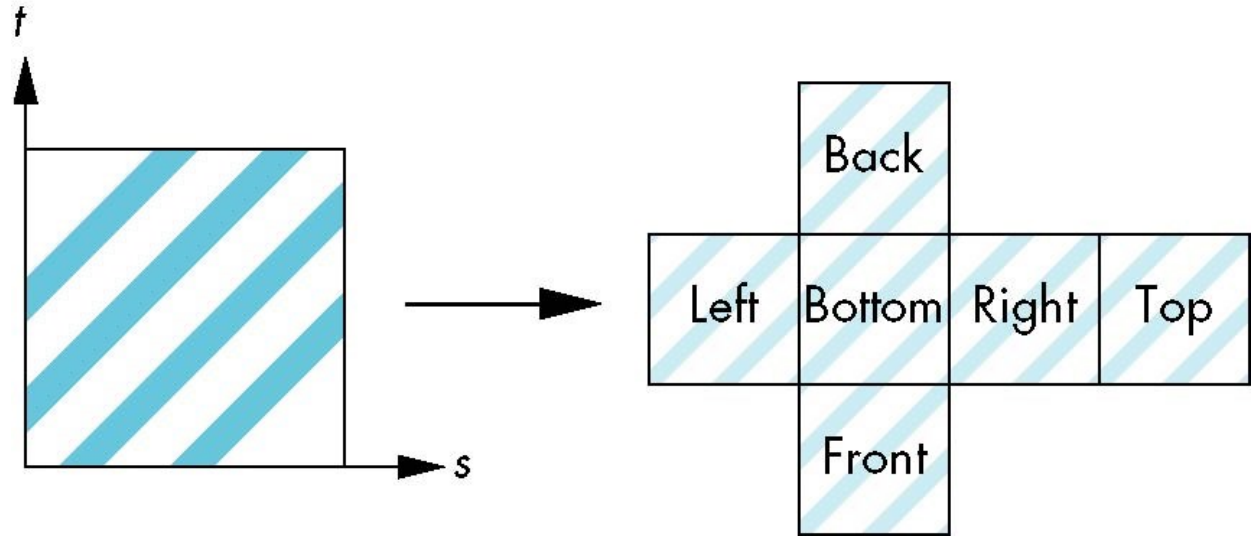


How can we map a texture onto an arbitrary object?

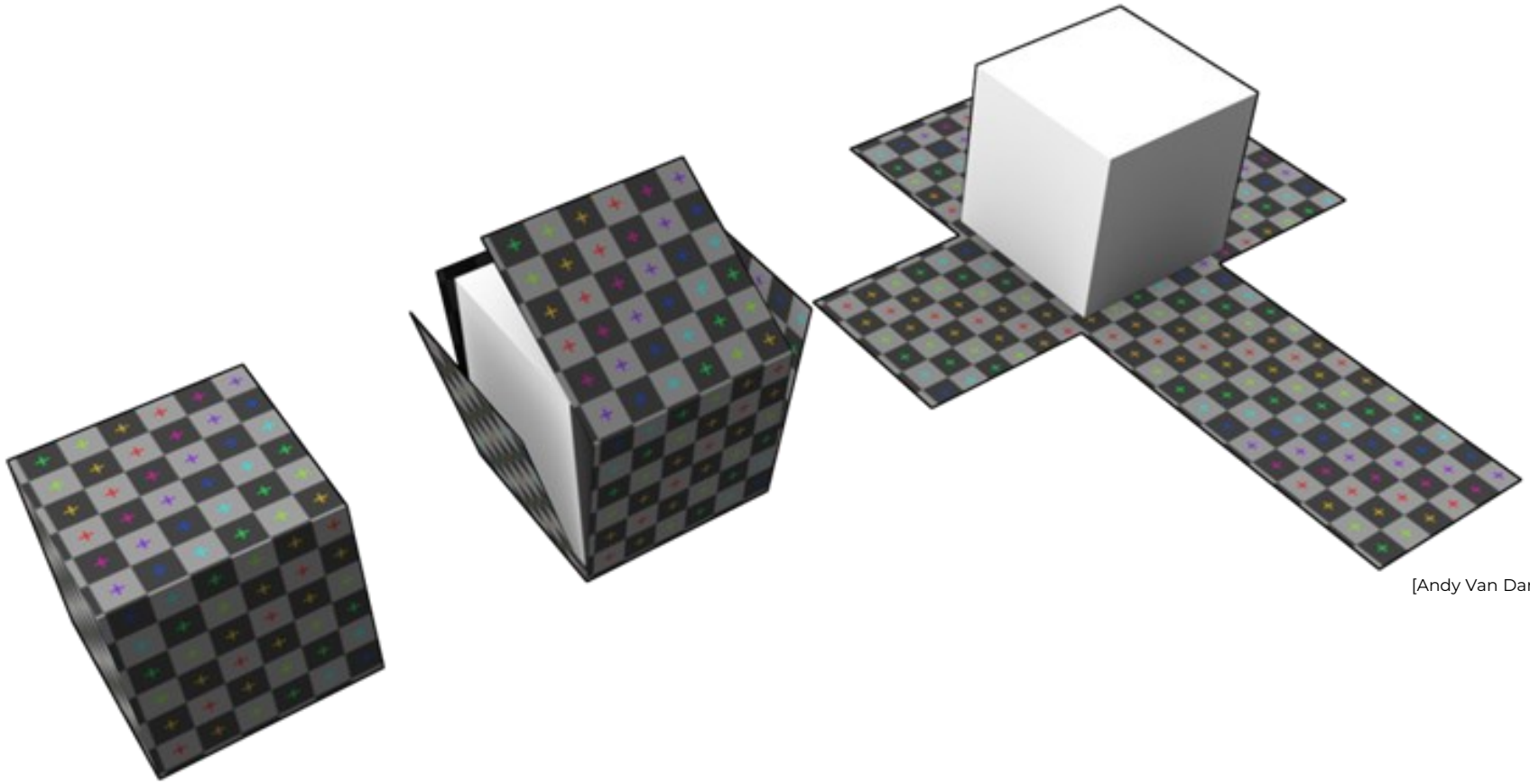
Box Mapping

Easy to use with simple orthographic projection

Also used in environment maps



Example

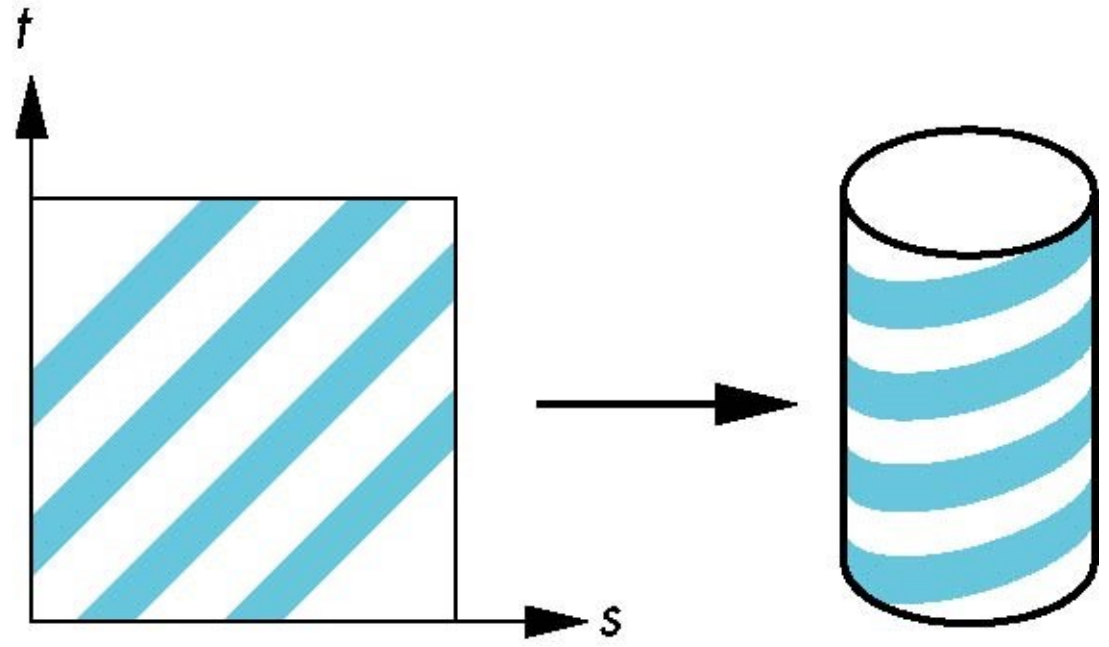


[Andy Van Dam]

Two-part mapping

- One solution to the mapping problem is to **first** map the texture to a simple **intermediate surface**

- Example: map to cylinder

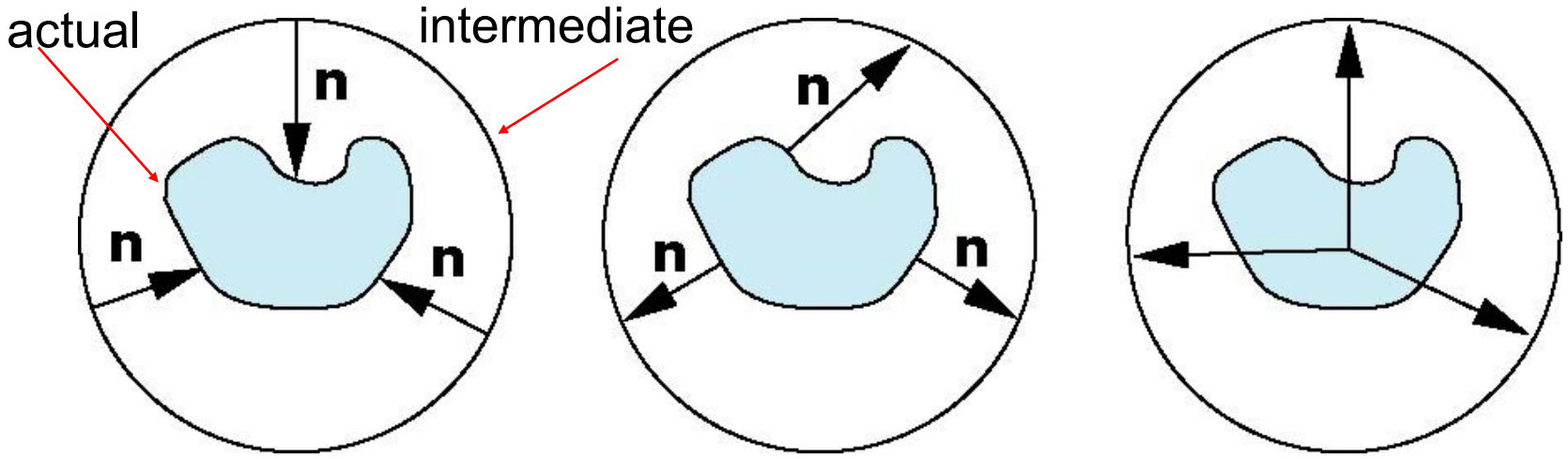


sss_cv_tex_v4

Two-part Mapping

Map from intermediate object to actual object. Different **alternatives**:

- Normals from intermediate to actual
- Normals from actual to intermediate
- Vectors from center of intermediate



Complex Geometry/Real Applications

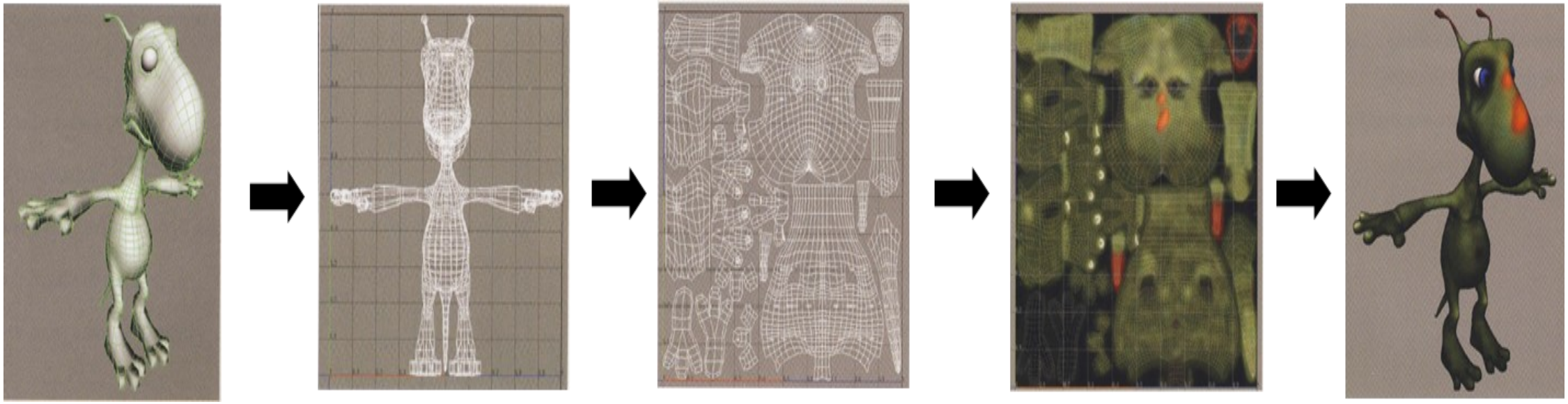
Texture mapping of **complicated objects**, not simple primitives

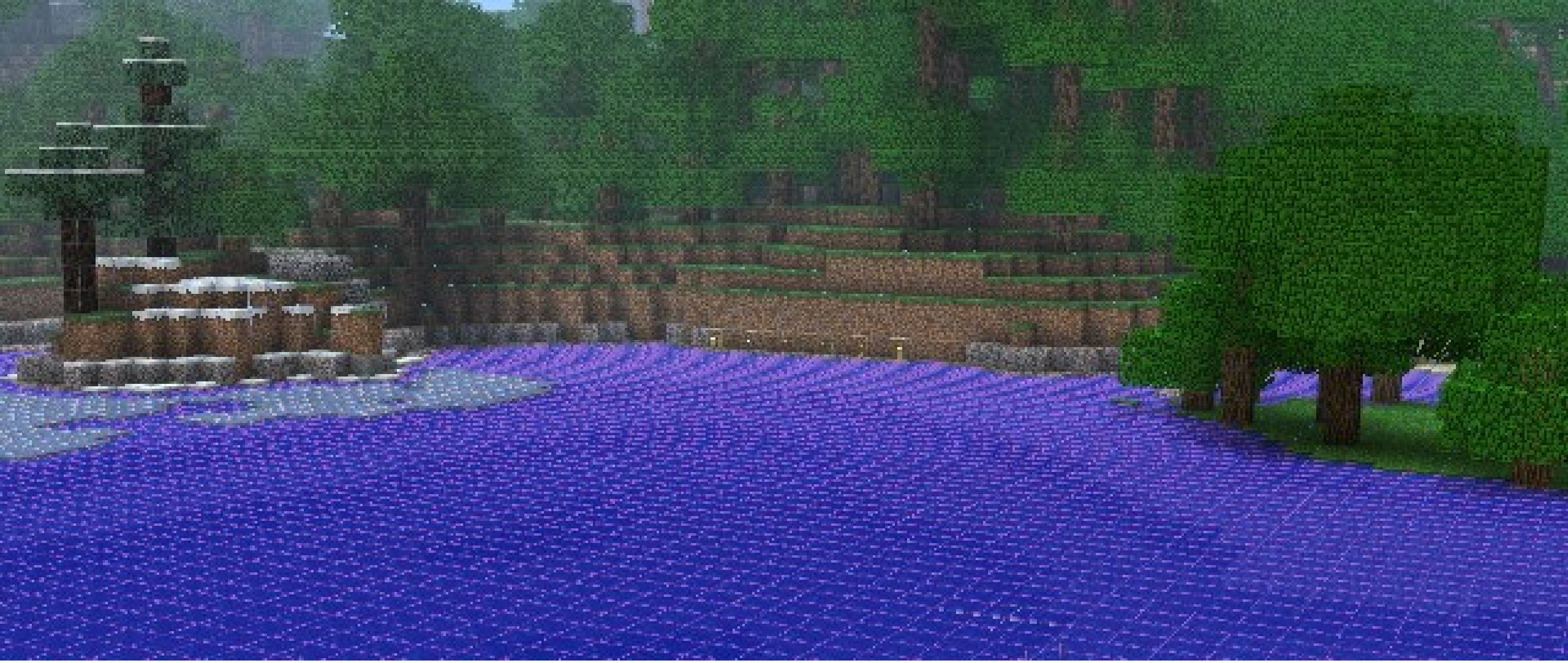
Need **precise control** over how the texture map looks on the object

Use 3D modelling programs

- E.g., **Maya**, Zbrush, Blender, …

Complex Geometry/Real Applications

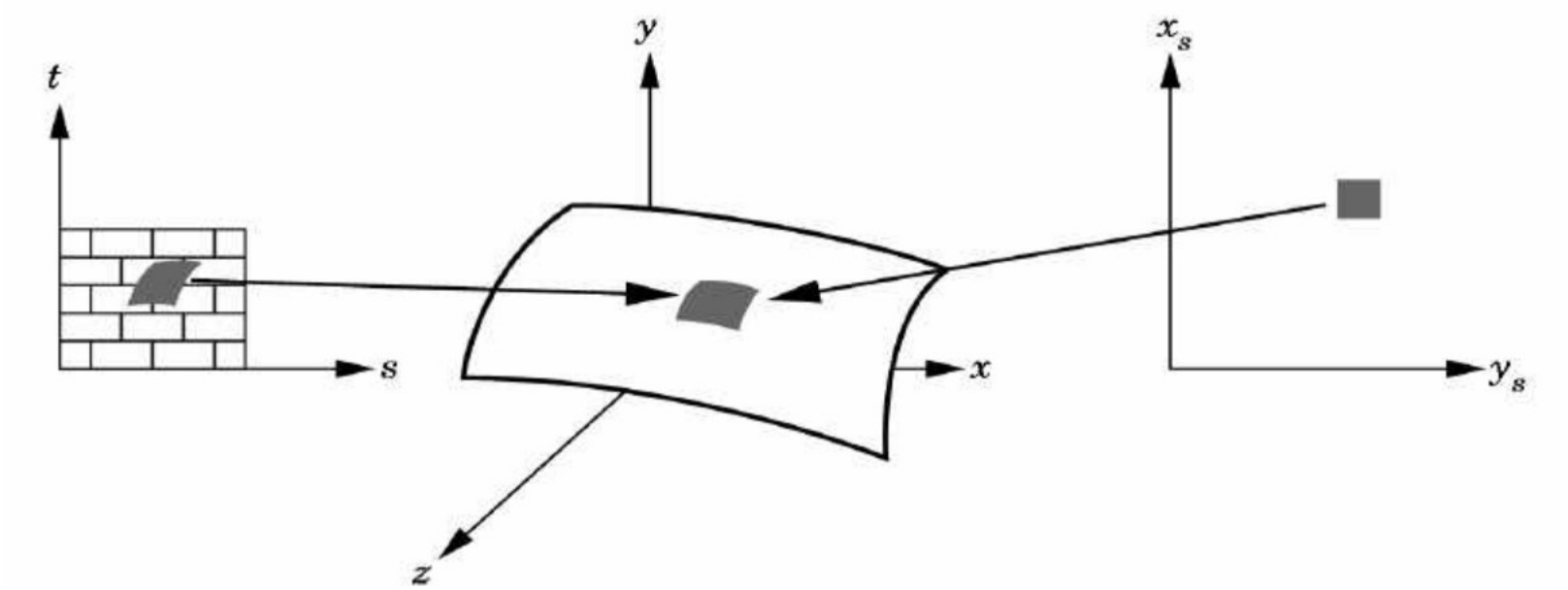




Aliasing

Texture to Screen

We need to map from screen to texture coordinates



Texture to Screen

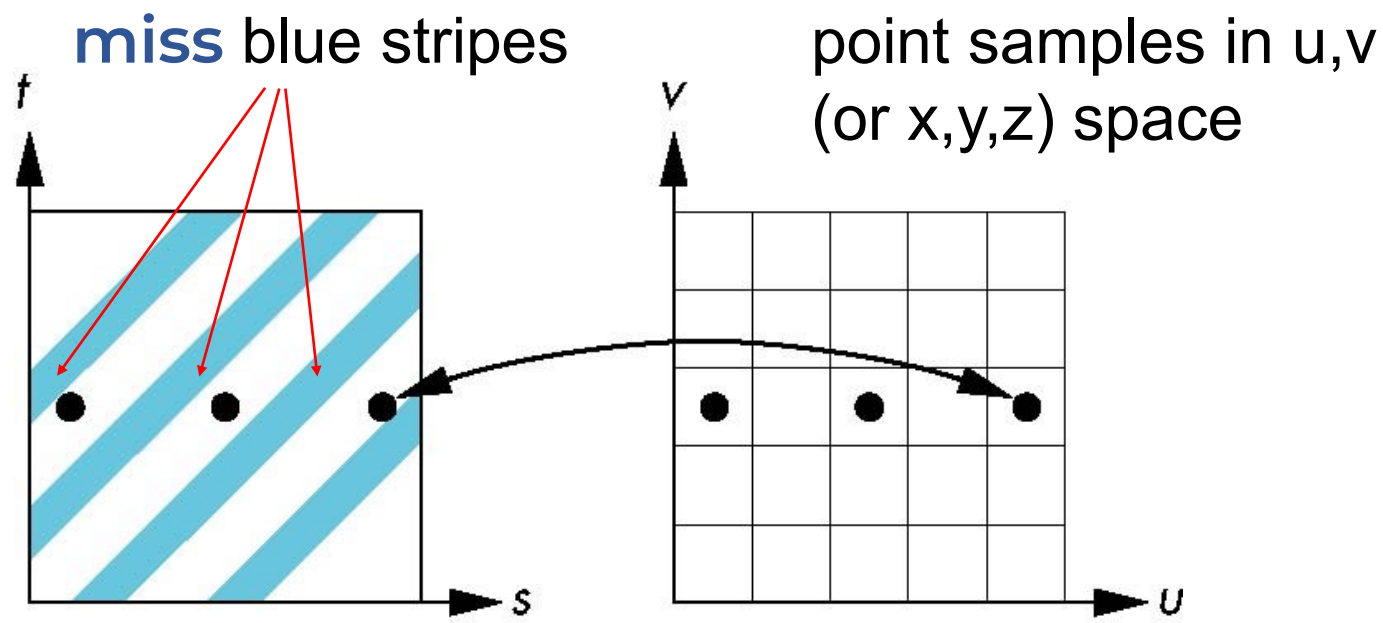
It is almost never the case that pixels in a texture match one-to-one with pixels on screen

Sometimes, a **single texel is stretched** to span over multiple screen pixels

Or several texels end up squished in a single screen pixel

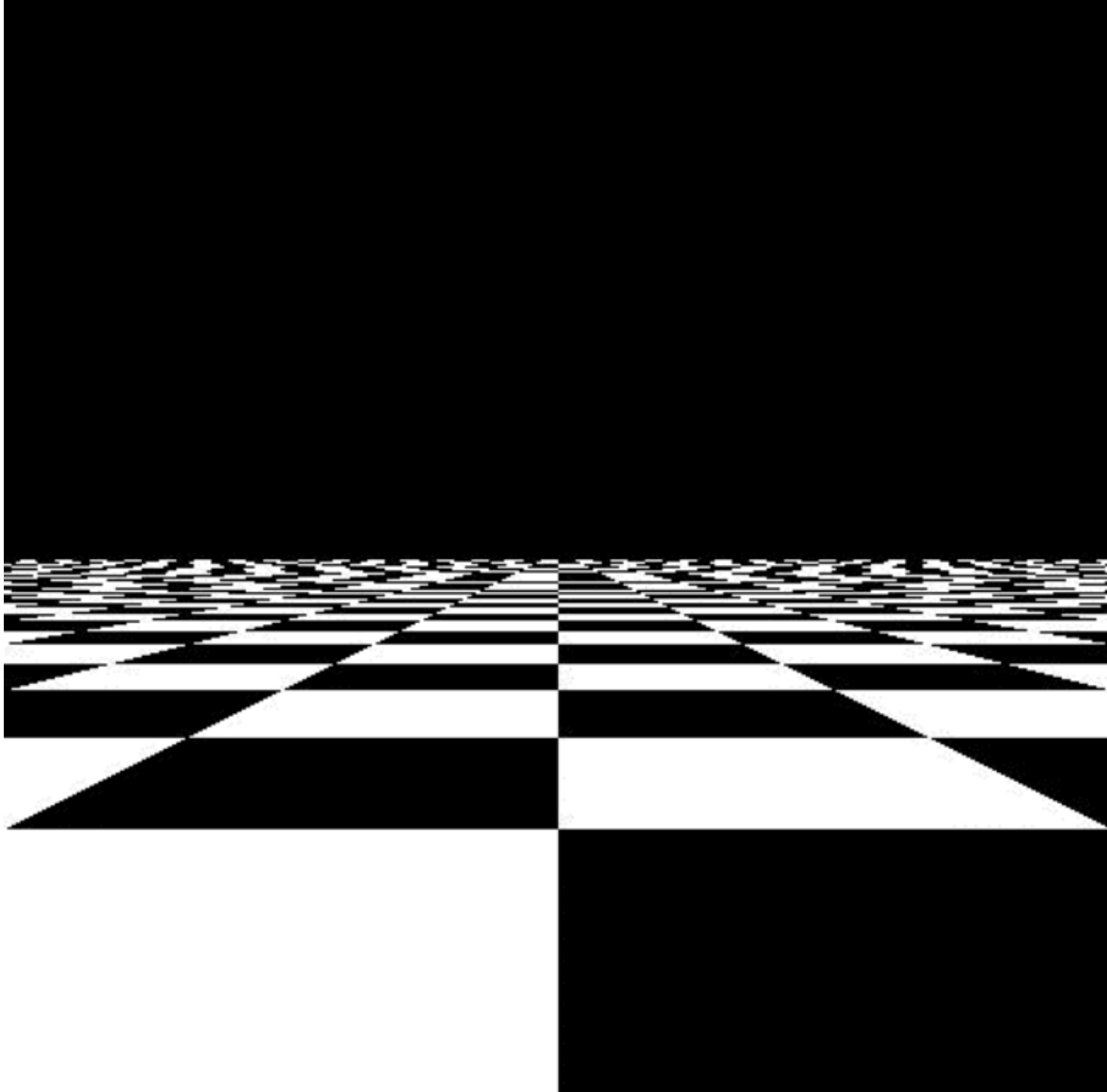
Aliasing

- Point sampling of the texture can lead to **aliasing errors**



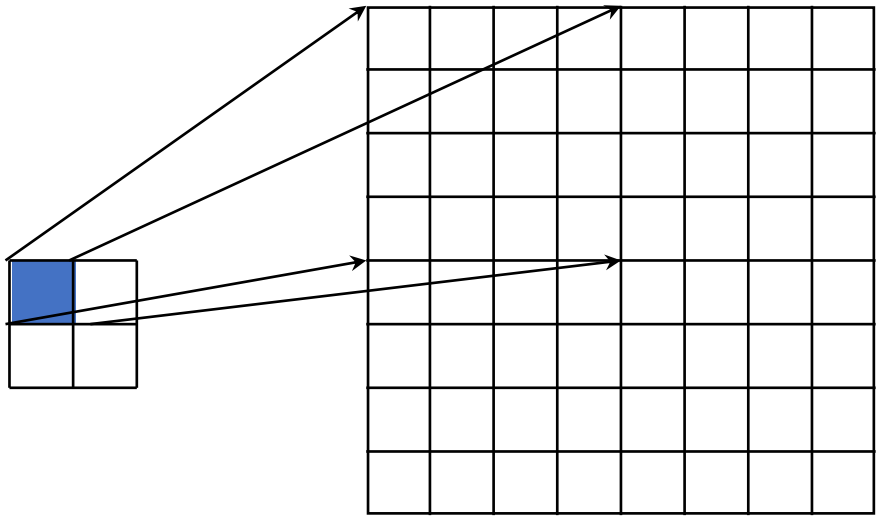
[Ed Angel]

Aliasing



Magnification and Minification

- *Magnification* : one texel is stretched over several pixels

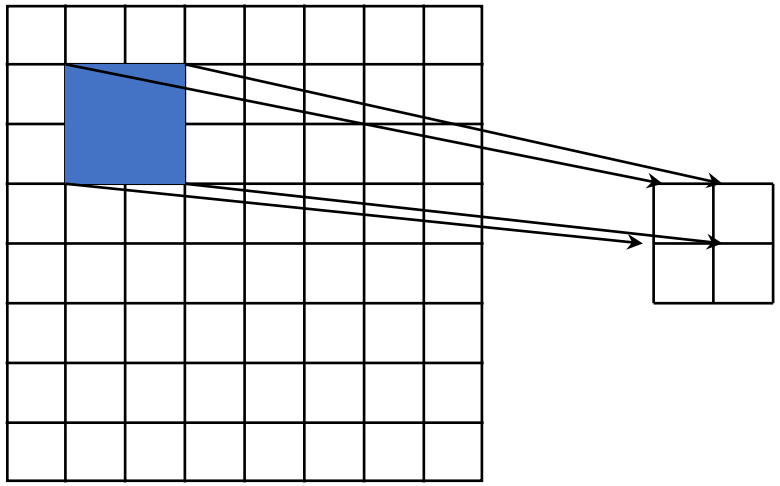


Texels / Pixels < 1

Texture Polygon

Minification

- ***Minification*** : several **texels** are squished in one screen pixel



Minification

Texels / Pixels > 1

[Ed Angel]

Texture Filters

How do we compute how the texture will look?

For **magnification** and **minification**

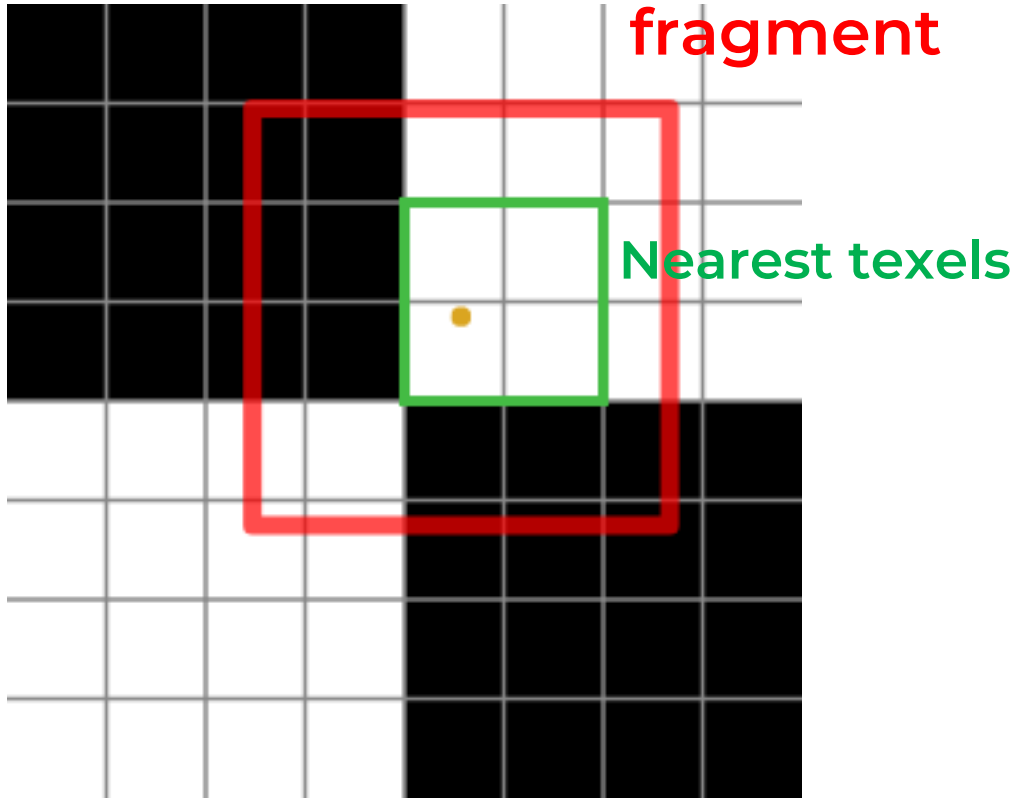
- Nearest neighbour
- Linear (sample the four nearest pixels) – **most used**

Just for minification:

- Mipmap this
- Mipmap that
- Mipmap all around
- **What??**

Minification in Detail

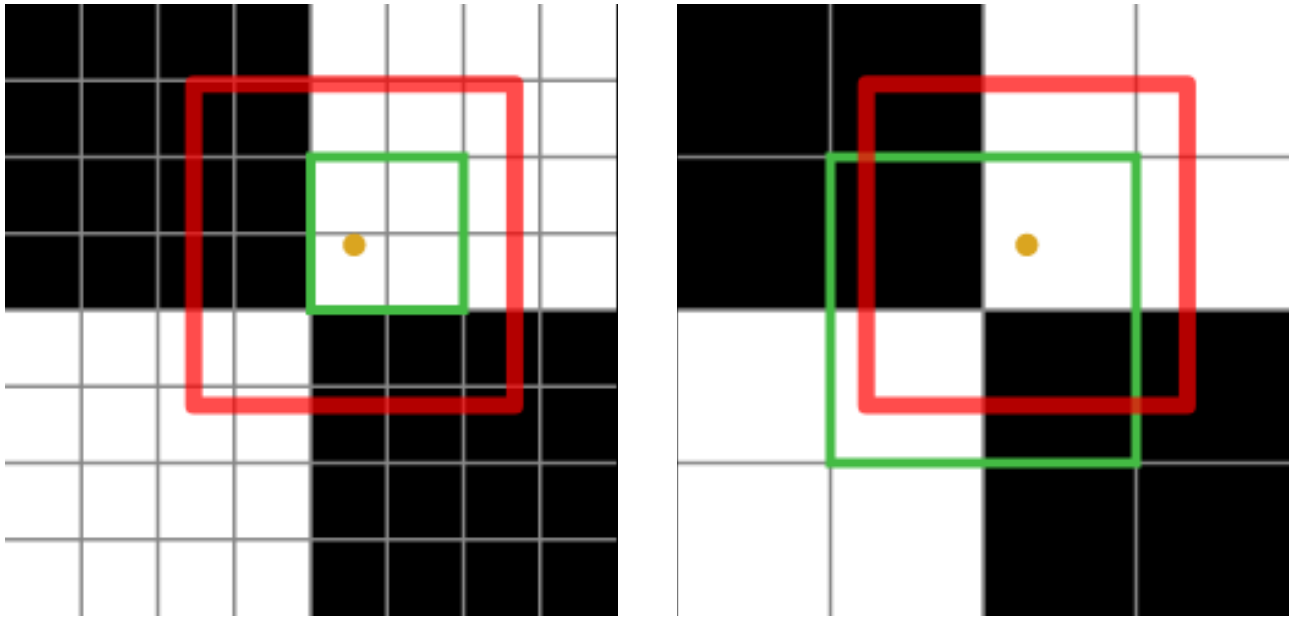
- Considering nearest neighbour sampling, for this example we get **white**
- However, it is easy to infer that **some shade of gray** would be more appropriate
- Solution?
- Maybe sample from more texels?
- Could be, but would get rather **computationally expensive**…



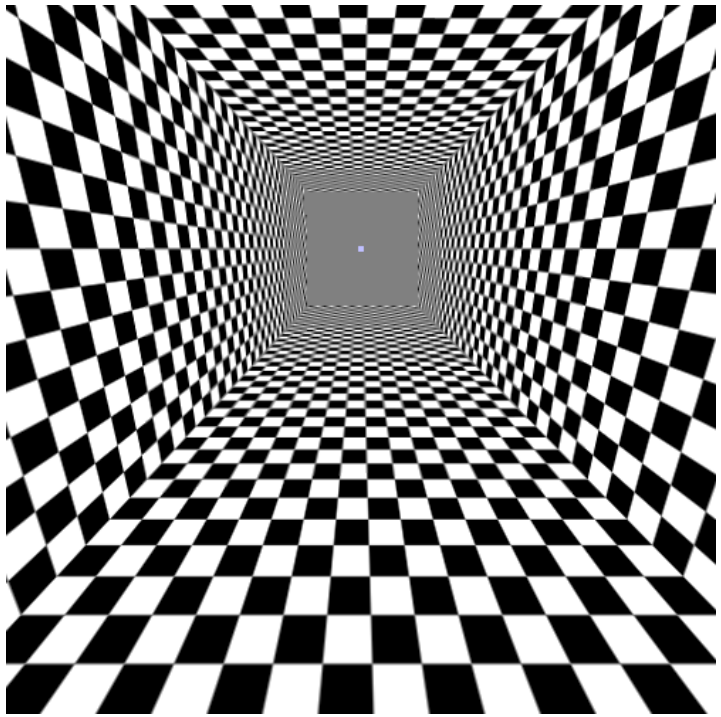
https://paroj.github.io/gltut/Texturing/Tut15%20Needs%20More%20Pictures.html

Minification in Detail

- What if we had **several lower resolution versions** of the texture?
- We could choose the one where the 4 nearest neighbours texels have a **size close to the fragment area**
- These smaller resolution textures are called **mipmaps**
- ***multum in parvo***
  - *Much in little*

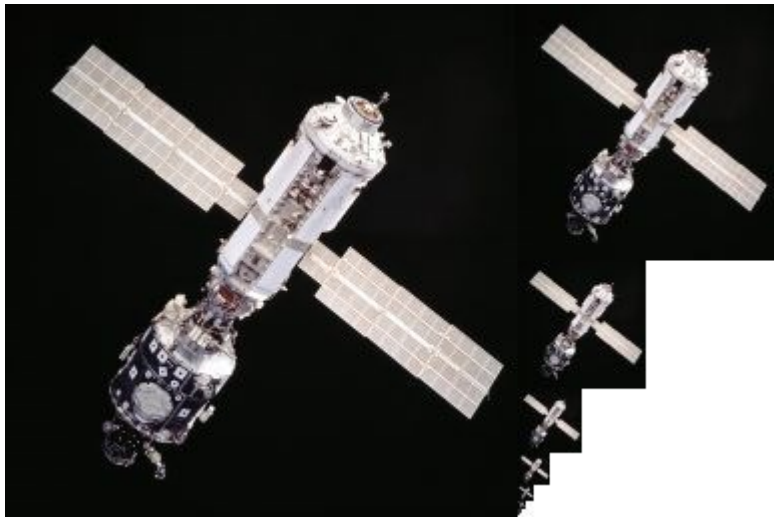


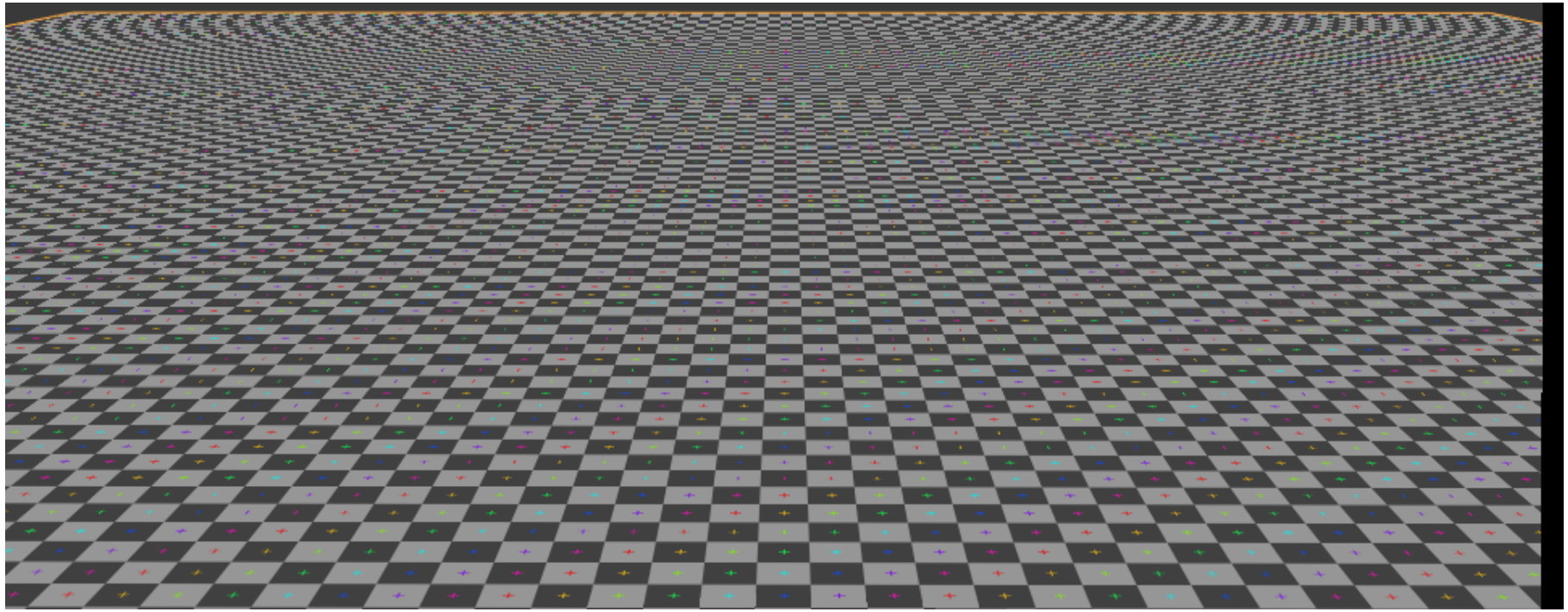
Mipmapped Textures

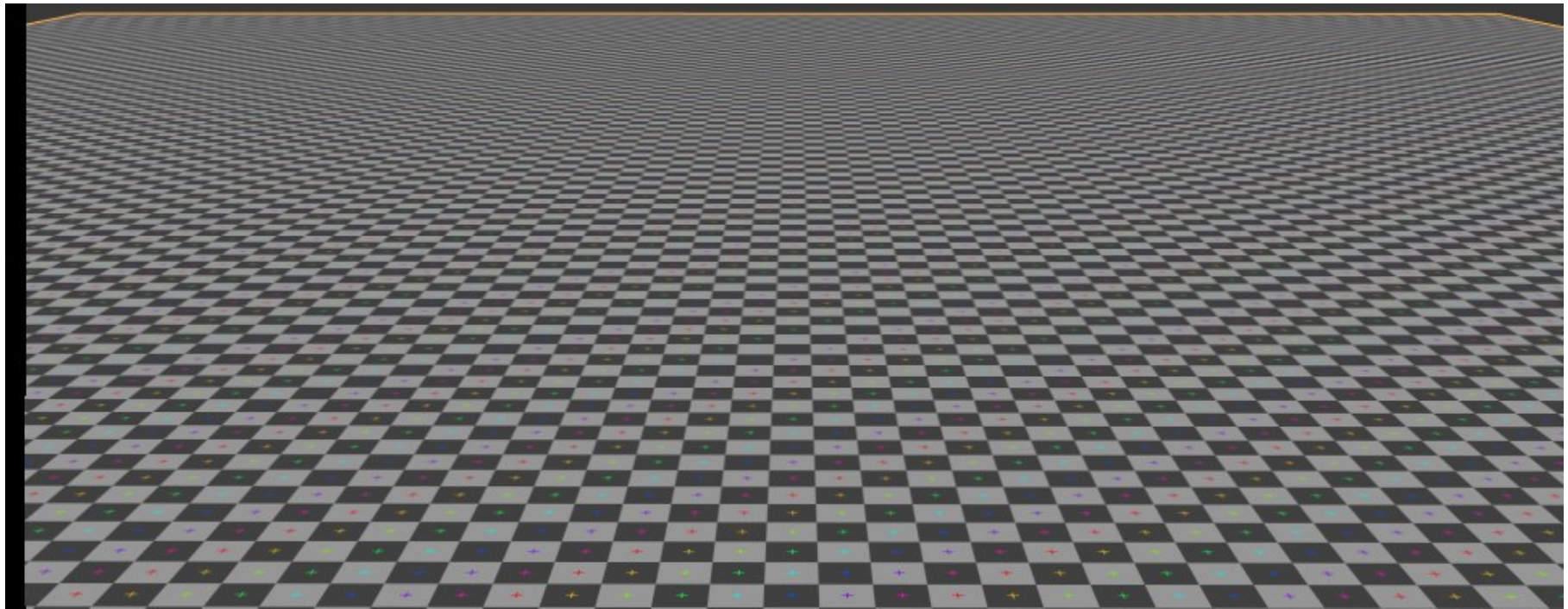


- ***Mipmapping*** allows for prefiltered texture maps of **decreasing resolutions**

- Lessens interpolation errors for smaller textured objects







Filtering

Average multiple texel to obtain the final RGB value

Filtering

Bilinear filtering

- Average 4 surrounding texels
- Cheap, but does not deal well with transitions between mipmaps

Filtering

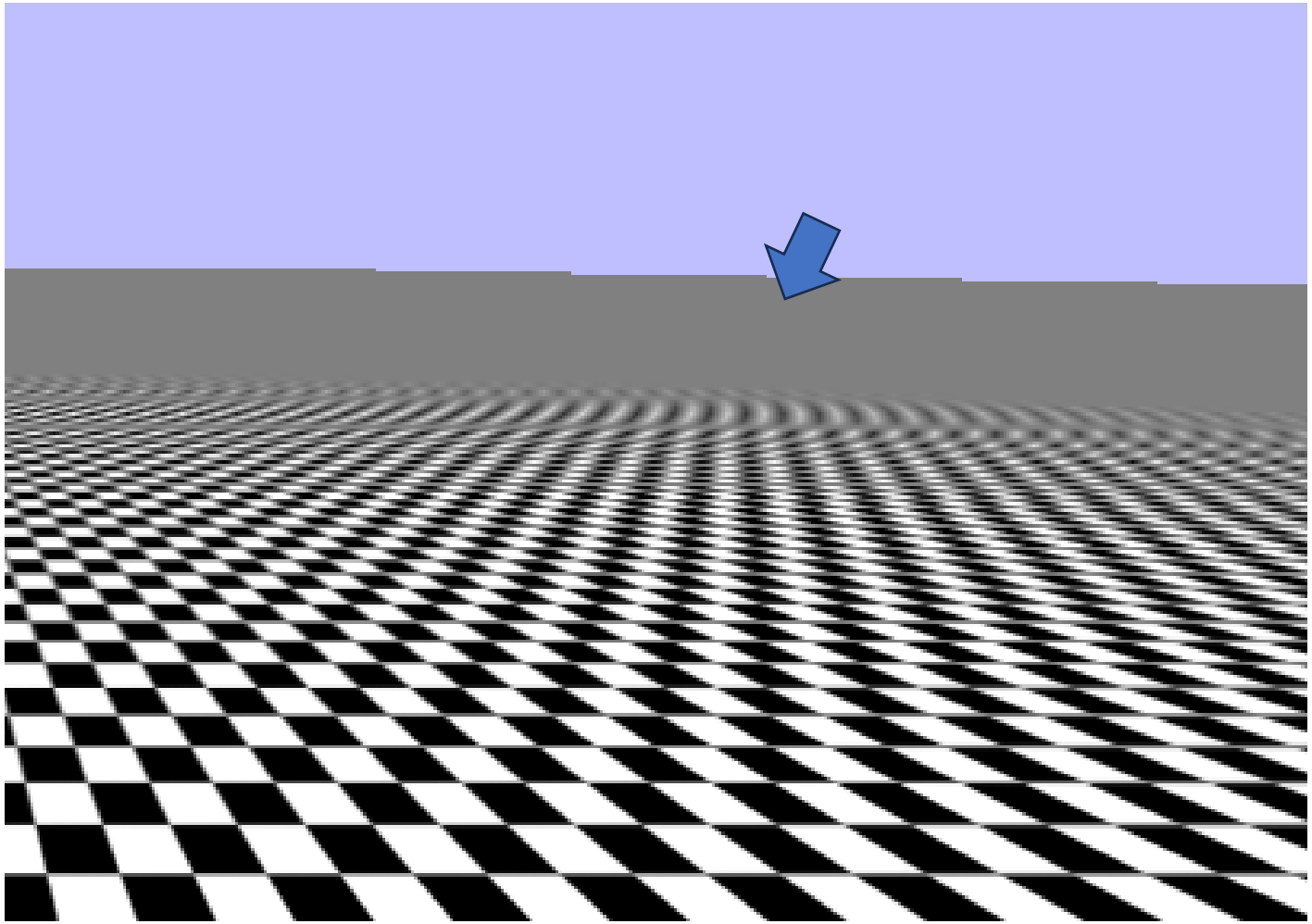
Trilinear filtering

- Interpolate between mipmaps
- Final RGB results from interpolation of a bilinear filtering at each mipmap
- Eliminates transition effect between mipmaps
- More “expensive”

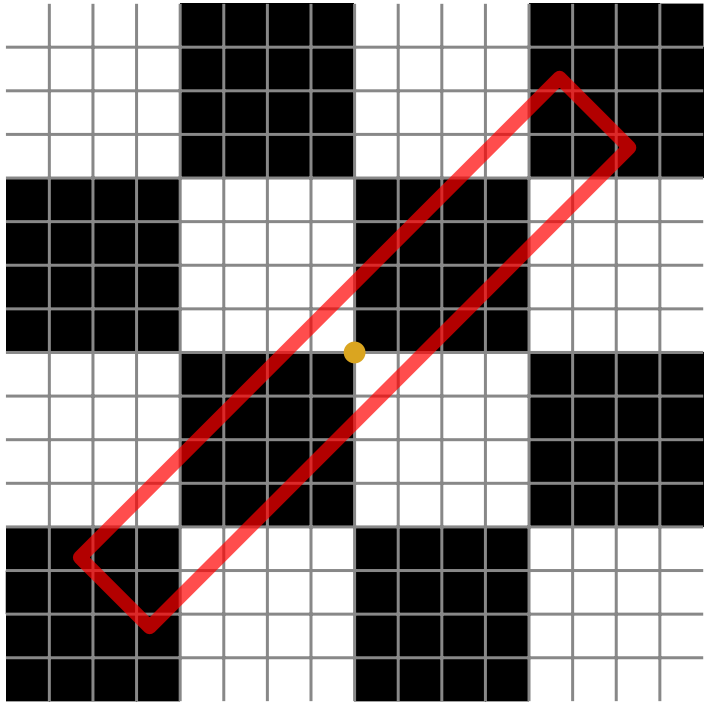
Filtering

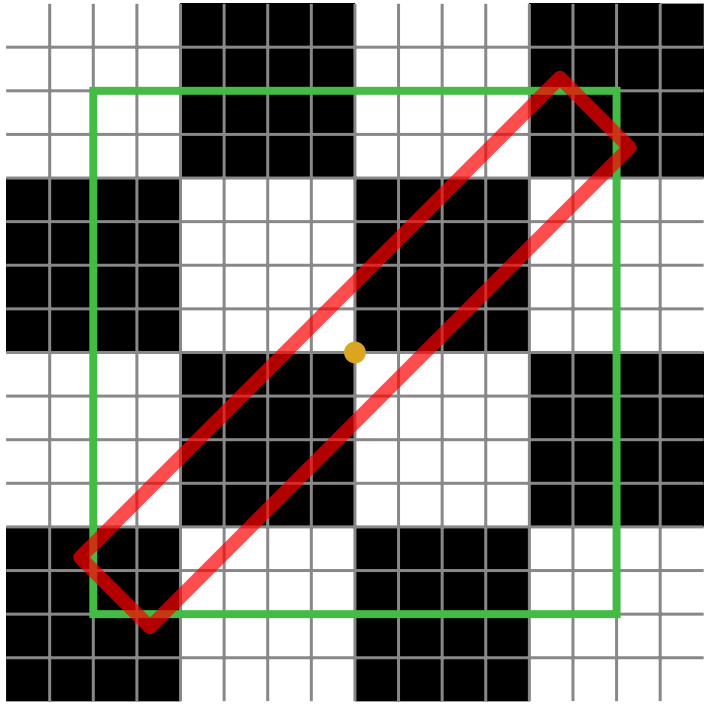
Mipmaps and trilinear filtering work nicely for square fragments

For oblong shapes, since mipmaps use similar sampling in both dimensions, things go wrong…



Filtering

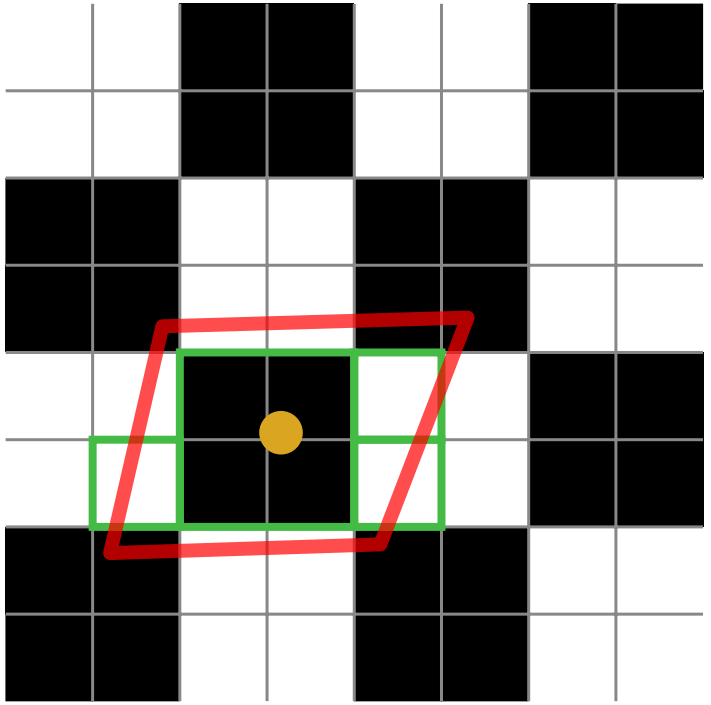




Filtering

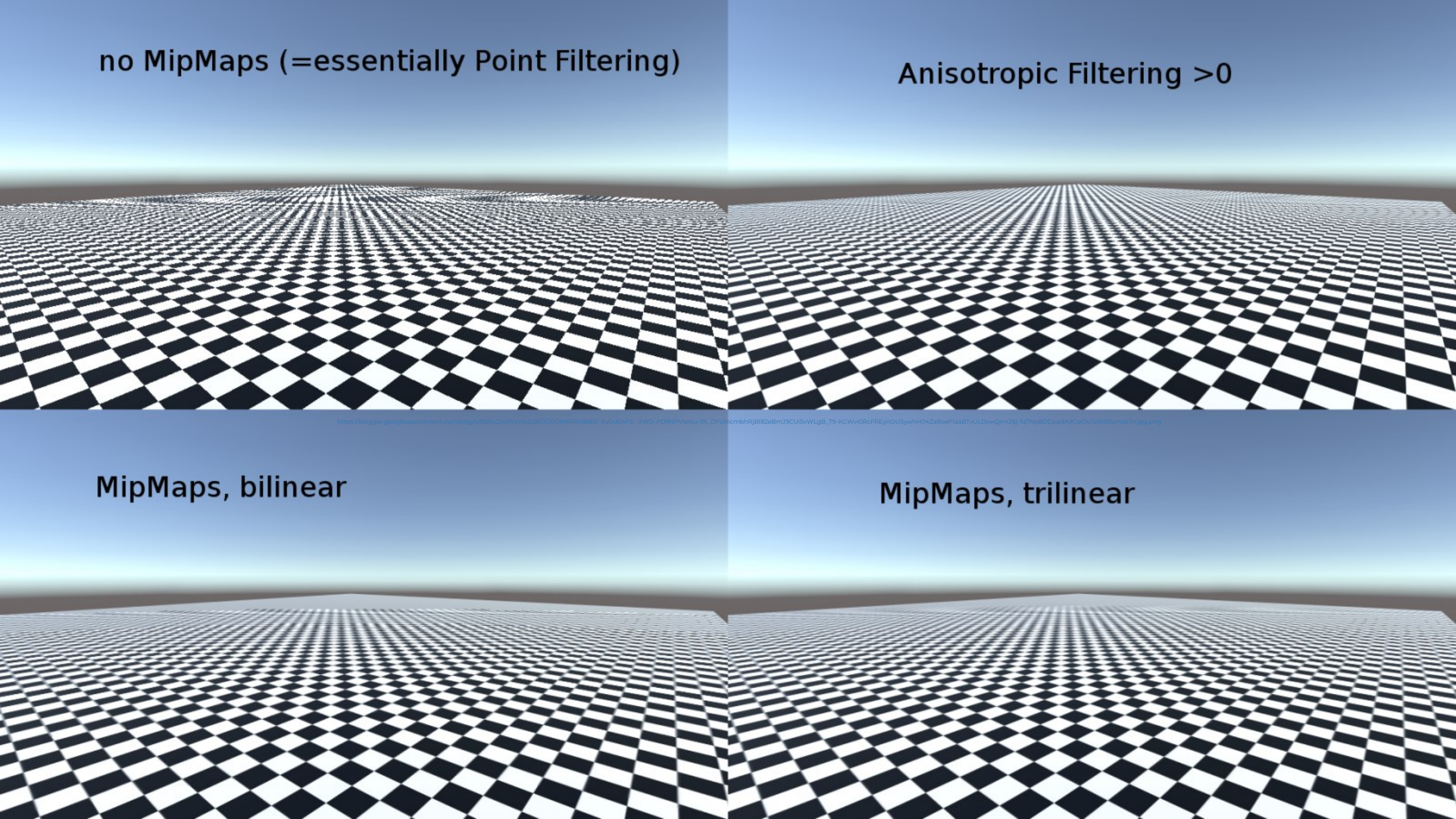
Previous approaches assume that a pixel corresponds to a square set of texels

Tilted surfaces actually do not follow that rule



Anisotropic Filtering

- A pixel may map onto an anisotropic region of texels
- Then uses bilinear or trilinear filtering
- Costly, but produces good results


no MipMaps (=essentially Point Filtering)
Anisotropic Filtering >0
MipMaps, bilinear
MipMaps, trilinear

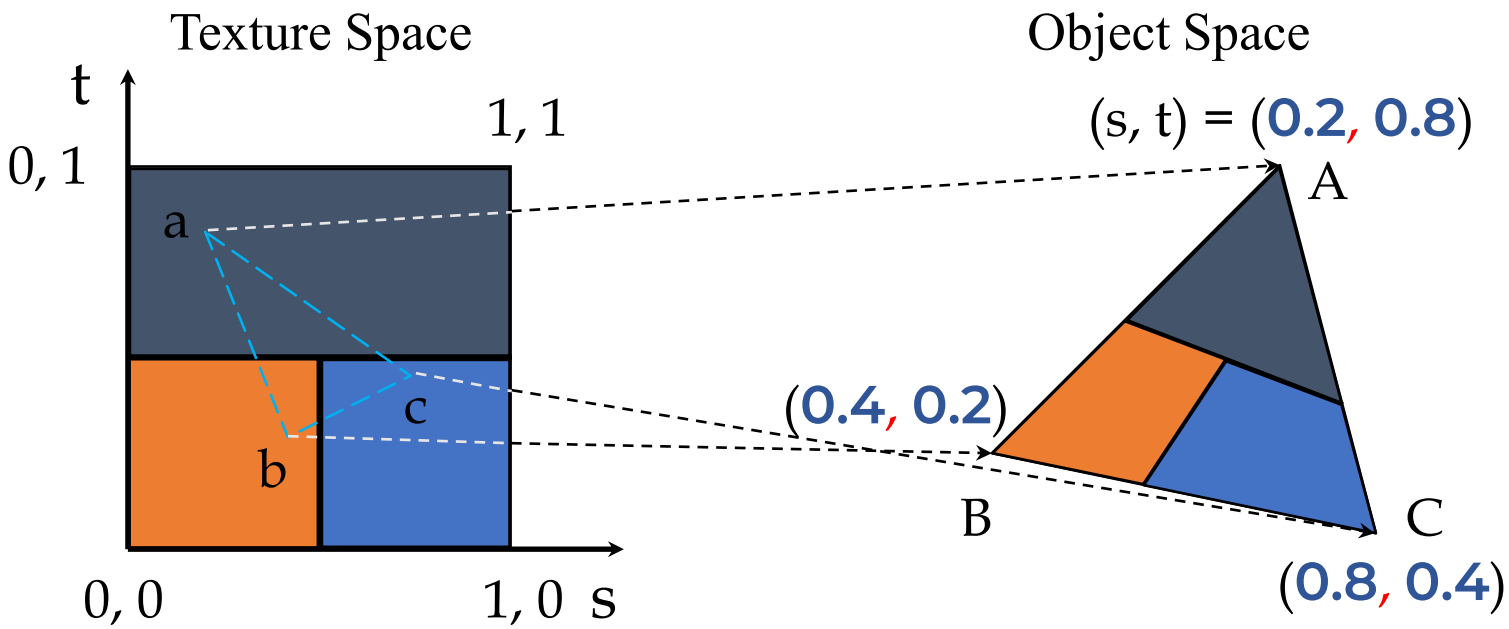
Applying textures

Textures – Basic Strategy

- Three steps to applying a texture
  1. specify the **texture**
     - read or generate image
     - assign to texture
     - enable texturing
  2. **assign** texture **coordinates** to **vertices**
     - Proper **mapping** function is left to **application**
  3. specify texture **parameters**
     - wrapping, filtering

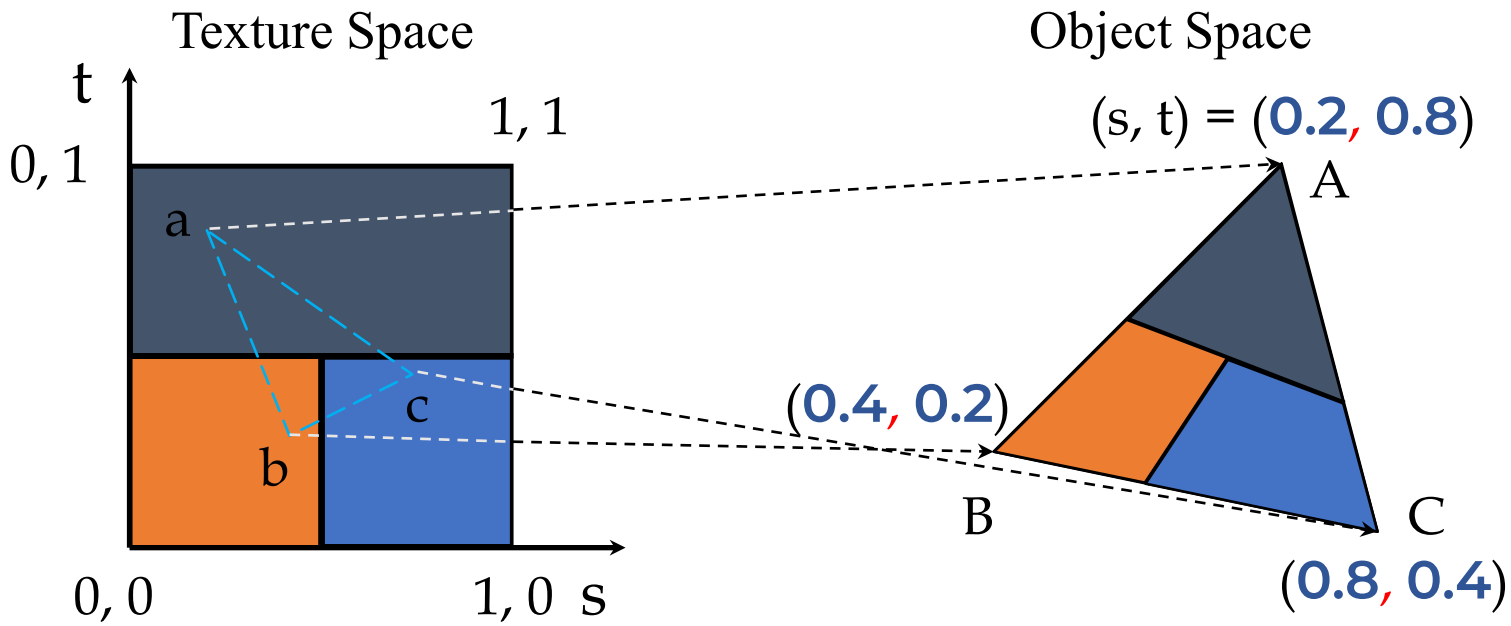
Mapping a Texture

Specify **texture coordinates** as a 2D **vertex attribute**

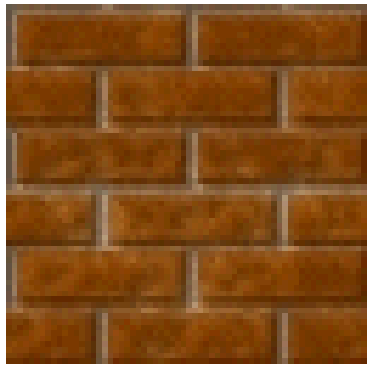


Mapping a Texture

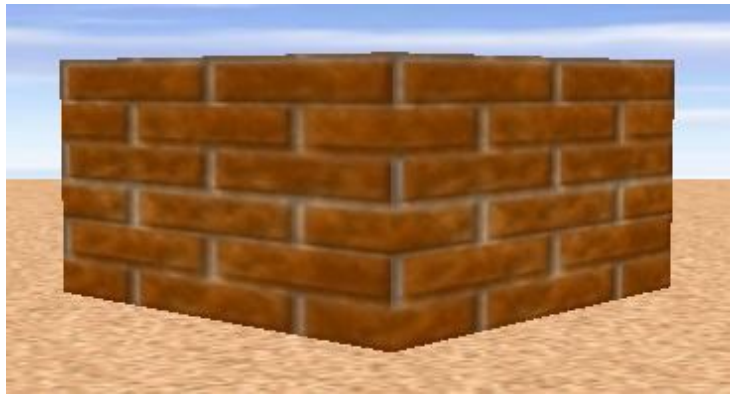
Texture coordinates are **linearly interpolated** across triangles



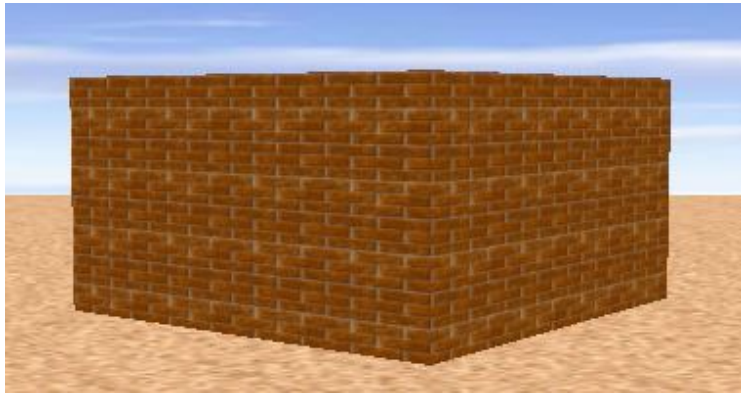
Texture Mapping Style – Tiling



Texture



Without Tiling



With Tiling

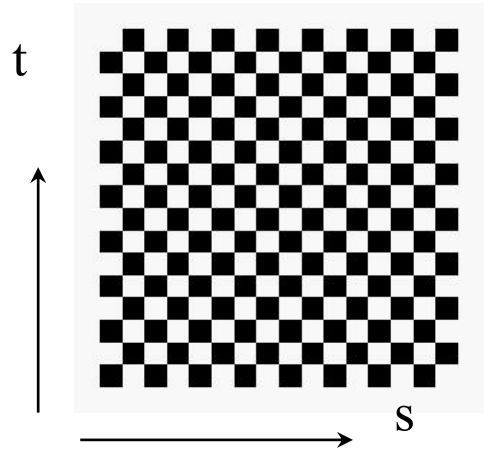
Texture Parameters

How is a texture applied ?

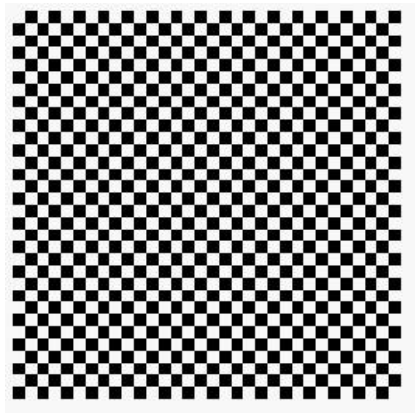
- **Wrapping** parameters determine what happens if s and t are outside the (0,1) range
- **Filter modes** allow us to use area averaging instead of point samples (for magnification and minification)
- **Mipmapping** allows us to use textures at multiple resolutions
- **Environment parameters** determine how texture mapping interacts with **shading**

Wrapping Mode

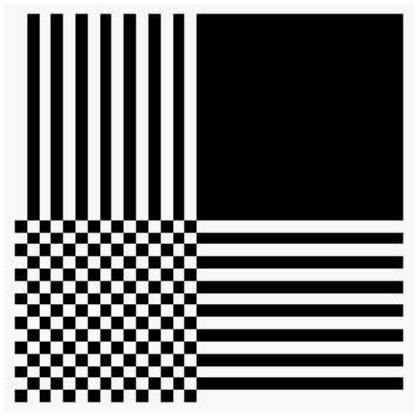
What to do if the texture coordinates go outside [0, 1] ?



texture



REPEAT
wrapping



CLAMP
wrapping

[Ed Angel]

Textures and OpenGL

Applying Textures

Textures can be applied in many ways

- A texture **fully determines color**
- A texture is **modulated** with a computed **color**
- A texture is blended with an **environmental color**

OpenGL – Vertex-shader

- The vertex-shader computes
  - Vertex **positions**
  - Vertex **colors**, if needed

- Usually, it will also output **texture coordinates**

OpenGL – Vertex-shader

```
#version 330 core
layout (location = 0) in vec3 aPos;
layout (location = 1) in vec3 aColor;
layout (location = 2) in vec2 aTexCoord;

out vec3 ourColor;
out vec2 TexCoord;

void main()
{
    gl_Position = vec4(aPos, 1.0);
    ourColor = aColor;
    TexCoord = aTexCoord;
}
```

Texture coordinates defined like any other attribute in our application

OpenGL – Applying textures

- Textures are applied during fragment shading by a **sampler**

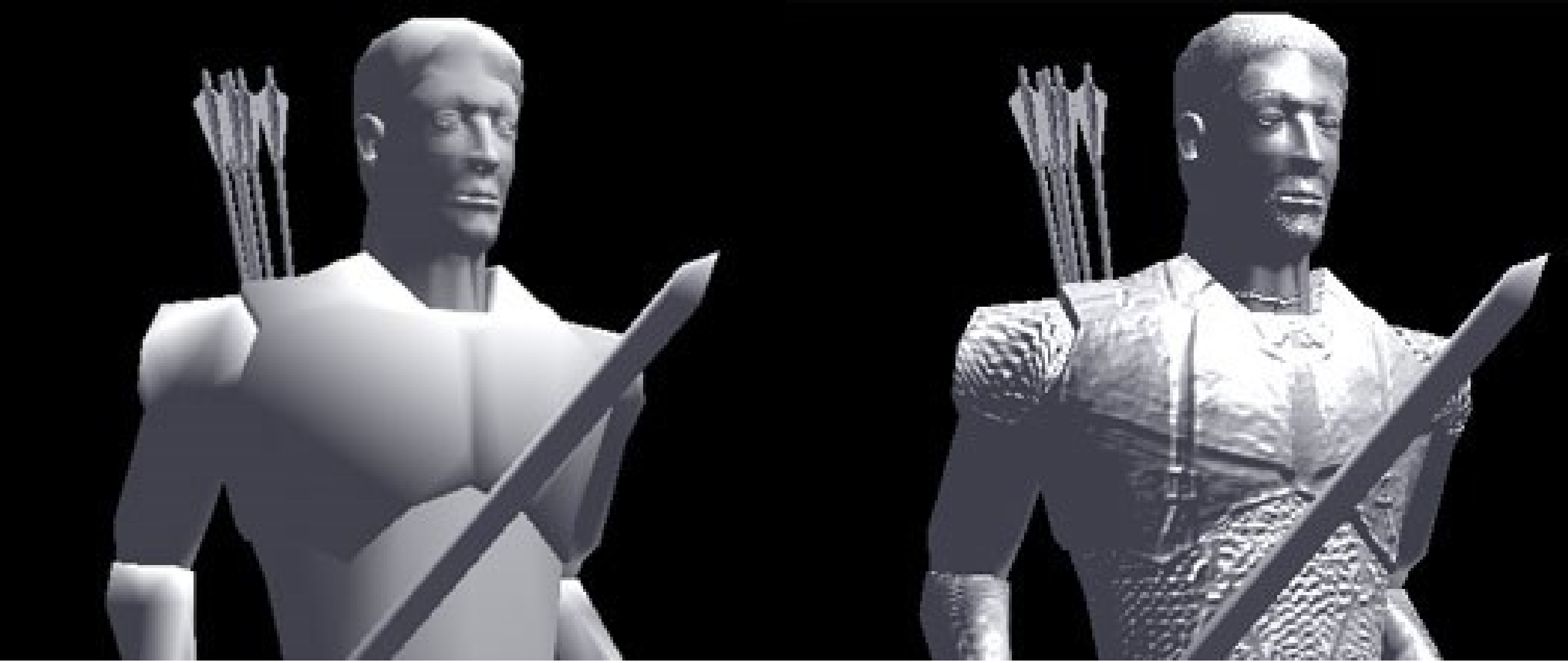
- Samplers return a **texture color** from a texture object

OpenGL – Fragment-shader

```
#version 330 core
out vec4 FragColor;

in vec3 ourColor;
in vec2 TexCoord;

uniform sampler2D ourTexture;

void main()
{
    FragColor = texture(ourTexture, TexCoord) * vec4(ourColor, 1.0);
}
```

these come from the vertex shader

what is this line doing?



Beyond Texture Mapping for Surface Color

Textures Beyond Surface Color

Textures can be used to represent aspects **beyond the surface's color**

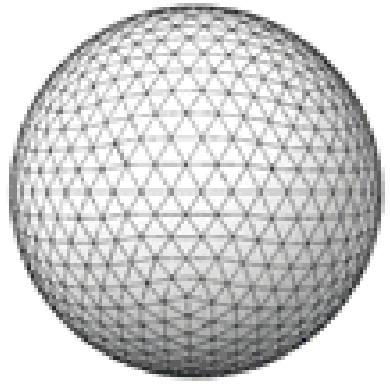
In fact, remember that textures are an array of values, often with **3 components per pixel** (R, G, B)

These values can be used to **store other data**, instead of surface color

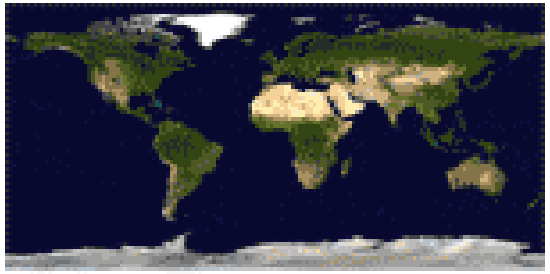
Texture Mapping

Paste the texture on a surface to **add detail** without adding more triangles

BUT, the surface stays **smoothly flat**...



Sphere with no texture



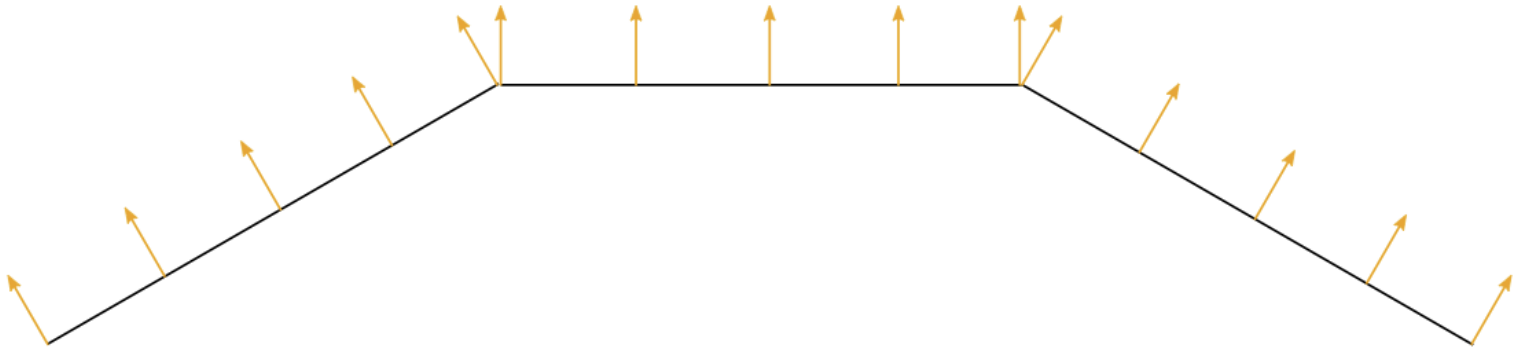
Texture image

Sphere with texture

Remember, remember, face Normals in November

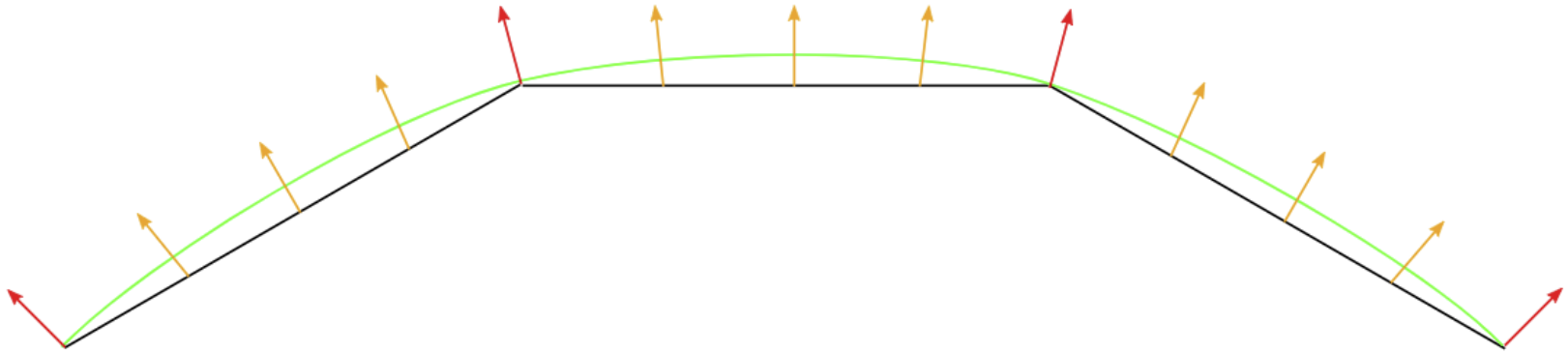
Before we move along to more interesting business…

Remember the role of normals for illumination computation



Remember, remember, face Normals in November

- And what kind of shading does this enable?

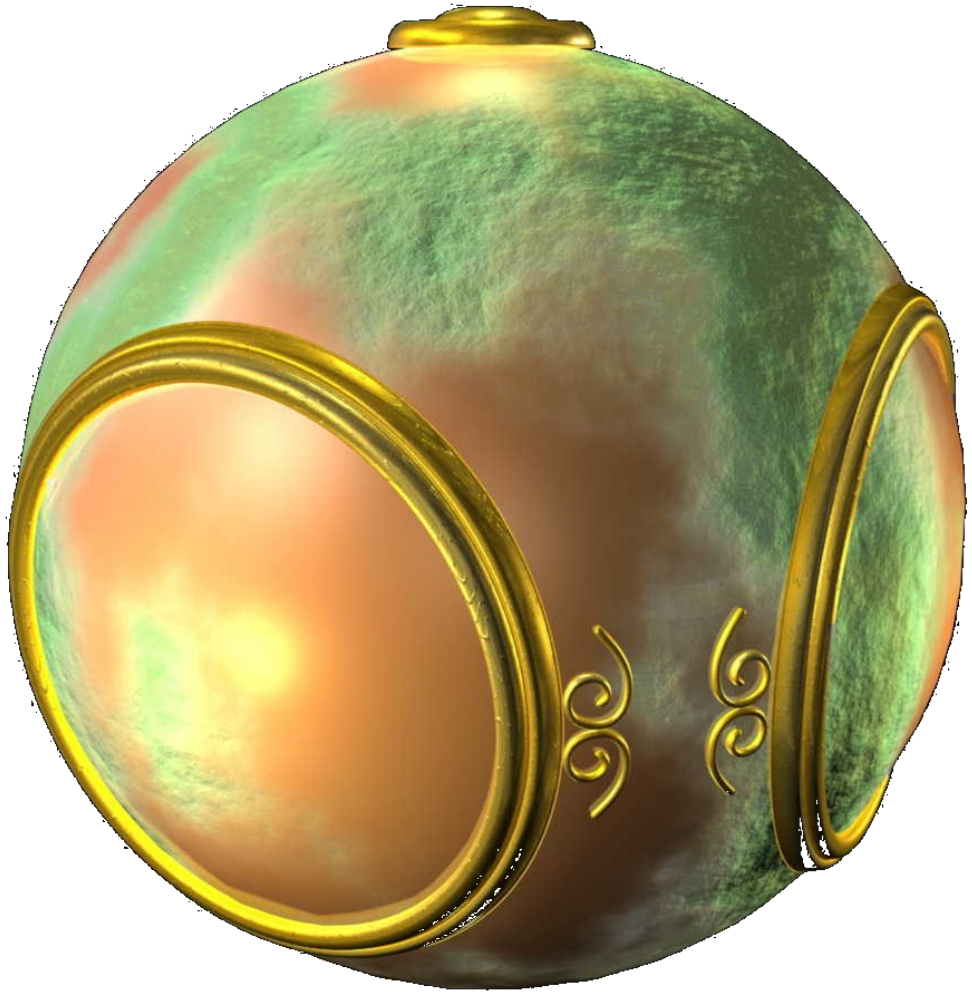


Bump Mapping

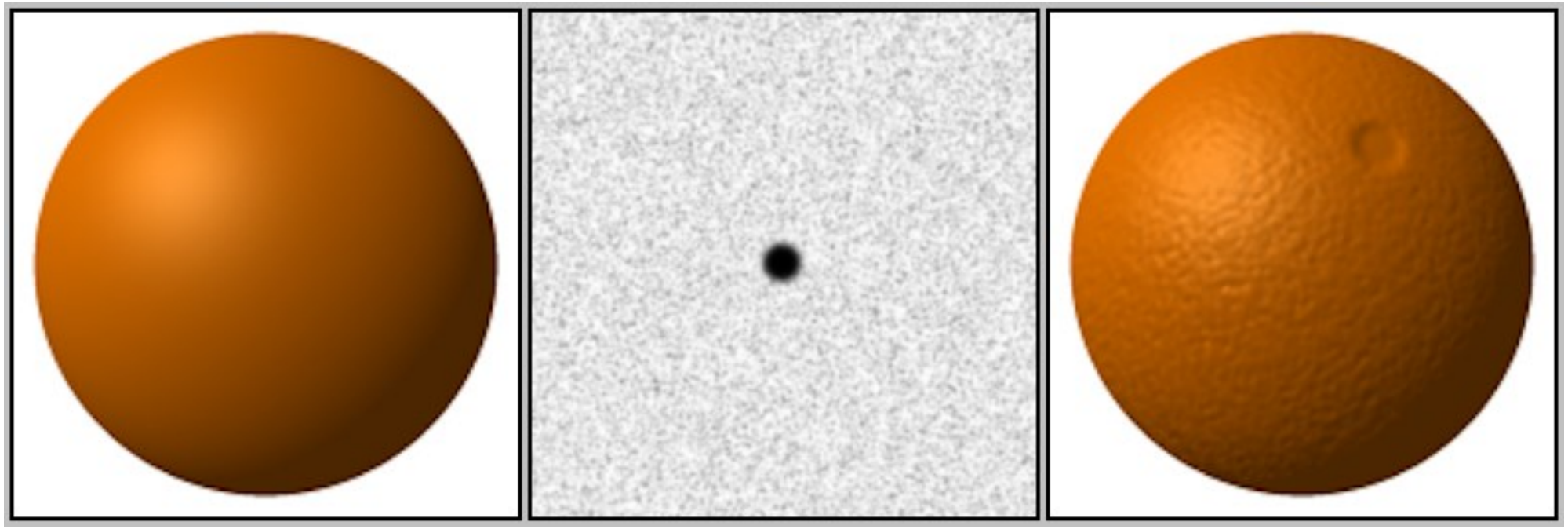
General name given to the technique of adding apparent fine detail to a surface through a texture

Blinn, 1978

Works by providing a map of **small displacements of the surface** and these are considered to affect the original normals



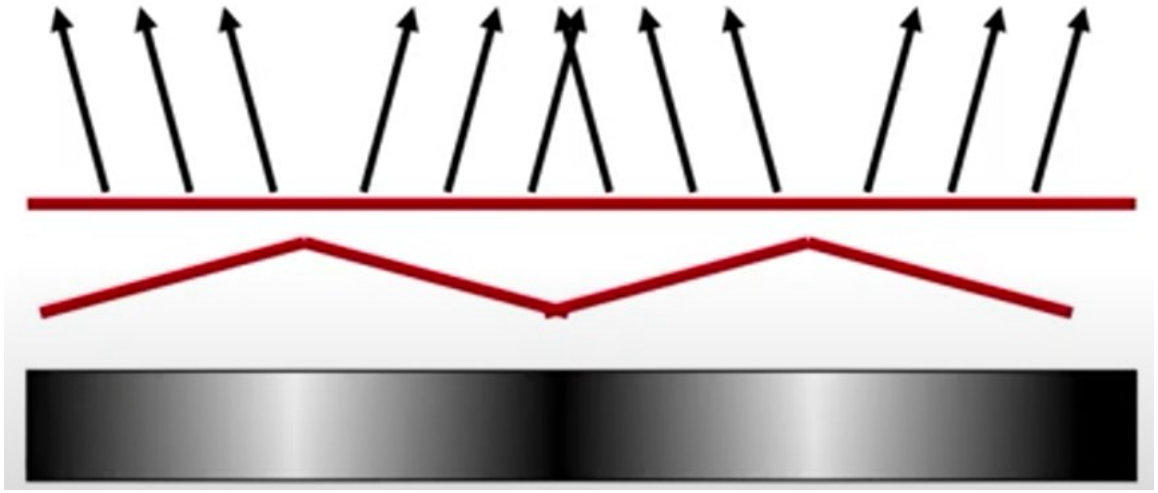
Bump Mapping



Original Rendering of Geometry

Bump Mapping

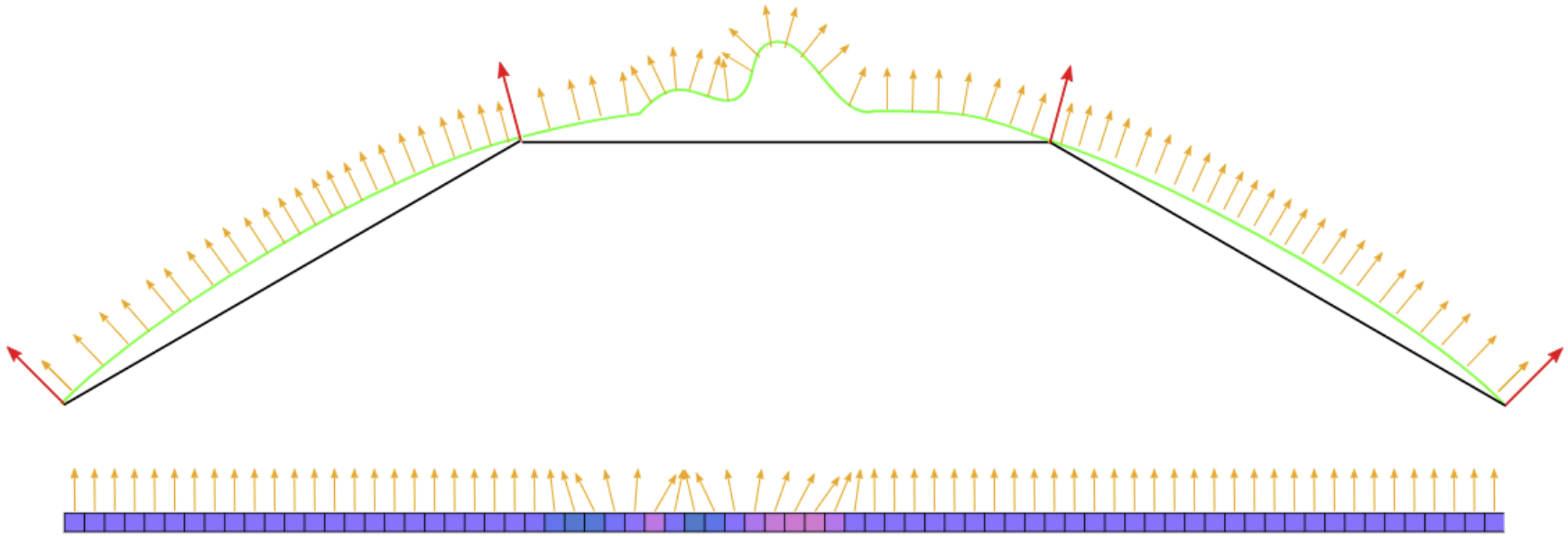
Bump Mapping



Normal Mapping

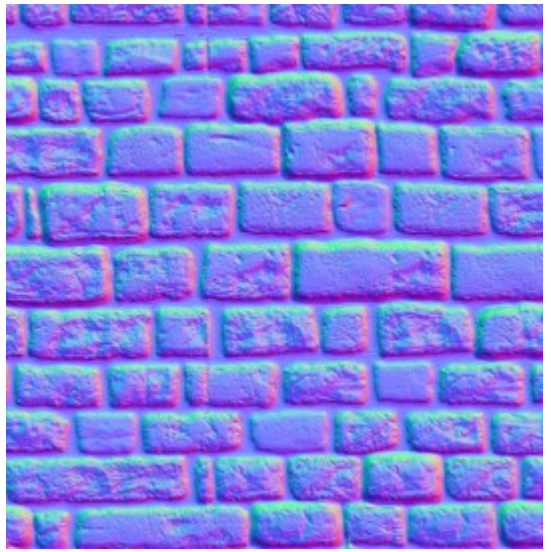
Provides a new set of normals for the surface

Normals are encoded in the RGB components of an image



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

Normal Mapping

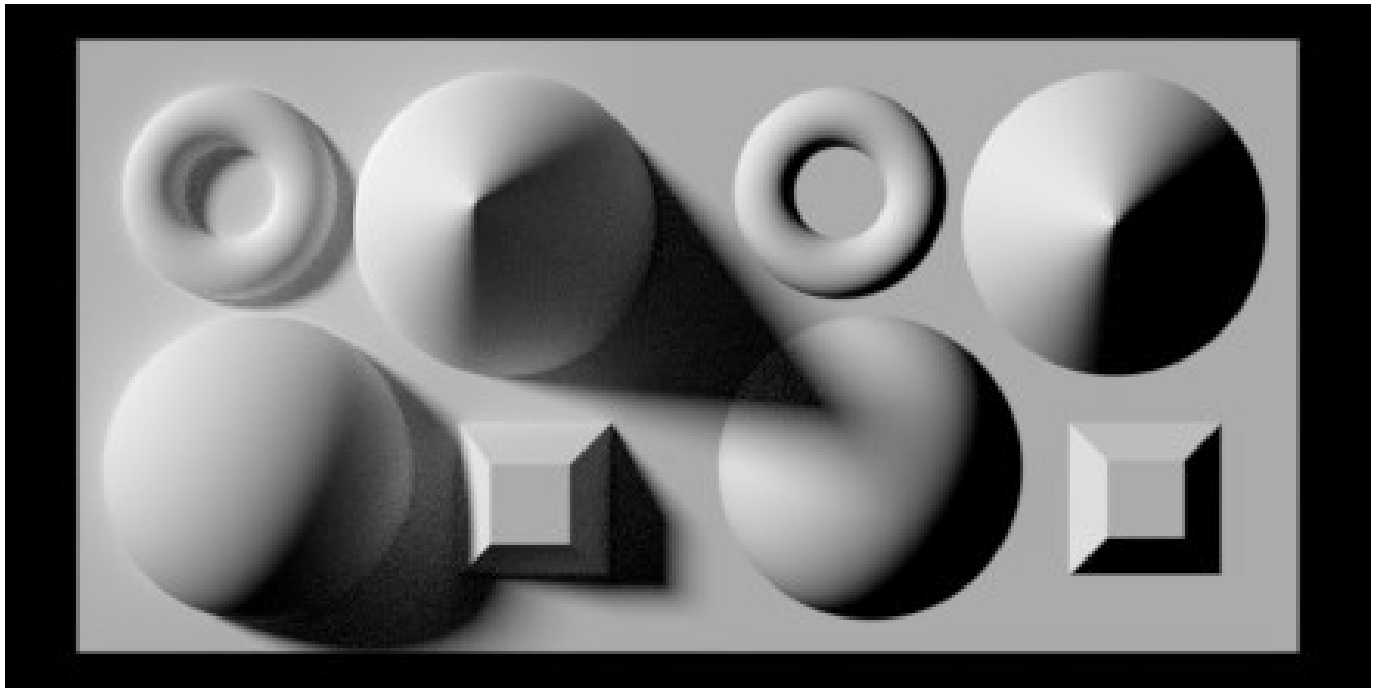


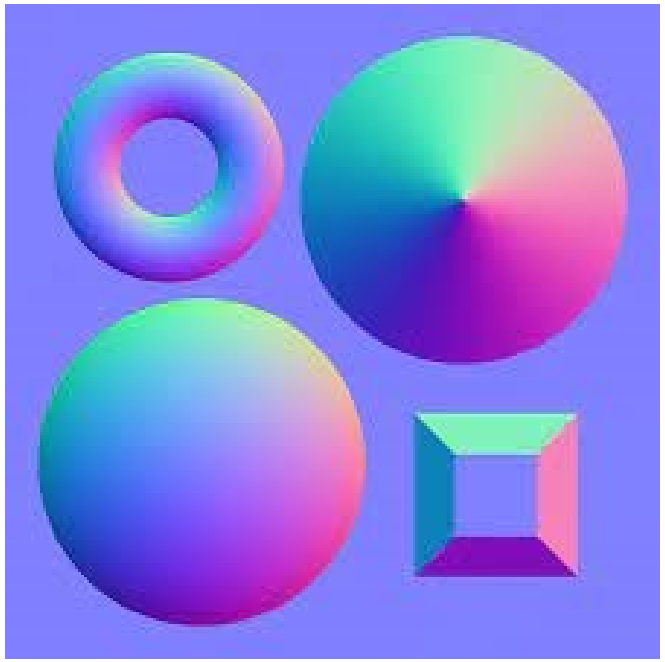
https://www.raywenderlich.com/

Normal Maps are bluish… why?

- Just **out of curiosity**, can you figure out why normal maps are typically bluish?
- RGB components **MUST** be positive, but the components of a normal vector can range from -1 to 1
- So… **-1 to 1** needs to be converted to **0 to 1**…
- …

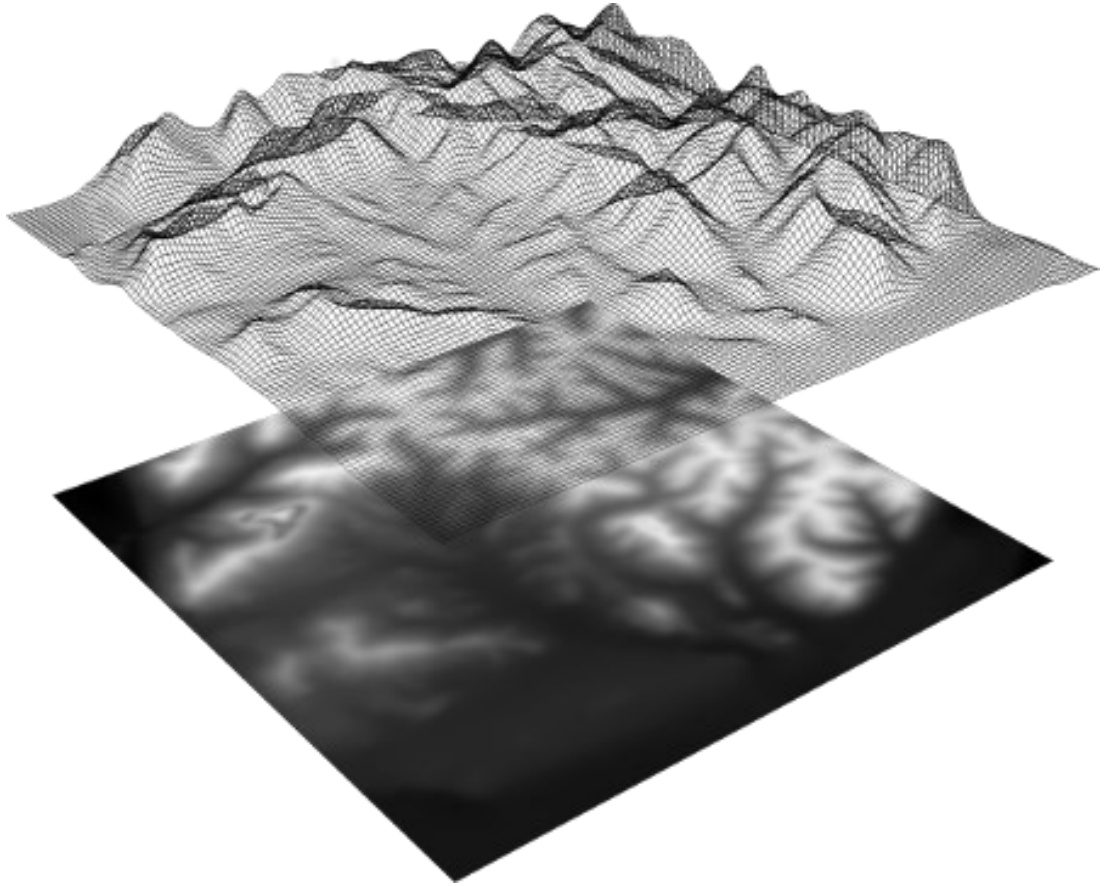
Normal Mapping



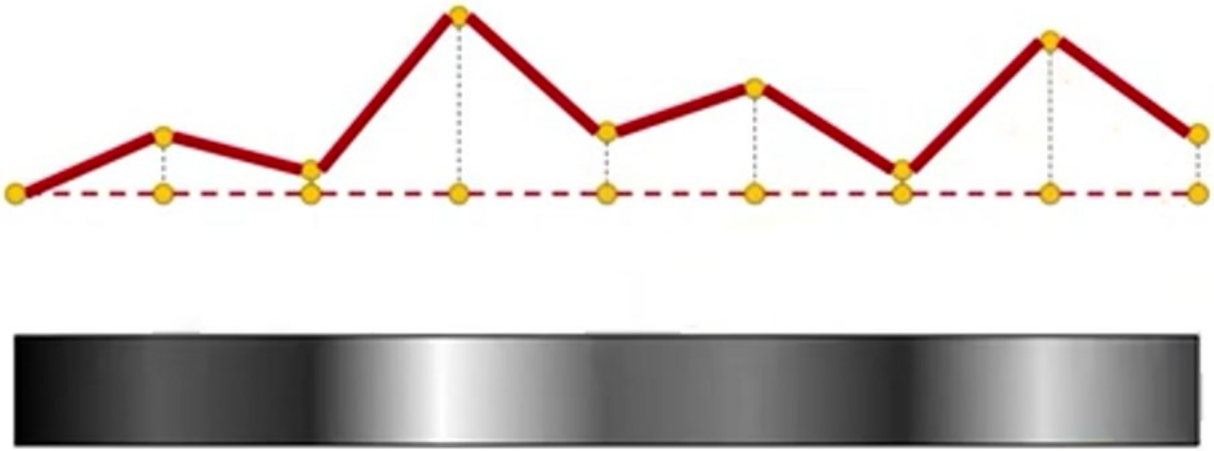


Displacement/Height Maps

- **Displacement maps** actually affect the geometry of the surface
- They are called displacement maps when they are used to add small detail in surfaces
- They are called **height maps** when they are applied in large displacements, e.g., of terrain



Displacement Maps



Bump Maps vs Displacement Maps

Notice that bump/normal maps do not affect the silhouette

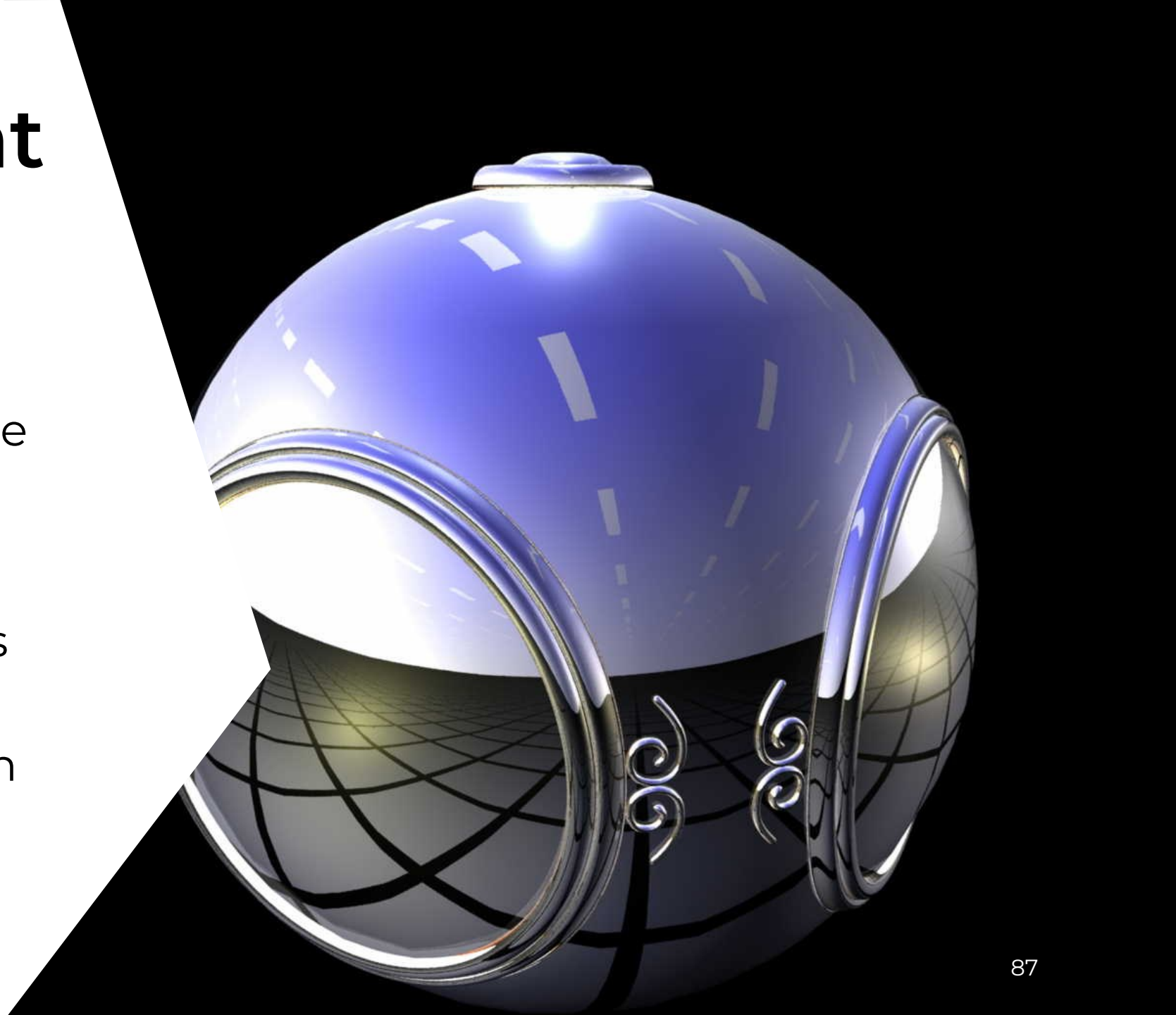
https://www.cgdirector.com/normal-vs-displacement-vs-bump-maps/

Environment Mapping

Global Illumination is very **resource consuming** to compute

This makes it hard to use in real-time settings, in many cases

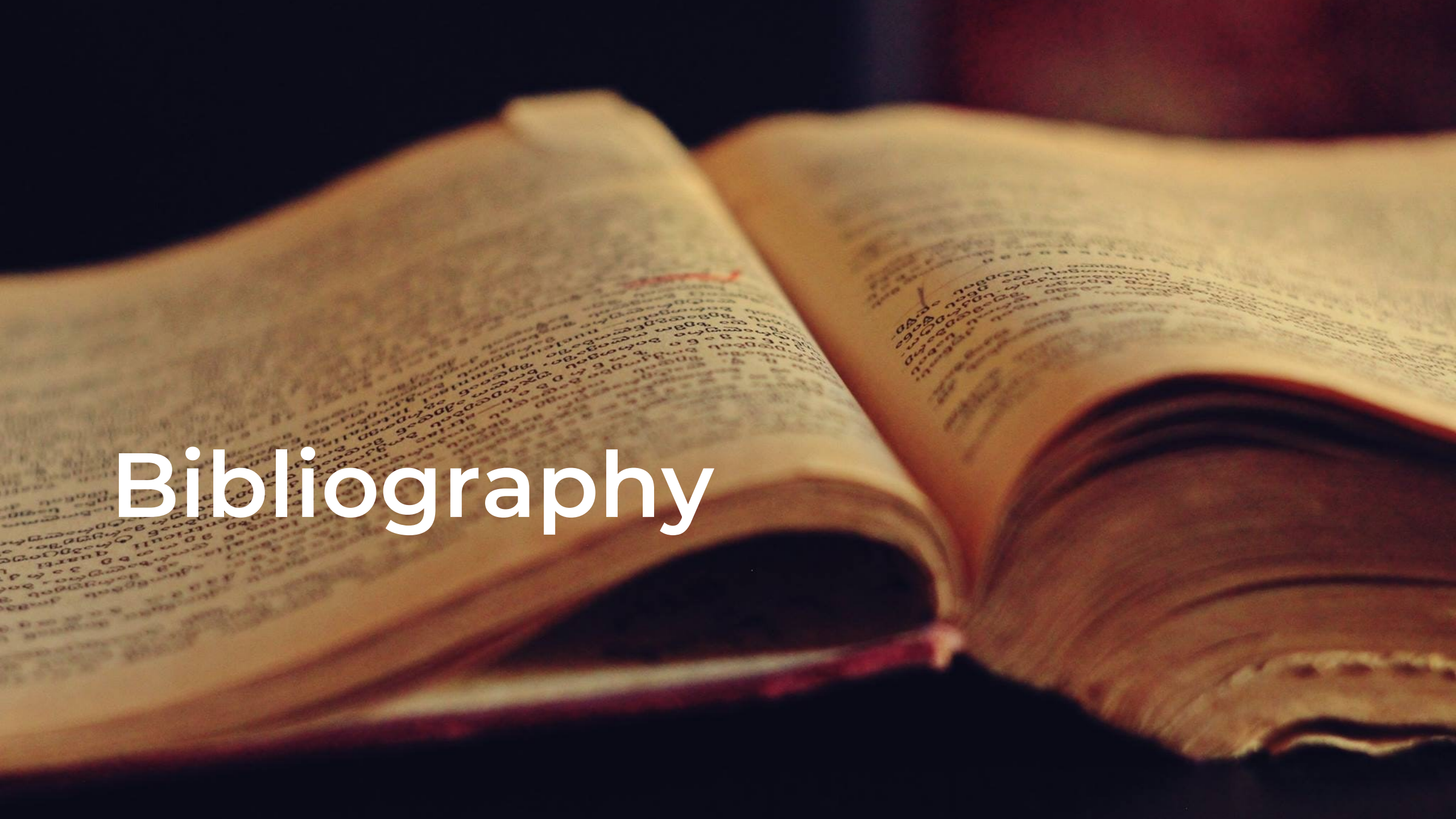
Environment maps can be considered to simulate reflection from environment



[http://www.okino.com]

Textures – Simulating Ray-Tracing

- Increased realism !!
  - 11 light sources + 25 texture maps
- “Baked” lighting



Bibliography

Bibliography

- Steve Marschner, Peter Shirley, “Texture Mapping”, chapter 11 in “Fundamentals of Computer Graphcs”, 4th ed, 2018 https://learning.oreilly.com/library/view/fundamentals-of-computer/9781482229417/K22616_C011.xhtml
- Edward Angel, Dave Shreiner, “Interactive Computer Graphics – A Top-Down Approach with Shader-based OpenGL, 6th ed., 2012 (sec. 7.4 + sec. 7.5)

sss_cv_tex_v4

Acknowledgments

Some ideas and figures have been taken from slides made available by Ed , Andy van Dam, and Joaquim Madeira