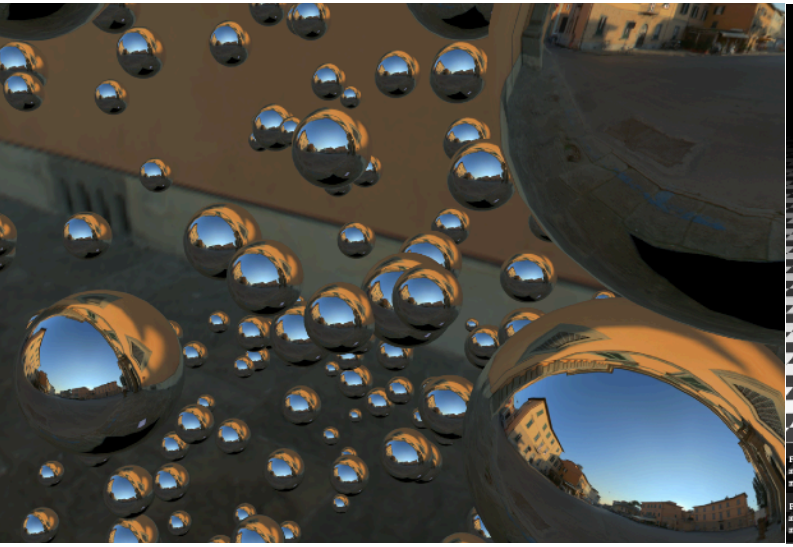
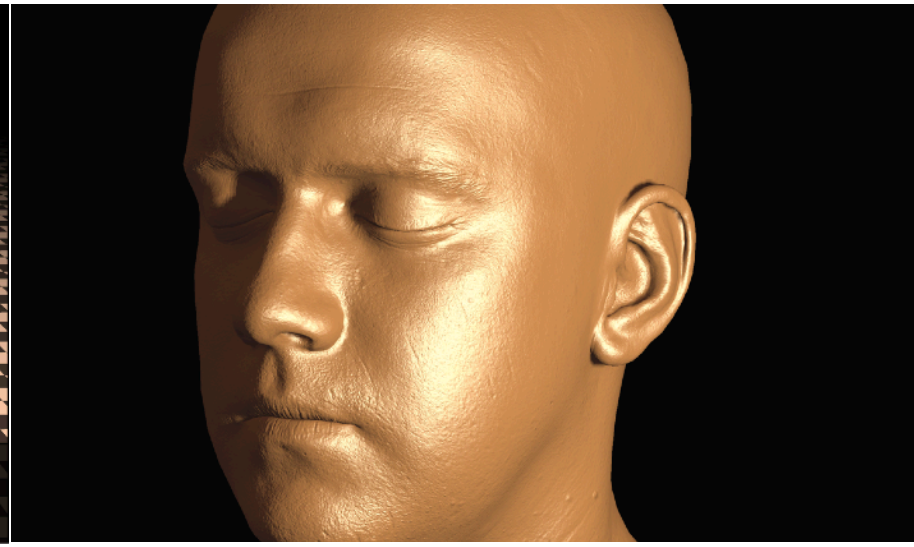
Visual Computing I:

Interactive Computer Graphics and Vision





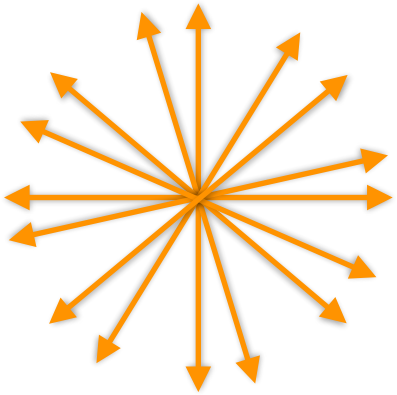
Texture Mapping

Stefanie Zollmann and Tobias Langlotz

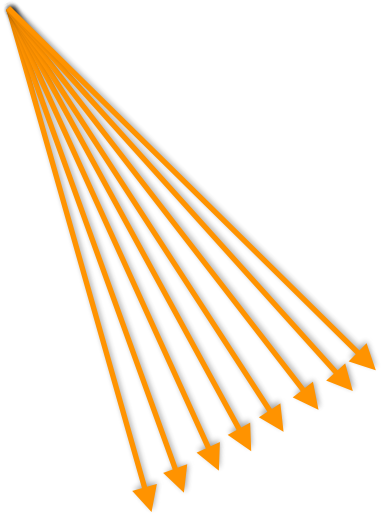
Recap: Why Illumination?

- Illumination is important for perception and understanding of 3D scenes
- Has visual cues for humans
- Provides information about
- Positioning of light sources
- Characteristics of light sources
  - Materials
  - Viewpoint

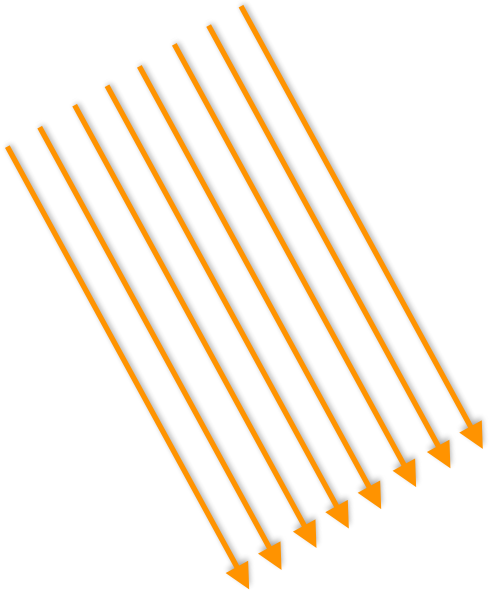
Recap: Light Sources



Point Light

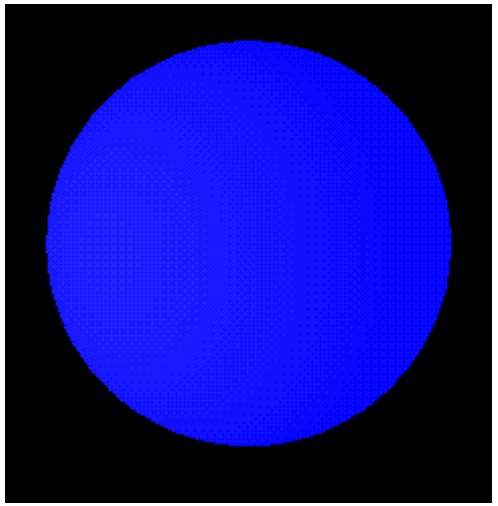


Spot Light

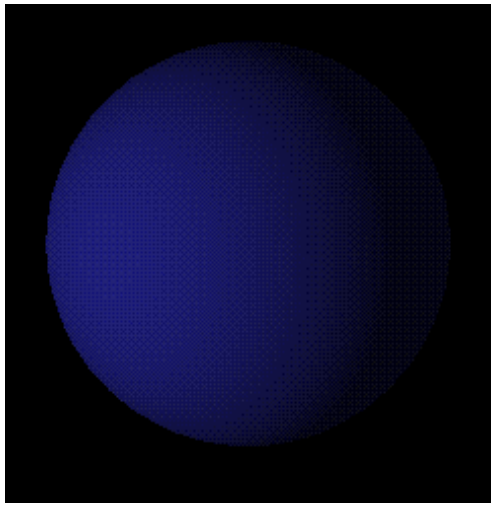


Directional Light

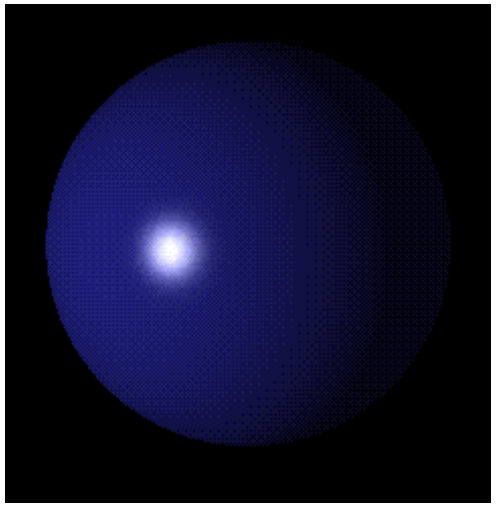
Recap: Reflection model



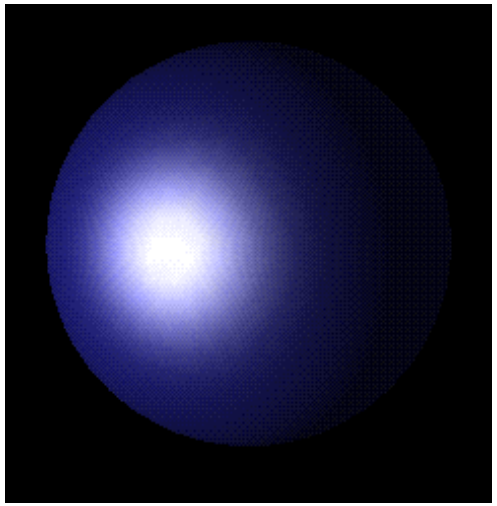
Ambient



Diffuse



Specular



Combined

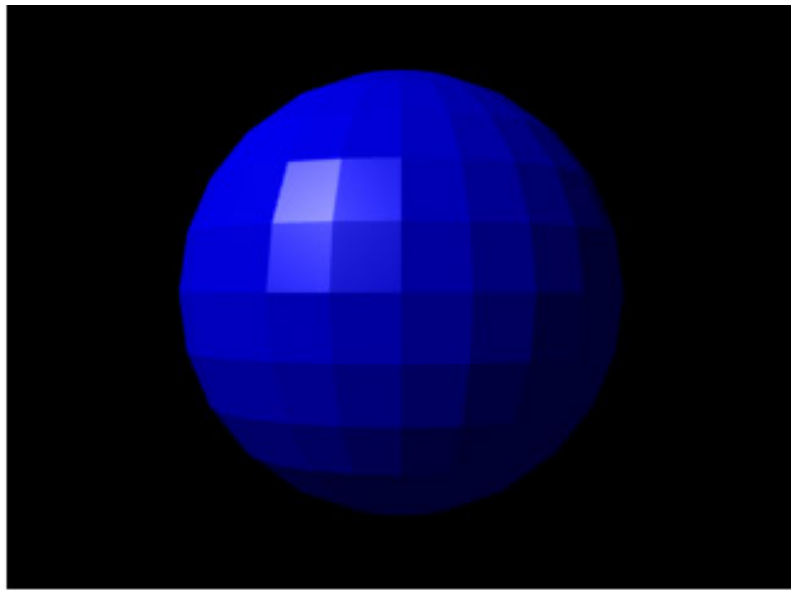
Recap: Phong Reflection Model

Illumination = $I_{ambient} = I_a k_a$ $I_{diffuse} = I_d k_d (\hat{N} \cdot \hat{L})$ $I_{specular} = I_s k_s (\hat{R} \cdot \hat{V})^{n_s}$

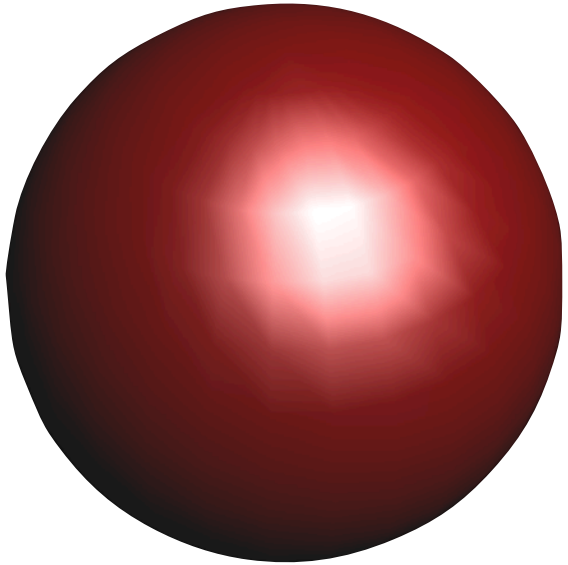
Illumination = $I_a k_a + I_d k_d (\hat{N} \cdot \hat{L}) + I_s k_s (\hat{R} \cdot \hat{V})^{n_s}$

Recap: Shading Models

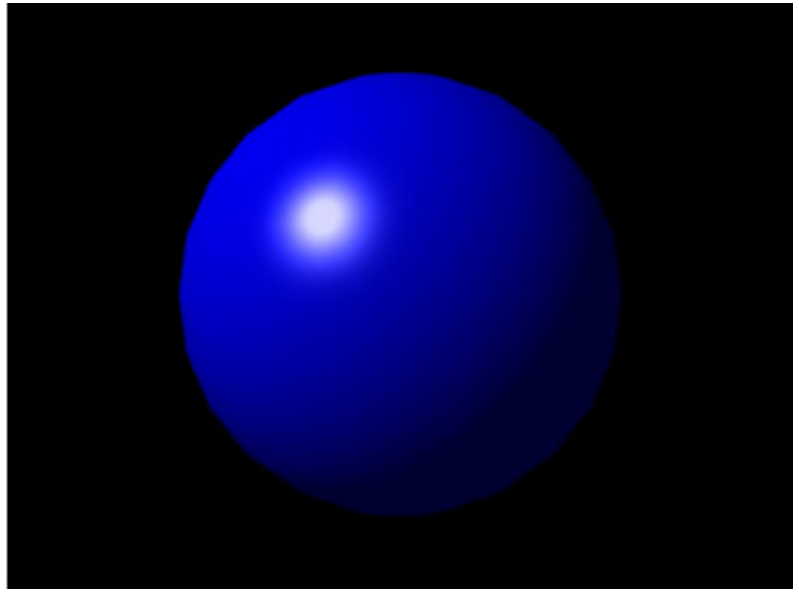
- Shading model determines on which shader stage and with which quality lighting for triangles is calculated



Flat Shading



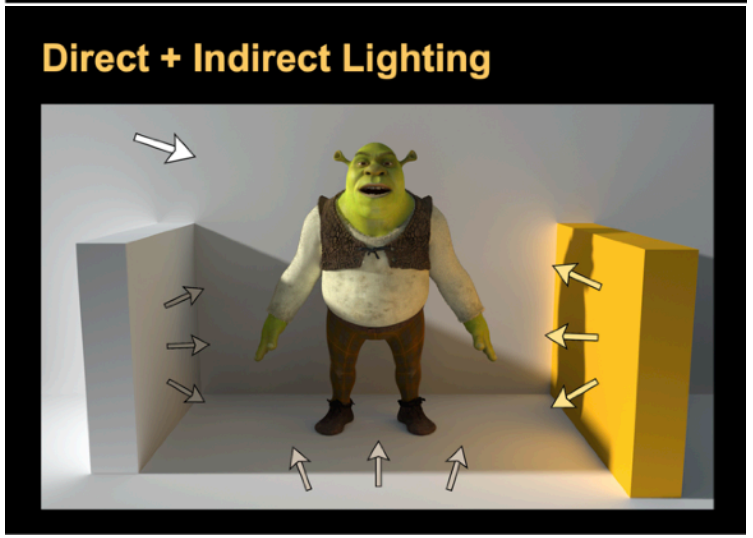
Gouraud Shading



Phong Shading

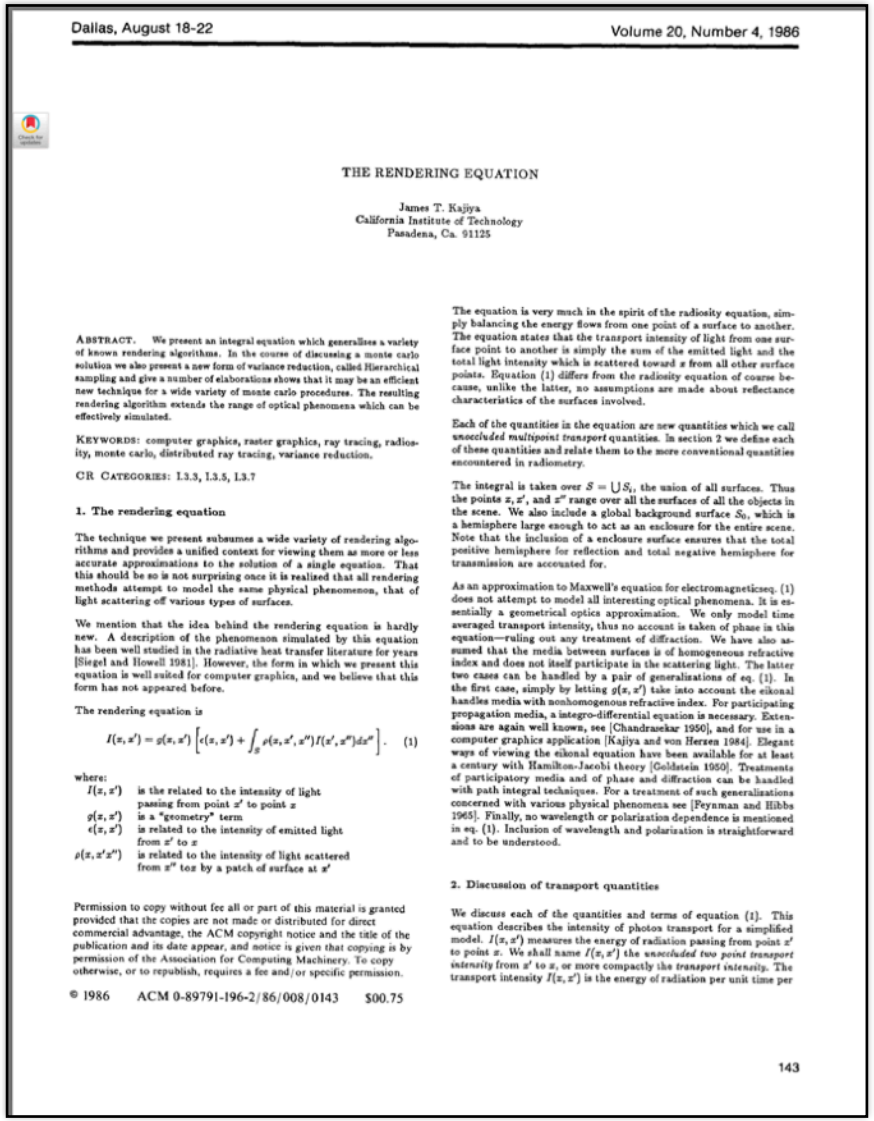
Global Illumination

- Direct and indirect lighting
- Option in Unity: Real-time Global Illumination
  - Shadows
  - Inter-object reflections
  - Rendering equation
  - Radiosity



SIGGRAPH 2010 Course: “Global Illumination Across Industries

Rendering Equation



Dallas, August 18-22 Volume 20, Number 4, 1986

THE RENDERING EQUATION

James T. Kajiya
California Institute of Technology
Pasadena, Ca. 91125

ABSTRACT. We present an integral equation which generalizes a variety of known rendering algorithms. In the course of discussing a monte carlo solution we also present a new form of variance reduction, called Hierarchical sampling and give a number of elaborations shows that it may be an efficient new technique for a wide variety of monte carlo procedures. The resulting rendering algorithm extends the range of optical phenomena which can be effectively simulated.

KEYWORDS: computer graphics, raster graphics, ray tracing, radiosity, monte carlo, distributed ray tracing, variance reduction.

CR CATEGORIES: I.3.3, I.3.5, I.3.7

1. The rendering equation

The technique we present subsumes a wide variety of rendering algorithms and provides a unified context for viewing them as more or less accurate approximations to the solution of a single equation. That this should be so is not surprising once it is realized that all rendering methods attempt to model the same physical phenomenon, that of light scattering off various types of surfaces.

We mention that the idea behind the rendering equation is hardly new. A description of the phenomenon simulated by this equation has been well studied in the radiative heat transfer literature for years [Siegel and Howell 1981]. However, the form in which we present this equation is well suited for computer graphics, and we believe that this form has not appeared before.

The rendering equation is

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]. \quad (1)$$

where:

$I(x,x')$	is the related to the intensity of light passing from point x' to point x
$g(x,x')$	is a "geometry" term
$\epsilon(x,x')$	is related to the intensity of emitted light from x' to x
$\rho(x,x'x'')$	is related to the intensity of light scattered from x'' toz by a patch of surface at x'

The equation is very much in the spirit of the radiosity equation, simply balancing the energy flows from one point of a surface to another. The equation states that the transport intensity of light from one surface point to another is simply the sum of the emitted light and the total light intensity which is scattered toward x from all other surface points. Equation (1) differs from the radiosity equation of course because, unlike the latter, no assumptions are made about reflectance characteristics of the surfaces involved.

Each of the quantities in the equation are new quantities which we call *unoccluded multipoint transport* quantities. In section 2 we define each of these quantities and relate them to the more conventional quantities encountered in radiometry.

The integral is taken over $S = \bigcup S_i$, the union of all surfaces. Thus the points x, x', and x'' range over all the surfaces of all the objects in the scene. We also include a global background surface S_0, which is a hemisphere large enough to act as an enclosure for the entire scene. Note that the inclusion of a enclosure surface ensures that the total positive hemisphere for reflection and total negative hemisphere for transmission are accounted for.

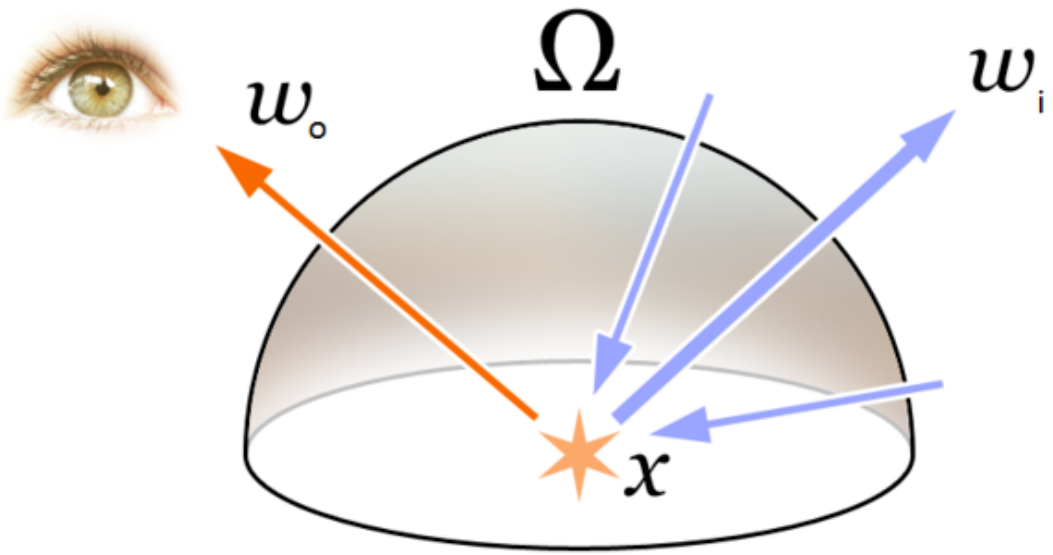
As an approximation to Maxwell's equation for electromagneticseq. (1) does not attempt to model all interesting optical phenomena. It is essentially a geometrical optics approximation. We only model time averaged transport intensity, thus no account is taken of phase in this equation—ruling out any treatment of diffraction. We have also assumed that the media between surfaces is of homogeneous refractive index and does not itself participate in the scattering light. The latter two cases can be handled by a pair of generalizations of eq. (1). In the first case, simply by letting $g(x,x')$ take into account the eikonal handles media with nonhomogenous refractive index. For participating propagation media, a integro-differential equation is necessary. Extensions are again well known, see [Chandraekar 1950], and for use in a computer graphics application [Kajiya and von Herzen 1984]. Elegant ways of viewing the eikonal equation have been available for at least a century with Hamilton-Jacobi theory [Goldstein 1950]. Treatments of participatory media and of phase and diffraction can be handled with path integral techniques. For a treatment of such generalizations concerned with various physical phenomena see [Feynman and Hibbs 1965]. Finally, no wavelength or polarization dependence is mentioned in eq. (1). Inclusion of wavelength and polarization is straightforward and to be understood.

2. Discussion of transport quantities

We discuss each of the quantities and terms of equation (1). This equation describes the intensity of photon transport for a simplified model. $I(x,x')$ measures the energy of radiation passing from point x' to point x. We shall name $I(x,x')$ the *unoccluded two point transport intensity* from x' to x, or more compactly the *transport intensity*. The transport intensity $I(x,x')$ is the energy of radiation per unit time per

Kajiya 1986

Rendering Equation



https://commons.wikimedia.org/wiki/File:Rendering_eq.png

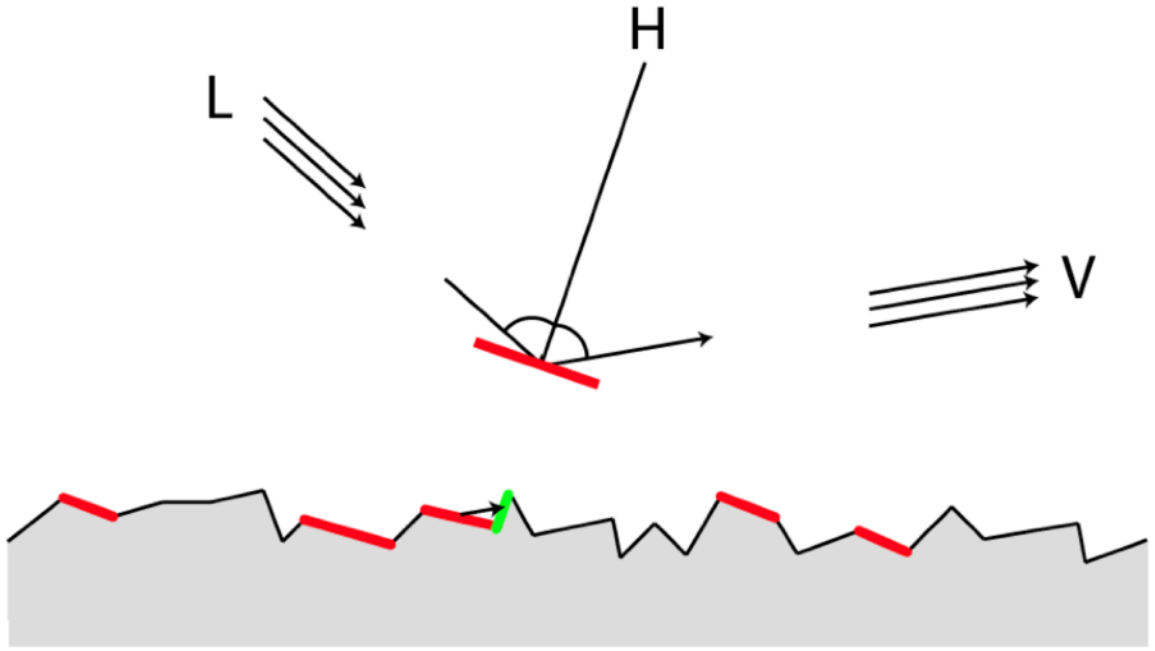
Kajiya 1986

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t) L_i(\mathbf{x}, \omega_i, \lambda, t)(\omega_i \cdot \mathbf{n}) \, \mathrm{d}\, \omega_i$$

Spectral radiance	Emitted spectral radiance	BRDF	Spectral radiance of wavelength (λ) coming inward toward point x
outgoing radiance (what we see)	emitted light (e.g., from a lamp or emissive material)	(bidirectional reflectance distribution function)	incoming radiance from direction

Physically Based Rendering (PBR)

- Approximation of Kajiya's equation
- Render materials based on real-world light behaviour, not ad-hoc artistic parameters.
- Core principles:
  - Energy conservation:
    - Light reflected ≤ light received
  - Microfacet model -> surfaces have tiny roughness details that shape reflections
  - Metallic–Roughness workflow -> materials defined by metalness and roughness, not specular color

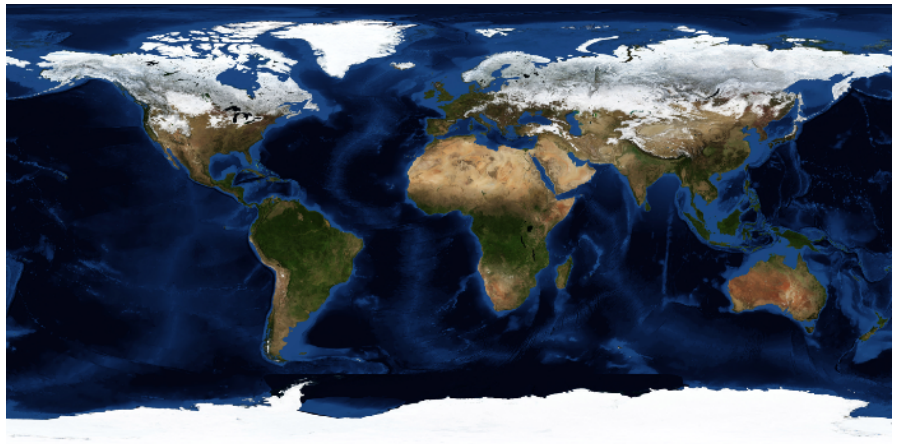


Material Settings Across 3D Model File Formats

- Base color / diffuse color
- Specular / metallic
- Roughness / glossiness
- Normal / bump maps
- Transparency / opacity
- Emission / self-illumination
- Texture references (image maps, UVs)

```
newmtl EarthMaterial

Ka 0.640000 0.640000 0.640000
Kd 0.640000 0.640000 0.640000
Ks 0.050000 0.050000 0.050000
Ns 30.0000
d 0.5
illum 2
map_Kd ColorMap.bmp
```

Example MTL

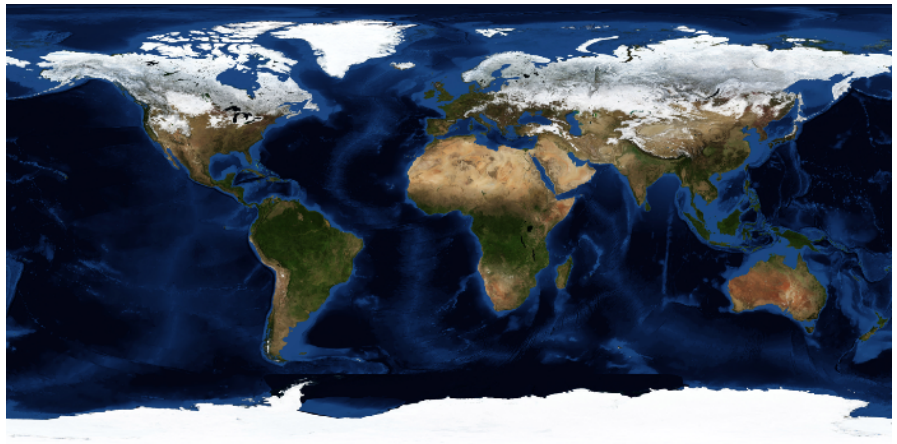


Material Definitions

- Material Template Library (MTL) can be used to define material settings
- Defines ambient (Ka), diffuse (Kd), specular (Ks) colours and the specular exponent (Ns)
- Also allows to define opacity (d) - 1.0 means fully opaque
- Set texture maps (map_Kd)

```
newmtl EarthMaterial

Ka 0.640000 0.640000 0.640000
Kd 0.640000 0.640000 0.640000
Ks 0.050000 0.050000 0.050000
Ns 30.0000
d 0.5
illum 2
map_Kd ColorMap.bmp
```

Example MTL



Material Definitions in GLTF

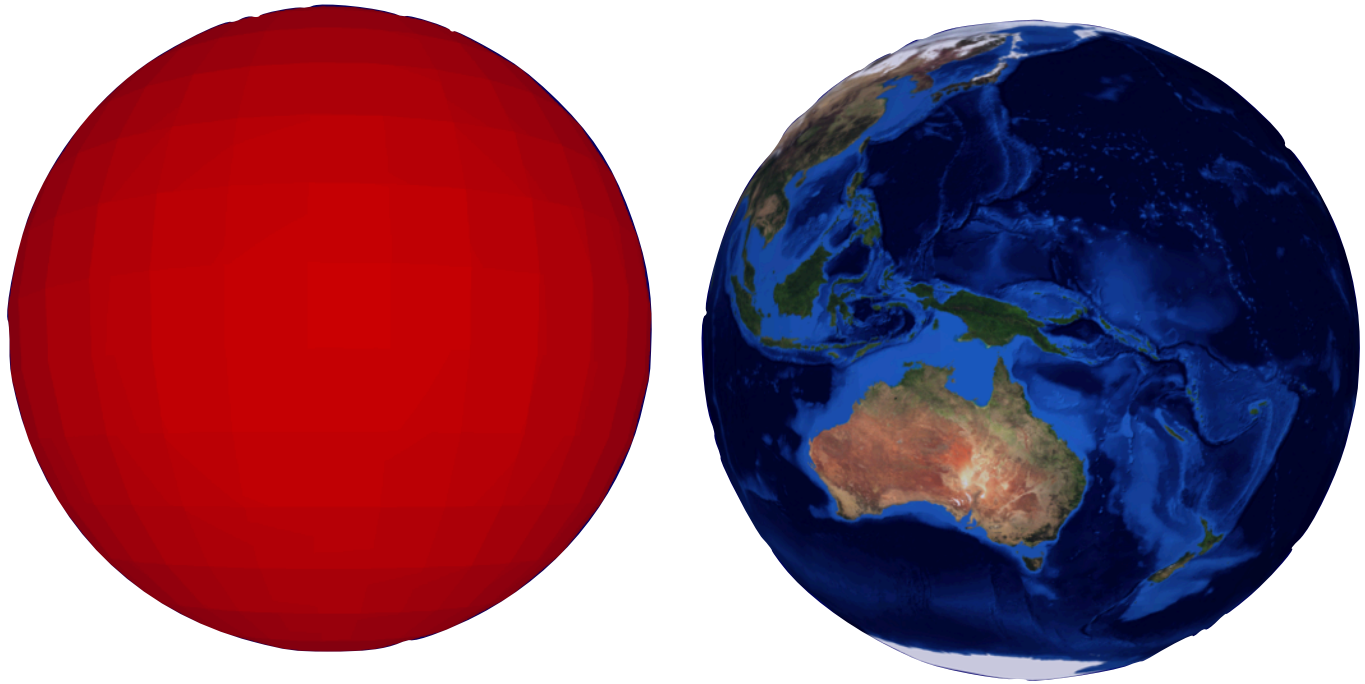
- Uses physically based rendering (PBR) parameters
- Separates metallic and roughness for realistic reflections
- Texture indices link to binary or image data (e.g., .glb or .bin)
- Enables consistent results across modern engines (WebGL, Unity, Unreal, etc.)

```
"materials": [
  {
    "name": "Bronze",
    "pbrMetallicRoughness": {
      "baseColorTexture": { "index": 0 },
      "baseColorFactor": [0.714, 0.428, 0.181, 1.0],
      "metallicFactor": 0.9,
      "roughnessFactor": 0.4
    },
    "normalTexture": { "index": 1 },
    "emissiveFactor": [0.0, 0.0, 0.0],
    "alphaMode": "OPAQUE"
  }
]
```

Example GLTF

Why Texturing?

- Enhance visual appearance of plain surfaces and objects by applying texture details
- Two spheres with identical geometry
  - The earth model appears to involve more complex materials for the objects in the scene
  - Detail is expensive to model
  - Often the surface of an object is viewed only from a distance
  - We can ‘paint’ the detail on the object instead of increasing the complexity of the object



Another texture mapping example

Image from the Blender Guru tutorial at http://www.blenderguru.com/tutorials/introduction-to-texture-nodes/

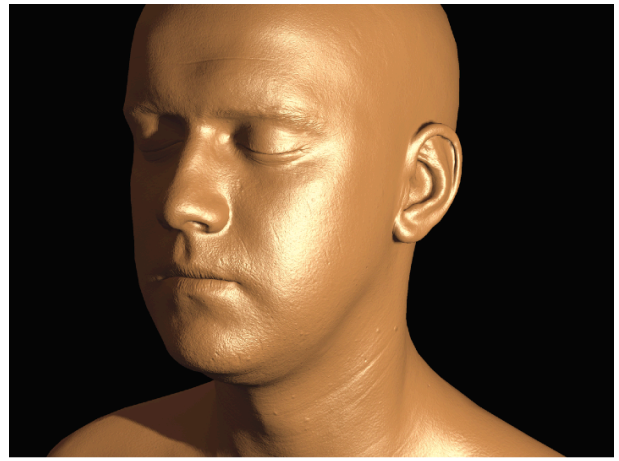
Terminology

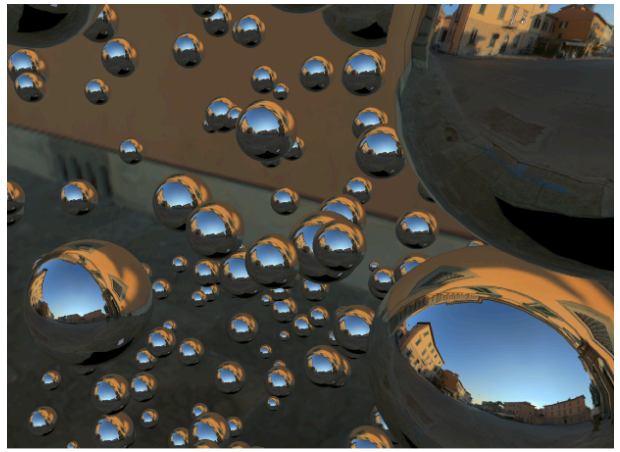
- Texture: Array of values:
  - 2D (most common), 1D, 3D
  - Colour, depth, alpha, normals
- Texels (texture elements): Elements of the texture map
- Texture coordinates:
  - Textures are defined in texture (u, v) coordinates
  - Coordinates u and v ranging from 0.0 - 1.0
  - Think of it as a continuous image

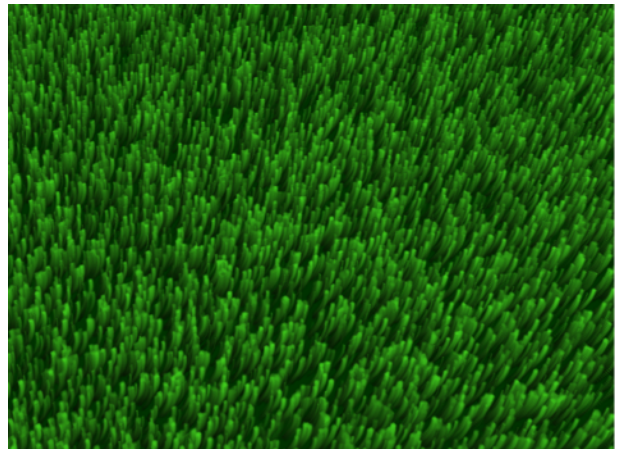
Texturing

Simulation of different material properties:

- Color
- Reflection
- Gloss
- Transparency
- Bumpiness







HOW?

- When drawing fragments, we use the local colour from the texture image
- Usually an image
- Can hold arbitrary information:
  - Interpreted by shader
  - Basis for most advanced rendering effects
- We need a coordinate system on the surface to find the right part of the texture

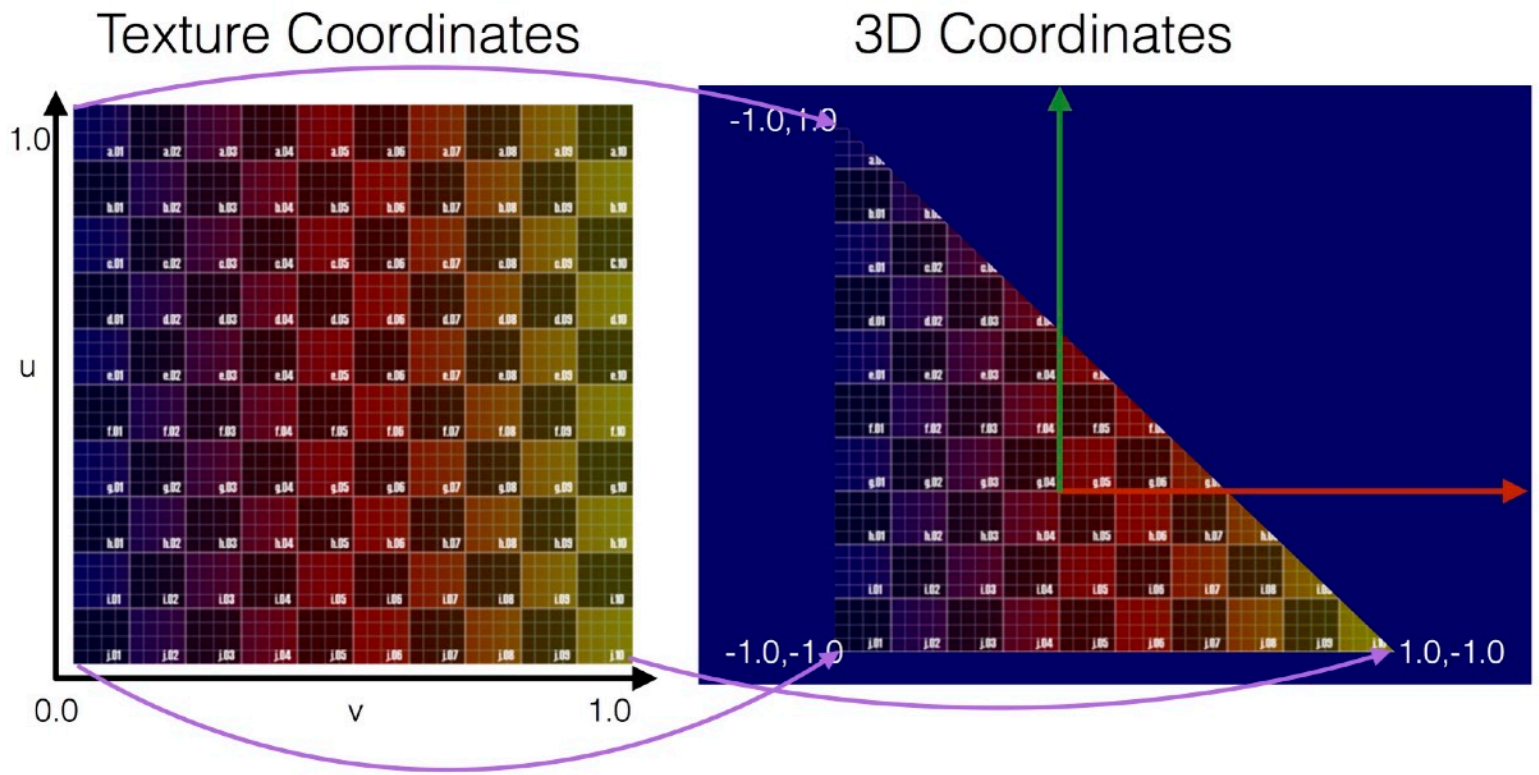
```
vec3 textureVal = texture( myTexture, UV ).rgb;
```

Object or World Coordinates?

Rosalee Wolfe: Teaching Texture Mapping Visually

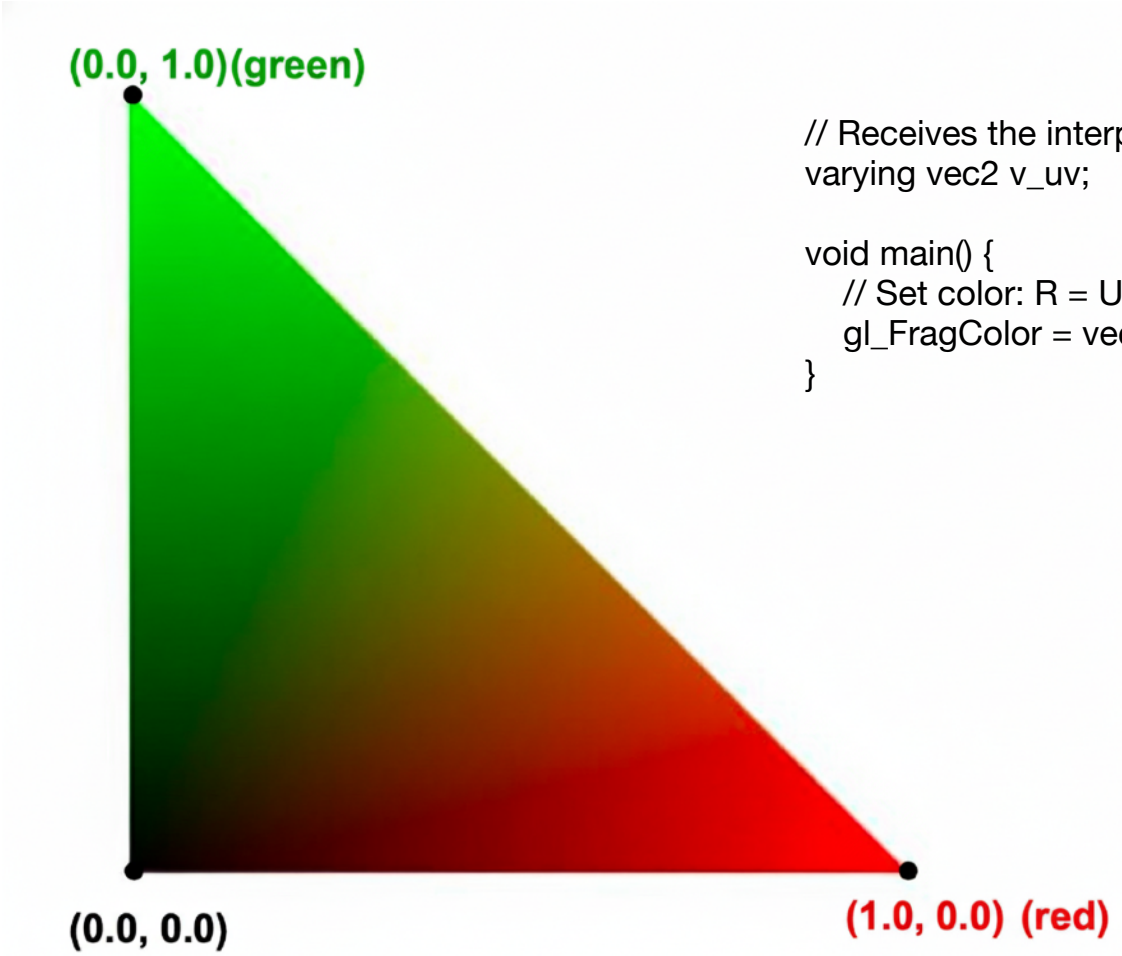
Texture Mapping

- Process of finding u,v coordinates for each vertex
- Texture is defined in a normalised space (e.g. for 2D textures: (u,v)∈[0...1,0...1]



Texture Coordinates

- Texture coordinates linearly interpolated over primitive



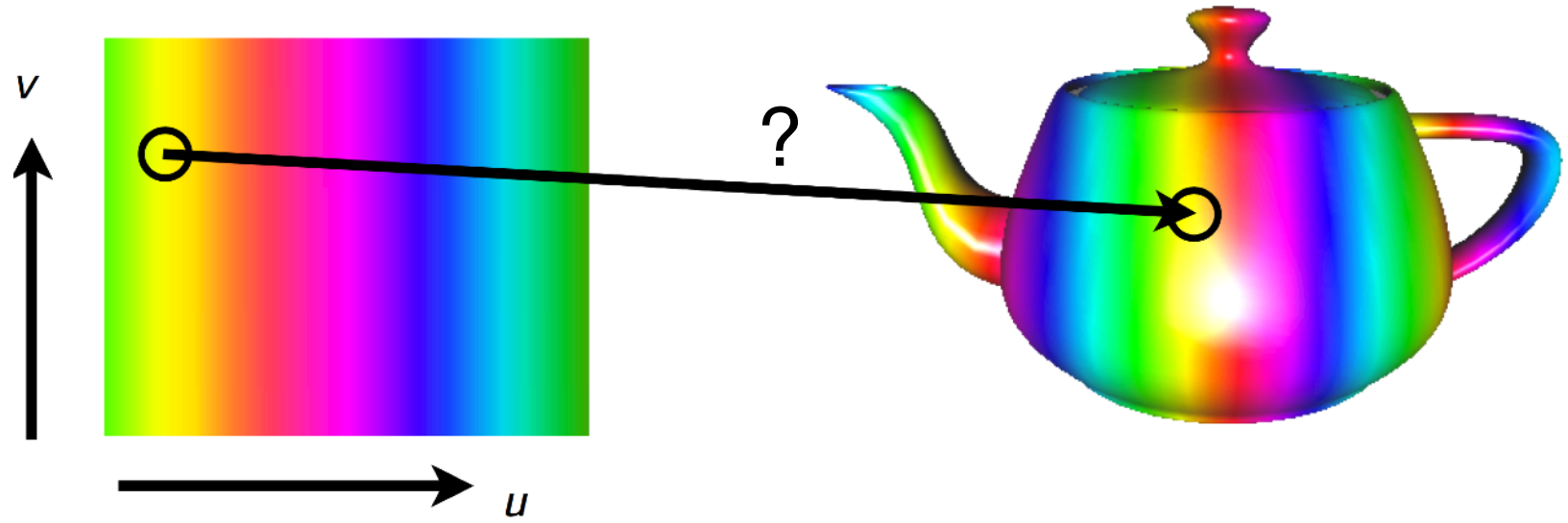

```
// Receives the interpolated UV from the vertex shader
varying vec2 v_uv;

void main() {
    // Set color: R = U (v_uv.x), G = V (v_uv.y)
    gl_FragColor = vec4(v_uv.x, v_uv.y, 0.0, 1.0);
}
```

3-Min Discussion:

What kinds of distortions can appear when mapping a 2D texture onto a 3D model, and what strategies can be used to reduce them?

Texture Mapping

- Often objects are not that simple

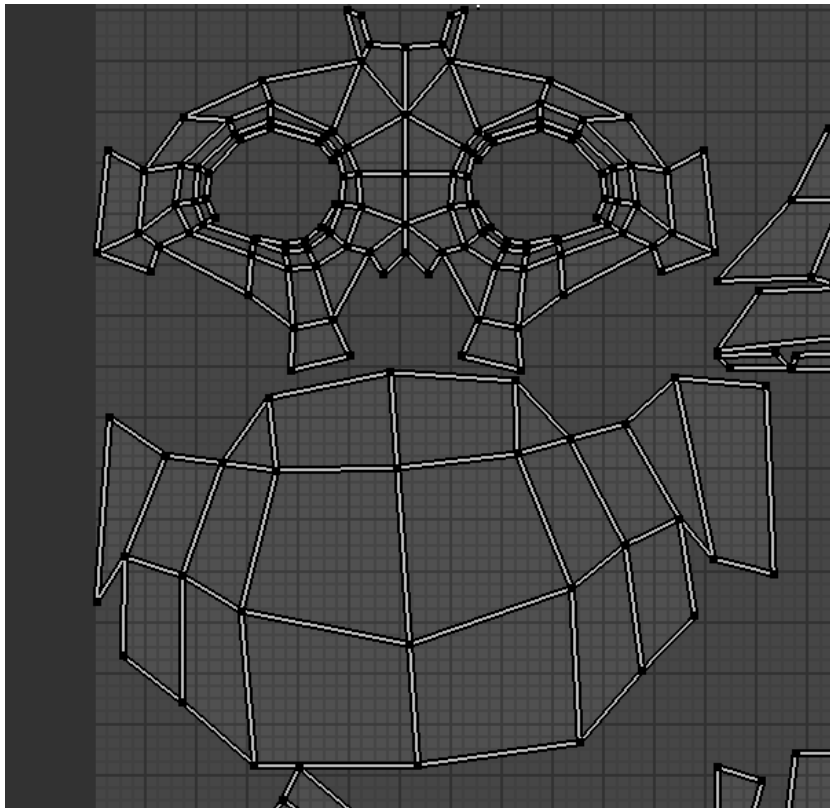


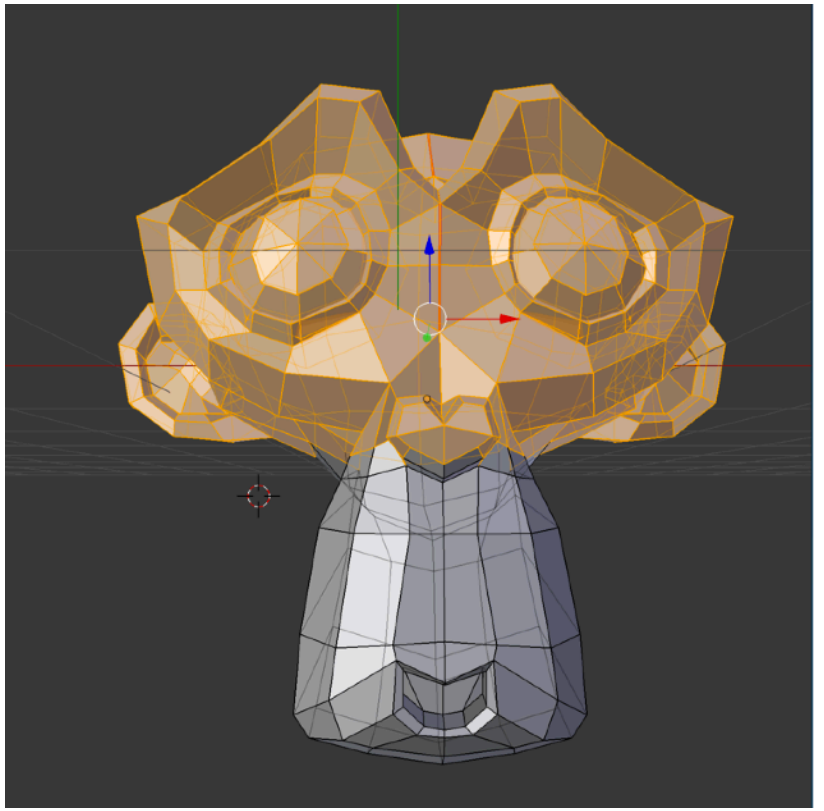
Rosalee Wolfe: Teaching Texture Mapping Visually

PARAMETERISATION

Parametrisation

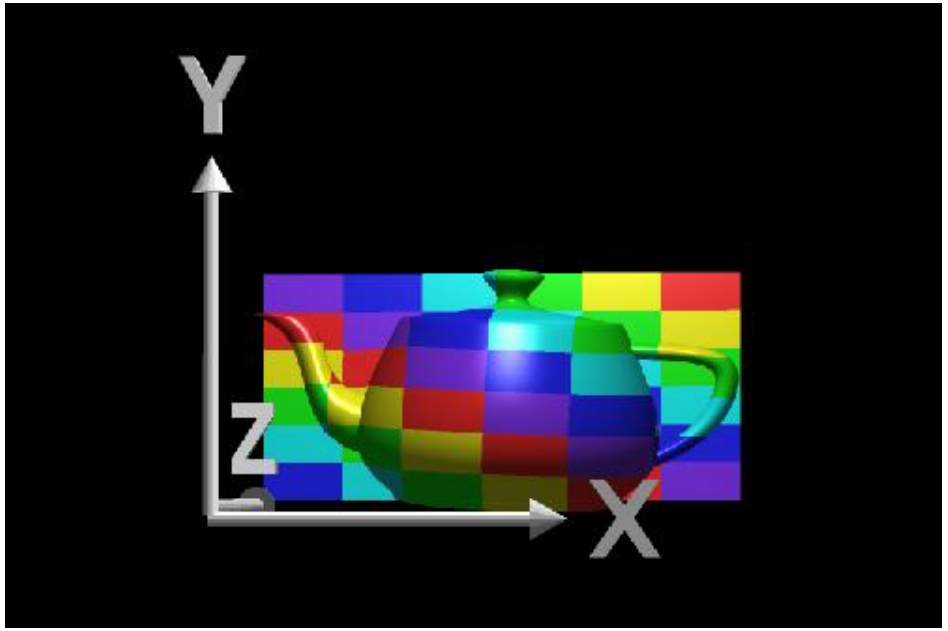
- Manual mapping
  - Unwrap object (e.g. in Blender)

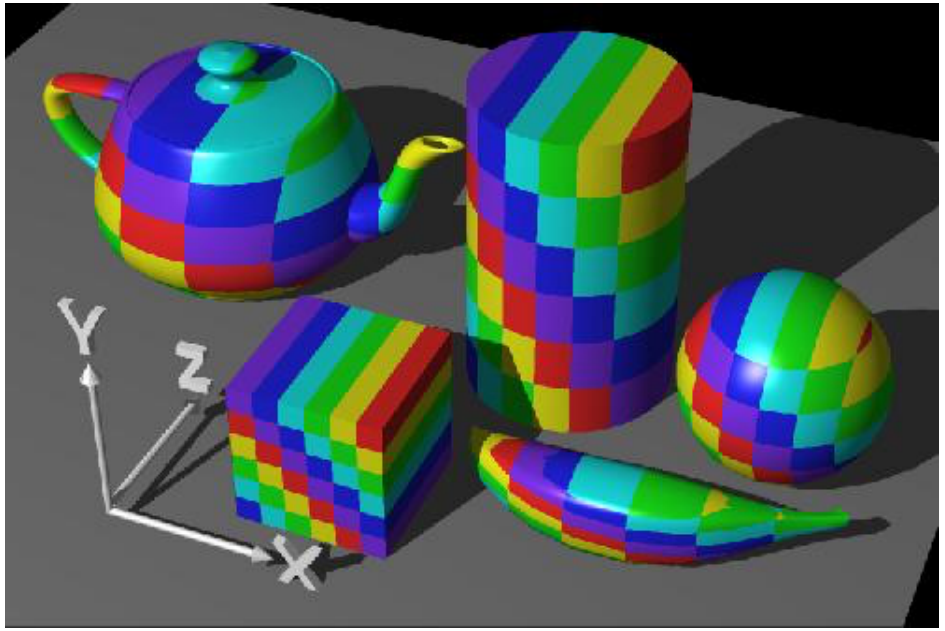




Parametrisation

- Planar mapping
  - Eliminates one axis (z) and use x and y directly: (u,v)=(x,y)
  - Looks only good from front

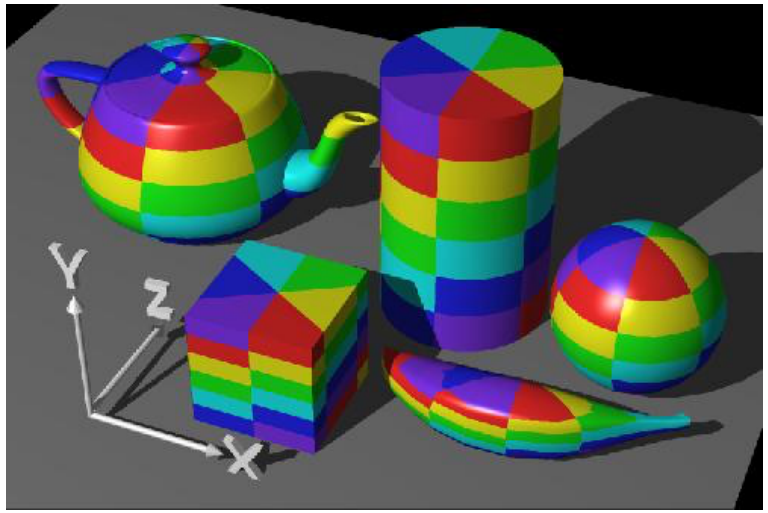




Rosalee Wolfe: Teaching Texture Mapping Visually

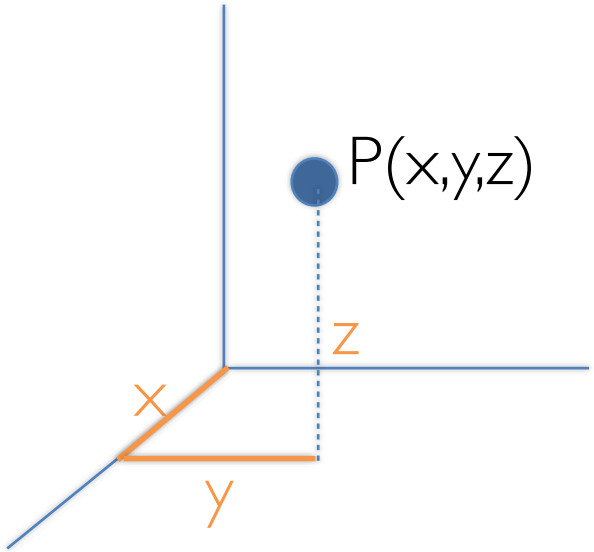
Parametrisation

- Cylindrical mapping
  - Compute angles between vertex and object centre
  - Converts point P(x,y,z) to cylindrical coordinates P(r, θ,z)
  - Wraps texture around the object

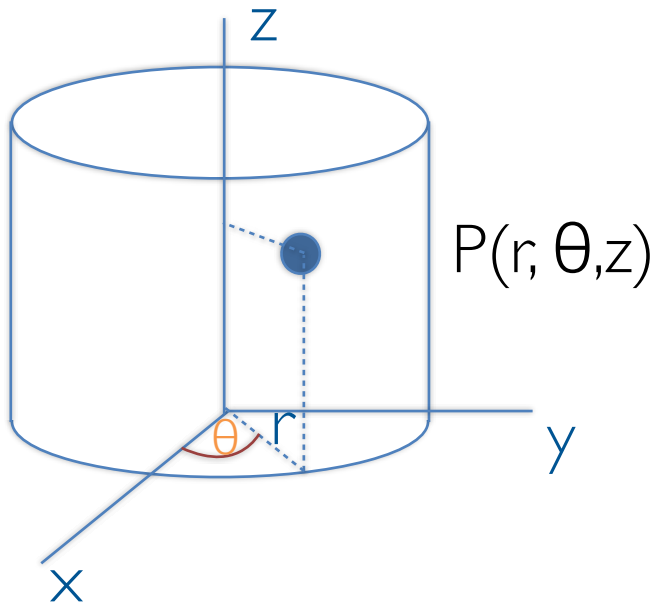


Rosalee Wolfe: Teaching Texture Mapping Visually

Parametrisation

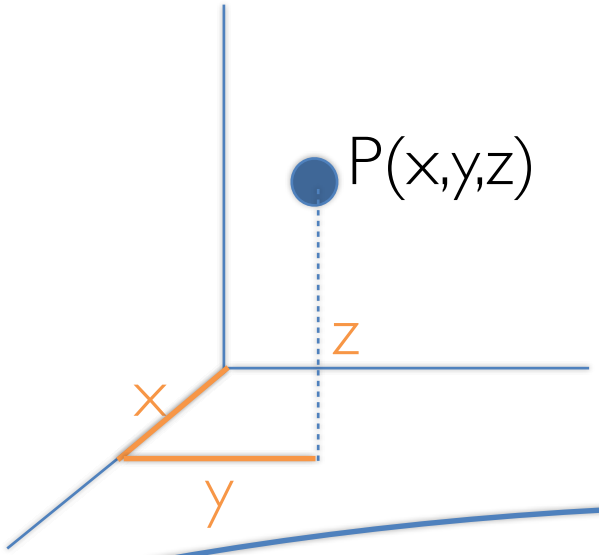


Cartesian Coordinates



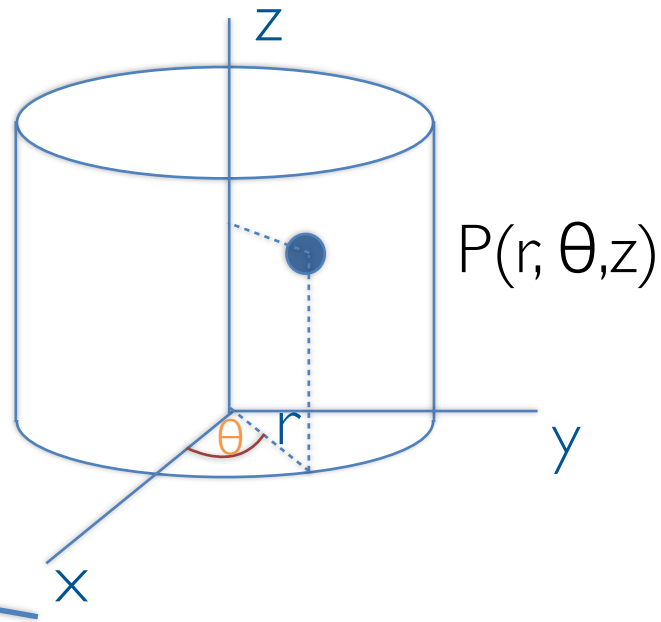
Cylindrical Coordinates

Parametrisation



Conversion:

- $x = r \cdot \cos \theta$,
- $y = r \cdot \sin \theta$,
- $z = v$



Cartesian Coordinates

Cylindrical Coordinates

Take care of quadrants:

$\tan \theta = y/x \Rightarrow \theta = \tan^{-1}(y/x)$

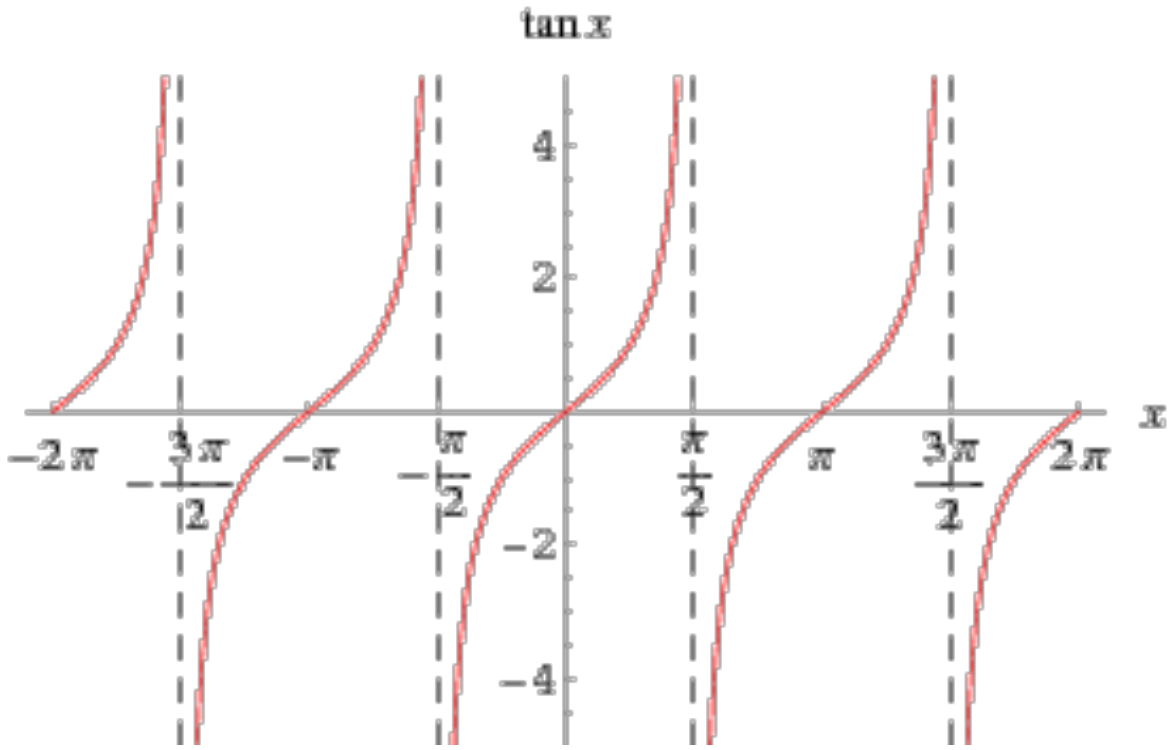
Conversion:

- $u = \theta = \tan^{-1}(y/x)$
- $v = z$

With:

- $u = \theta$, with $0 \leq \theta \leq 2\pi$
- $v = z$ with $0 \leq z \leq 1$

Parametrisation

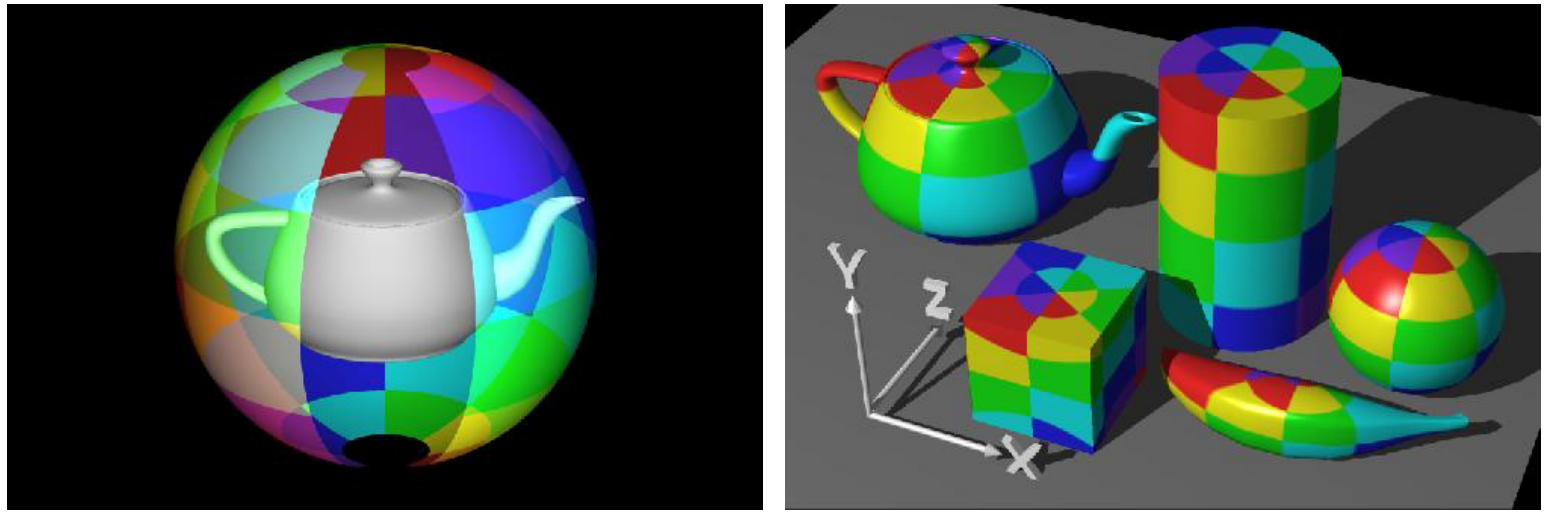


https://mathworld.wolfram.com/Tangent.html

- Take care of quadrants:
  - tan θ =y/x => θ = tan-1(y/x)

Parametrisation

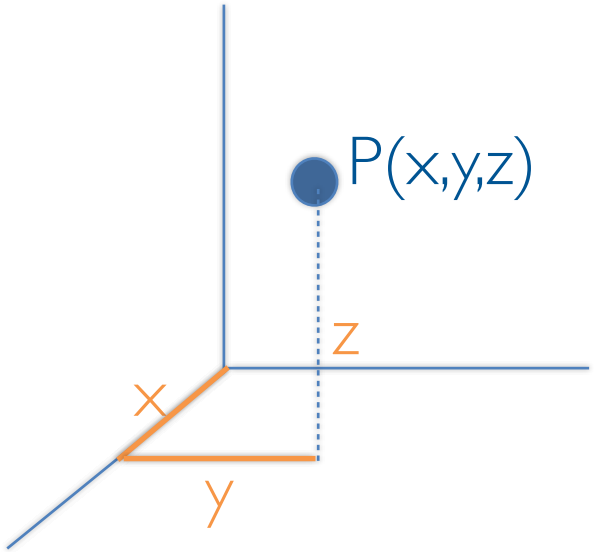
- Spherical mapping
  - $p(x, y, z)$ value of a point is converted into spherical coordinates ((θ), (ϕ))
  - Wraps texture around the object



Rosalee Wolfe: Teaching Texture Mapping Visually

Notes: Parametrisation

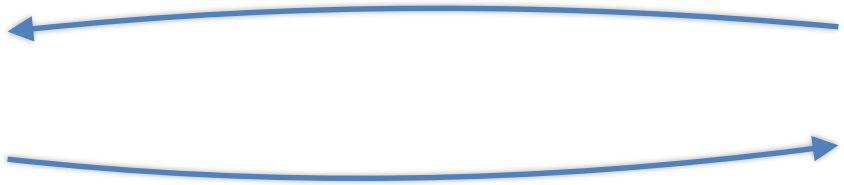
ρ Rho
θ Theta
ϕ phi



Cartesian Coordinates

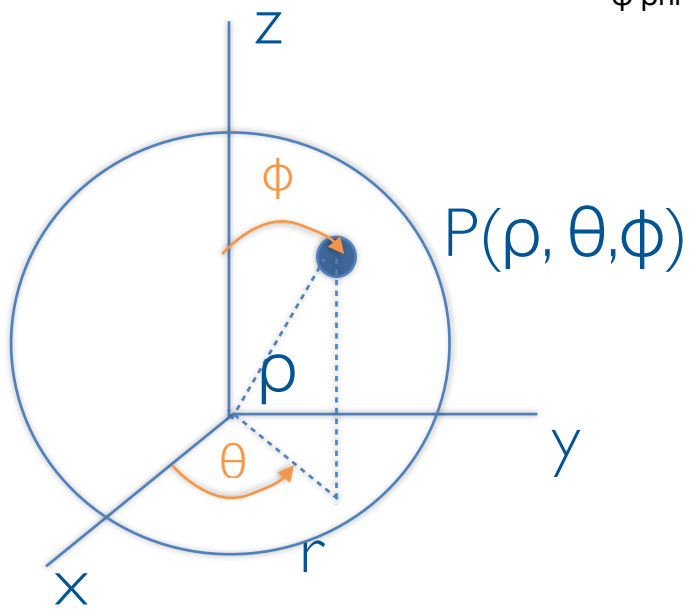
Conversion:

- $x = \rho \sin\phi \cos\theta$
- $y = \rho \sin\phi \sin\theta$
- $z = \rho \cos\phi$



Conversion:

- $u = \theta = \tan^{-1}(y/x)$
- $v = \phi = \cos^{-1}(z/\rho)$



Spherical Coordinates

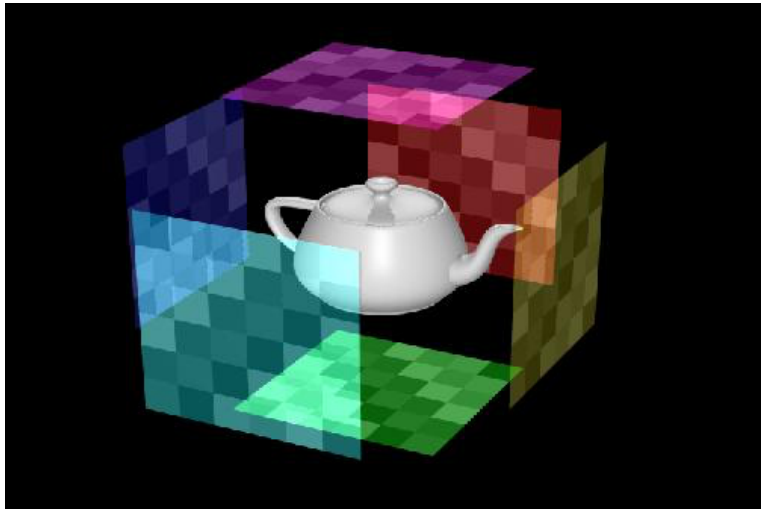
With:

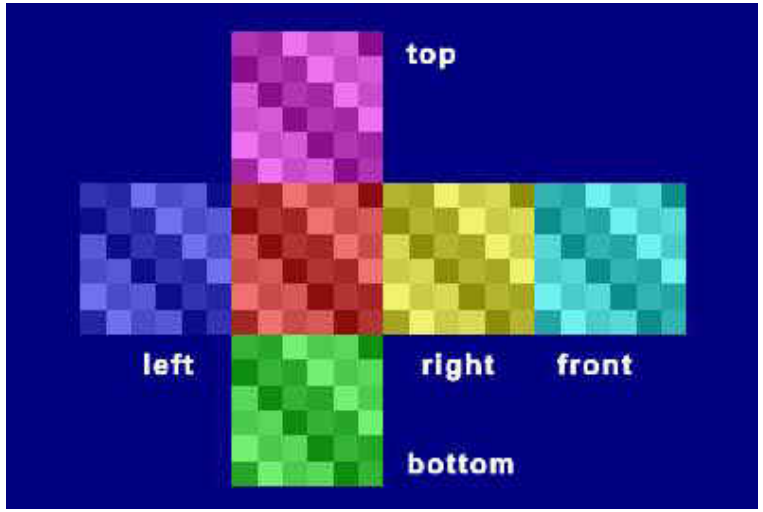
- $u = \theta$, with $0 \le \theta \le 2\pi$
- $v = \phi$ with $0 \le \phi \le 2\pi$

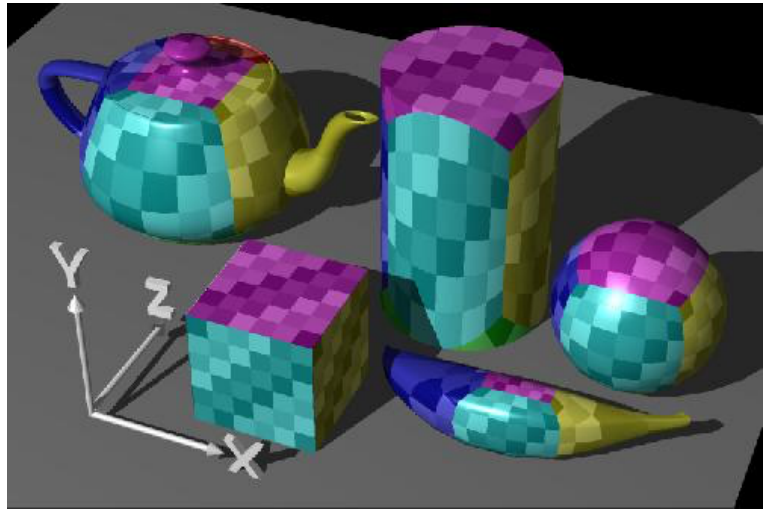
- u: wraps around the sphere horizontally (longitude)
- v: runs vertically from pole to pole (latitude)

Parametrisation

- Box mapping
  - Similar to planar mapping
  - But uses 6 textures instead of 1
  - Used mainly for environment mapping







Rosalee Wolfe: Teaching Texture Mapping Visually

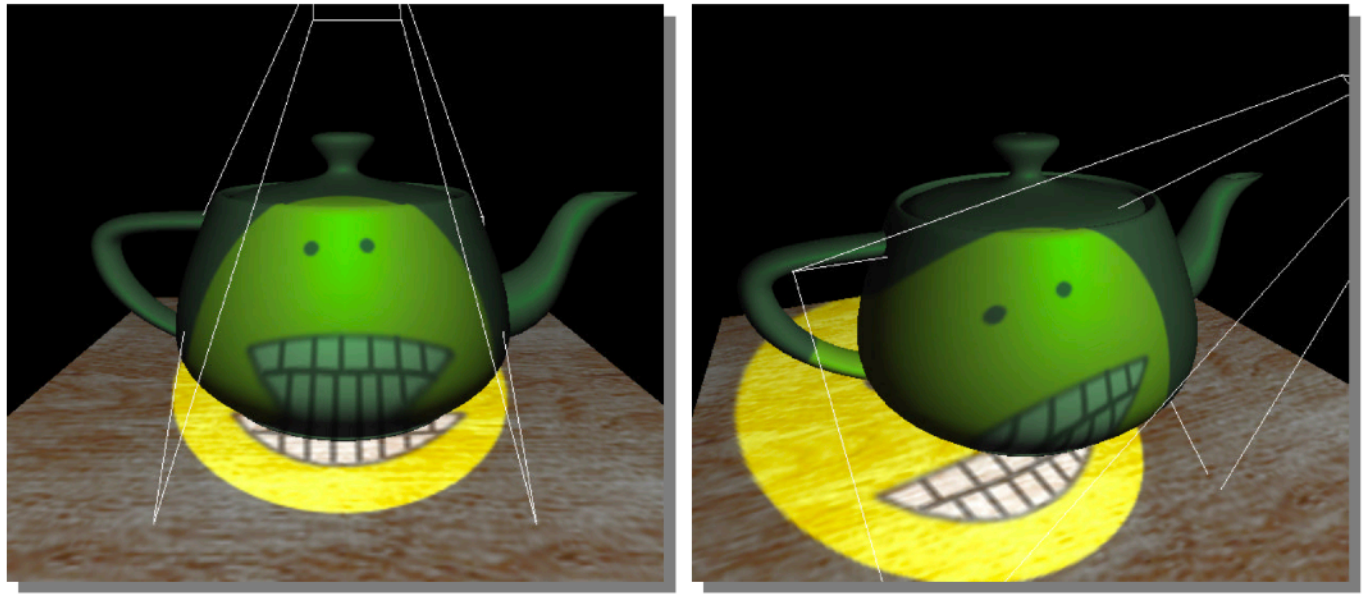
Parametrisation

- Box mapping
  - Similar to planar mapping
  - But uses 6 textures instead of 1
  - Used mainly for environment mapping

Rosalee Wolfe: Teaching Texture Mapping Visually

Projective Texture mapping

- Texture is used as light source
- Like a slide projector
- A good model for shading variations due to illumination
- But also has other use cases



Everitt: Projective Texture Mapping

Projective Texture mapping

- Other use case:
  - Create billboard models from photos
  - Uses input photos as textures with projective texture mapping

Zollmann and Reitmayr: Dense Depth Maps

Texture addressing

- At some point, texture coordinates have to be mapped to texels
- What happens if coordinates are outside of the image?
  - -> “texture addressing”

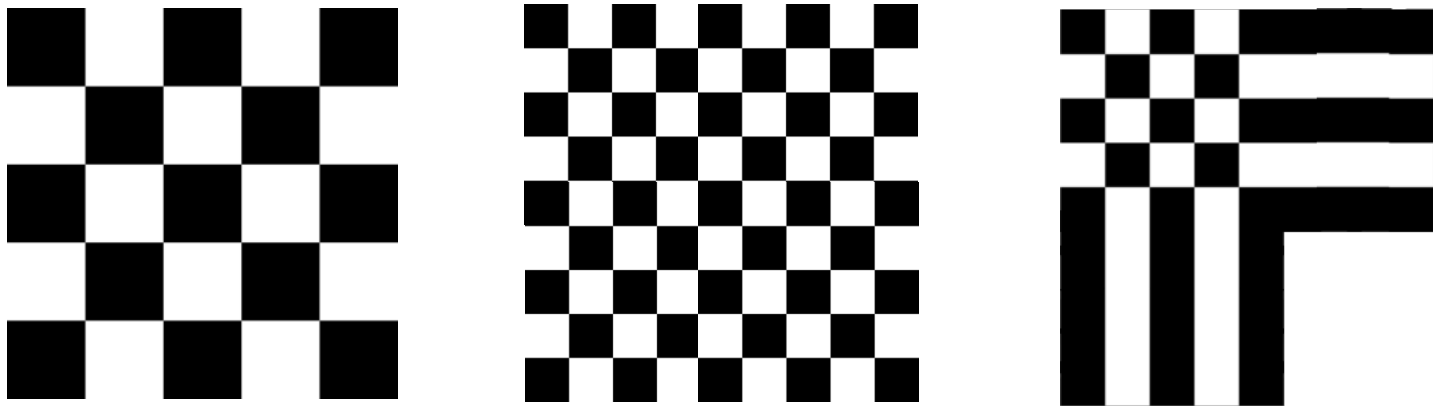
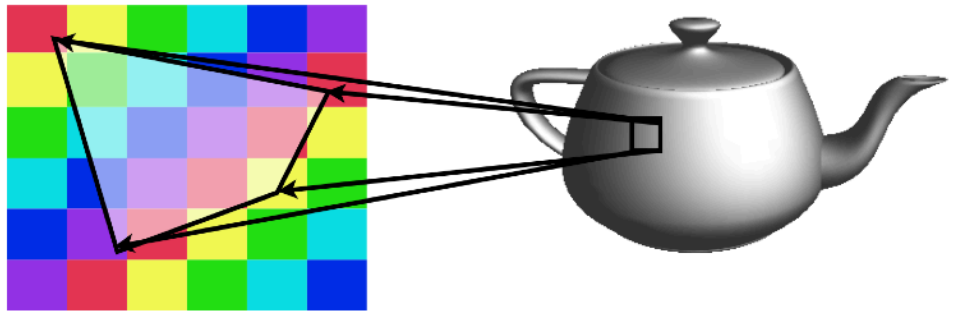


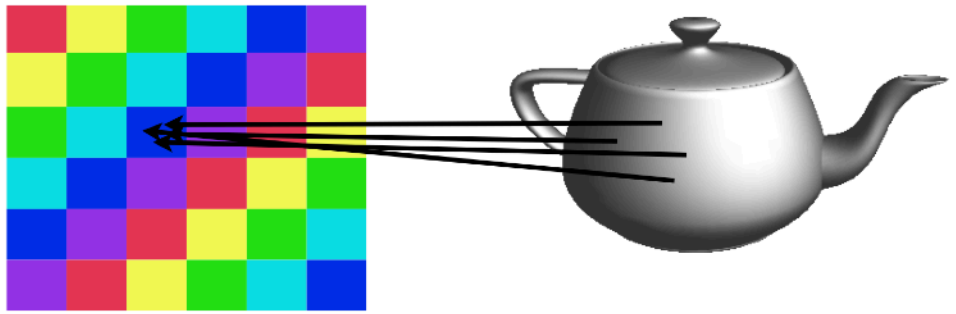
Image GL_WRAP GL_CLAMP

Challenges

- Undersampling: one fragment maps to an area covering many texel (Minification)

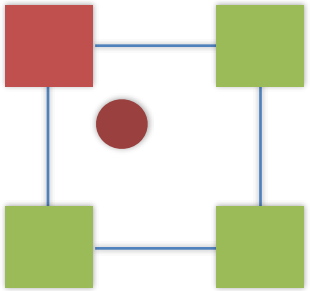


- Oversampling: many fragments map to an area contained by only one texel (Magnification)

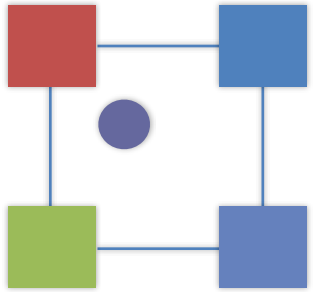


Texture Filtering

- Interpolate texel value from neighbours



Nearest Neighbour (lower quality)

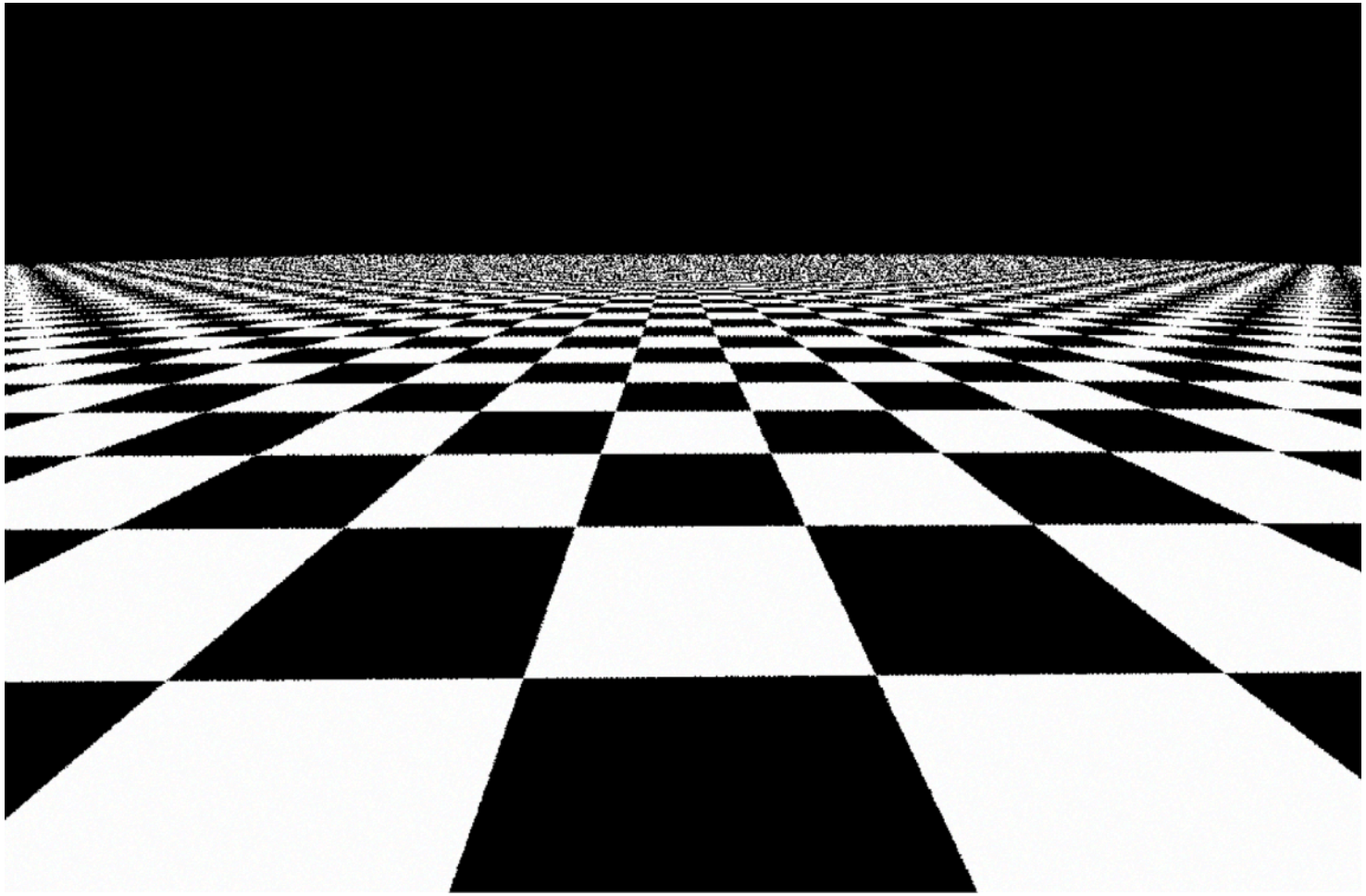


Linear interpolate the neighbours (better quality, slower)

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
```

Texture Filtering

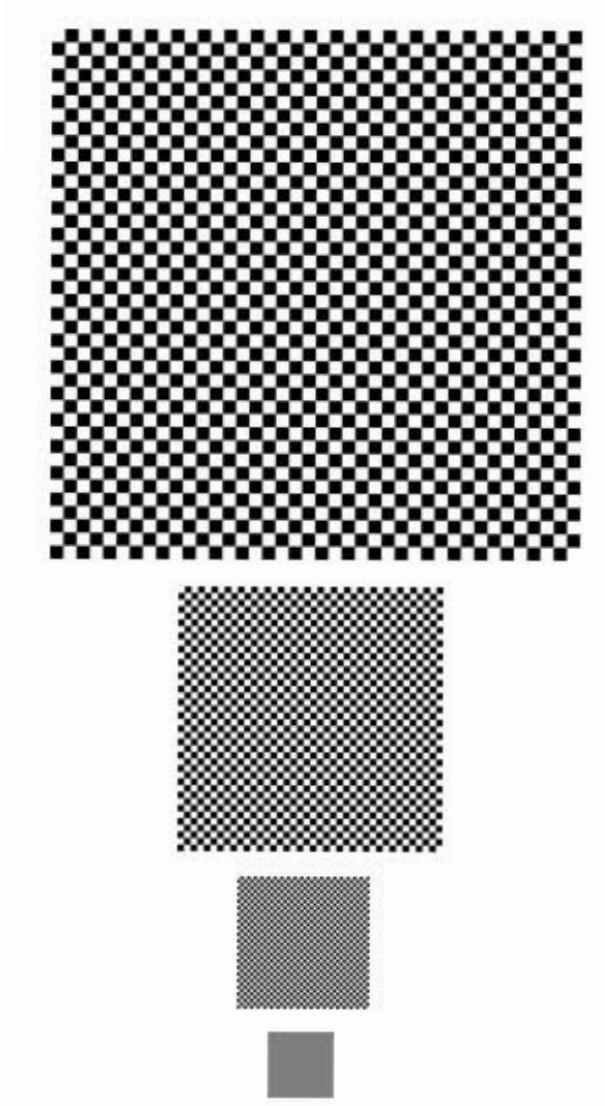
MipMaps



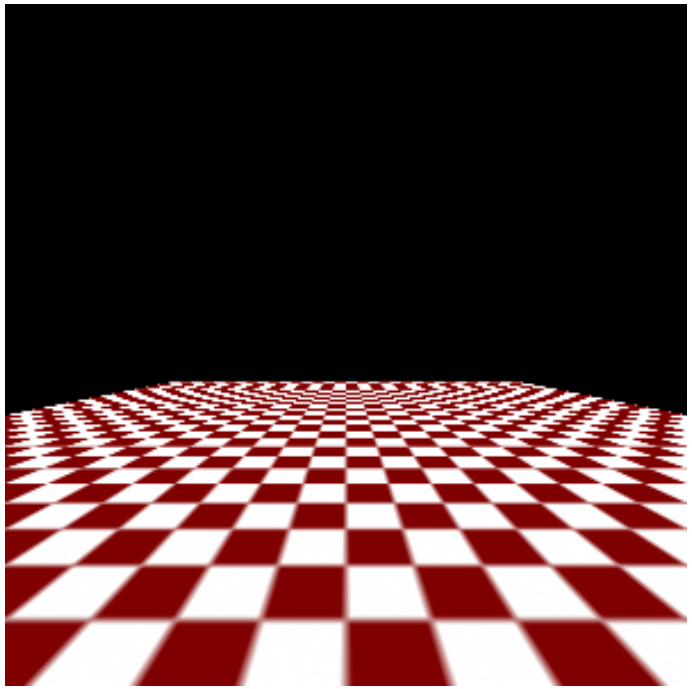
Problem of undersampling

MipMaps

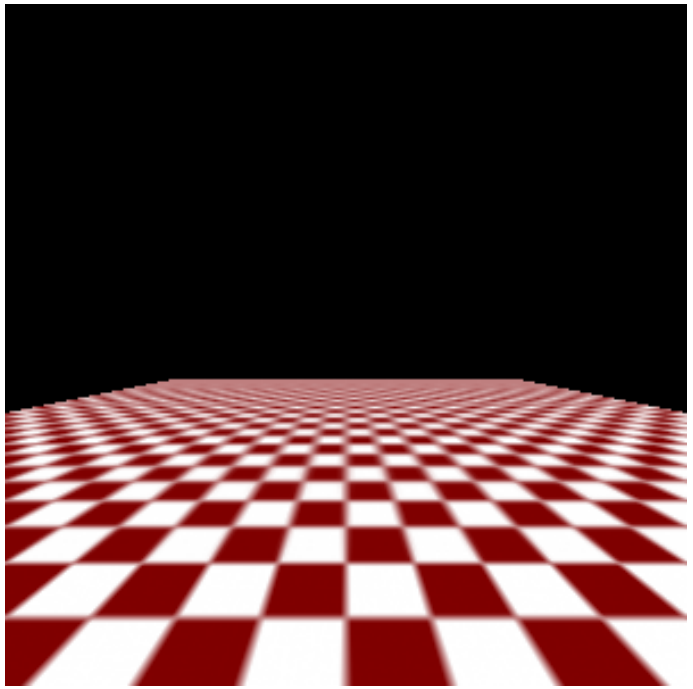
- Start with most detailed texture
- Reduce to half size by combining (averaging) adjacent four texels
- Continue to minimum
- During rendering choose texture level according to distance



MipMaps



No mipmaps



Mipmaps

MipMaps


Floor (128x128)
mag: Linear
min: LinearMipMapLinear
Painting (748x600)
mag: Linear
min: Linear
Floor
mag: Nearest
min: NearestMipMapNearestFilter
Painting
mag: Nearest
min: Nearest

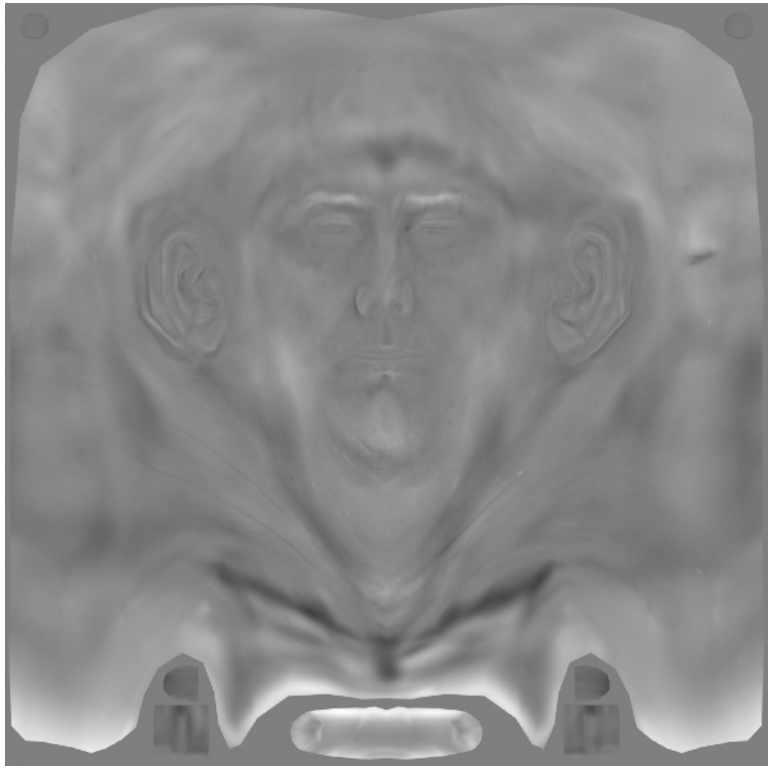
Texture Types

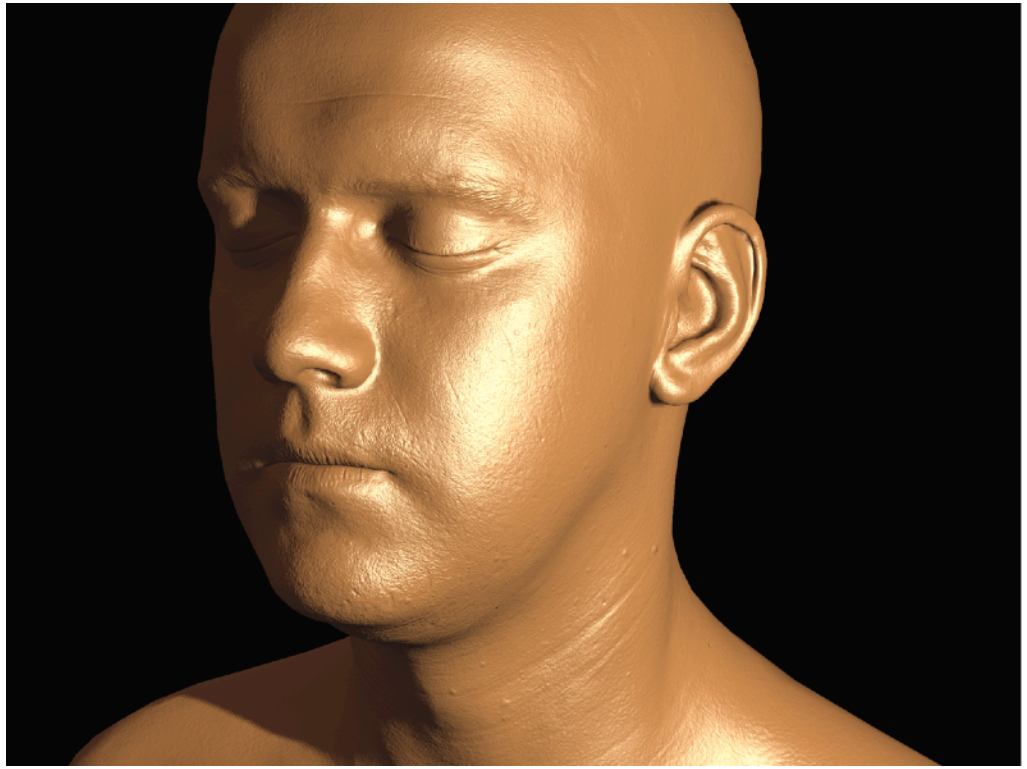
- One-dimensional: T(s)
  - Parameter can have arbitrary domain (e.g. incident angle)
- Two-dimensional: T(s, t)
  - Most often: raster images
  - Painted by artist, scanned, photographed, generated procedurally,
- Pixels of a texture image are called “texels”.
- Three-dimensional: T(s, t, r)

Applications

Bump Mapping

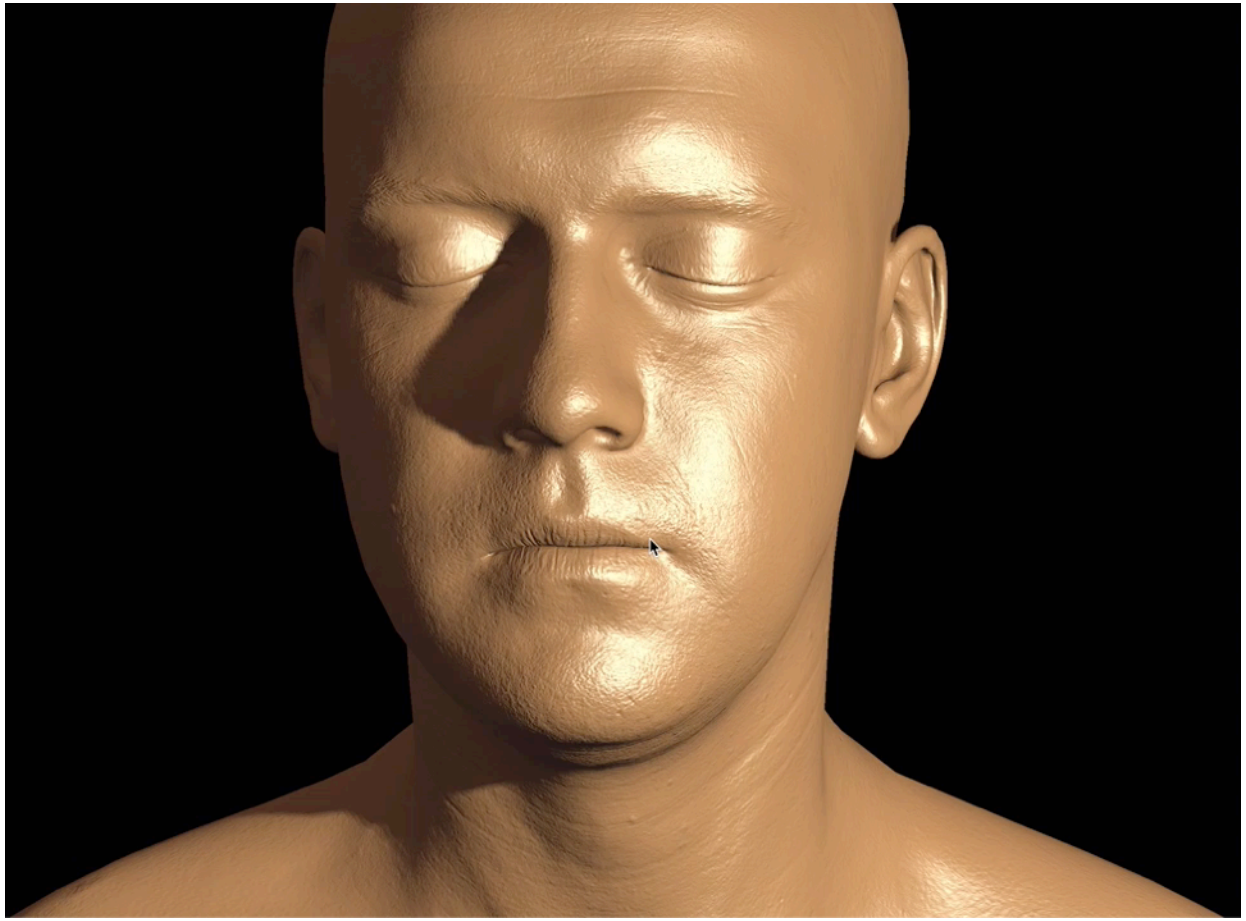
- Create details (e.g. bumps, wrinkles) with changing normals
- Modifies the normals used for computing illumination





Source: https://github.com/mrdoob/three.js/

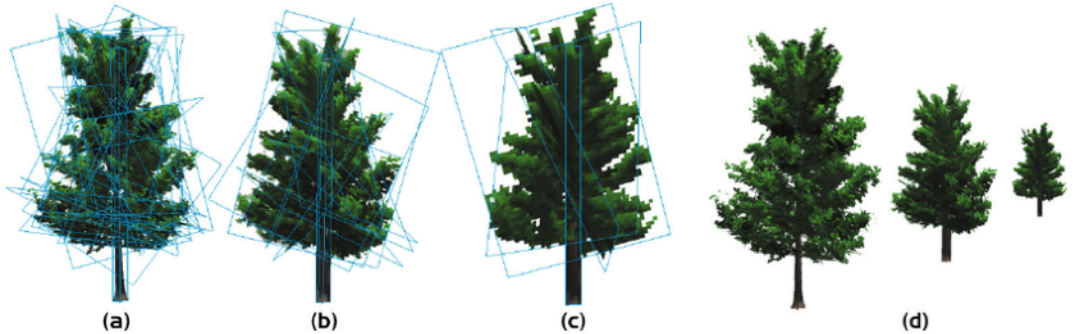
Bump Mapping



Source: https://github.com/mrdoob/three.js/

Alpha Mapping

- Alpha map contains opacity values
- Use for blending between multiple textures
- Creating transparency effects
- Billboard rendering



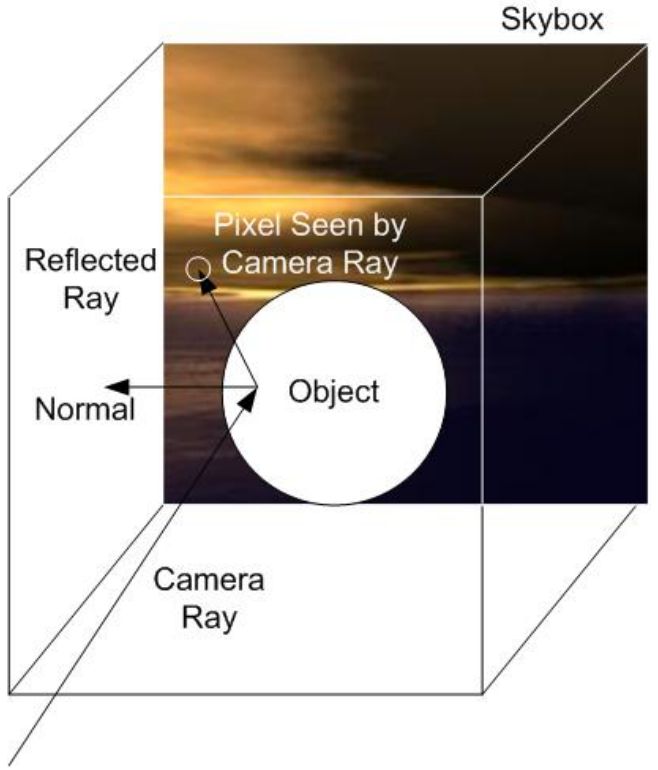
Fuhrman et al.: Extreme Model Simplification for Forest Rendering.

Source: http://jeromeetienne.github.io/

Environment Mapping

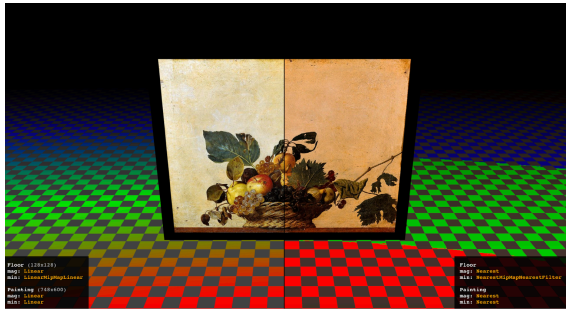
Environment Mapping

- Reflection mapping
  - Defined over surface of enclosing object (e.g. Sphere, Cube, Cylinder)
- Differences to previous techniques:
  - Use the direction of the reflection ray to compute point on enclosing object
  - Texture contains information about:
    - Light sources
    - Sky
    - Background objects



https://commons.wikimedia.org/wiki/File%3ACube_mapped_reflection_example.jpg

Time for some examples



The end!