Universitat de Girona

Texture Mapping

Gonzalo Besuievsky

IMAE - UdG

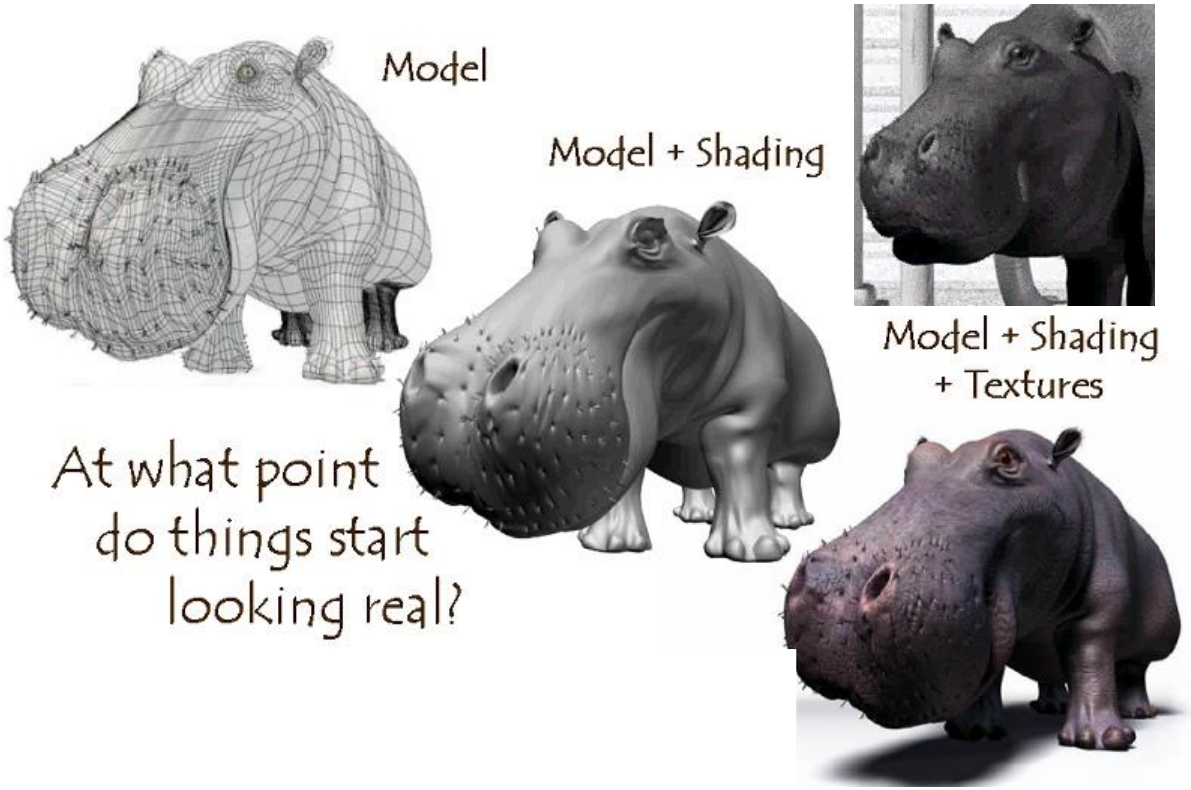
Universitat de Girona

Materials i Textures

- Introduction and basic concepts
- Mapping functions
- Bump mapping
- Environment maps
- Illumination maps
- Textures en WebGL

Universitat de Girona

Introduction



Universitat de Girona

Hierarchical Scale (Westin et al 92)

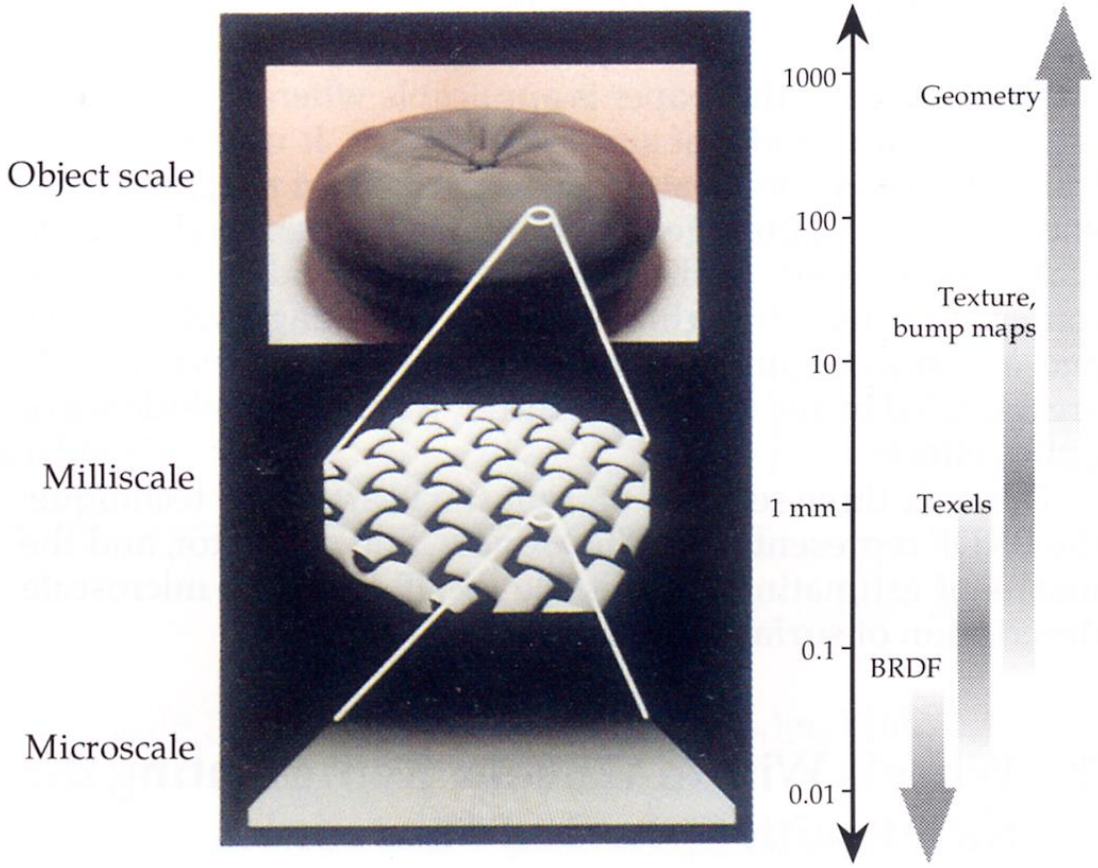


Figure 1: Applicability of Techniques

Universitat de Girona

Hierarchical Scale

Physics

Geometrical optics

Macro-structures

Transport

Micro-structures

Microfacets

Physical optics

Quantum optics

Computer Graphics

Geometry

Displacement (P) maps

Bump (N) maps

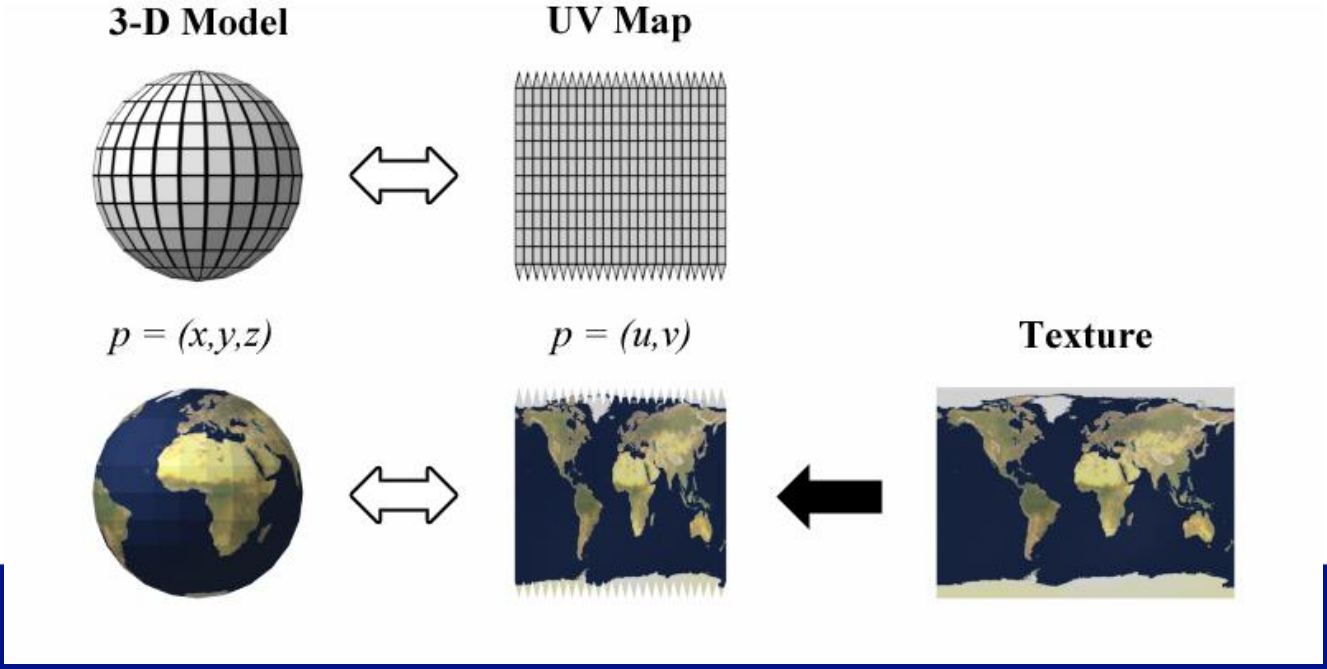
Reflection

Texture

Universitat de Girona

Texture Maps

- Com es pot mapejar una textura a una superfície?
  - Dimensió: 1D, 2D, 3D
  - Coordenades de Texture (s,t)
    - Paràmetres de superfície (u,v)
    - Direcció de vectors: reflexió R, normal N, halfway H
    - Projecció: cilindro, esfera, cub



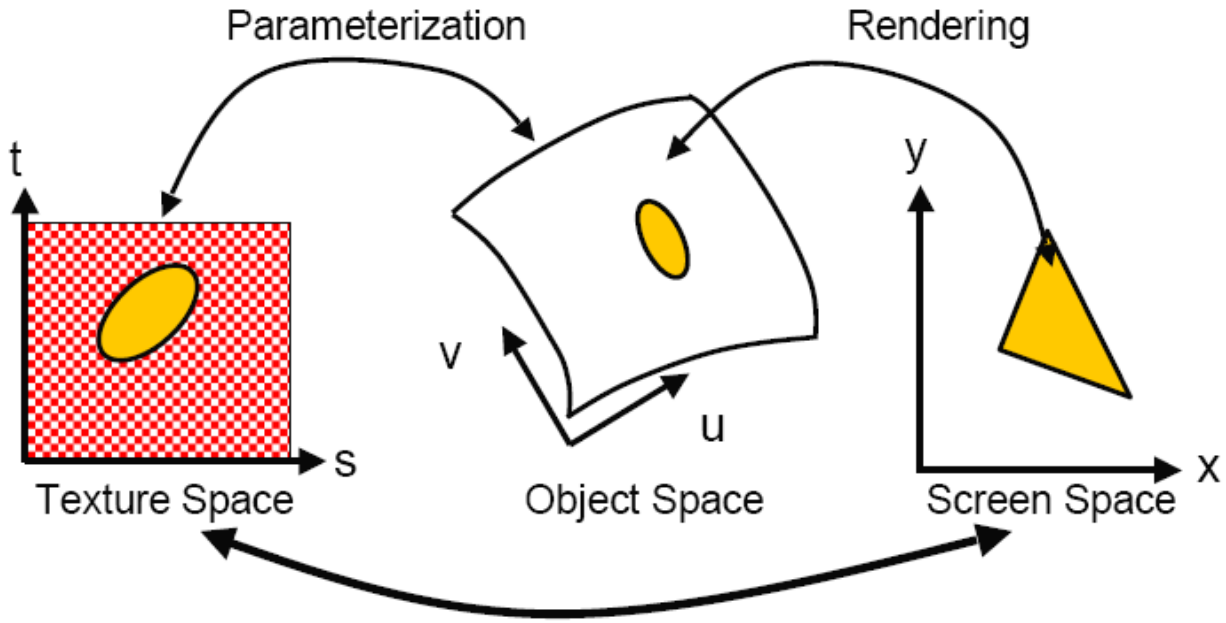
Universitat de Girona

Texture Maps

- What can be controlled ?
  - Color and opacity of surfaces
  - Illumination function: environment maps, shadow maps
  - Reflection function: reflectance maps
  - Geometry: bump maps and displacement maps

Universitat de Girona

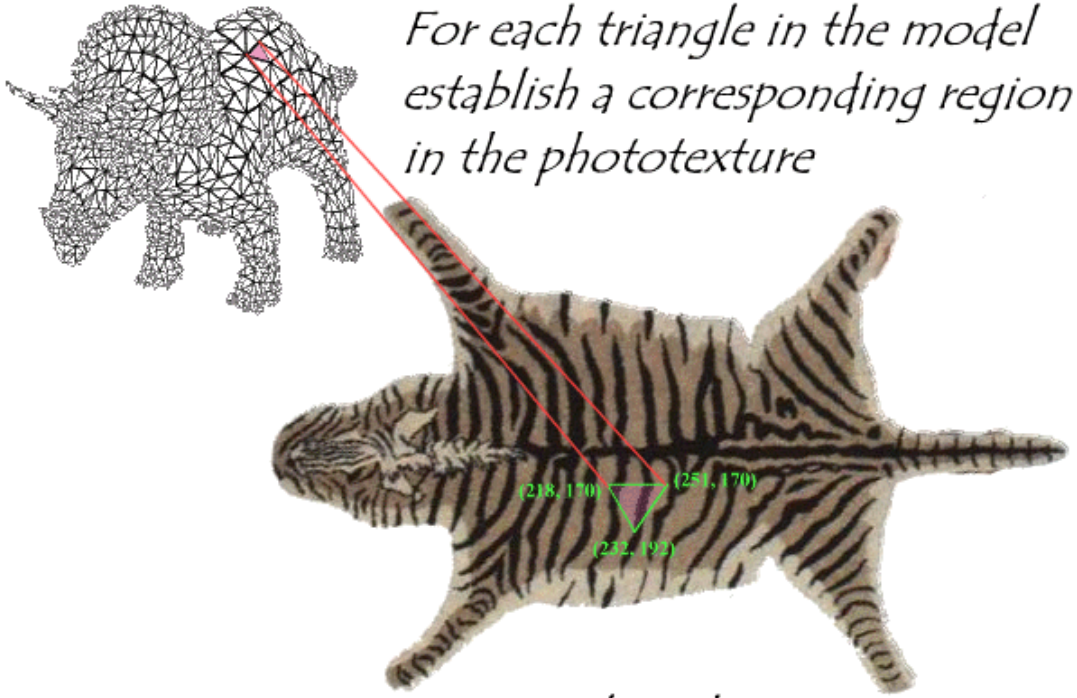
Texture mapping Spaces



Projection composition

Universitat de Girona

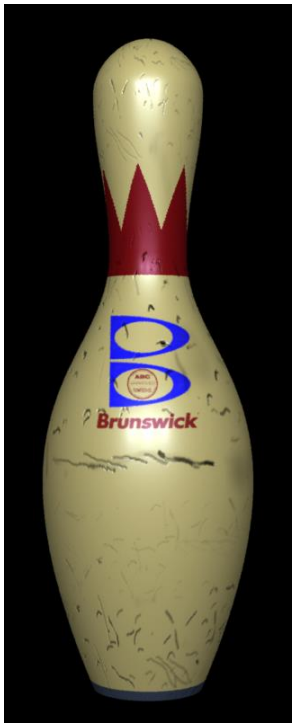
Imatge Textures



Universitat de Girona

Material color and layers

Tom Porter's Bowling Pin



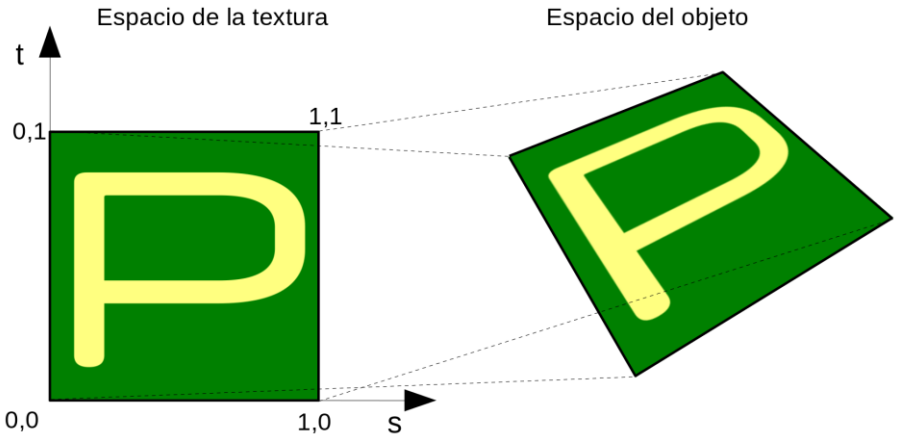
RenderMan Companion, Pls. 12 & 13

Universitat de Girona

Texture mapping

- How to map an images ?

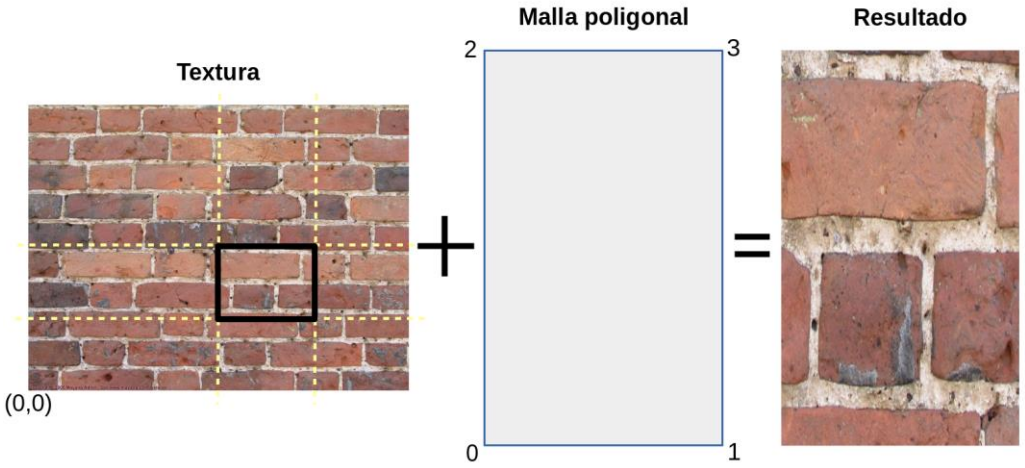
- Assign texture coordinates
  - –Attributes for each vertex
  - –Texture space range [0..1]
  - –Coordinate are provide by:
    - The geometric model
    - The program



Universitat de Girona

Example

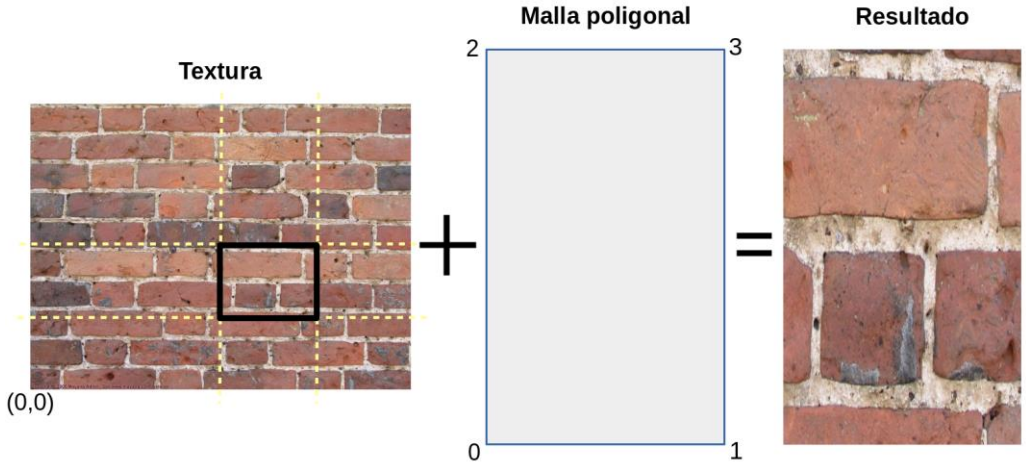
- If the size of the texture portion is ¼ of the height and weight size,
- Which are the texture coordinates for the polygon ?



Universitat de Girona

Example

- If the size of the texture portion is ¼ of the height and weight size,
- Which are the texture coordinates for the polygon ?

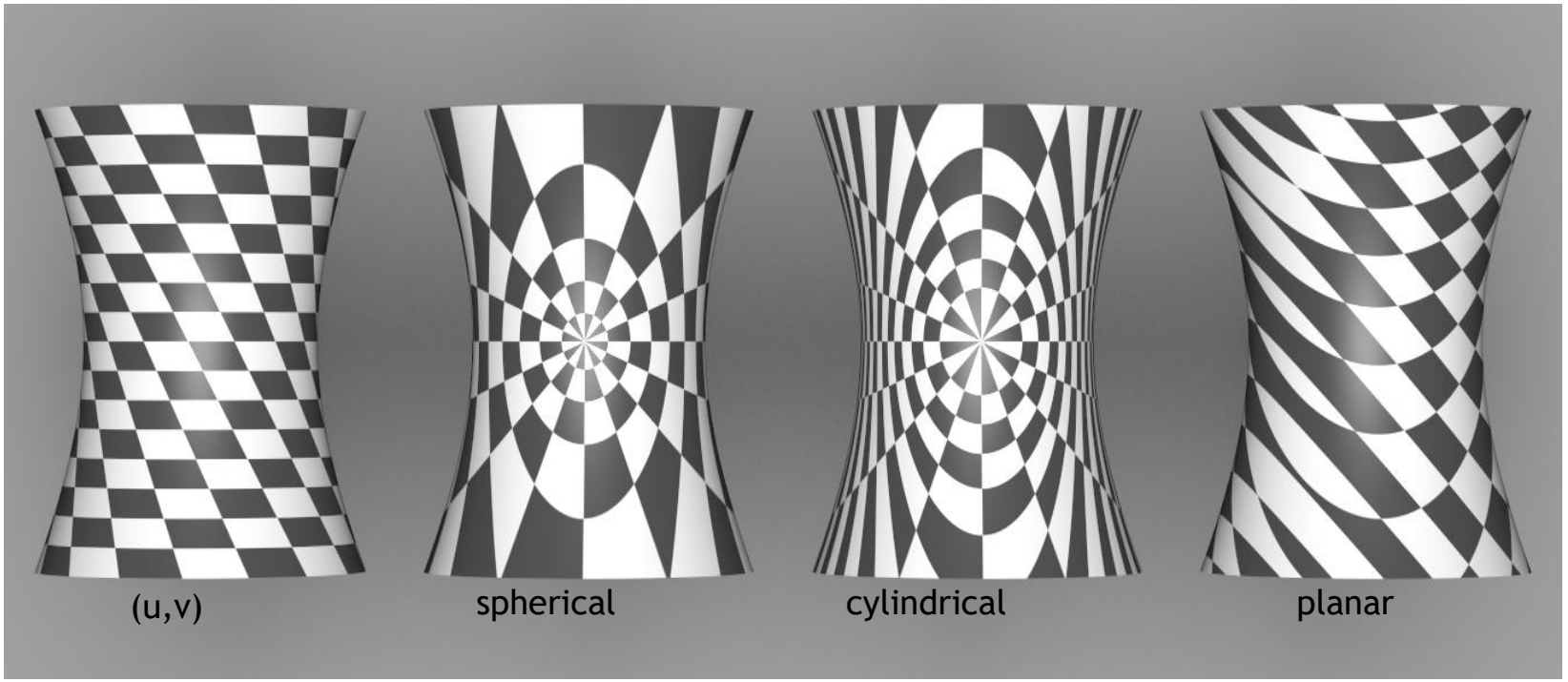


Solution:
0 -> 0.5, 0.25
1-> 0.75, 0.25
2-> 0.5, 0.5
3-> 0.75. 0.5

Universitat de Girona

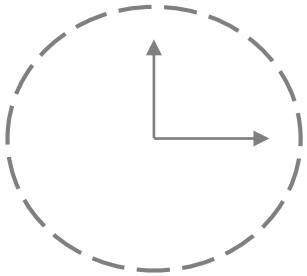
Texture Coordinate generation

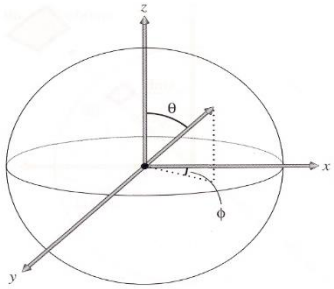
Basic Projections

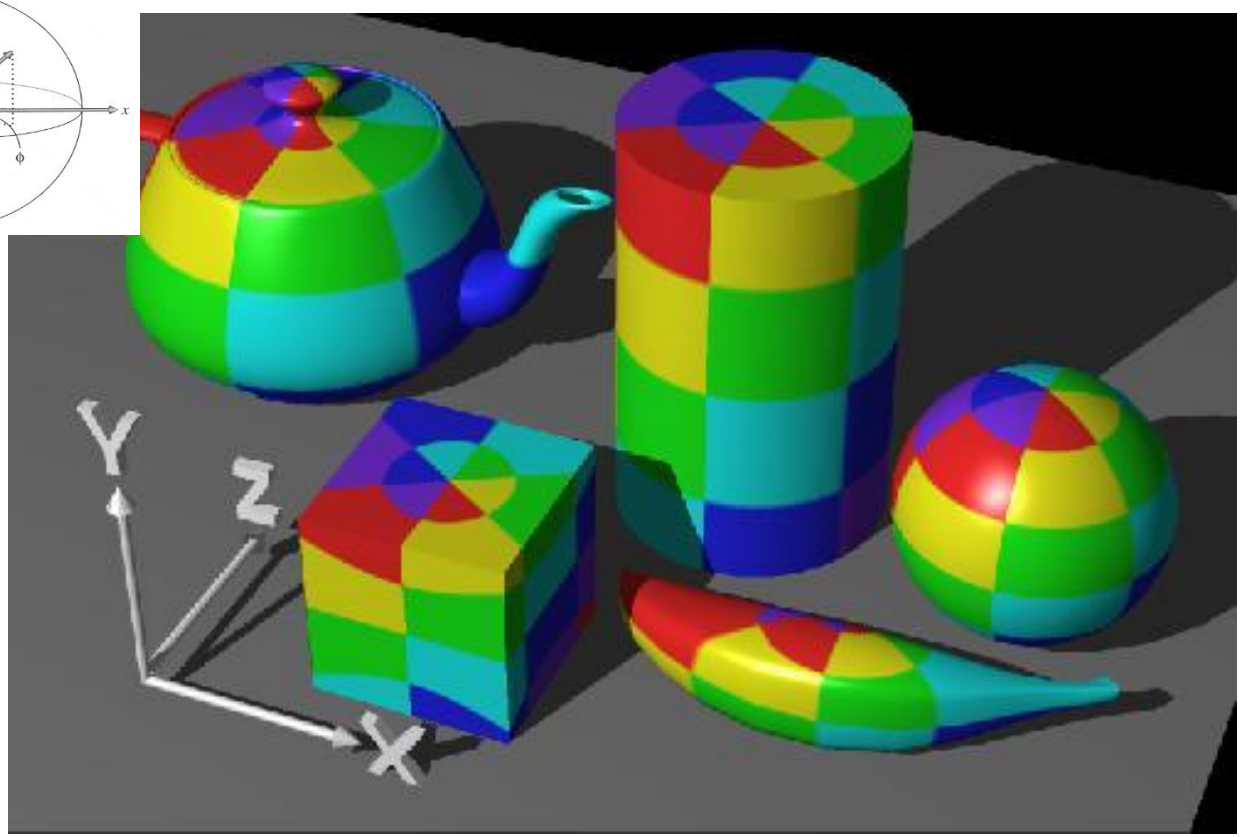


Universitat de Girona

Spherical mapping







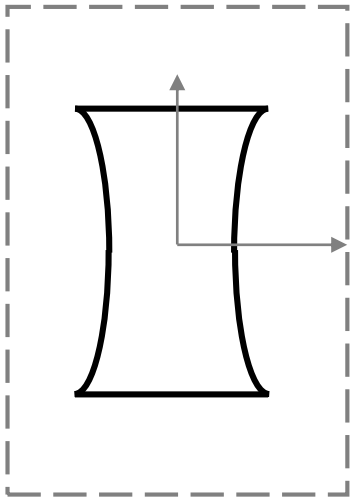
$x = r \cos 2\pi u$

$y = r \sin 2\pi u \cos 2\pi v$

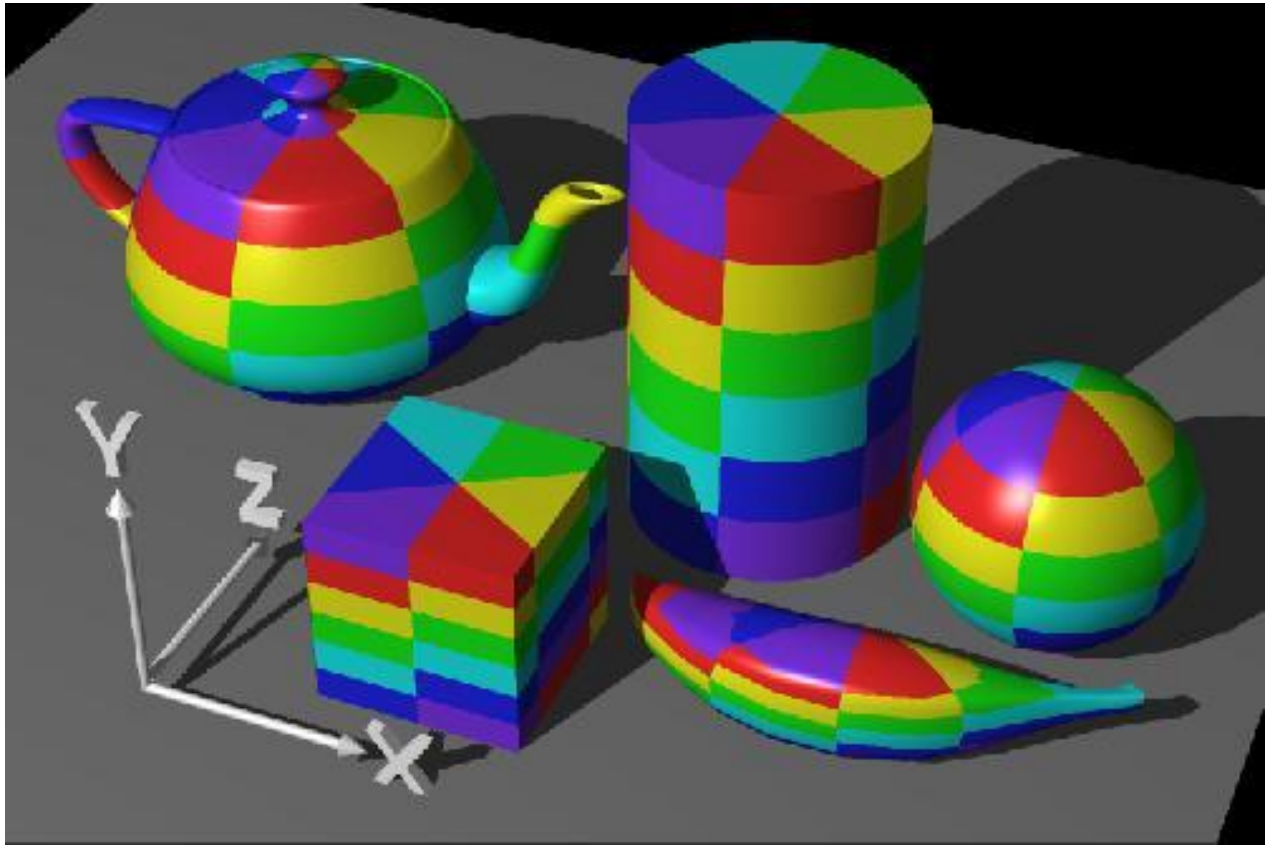
$z = r \sin 2\pi u \sin 2\pi v$

Universitat de Girona

Cylindrical mapping

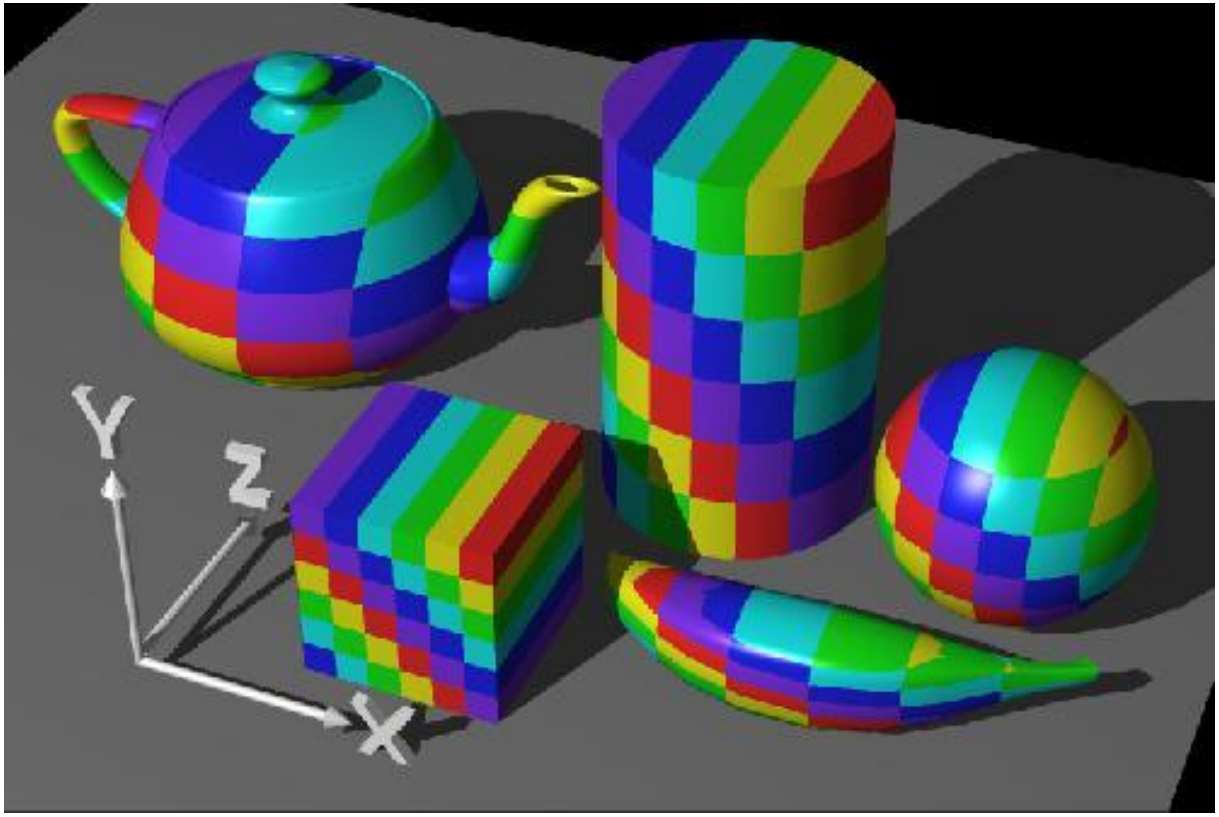


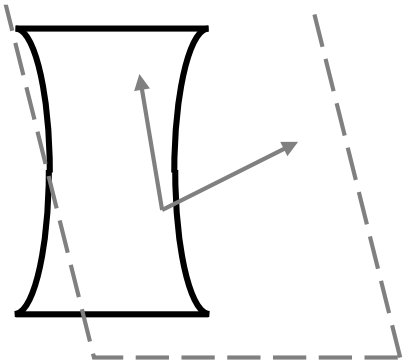
$x = r \cos 2\pi u$
$y = r \sin 2\pi u$
$z = v/h$



Universitat de Girona

Planar mapping

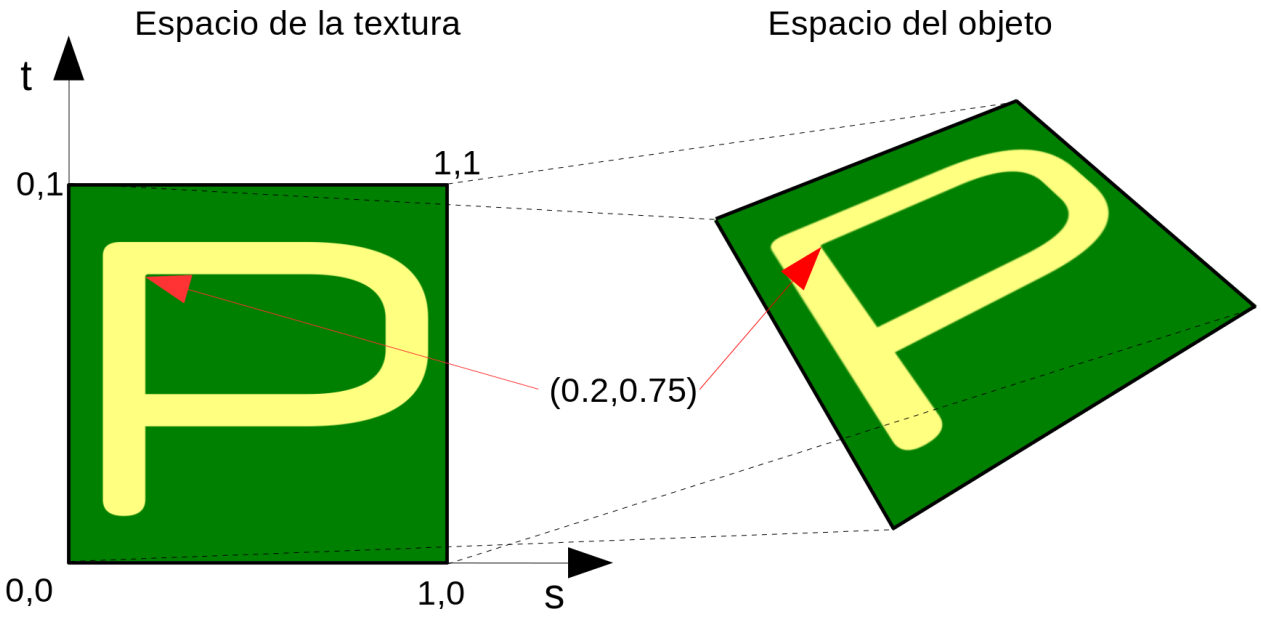




Universitat de Girona

Coordinates at interior points

- Coordinates are interpolated for each fragment



Universitat de Girona

Basic Shaders for Texture 2D

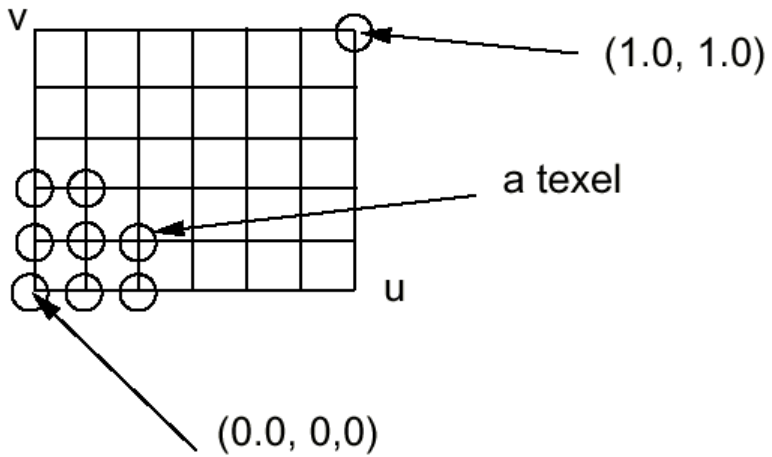
- Send texture coordinates to fragment shader for interpolation (out)
- Use `sampler2D` type

```
// Vertex shader
...
in  vec2 VertexTexcoords;  // nuevo atributo
out vec2 texCoords;

void main()  {
   ...
   // se asignan las coordenades de textura del vertice a la variable texCoords
   texCoords   = VertexTexcoords;
}

// Fragment shader
...
uniform sampler2D myTexture;     // la textura
in vec2 texCoords;               // coordenadas de textura interpoladas

void main()
{
   ...
   // acceso a la textura para obtener un valor de color RGBA
   fragmentColor = texture(myTexture, texCoords);
}
```

Universitat de Girona

Texture and Texel

- Each pixel in a texture map is called a Texel (TEXture-ELement)
- Each Texel is associated with a (u,v) 2D texture coordinate
- The range of u, v is [0.0,1.0] due to normalization



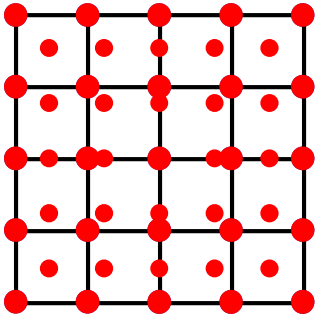
Universitat de Girona

How do we get **F**(u,v) from F[i,j]?

- We are given a discrete set of values:
  - **F**[i,j] for i=0,…,N, j=0,…,M
- Nearest neighbor:
  - **F**(u,v) = **F**[round(N*u), round(M*v)]
- Linear Interpolation:
  - i = floor(N*u), j = floor(M*v)
  - interpolate from **F**[i,j], **F**[i+1,j], **F**[i,j+1], **F**[i+1,j]
- This is a filtering problem !

Universitat de Girona

Interpolation

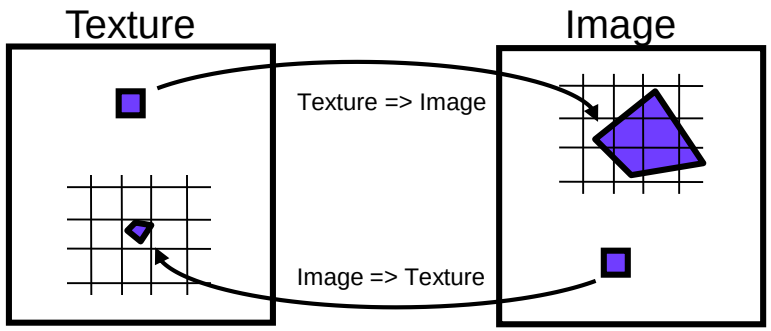


Nearest neighbor

Linear Interpolation

Universitat de Girona

Filtering Textures



Footprint changes from pixel to pixel: no single filter

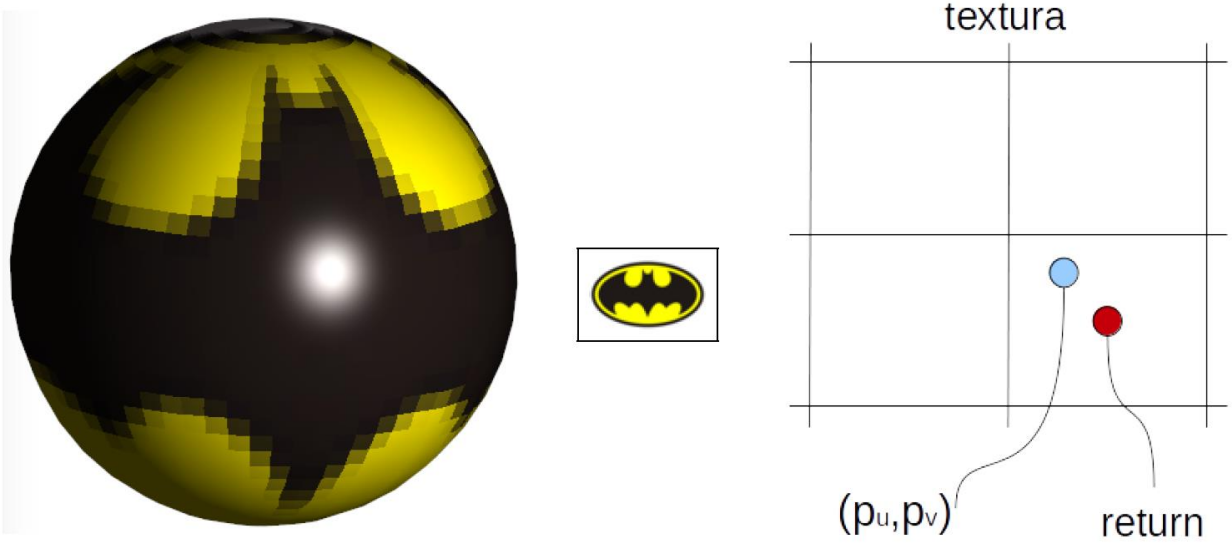
Resampling theory:

- Magnification: Interpolation
  - One texel correspond to many pixels
- Mignification: Averaging
  - One pixel correspond to many texels

Universitat de Girona

Magnification

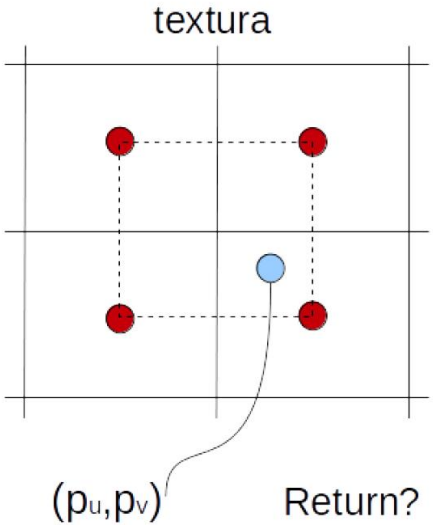
- Box Filter
  - –Use closest texel
  - –Effect: “pixelated”



Universitat de Girona

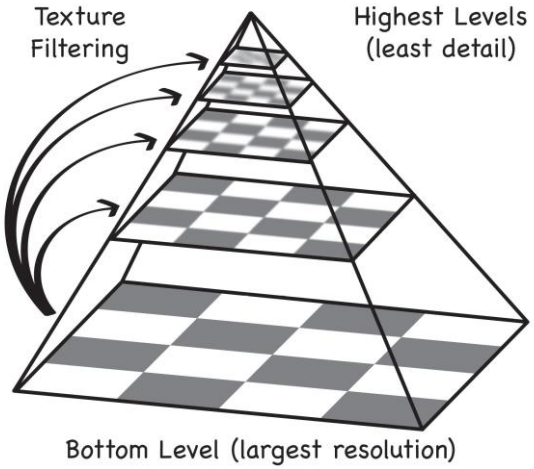
Magnification

- Bi-linear interpolation
  - –Interpolate linearly through 4 values
  - –Effect: “Blurring”

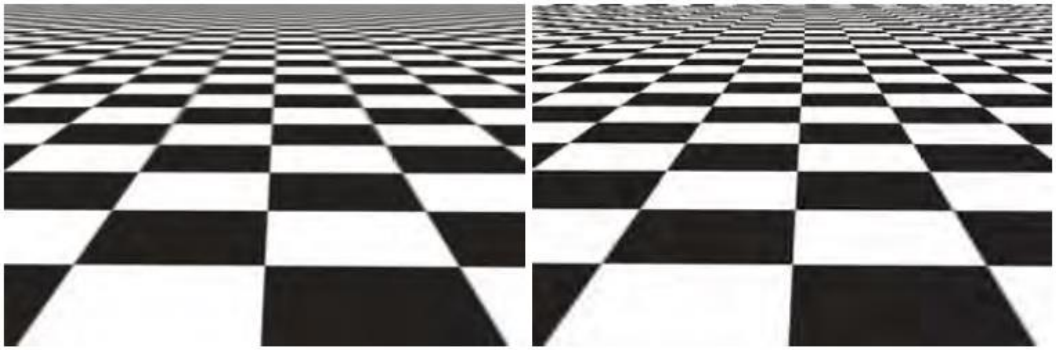


Universitat de Girona

Minification



- Same filters as Magnification
- Mipmaping
  - –Use a set of textures at different resolutions
  - –Select which texture use at run time



Mipmapping

Bilinear

Universitat de Girona

Create texture from algorithms: “on the fly”

Do not use fixed images

PROCEDURAL TEXTURE

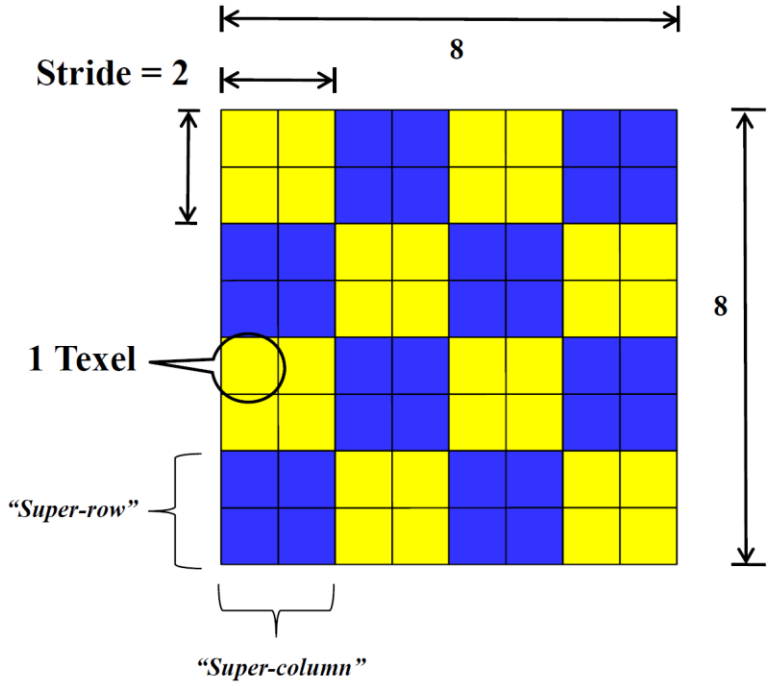
Universitat de Girona

Procedural Textures

- Advantages
  - –Extremely compact in size
  - –No fixed resolution (no matter how close you look at it)
  - –Can cover arbitrarily large areas (without repetition patterns)
  - –Can be parametric, generate classes of texture
- Disadvantages
  - –Difficult to build and debug
  - –Could be unpredictable
  - –Can be slow to compute
  - –Can produce aliasing

Universitat de Girona

Checkerboard Texture



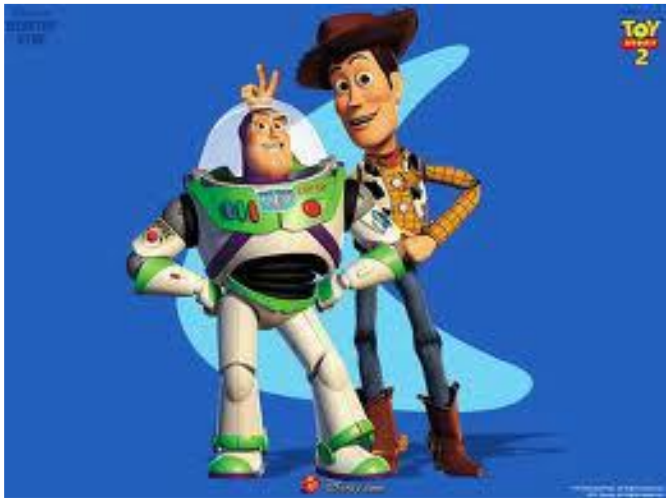
Observations:

```
if (superCol==superRow)
    color = yellow;

if (even(superCol) && even(superRow))
    color = yellow;

if (odd(superCol) && odd(superRow))
    color = blue;

if (even(superCol+superRow))
    color = yellow ;
else color = blue ;
```

Universitat de Girona

Toy Ball Shader

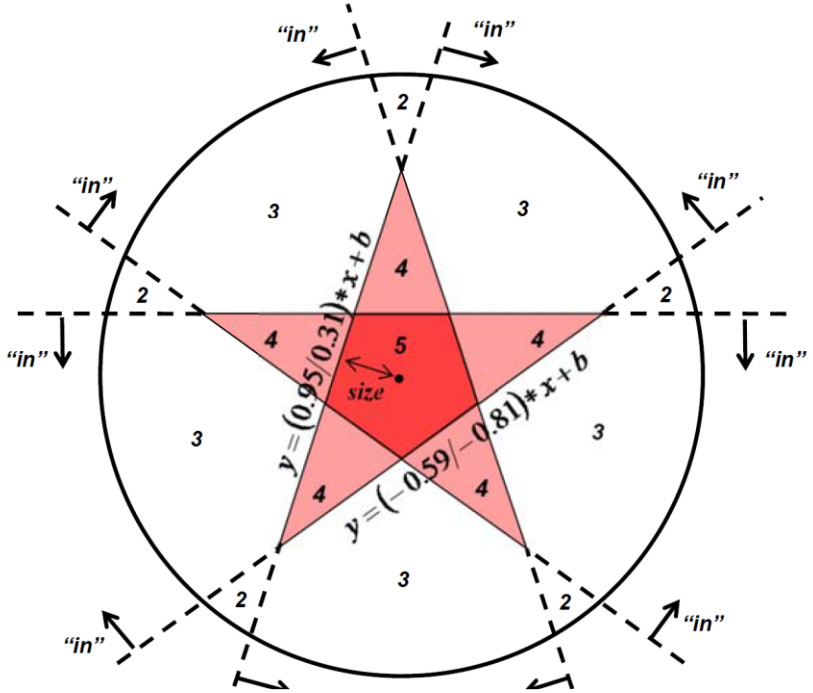
- Pixar classic Toy Ball



Universitat de Girona

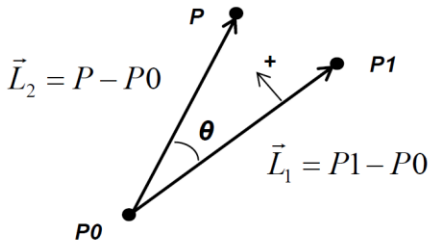
Toy Ball Shader

- Finding points inside the Star



A point is in the star if and only if it lies on the "in" side of at least 4 "half-space" lines.

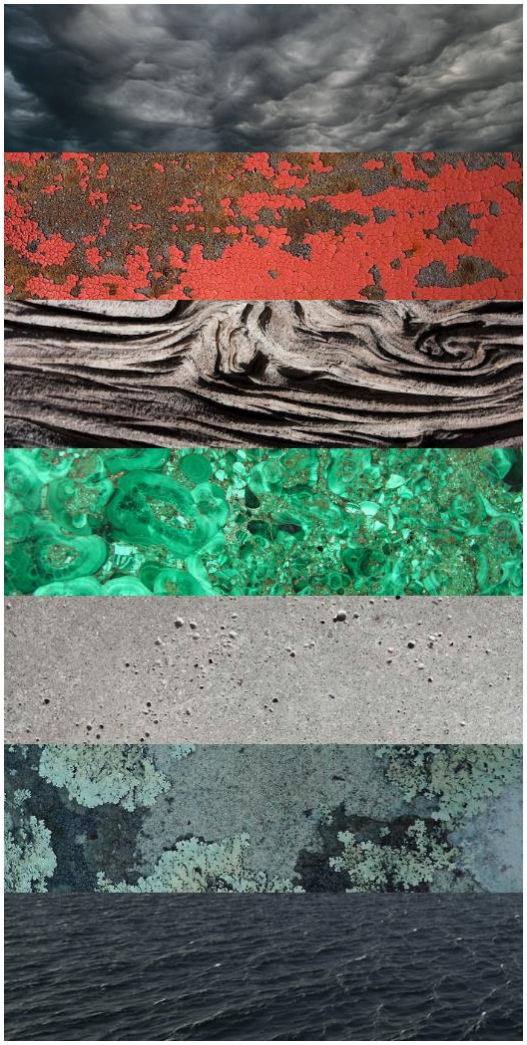
A point P lies on the positive side of a line P0-P1 if and only if $L_1 \bullet L_2$ is positive:



Universitat de Girona

Noise

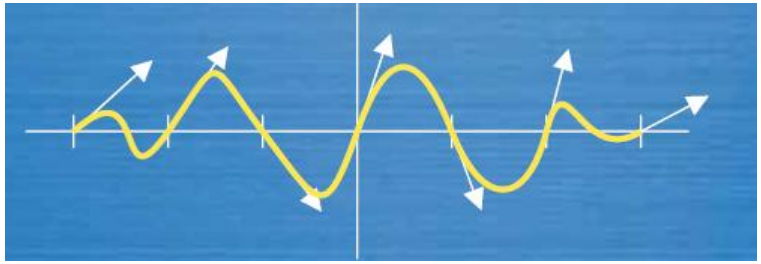
- Problem
  - How to build noise functions to simulate natural processes
- Ken Perlin
  - *"An Image Synthesizer"* SIGGRAPH 1985
  - He was trying to make realistic effects for the movie *"TRON"*
  - Later he won an Oscar (1997)!

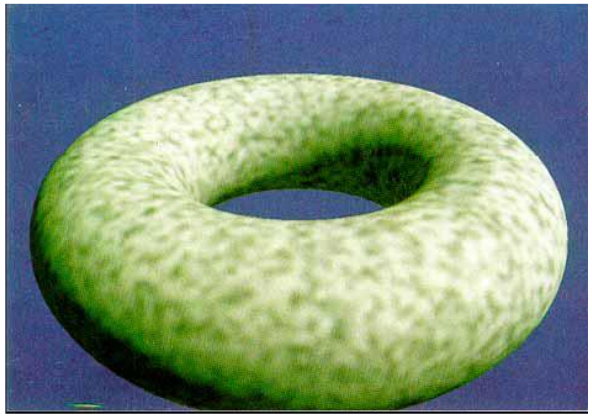


Universitat de Girona

Making Noise

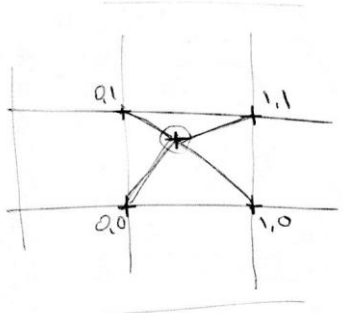
- Use random number to generate a stochastic process
- When x,y,z integer = integer lattice
- On lattice
  - Noise=0
  - Random gradient
    - Hash(x,y,z)
- Off-lattice
  - Smooth interpolation
  - Cubic interpolation



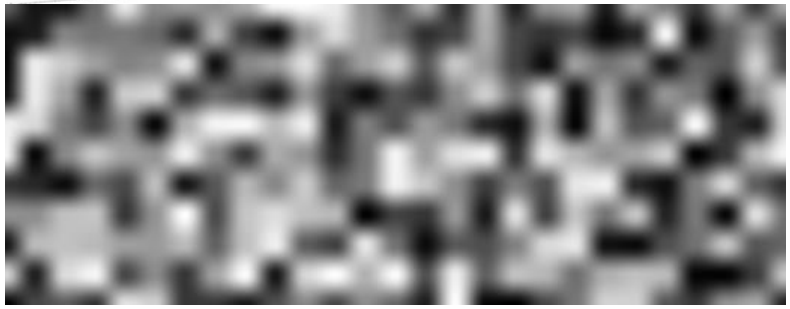


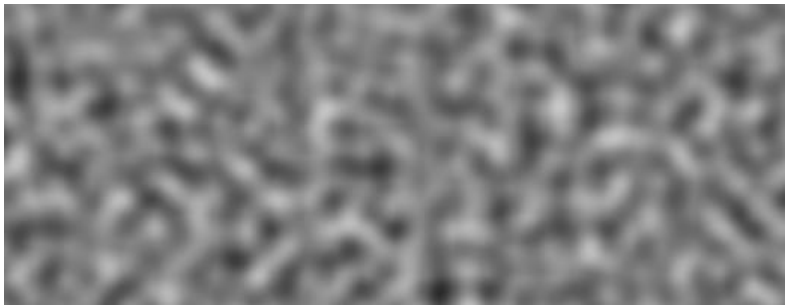
Universitat de Girona

Making Noise



- Value Noise
  - Cubic interpolation for value
  - 2D Noise
    - 4 points
  - 3D Noise
    - 8 points
- Gradient Noise
  - Interpolate directions

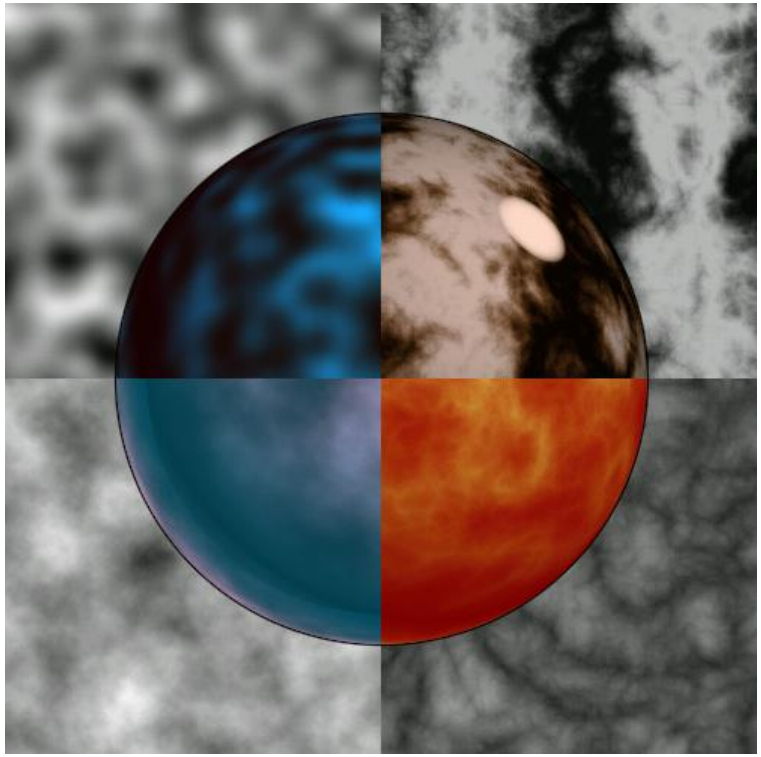




Universitat de Girona

Weighted Sums: Different effects

noise:
-Worn metal
-Water wave (gradient)



Sin(x + Sum[1/f *|noise|]):
-Turbulent flows
-Fire
-Marble

Sum[1/f * noise]:
-Rock
-Mountains
-Clouds

Sum[1/f * |noise|]:
-Turbulent flows
-Fire
-Marble
-Clouds

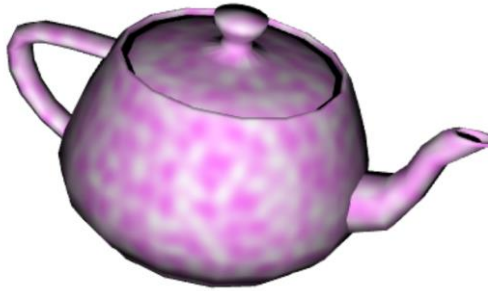
K. Perlin

Universitat de Girona

WebGL examples

- Implemented as Fragment Shaders
  - http://learnwebgl.brown37.net/10_surface_properties/texture_mapping_procedural.html
  - http://math.hws.edu/graphicsbook/demos/c7/procedural-textures.html

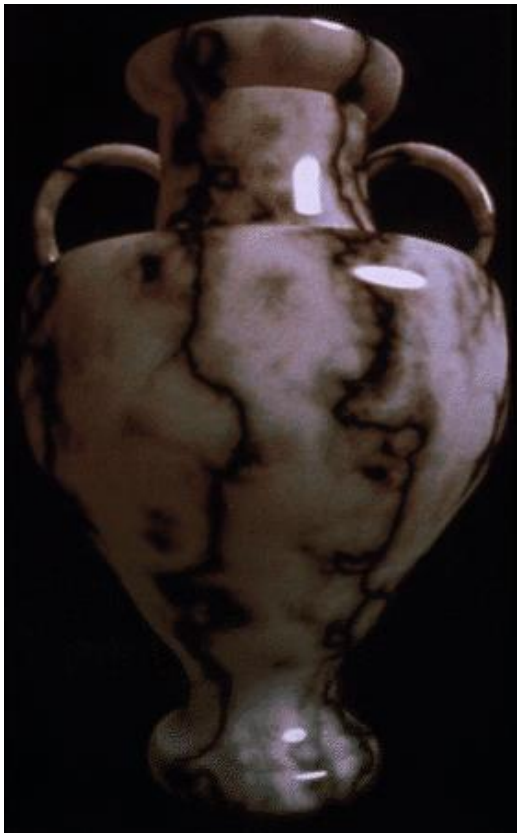
  - The Book of Shaders
  - https://thebookofshaders.com/11/

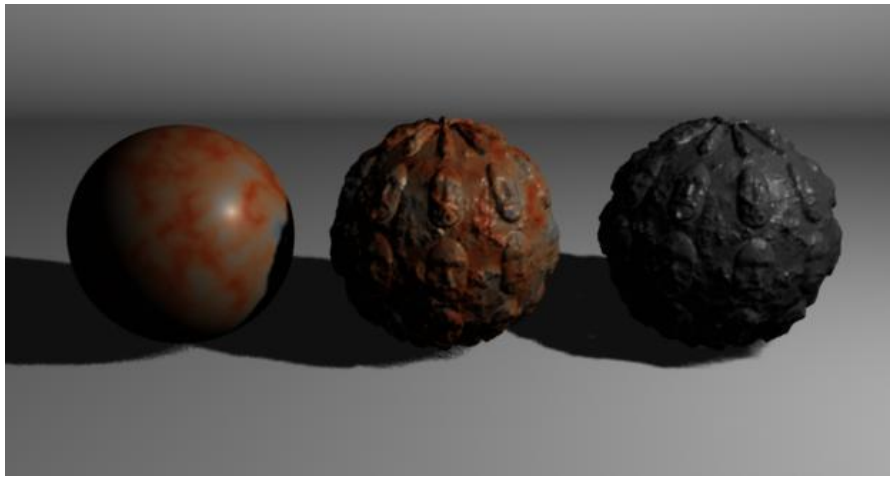


Universitat de Girona

Solid Textures

- 3D maps: extend coordinate texture (s, t, r)
- En general procedurales: marmol, madera



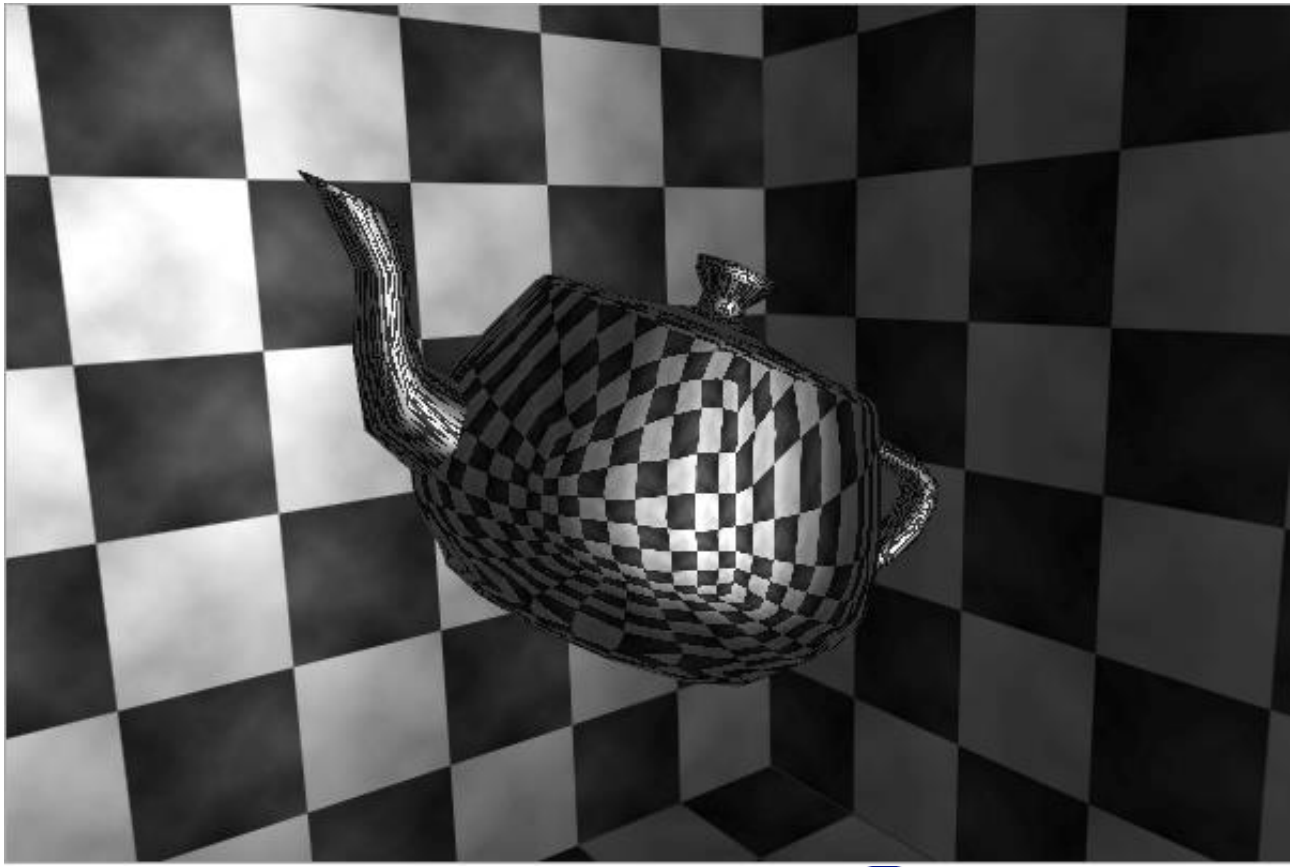


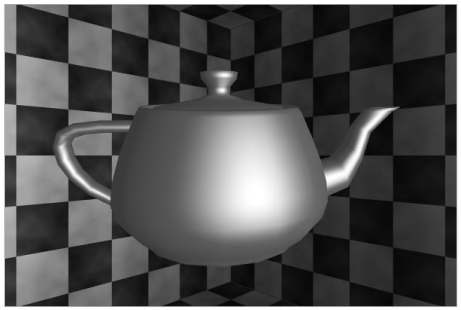
Universitat de Girona

REFLECTIONS AND LIGHTING EFFECTS

Universitat de Girona

Reflections





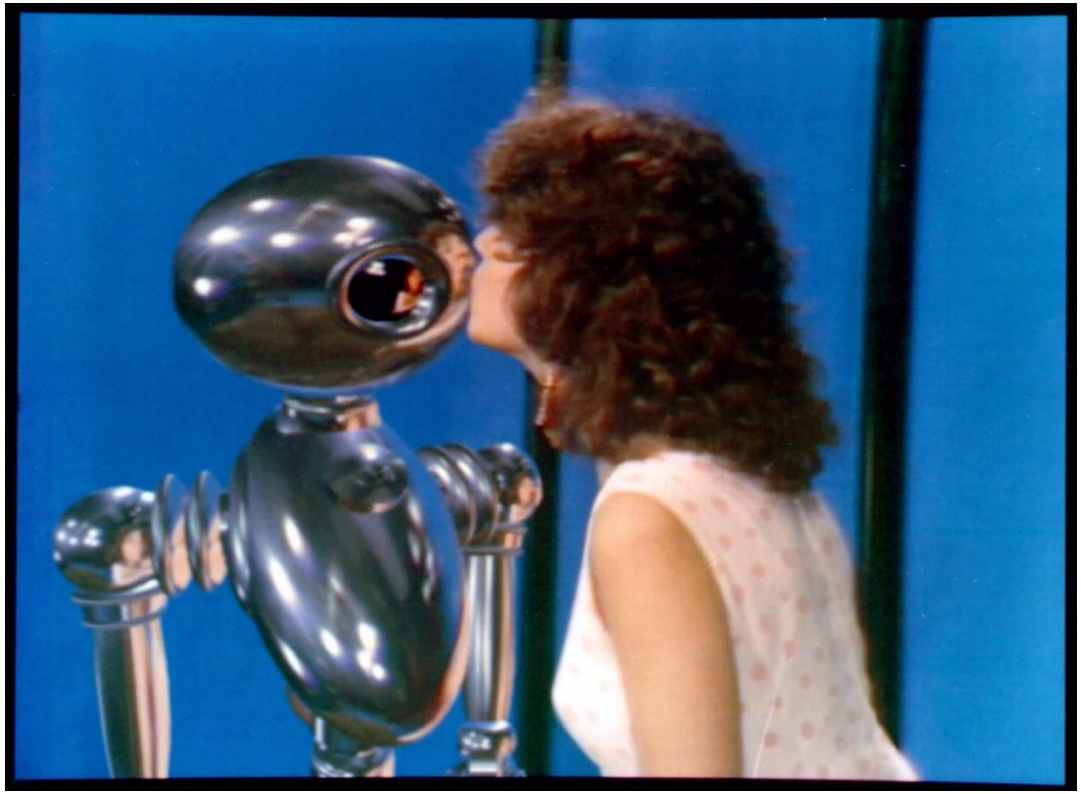
Universitat de Girona

Reflection Maps

Blinn and Newell, 1976

Universitat de Girona

Environment Maps



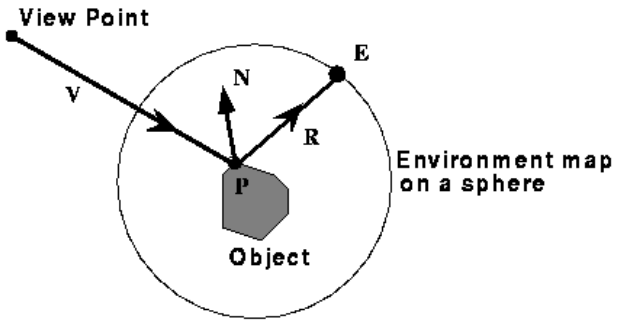
***Interface*, Chou and Williams (ca. 1985)**

https://youtu.be/J7U_ZQop8b4

Universitat de Girona

Environment Maps

- Consider illumination distant: light only depends on reflection
- Index to a texture map from reflection map
  - No texture coordinates !
- Maps could also be pre computed as irradiance



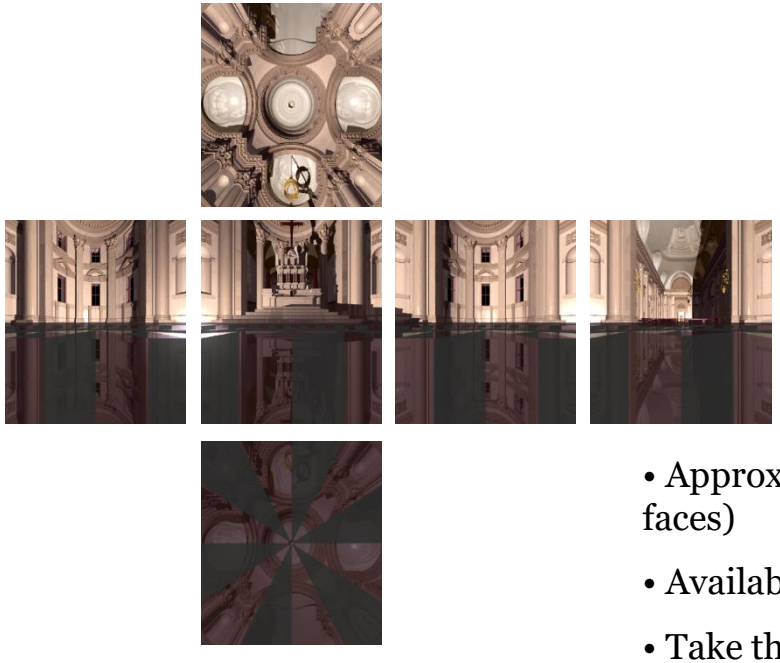
Universitat de Girona

Environment Maps

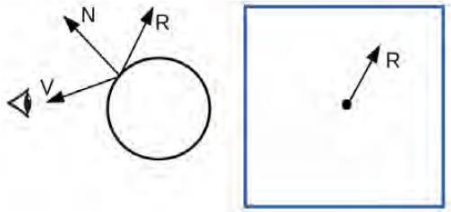
- Es una aproximació, problemes:
  - No considera interreflexions entre objectes de l'escena
  - No considera autoreflexions
  - Problemes amb objectes còncaves
- Funciona be per objectes corbats
- Cada objecte podria tenir el seu mapa
- Diferents formes de mapear el entorn
  - Cúbic
  - Esfèric
  - Paraboloide

Universitat de Girona

Environment Map Cubic



- Approximate environament with a cube (6 faces)
- Available in WebGL 2 (SamplerCube)
- Take the corresponding imatge from reflection direction



Universitat de Girona

Spherical Maps

- Es poden representar amb una foto d'una esfera que reflectí tot el entorn
- Paul DeBevec, www.debevec.org

Universitat de Girona

Environment Map Approximation

Ray tracing

Environment Map

Auto reflexions no es consideren en environment maps

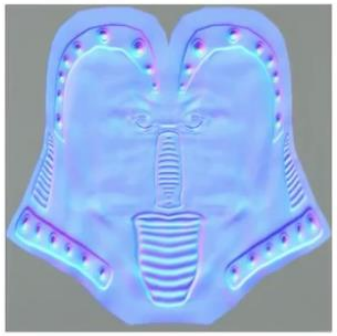
Universitat de Girona

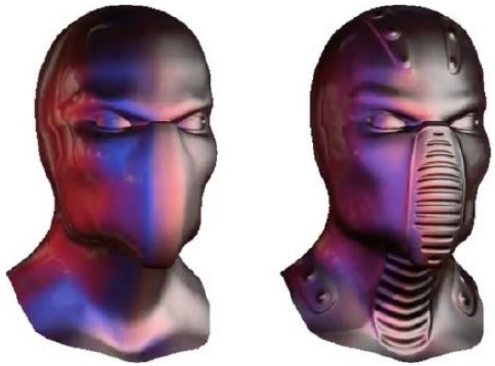
Reflections @ Grand Turismo 5

Universitat de Girona

Bump Mapping (or Normal Mapping)

- Modify normal from a texture map
- Texture used as field of values
- Provides relief effects

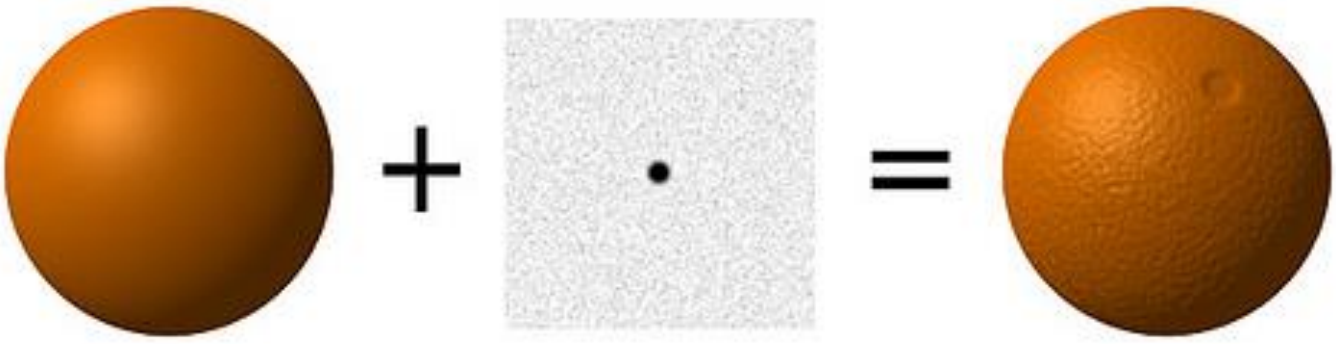




Universitat de Girona

Bump Mapping

- Bump mapping is a technique used to add detail to a surface
  - Think roughness / texture

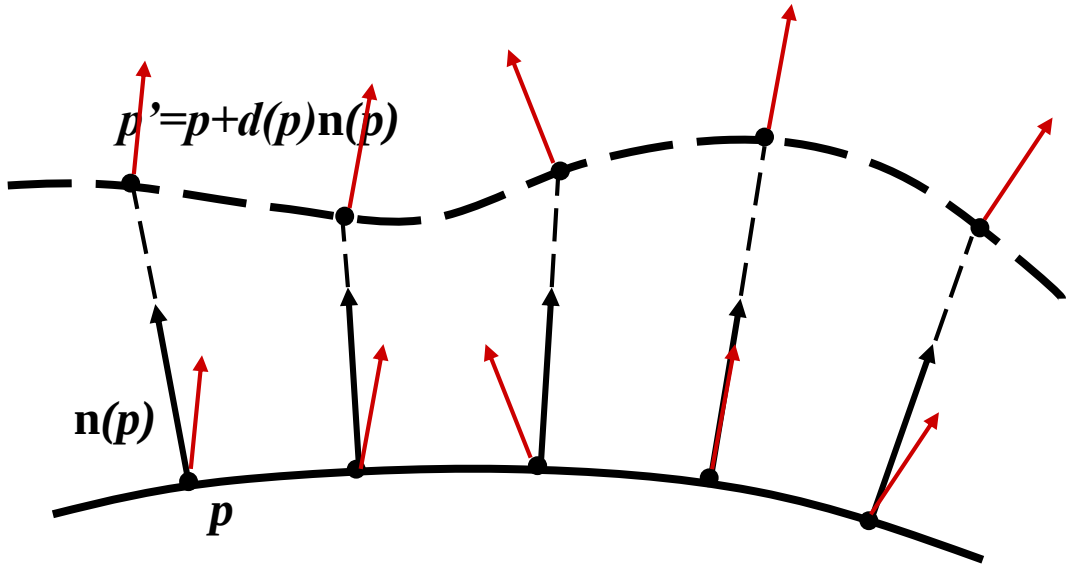


Does not change silhouette edges

Universitat de Girona

Bump mapping and Displacement mapping

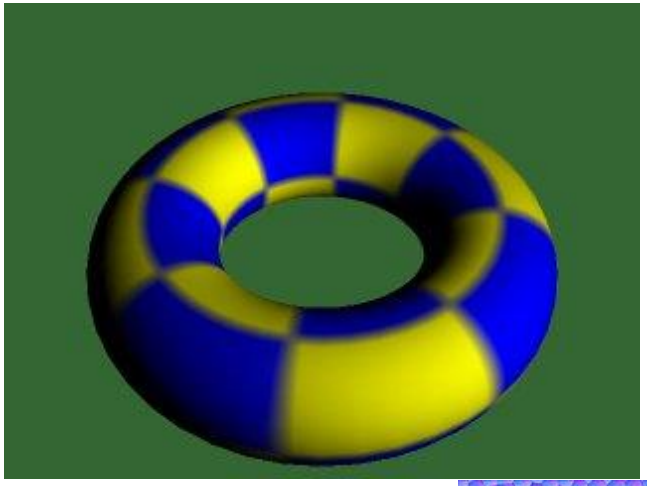
- Displacement mapping adds geometrical details to surfaces

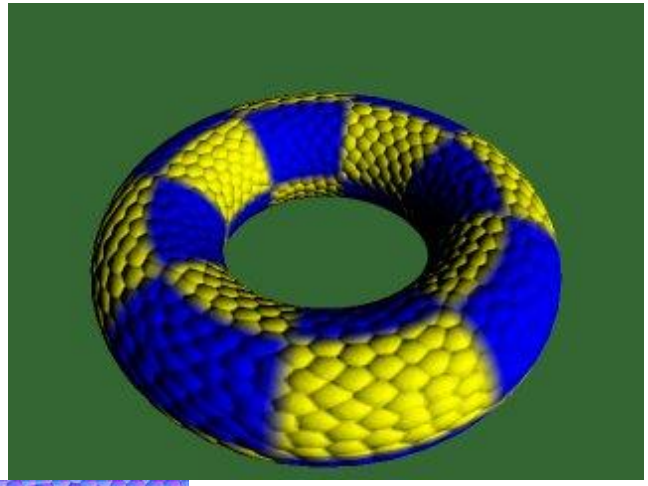


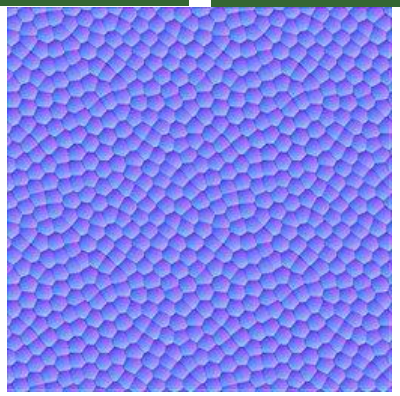
- Bump mapping: augments geometrical details to a surface without changing the surface itself
- It works well when the displacement is small

Universitat de Girona

Example Bump Mapping





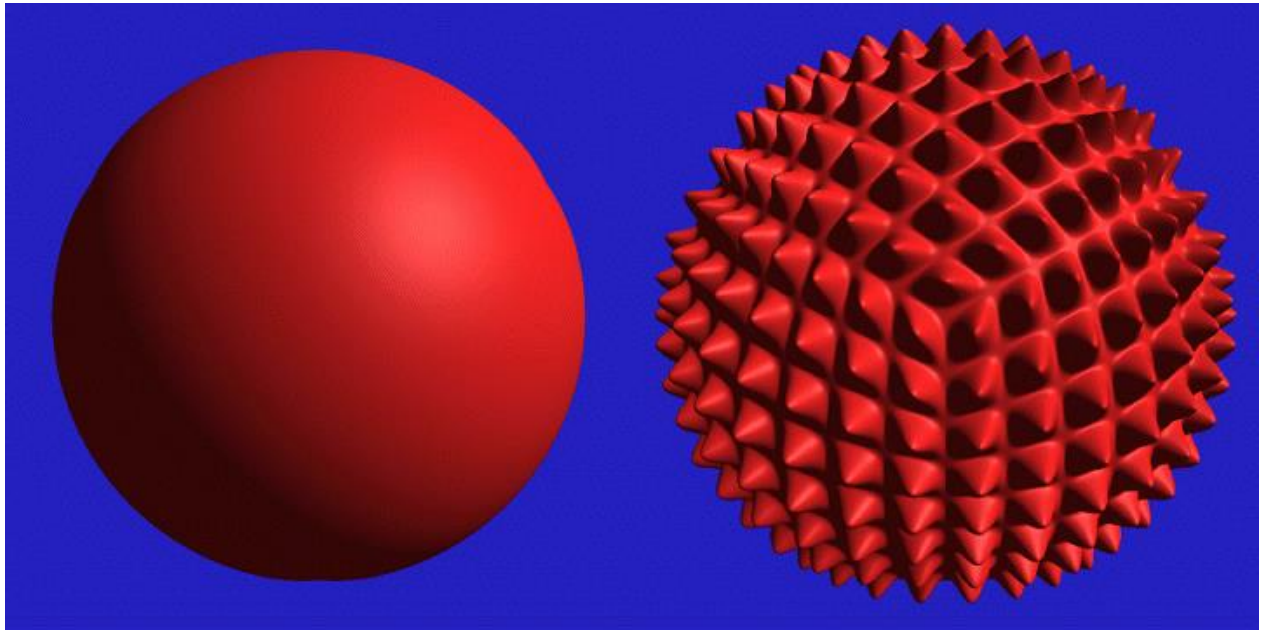


https://webglfundamentals.org/webgl/lessons/webgl-qna-apply-a-displacement-map-and-specular-map.html
Example

Universitat de Girona

Examples Displacement Mapping

Modify geometry



Universitat de Girona

Displacement Mapping

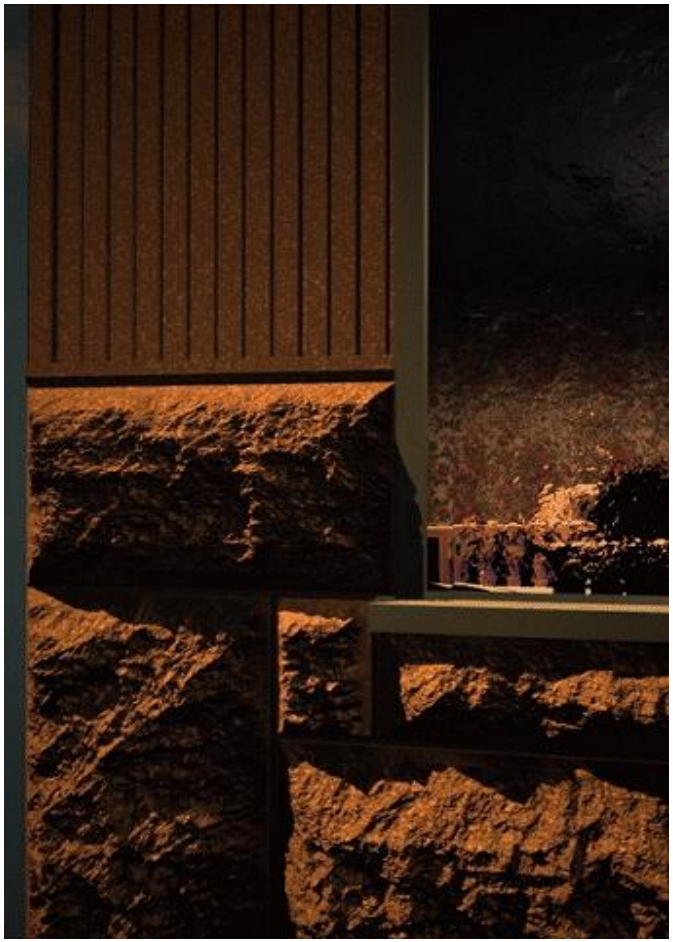


Image from

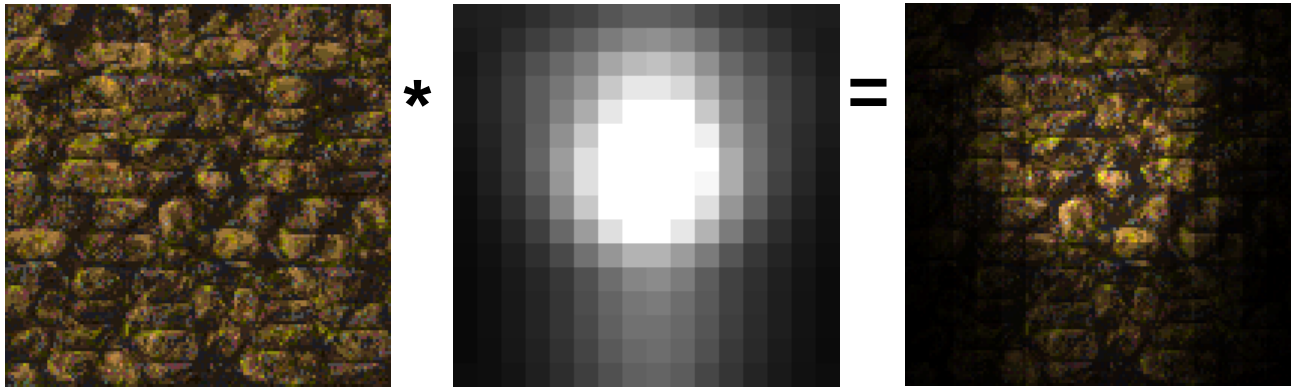
Geometry Caching for Ray-Tracing Displacement Maps

by Matt Pharr and Pat Hanrahan.

Universitat de Girona

Illumination Maps

- Store illumination computed in texture maps
- Combines with texture

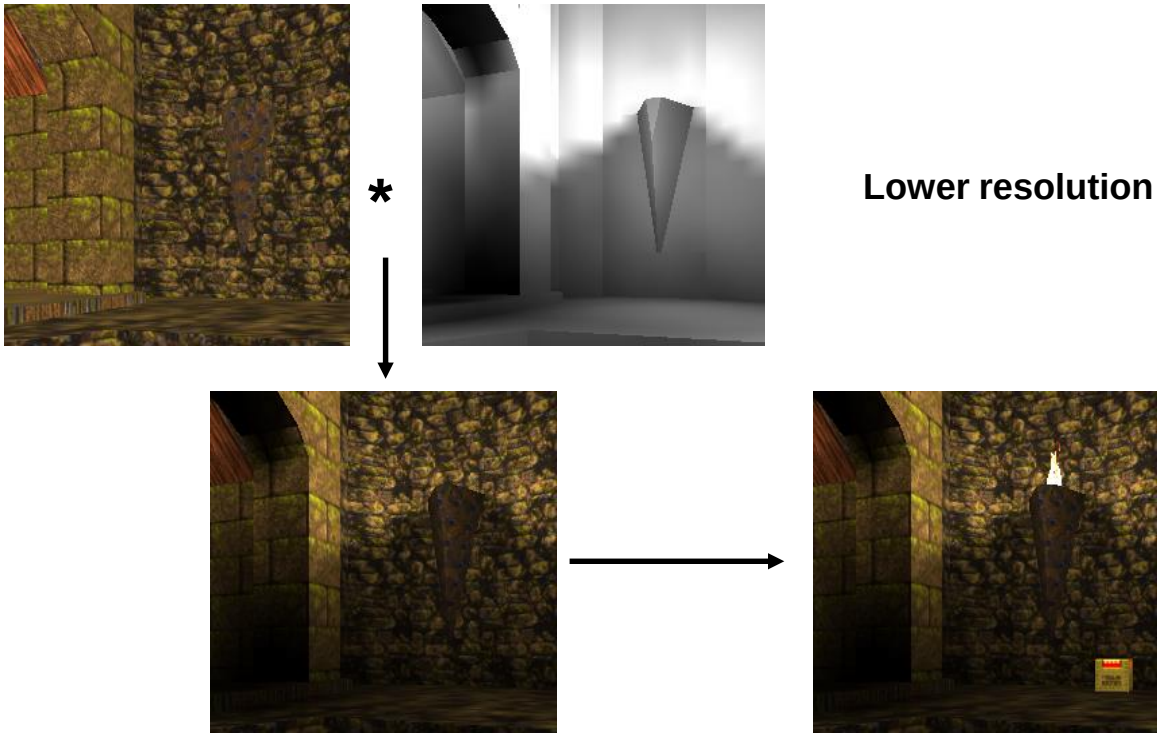


Reflectance **Irradiance** **Radiosity**

$$\rho(x) \qquad E(x) \qquad B(x)$$

Universitat de Girona

Quake Light Maps

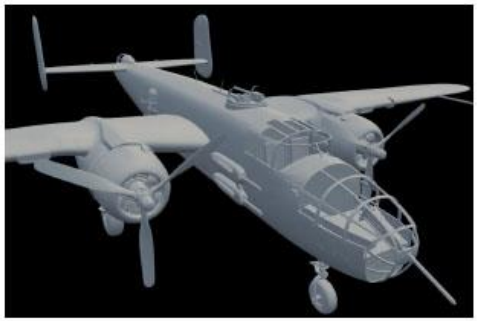



without bump mapping
with bump mapping

Universitat de Girona

Ambient Occlusion Maps

- Efectes de il·luminació global amb mapes precalculats



From Production ready global illumination, Hayden Landis, ILM

Universitat de Girona

Final Remarks

- Texture mapping techniques
  - Material texture
  - Bump and displacement mapping
  - Environment mapping
  - Lighting maps
- All are crucial for realistic interactive techniques
- NEXT:
  - Shadow maps
  - Raytracing
  - Global Illumination

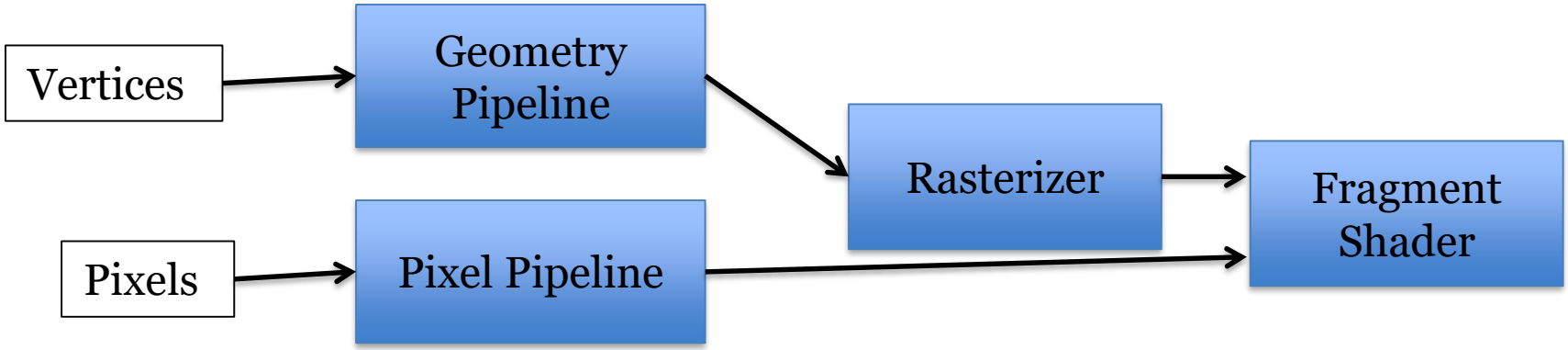
Universitat de Girona

TEXTURE IN WEBGL

Universitat de Girona

Texture Mapping and the OpenGL Pipeline

- Images and geometry flow through separate pipelines that join at the rasterizer
  - "complex" textures do not affect geometric complexity



Universitat de Girona

Applying Textures procedure

- Three basic steps for applying a texture
  1. specify the texture

»read (or generate) the image

»assign to texture

»enable texturing

  2. assign texture coordinates to vertices
  3. specify texture parameters

»wrapping, filtering

Universitat de Girona

Specifying a Texture Image

- Define a texture image from an array of *texels* (texture elements) in CPU memory
- Use an image in a standard format (such as JPEG)
  - Scanned image
  - Generate by application code
- WebGL 1.0 supports only 2 dimensional texture maps
  - no need to enable as in desktop OpenGL
- WebGL 2 supports 3 dimensional texture maps
- Desktop OpenGL supports 1-4 dimensional texture maps

Universitat de Girona

Define Image as a Texture

```
gl.texImage2D( target, level, components,
    w, h, border, format, type, texels );
```

`target:` type of texture, e.g. `gl.TEXTURE_2D`
`level:` used for mipmapping
`components:` elements per texel
`w, h:` width and height of `texels` in pixels
`border:` used for smoothing
`format and type:` describe texels
`texels:` pointer to texel array

Example:

```
gl.texImage2D(gl.TEXTURE_2D, 0, 3, 512, 512, 0, gl.RGB, gl.UNSIGNED_BYTE,
my_texels);
```

Universitat de Girona

Texture Objects

- Have WebGL store your images
  - one image per texture object
- Create an empty texture object

```
var texture = gl.createTexture();
```

- Bind textures before using

```
gl.bindTexture( gl.TEXTURE_2D, texture );
```

Universitat de Girona

Specifying a Texture Image

•Define a texture image from an array of *texels* in CPU memory

```
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, texSize,
        texSize, 0,  gl.RGBA, gl.UNSIGNED_BYTE, image);
```

• Define a texture image from an image in a standard format memory specified

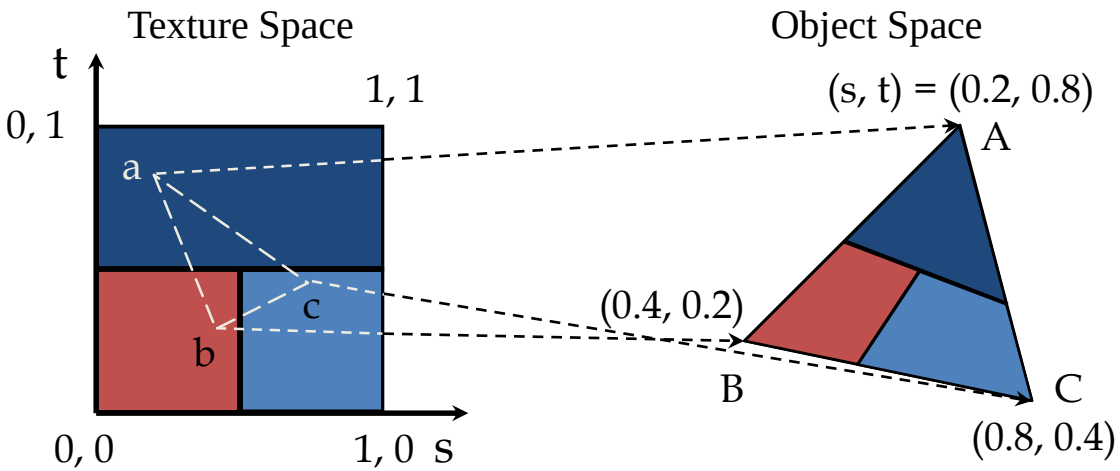
–For example, with the <image> tag in the HTML file

```
var image = document.getElementById("texImage");

gl.texImage2D( gl.TEXTURE_2D, 0, gl.RGB, gl.RGB, gl.UNSIGNED_BYTE, image );
```

Universitat de Girona

Mapping a Texture

- Based on parametric texture coordinates
- Specify as a 2D vertex attribute



Universitat de Girona

Cube Example

```
var exampleCube = {  // 24 vértices, 12 triángulos

  "vertices" : [-0.5, -0.5,  0.5,   0.0,  0.0,  1.0,   0.0, 0.0,
                 0.5, -0.5,  0.5,   0.0,  0.0,  1.0,   1.0, 0.0
                 0.5,  0.5,  0.5,   0.0,  0.0,  1.0,   1.0, 1.0,
                -0.5,  0.5,  0.5,   0.0,  0.0,  1.0,   0.0, 1.0
                 0.5, -0.5,  0.5,   1.0,  0.0,  0.0,   0.0, 0.0,
                 0.5, -0.5, -0.5,   1.0,  0.0,  0.0,   1.0, 0.0,
                 0.5,  0.5, -0.5,   1.0,  0.0,  0.0,   1.0, 1.0,
                 0.5,  0.5,  0.5,   1.0,  0.0,  0.0,   0.0, 1.0,
                 0.5, -0.5, -0.5,   0.0,  0.0, -1.0,   0.0, 0.0,
                -0.5, -0.5, -0.5,   0.0,  0.0, -1.0,   1.0, 0.0,
                -0.5,  0.5, -0.5,   0.0,  0.0, -1.0,   1.0, 1.0,
                 0.5,  0.5, -0.5,   0.0,  0.0, -1.0,   0.0, 1.0,
                -0.5, -0.5, -0.5,  -1.0,  0.0,  0.0,   0.0, 0.0,
                -0.5, -0.5,  0.5,  -1.0,  0.0,  0.0,   1.0, 0.0,
                -0.5,  0.5,  0.5,  -1.0,  0.0,  0.0,   1.0, 1.0
                -0.5,  0.5, -0.5,  -1.0,  0.0,  0.0,   0.0, 1.0,
                -0.5,  0.5,  0.5,   0.0,  1.0,  0.0,   0.0, 0.0,
                 0.5,  0.5,  0.5,   0.0,  1.0,  0.0,   1.0, 0.0,
                 0.5,  0.5, -0.5,   0.0,  1.0,  0.0,   1.0, 1.0,
                -0.5,  0.5, -0.5,   0.0,  1.0,  0.0,   0.0, 1.0,
                -0.5, -0.5, -0.5,   0.0, -1.0,  0.0,   0.0, 0.0,
                 0.5, -0.5, -0.5,   0.0, -1.0,  0.0,   1.0, 0.0,
                 0.5, -0.5,  0.5,   0.0, -1.0,  0.0,   1.0, 1.0
                -0.5, -0.5,  0.5,   0.0, -1.0,  0.0,   0.0, 1.0],

  "indices" :  [  0,  1,  2,  0,  2,  3,
                  4,  5,  6,  4,  6,  7,
                  8,  9, 10,  8, 10, 11,
                 12, 13, 14, 12, 14, 15,
                 16, 17, 18, 16, 18, 19,
                 20, 21, 22, 20, 22, 23]

};
```

Texture coordinates

Universitat de Girona

Basic Shaders for 2D Texture

```
// Vertex shader
...
in  vec2 VertexTexcoords;  // nuevo atributo
out vec2 texCoords;

void main()  {
   ...
   // se asignan las coordenades de textura del vertice a la variable texCoords
   texCoords    = VertexTexcoords;
}

// Fragment shader
...
uniform sampler2D myTexture;     // la textura
in vec2 texCoords;               // coordenadas de textura interpoladas

void main()
{
   ...
   // acceso a la textura para obtener un valor de color RGBA
   fragmentColor = texture(myTexture, texCoords);
}
```

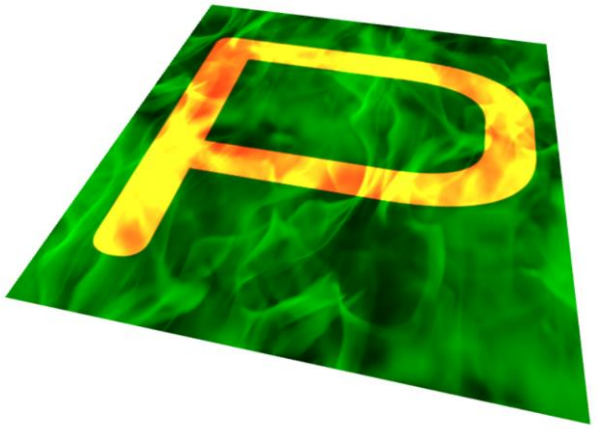
type

GLSL function

Universitat de Girona

Multiple textures

```
// Fragment shader
...
uniform sampler2D myTexture1, myTexture2;     // las texturas
in vec2 texCoords;     // coordenadas de textura interpoladas

void main()
{
   ...
   // acceso a la textura para obtener un valor de color RGBA
   fragmentColor = texture(myTexture1, texCoords) * texture(myTexture2, texCoords);
}
```



Universitat de Girona

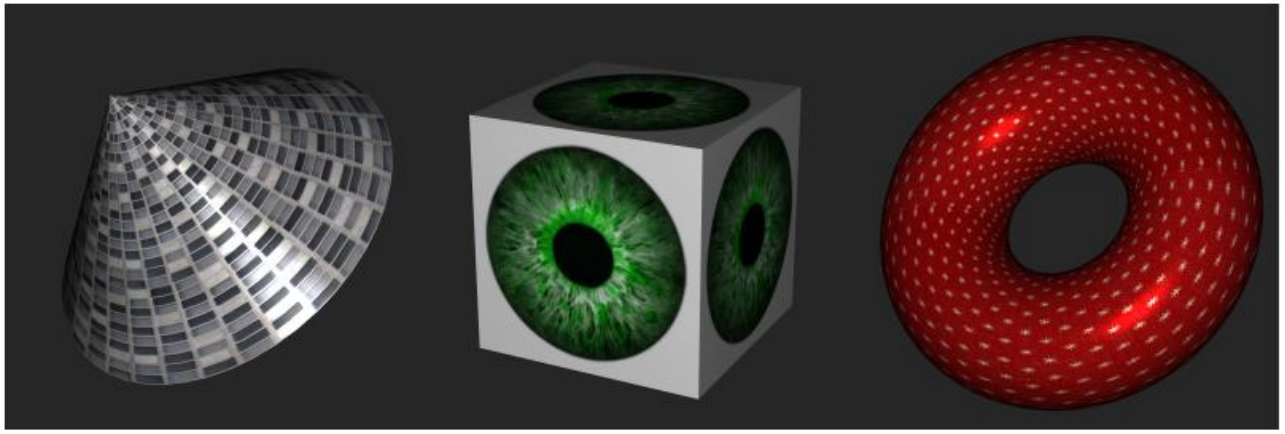
Texture Parameters

- WebGL has a variety of parameters that determine how texture is 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
  - Mipmapping allows us to use textures at multiple resolutions
  - Environment parameters determine how texture mapping interacts with shading

Universitat de Girona

Class examples: Applying Textures

1. Create a texture object
2. Assign the unit texture
3. Specify vertex coordinate and texture coordinates



Universitat de Girona

Create texture

```
// Crea un objeto textura
texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);

// Especifica la textura RGB
gl.texImage2D (gl.TEXTURE_2D, 0, gl.RGB, image.ancho, image.alto, 0, gl.RGB, gl.UNSIGNED_BYTE,
      image);

// Repite la textura tanto en s como en t
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.REPEAT);
gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.REPEAT);

// Filtrado
gl.texParameteri (gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR_MIPMAP_LINEAR);
gl.texParameteri (gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);
gl.generateMipmap(gl.TEXTURE_2D);
```

Assign texture: each object can have different textures

```
// Selecciona la unidad de textura 0
gl.activeTexture(gl.TEXTURE0);

// Asigna el objeto textura a la unidad de textura seleccionada
gl.bindTexture  (gl.TEXTURE_2D, texture);
```

Universitat de Girona

Link the shader variable with the texture unit

```
// Obtiene el indice de la variable del Shader de tipo sampler2D
program.textureIndex = gl.getUniformLocation(program, 'myTexture');

// Indica que myTexture del Shader use la unidad de textura 0
gl.uniform1i(program.textureIndex, 0);
```

Enable texture coordinate attribute

```
// se obtiene la referencia al atributo
program.vertexTexcoordsAttribute =
  gl.getAttribLocation ( program, "VertexTexcoords");

// se habilita el atributo
gl.enableVertexAttribArray(program.vertexTexcoordsAttribute);
```

Universitat de Girona

Procedural texture

- Examples
  - Damas
  - Enrejado
  - Rayado
  - Nubes