UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

CG Basics III – Textures

Bachelor's Degree in Video Game Design and Development

Textures

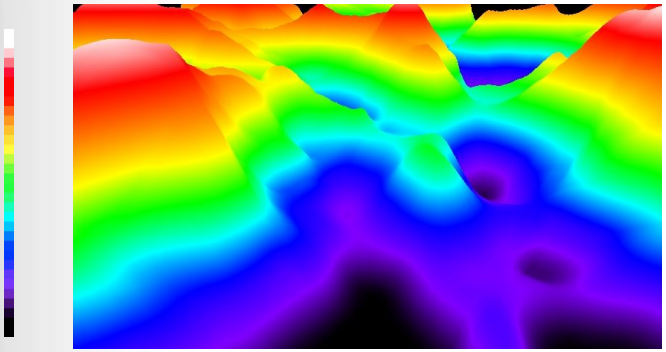
Geometry Image Textured geometry

Definition

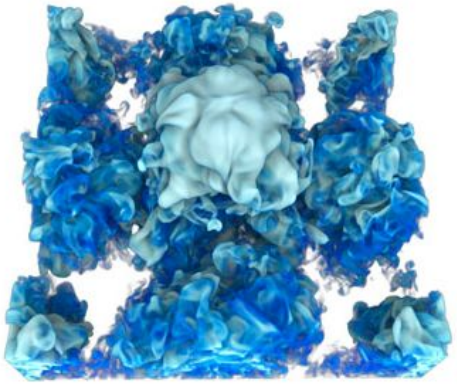
- A texture is an image (1D, 2D or 3D) that represents the features of a surface through a table
  - Each element of texture is called texel
  - The table can be:
    - Look-up table, map, bitmap

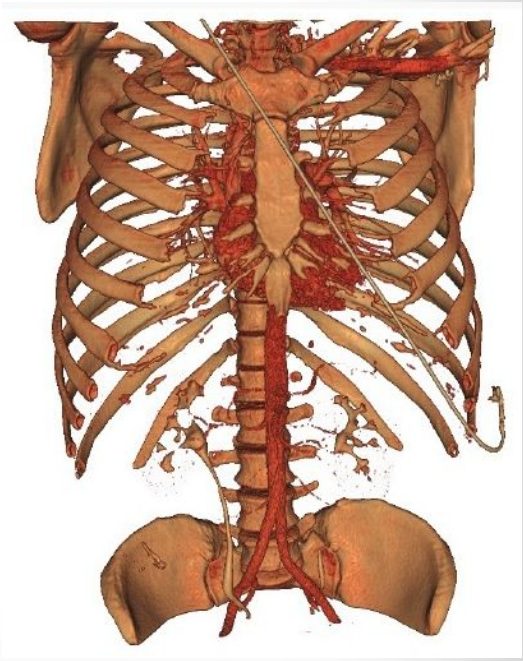
- Dimension of a texture
  - Number of index used to reference the elements of a texture
    - 1D → 1 value
    - 2D → 2 values (an image)
    - 3D → 3 values (a volume)

Types



1D texture

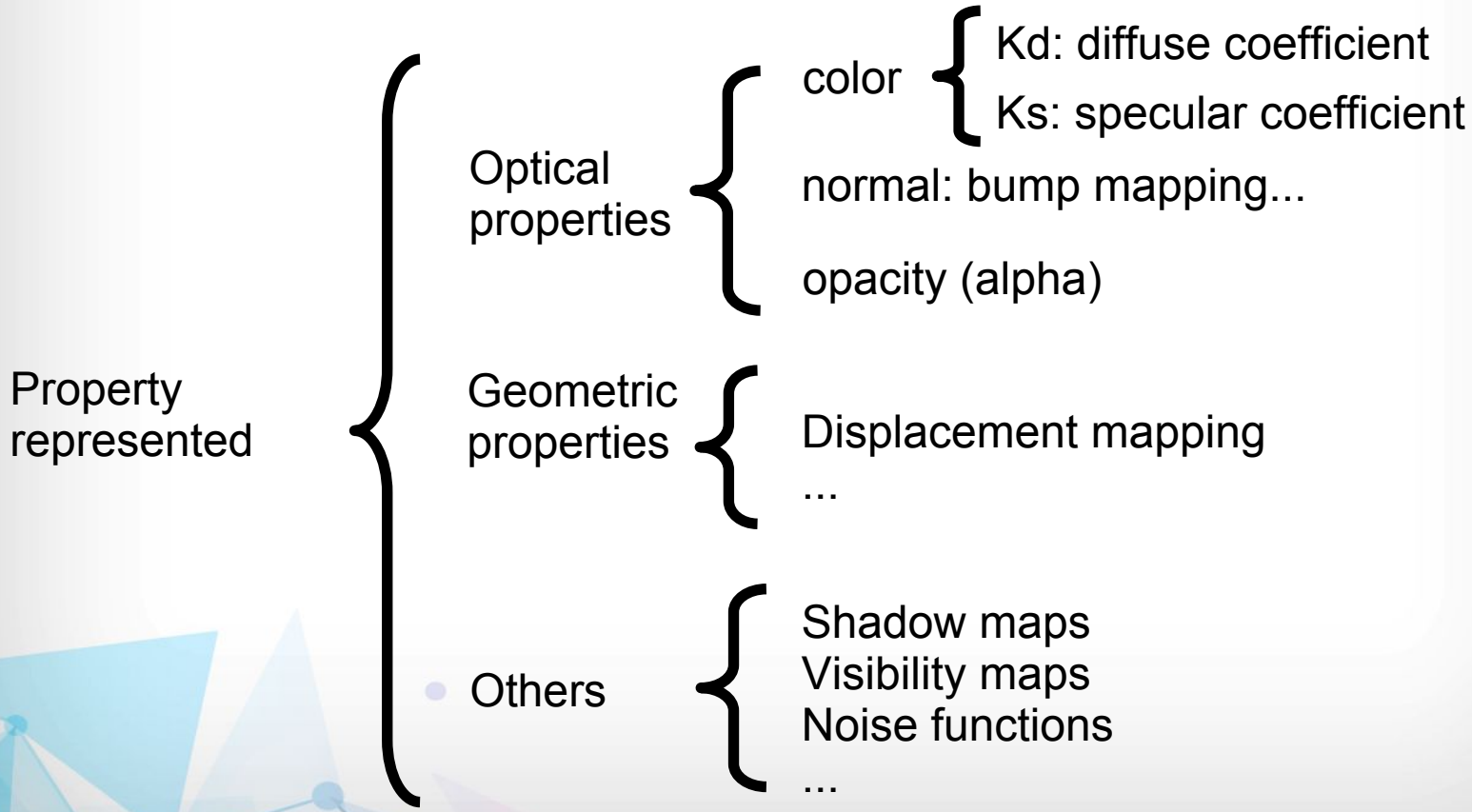




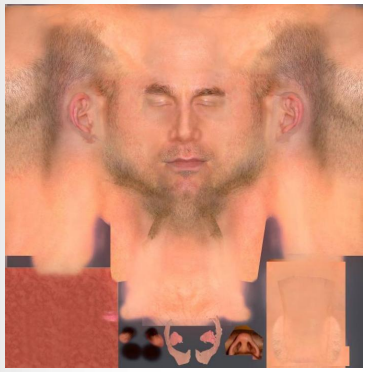
2D texture

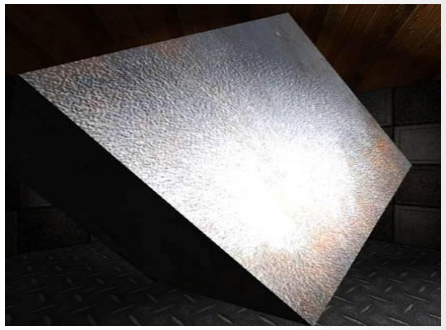
3D textures

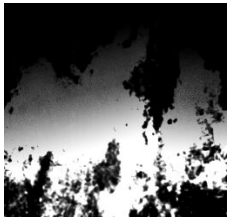
Properties Representation

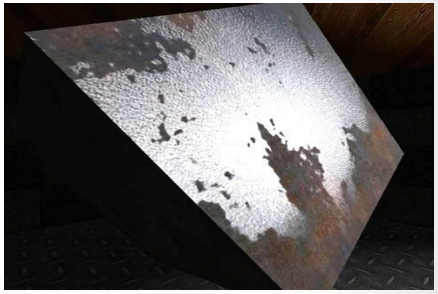


Properties Representation





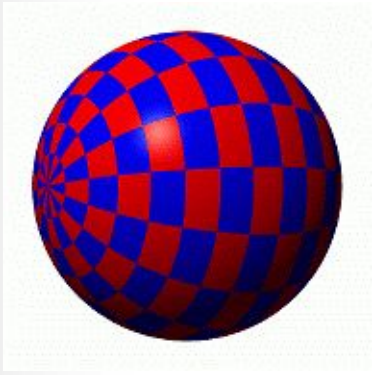




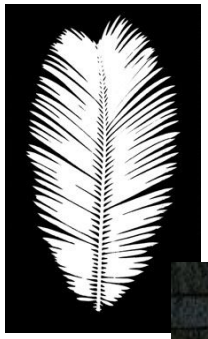
Kd (color map)

Ks (gloss map)

Properties Representation

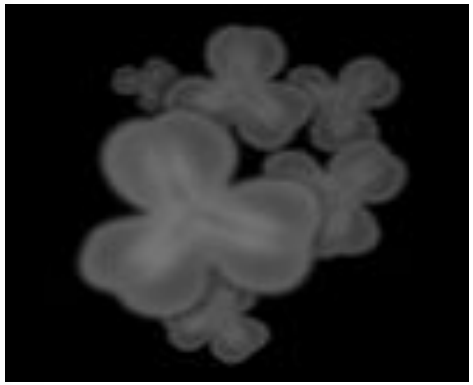


Bump mapping



Opacity map

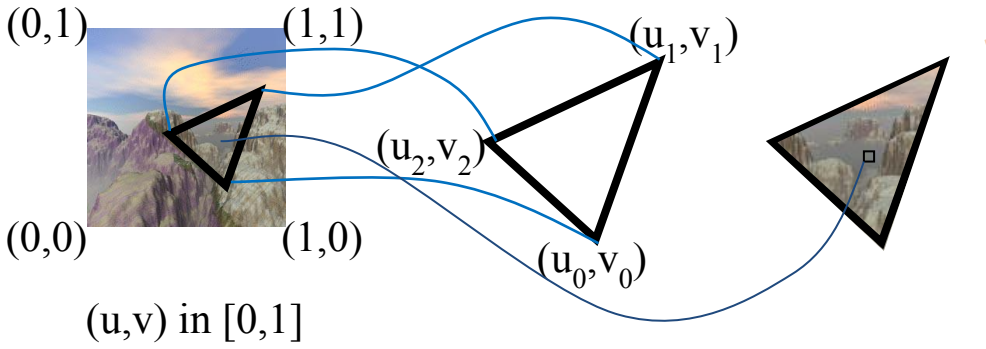
Properties Representation



Displacement map

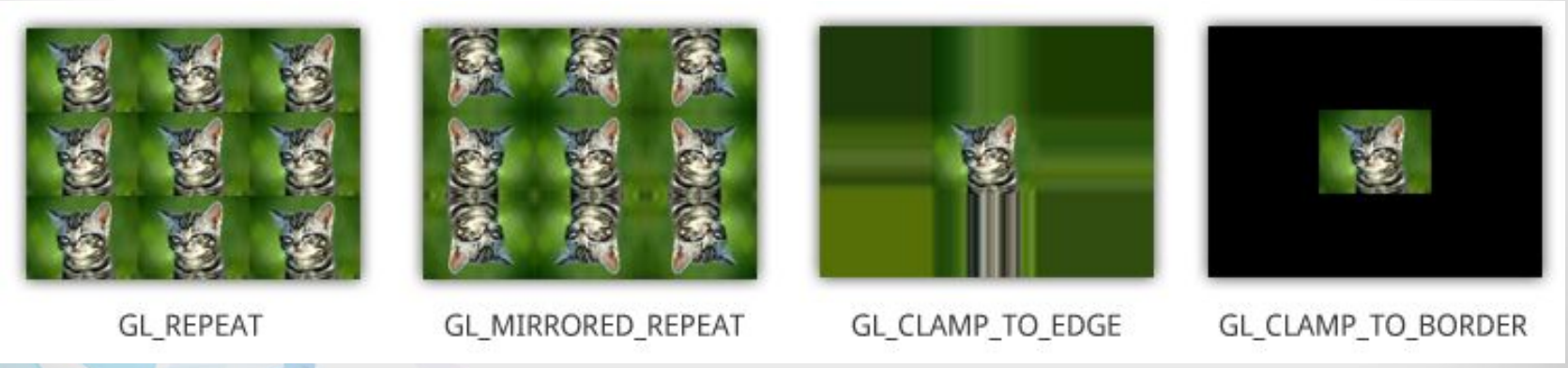
Texture Coordinates

- **Texture space**
  - Set of valid index
  - Texture coordinates are in the range
    - 1D: [0, 1] 2D: [0, 0] → [1, 1] 3D: [0, 0, 0] → [1, 1, 1]
  - Coordinates known as (s,t) or (u, v) in 2D



Wrapping

- What if (u, v) > 1.0 or < 0.0?
  - Different methods can be applied in this case
  - Repeat the texture
  - Texture mirrored
  - Texture clamped
  - Border clamped



Mapping

- Correspondence between the texels of a texture and the pixels of the image
- The most used method is known as inverse mapping
  - Based on setting a texture coordinate (u,v) for each vertex of the model
  - Can be done
    - During modeling
    - During visualization → sent to the pipeline along with the other attributes of a vertex (coordinates, normal, color…)

Filtering

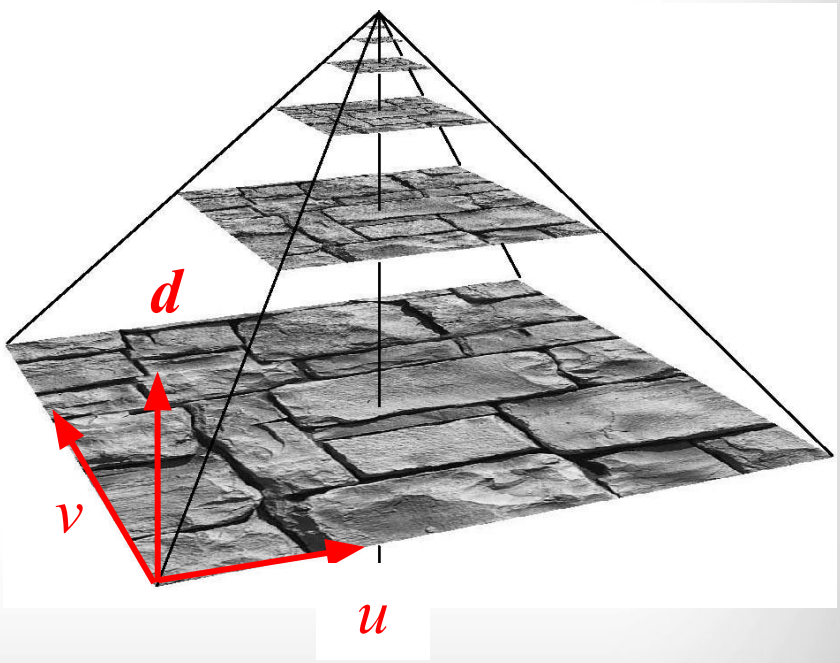
- Texture coordinates do not depend on the resolution of the image, as a consequence, they won't always match a pixel exactly
- To solve this, upsampling or downsampling of the texture is applied
  - OpenGL offers different filters to do this
    - NEAREST → returns the pixel that is closest to the coordinates
    - LINEAR → returns the average of the closest pixels to the coordinates

GL_NEAREST

GL_LINEAR

Filtering

- **Mipmapping**
  - Smaller copies of the texture that have been sized down and filtered in advanced
  - They are represented as a pyramid of images
  - They result in both
    - Higher quality
    - Higher performance



Textures in WebGL

- To use textures in WebGL, we must follow different steps:
  - Loading the texture on the GPU (*webGLStart()*)
    - Wrapping and filtering modes are defined in this step, by calling the function setTextureParams(…)

```
var myTexture;

function loadTextureOnGPU() {
    myTexture = gl.createTexture();
    myTexture.image = new Image();
    myTexture.image.onload = function () {
        setTextureParams(myTexture)
    }
    myTexture.image.src = "exampleImg.png";
}
```

```
function setTextureParams(texture) {
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA,
                  gl.UNSIGNED_BYTE, texture.image);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
                     gl.LINEAR);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
                     gl.LINEAR);
    gl.bindTexture(gl.TEXTURE_2D, null);
}
```

Textures in WebGL

- **To use textures in WebGL, we must follow different steps:**
  - **Define the mapping (*loadSceneOnGPU()*)**
    - **Set the texture coordinates for each vertex of the model**
    - **The procedure is the same as if we were specifying the colors of the vertices (create a new buffer and bind it, create the list with the texture coordinates…)**
  - **Send the texture coordinates to the rendering pipeline (*drawScene()*)**
    - **The procedure is the same as if we were sending the colors of the vertices (bind the texCoords buffer and send them to an attribute of the vertex shader)**

Textures in WebGL

- To use textures in WebGL, we must follow different steps:
  - As well, we must send the texture itself (*drawScene()*)
    - OpenGL/WebGL has 32 texture units (from 0 to 31)
    - We must activate one texture
    - Bind the texture to use
    - Send it to the fragment shader

```
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, myTexture);
gl.uniform1i(shaderProgram.samplerUniform, 0);
```


Textures in WebGL

- To use textures in WebGL, we must follow different steps:
  - Modify the shaders to receive the texture coordinates of each vertex
  - Pass those texture coordinates to the fragment shader
  - Use the colors of the texture to compute the color of each fragment

Vertex shader

```
attribute vec3 aVertexPosition;
attribute vec2 aVertexTexCoords;

uniform mat4 uMVMatrix;
uniform mat4 uPMatrix;

varying vec2 texCoords;

void main(void) {
    gl_Position = uPMatrix * uMVMatrix *
                  vec4(aVertexPosition, 1.0);
    texCoords = aVertexTexCoords;
}
```

Remember to obtain the location of these new variables (*initShaders()*)!!

Fragment shader

```
precision mediump float;

varying vec2 texCoords;
uniform sampler2D texture;

void main(void) {
    gl_FragColor = texture2D(texture, texCoords);
}
```

Textures in WebGL

- To use textures in WebGL, we must follow different steps:
  - At the end, we must redraw the scene every time is needed (changes are made), and not just when loading the web page
  - To do this, we create the reDraw() function that is called in the webGLStart() function

```
function reDraw() {
    requestAnimFrame(reDraw);
    drawScene();
}

function webGLStart() {
    var canvas = document.getElementById("webGL-canvas");
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    initGL(canvas);
    initShaders();
    loadSceneOnGPU();
    loadTextureOnGPU();

    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.enable(gl.DEPTH_TEST);
    gl.enable(gl.CULL_FACE);
    gl.cullFace(gl.BACK);

    reDraw();
}
```

Questions?

www.citm.upc.edu