# Texturing the 3D Cube in WebGL

We continue with the spot light example and add a texture to the cube.

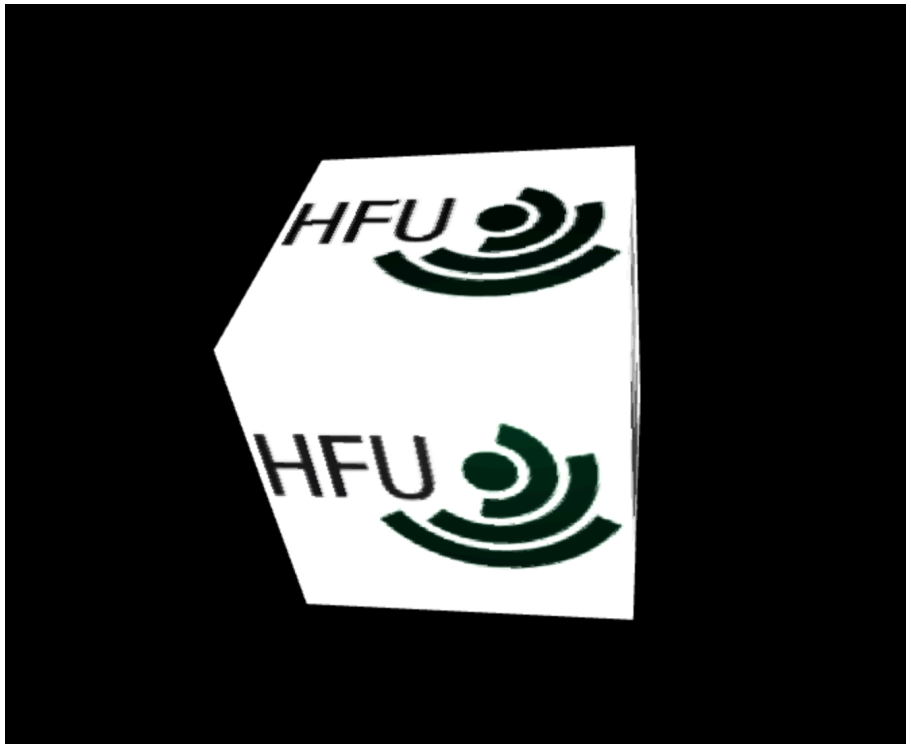
# Explanation

👀 means that the code is already in the repository and you just need to look at it.
📋 means you can copy-paste the code and it should work.
📝 means that you need to create a new file
⭕ indicates that you need to do more than just copy-paste the code.
❌ indicates that you need to replace the old code with something new.

**In any case you need to understand what you are doing.**

# Geometry

We now need texture coordinates to map the texture onto the cube.



## Texture coordinates on the cube

👀 The new cube class from `utils.zip` contains a method that defines the texture coordinates.

```
generateTexcoords() {
    const texcoords = [];
    // front face
    let numFaces = 6; // Each face has 4 vertices
    for (let i = 0; i < numFaces; i++) {
        texcoords.push(...[0, 0]);
        texcoords.push(...[0, 1]);
        texcoords.push(...[1, 1]);
        texcoords.push(...[1, 0]);
    }
    return texcoords;
}
```

## Get texture coordinates to the vertex shader

📋 We update the vertex shader code and add the textures as attribute and as output.

```
in vec2 aTexcoord;
// a varying to pass the texture coordinates to the fragment shader
out vec2 vTexcoord;

void main() {
    ...
    // Pass the texcoord to the fragment shader.
    vTexcoord = aTexcoord;
}
```

## Retrieve the texture coordinates in the fragment shader

📋 We need to retrieve the texture coordinates in the fragment shader.

```
in vec2 vTexcoord;
uniform sampler2D uTexture;
void main() {
    ...
    // Sample the texture using the texture coordinates
    outColor = texture(uTexture, vTexcoord);
}
```

## Connect the texture coordinates to the vertex shader

📋 In the script we read the texture coordinates into a buffer:

```
const texcoordBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, texcoordBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(cube.texcoords), gl.STATIC_DRAW);
```

⭕ Use `connectShaderAttributes` to connect this buffer to the vertex shader.

# Create a texture

⭕ Insert this code lines at the right positions.

```
// Create a texture.
let texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);

// Fill the texture with a 1x1 blue pixel.
gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 1, 1, 0, gl.RGBA, gl.UNSIGNED_BYTE,
    new Uint8Array([0, 0, 255, 255]));

// Asynchronously load an image
let image = new Image();
image.src = "../resources/logo-hfu.png";
image.addEventListener('load', function () {
    // Now that the image has loaded, copy it to the texture.
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, image);
    gl.generateMipmap(gl.TEXTURE_2D);
});
```

# Result: Textured Cube

You should be able to see a textured cube and also change the texture by loading a different image.