

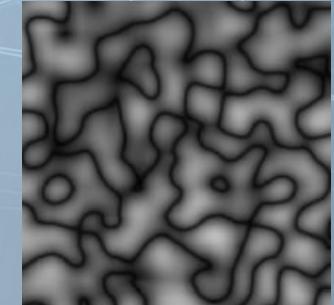
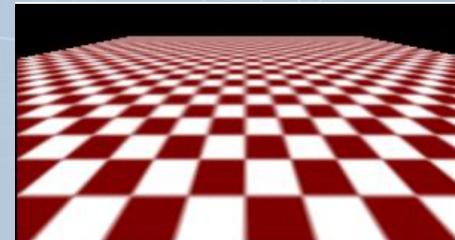
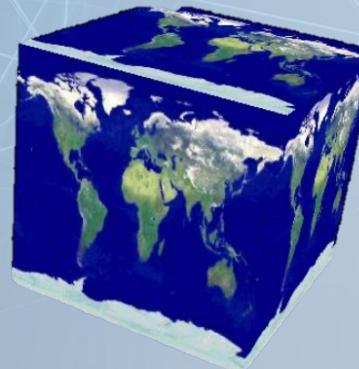


Texture

CSU0021: Computer Graphics

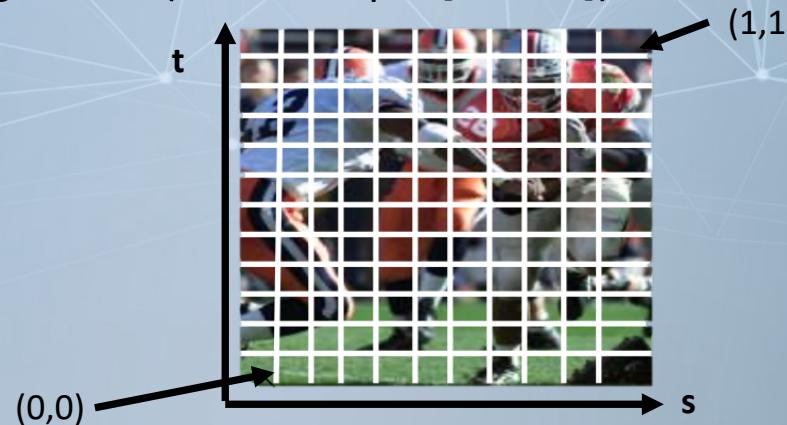
Texture Mapping

- A way of adding surface detail
- Two ways can achieve the goal
 - Surface detail polygon : create extra polygon to model object details
 - Add scene complexity and slow down the graphics rendering speed
 - Map a texture to the surface (popular method)
 - Complexity of images does not affect the complexity of geometry



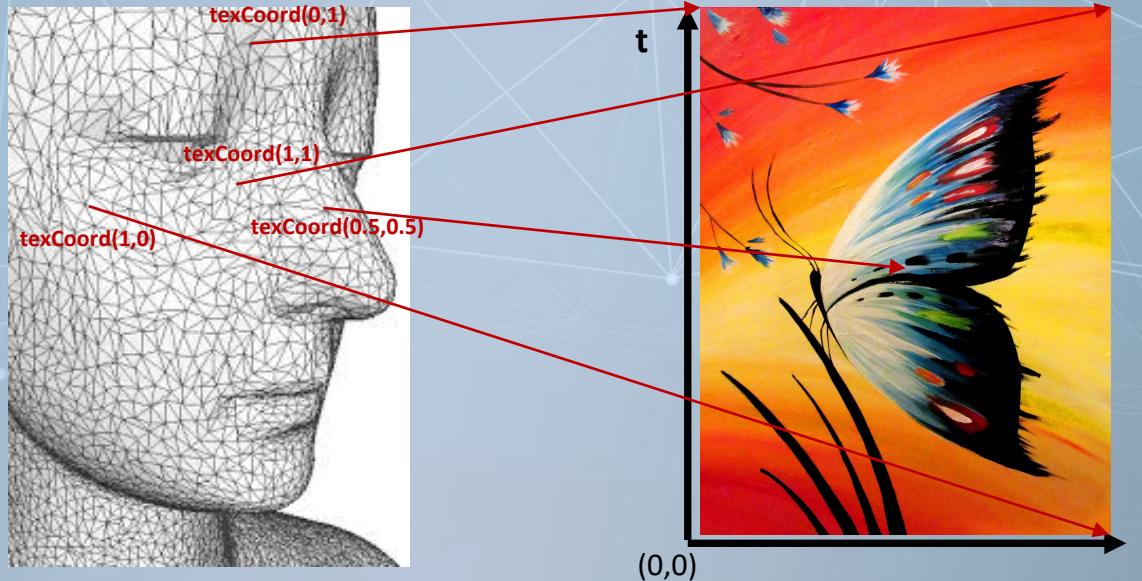
Texture Representation

- Bitmap texture: pixels
 - A 2D image: a 2D array
 - Each pixel (or texel) by a unique pair texture coordinate (s, t)
 - The s and t are usually normalized to a $[0, 1]$ range
 - For any given (s, t) in the normalize range, there is a unique image value (i.e., a unique $[R, G, B]$)



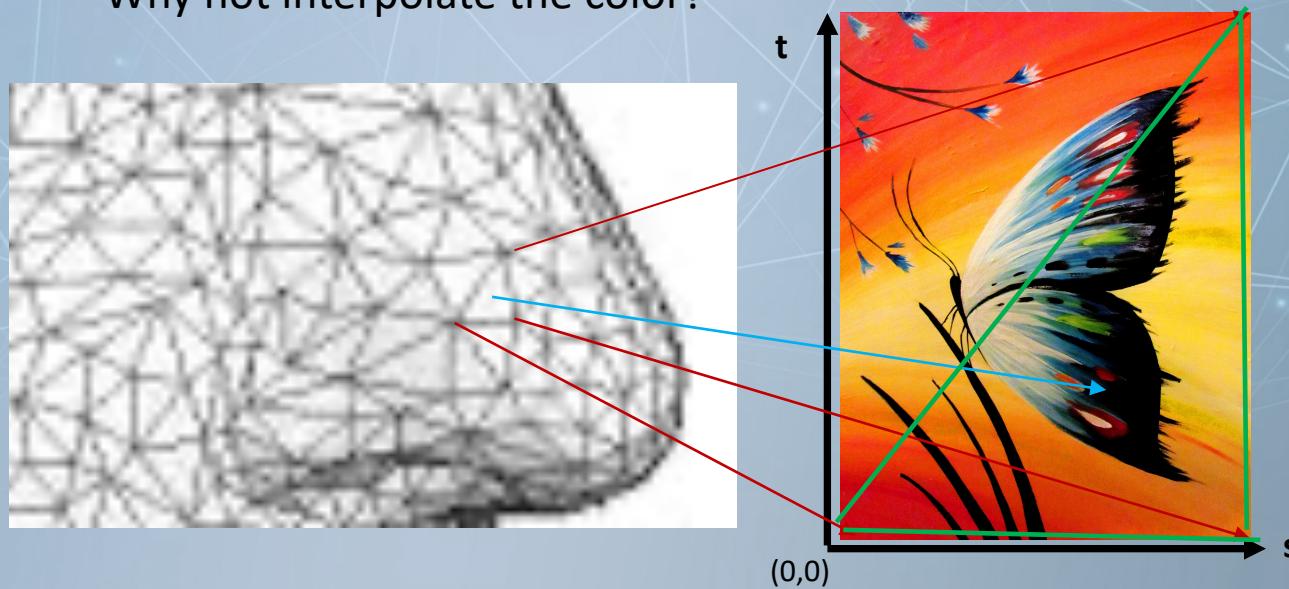
Map Textures to Surfaces

- Vertex: using texture coordinate to establish mapping from texture to surfaces (polygons)



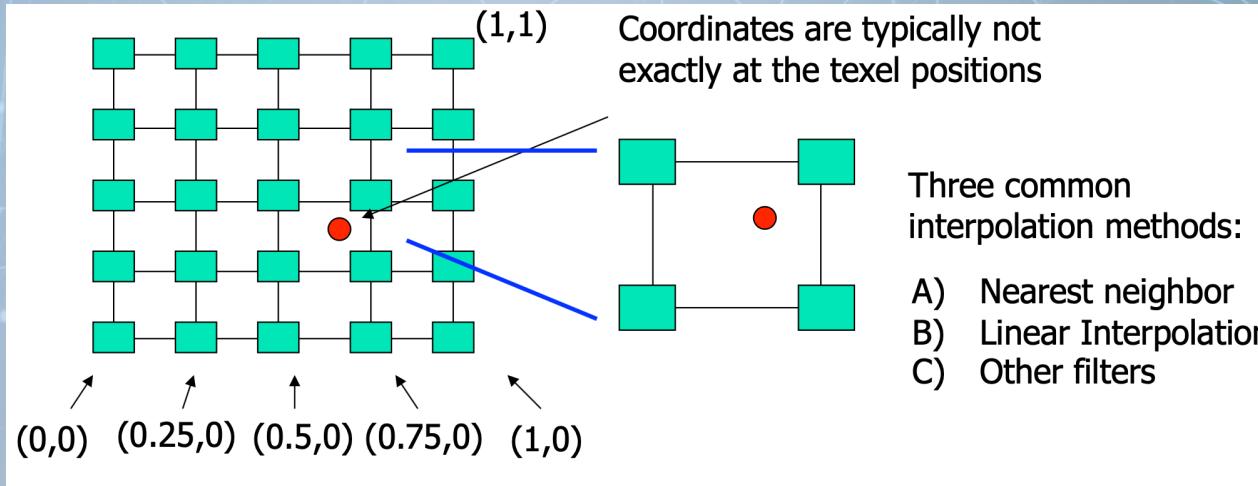
Map Textures to Surfaces

- Inside a triangle: mapping is performed in rasterization
- The texture coordinate is calculated by interpolating the texture coordinates on vertices of the triangle
 - Why not interpolate the color?



Map Textures to Surfaces

- If we would like to have the texture coordinates within a polygon, like other vertex attributes, the texture coordinate is interpolated from vertices of the polygon
- For the given texture coordinate (s, t) , we can find a unique image value from the texture map

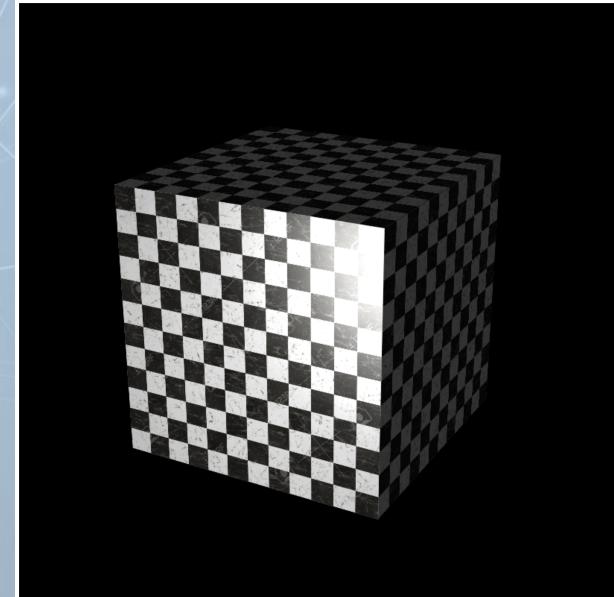
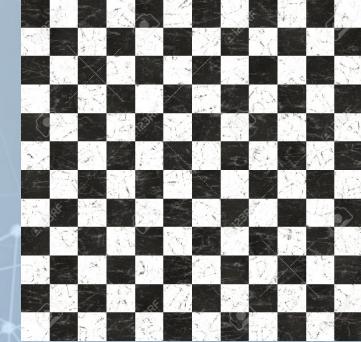


WebGL: Steps to Use Texture

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a ‘sampler2D’ in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`

Example (Ex08-1)

- Use chessboard texture to color the cube
- Files
 - Index.html
 - WebGL.js
 - cuon-matrix.js
 - cube.obj
 - chess.jpg



Example (Ex08-1)

- main() in WebGL.js

```
var objComponents = [];
var textures = {};
var texCount = 0;
var numTextures = 1; //brick

async function main(){
    canvas = document.getElementById('webgl1');
    gl = canvas.getContext('webgl2');
    if(!gl){
        console.log('Failed to get the rendering context for WebGL');
        return ;
    }

    program = compileShader(gl, VSHADER_SOURCE, FSHADER_SOURCE);

    gl.useProgram(program);
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a 'sampler2D' in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()

```
program.a_Position = gl.getAttribLocation(program, 'a_Position');
program.a_TexCoord = gl.getAttribLocation(program, 'a_TexCoord');
program.a_Normal = gl.getAttribLocation(program, 'a_Normal');
program.u_MvpMatrix = gl.getUniformLocation(program, 'u_MvpMatrix');
program.u_modelMatrix = gl.getUniformLocation(program, 'u_modelMatrix');
program.u_normalMatrix = gl.getUniformLocation(program, 'u_normalMatrix');
program.u_LightPosition = gl.getUniformLocation(program, 'u_LightPosition');
program.u_ViewPosition = gl.getUniformLocation(program, 'u_ViewPosition');
program.u_Ka = gl.getUniformLocation(program, 'u_Ka');
program.u_Kd = gl.getUniformLocation(program, 'u_Kd');
program.u_Ks = gl.getUniformLocation(program, 'u_Ks');
program.u_shininess = gl.getUniformLocation(program, 'u_shininess');
program.u_Sampler0 = gl.getUniformLocation(program, "u_Sampler0")

response = await fetch('cube.obj');
text = await response.text();
obj = parseOBJ(text);

for( let i=0; i < obj.geometries.length; i ++ ){
    let o = initVertexBufferForLaterUse(gl,
        obj.geometries[i].data.position,
        obj.geometries[i].data.normal,
        obj.geometries[i].data.texcoord);
    objComponents.push(o);
}

Load "chess.jpg" image

var imageChess = new Image();
imageChess.onload = function() initTexture(gl, imageChess, "chessTex");
imageChess.src = "chess.jpg";
```

When the loading is done,
call initTexture() to create a texture

```
mvpMatrix = new Matrix4();
modelMatrix = new Matrix4();
normalMatrix = new Matrix4();
gl.enable(gl.DEPTH_TEST);

canvas.onmousedown = function(ev){mouseDown(ev)};
canvas.onmousemove = function(ev){mouseMove(ev)};
canvas.onmouseup = function(ev){mouseUp(ev)};
```

Example (Ex08-1)

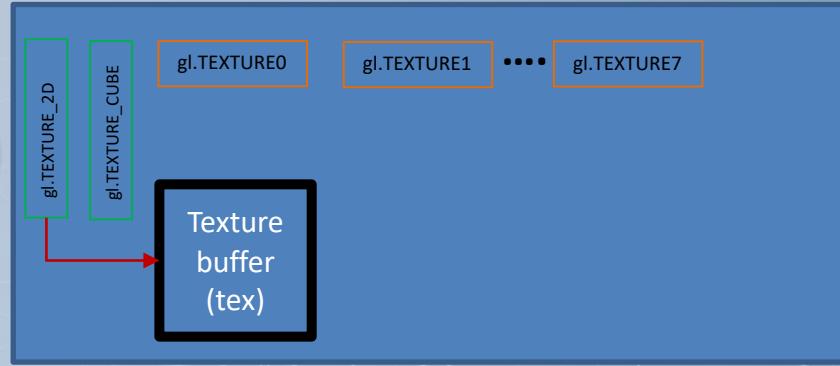
- `gl.createTexture()`
 - Create a texture object
 - <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/createTexture>
 - (You will have at least 8 texture units)



- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a 'sampler2D' in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`

```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

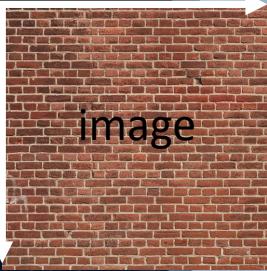
Example (Ex08-1)



- `gl.bindTexture(target, texture)`
 - target: `gl.TEXTURE_2D` or `gl.TEXTURE_CUBE`
 - texture: texture object
 - <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/bindTexture>

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a 'sampler2D' in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`

```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```



Example (Ex08-1)



- `gl.pixelStorei(pname, param)`
 - <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/pixelStorei>
- if `pname: gl.UNPACK_FLIP_Y_WEBGL` and `param = 1`, the image is flipped along the y-axis

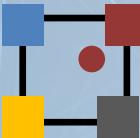
```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a 'sampler2D' in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`

Example (Ex08-1)

- gl.texParameteri(target, pname, param)
 - <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/texParameter>

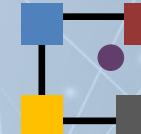
(1) Nearest neighbor (low quality, fast)



gl.texParameter(..., gl.TEXTURE_MIN_FILTER, gl.NEAREST)



(2) Linear interpolation (high quality, slow)



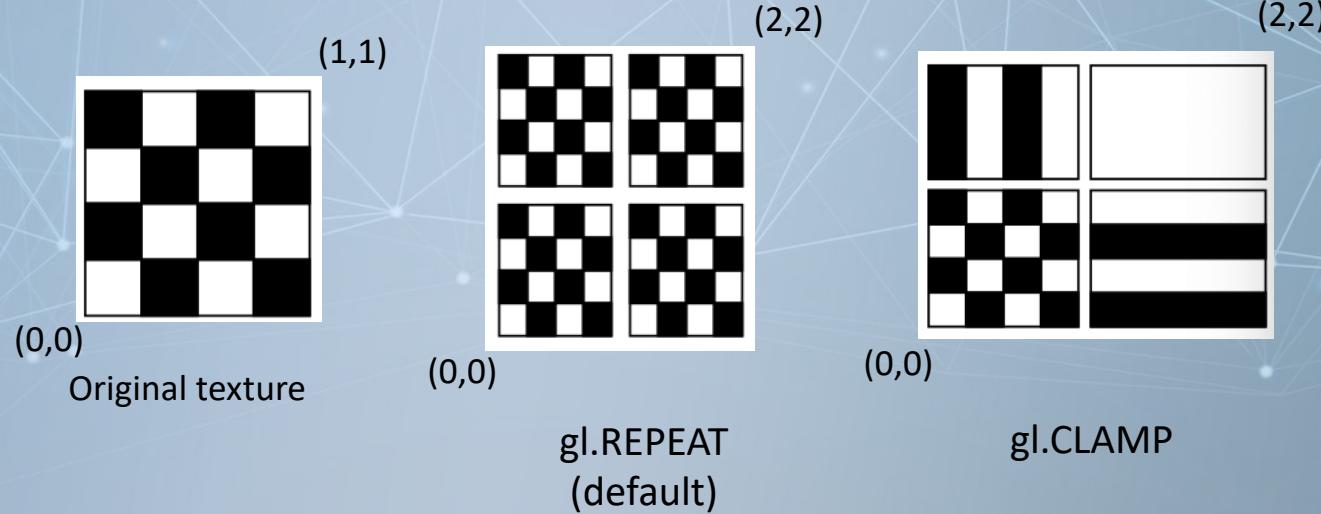
gl.texParameter(..., gl.TEXTURE_MIN_FILTER, gl.LINEAR)

```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a 'sampler2D' in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()

Example (Ex08-1)

- What if the given texture coordinates (s, t) are outside $[0,1]$ range?
 - Ex:
 - `gl.texParameteri(..., gl.TEXTURE_WRAP_S, gl.CLAMP)`
 - `gl.texParameteri(..., gl.TEXTURE_WRAP_T, gl.CLAMP)`

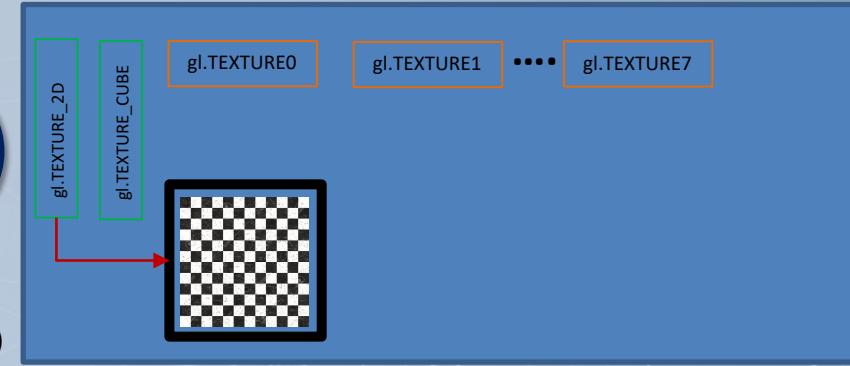


Example (Ex08-1)

gl.texImage2D(target, level, internalformat, format, type, image)

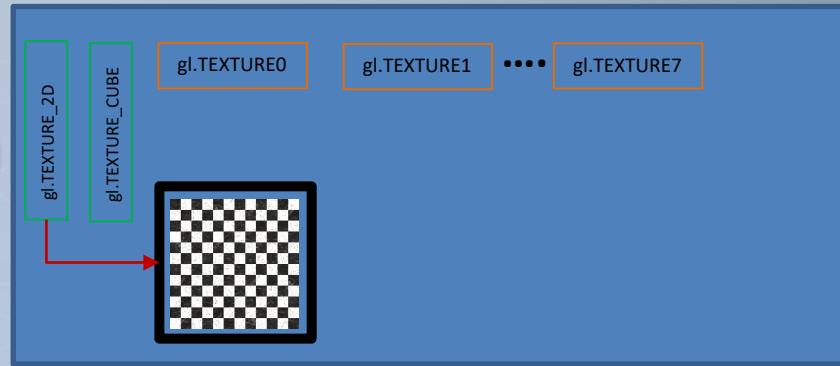
- <https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/texImage2D>
- target: gl.TEXTURE_2D or gl.TEXTURE_CUBE_MAP
- level: level of detail
- internalformat: image's format
- format: texture format (the same as the internal format)
- type: type of the texture (per channel)
- image: input texture image

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a 'sampler2D' in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()



```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

Example (Ex08-1)



Render the first frame
if the loading of all textures is done

Save the texture we create

```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a 'sampler2D' in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`

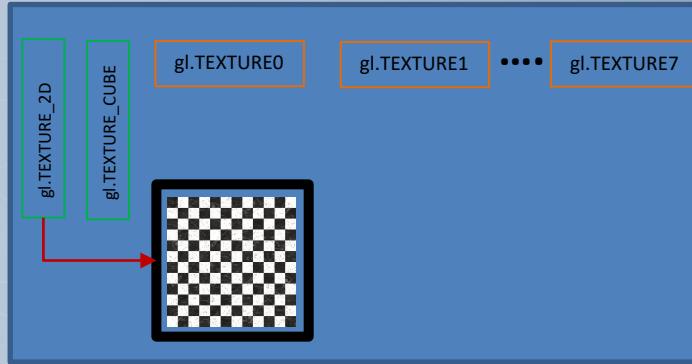
Example (Ex08-1)

- draw() in WebGL.js
- Of course, beside pass vertex position and normal vector information, we also have to pass texture coordinates of the object vertices to shader (to an attribute variable)

```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.scale(objScale, objScale, objScale);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();
```



- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a 'sampler2D' in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()

```
gl.uniform3f(program.u_LightPosition, 0, 0, 3);
gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
gl.uniform1f(program.u_Ka, 0.2);
gl.uniform1f(program.u_Kd, 0.7);
gl.uniform1f(program.u_Ks, 1.0);
gl.uniform1f(program.u_shininess, 10.0);
gl.uniform1i(program.u_Sampler0, 0);

gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textures["chessTex"]);

for( let i=0; i < objComponents.length; i ++ ){
    initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
    initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
    initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);
    gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
}
```

Example (Ex08-1)

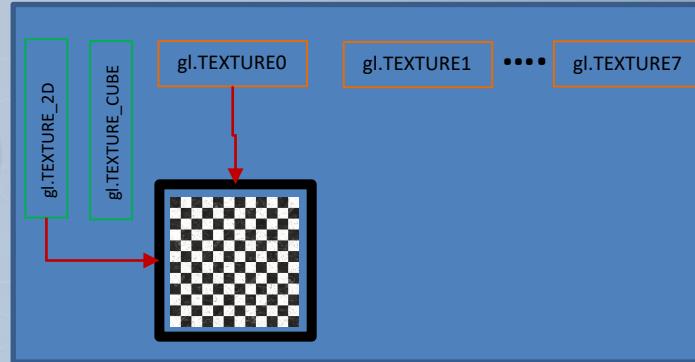
- draw() in WebGL.js

```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.scale(objScale, objScale, objScale);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a 'sampler2D' in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()



```
gl.uniform3f(program.u_LightPosition, 0, 0, 3);
gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
gl.uniform1f(program.u_Ka, 0.2);
gl.uniform1f(program.u_Kd, 0.7);
gl.uniform1f(program.u_Ks, 1.0);
gl.uniform1f(program.u_shininess, 10.0);
gl.uniform1i(program.u_Sampler0, 0);

gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textures["chessTex"]);

for( let i=0; i < objComponents.length; i ++ ){
    initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
    initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
    initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);
    gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
}
```

Example (Ex08-1)

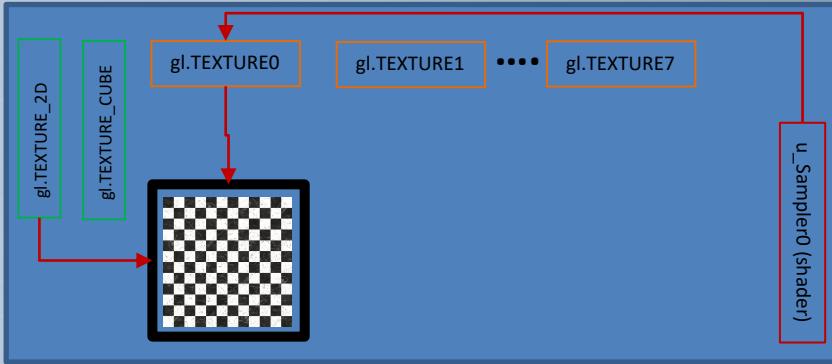
- draw() in WebGL.js

```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.scale(objScale, objScale, objScale);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a ‘sampler2D’ in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()



```
gl.uniform3f(program.u_LightPosition, 0, 0, 3);
gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
gl.uniform1f(program.u_Ka, 0.2);
gl.uniform1f(program.u_Kd, 0.7);
gl.uniform1f(program.u_Ks, 1.0);
gl.uniform1f(program.u_shininess, 10.0);
gl.uniform1i(program.u_Sampler0, 0); ? Because we bind the texture to TEXTURE0

gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textures["chessTex"]);

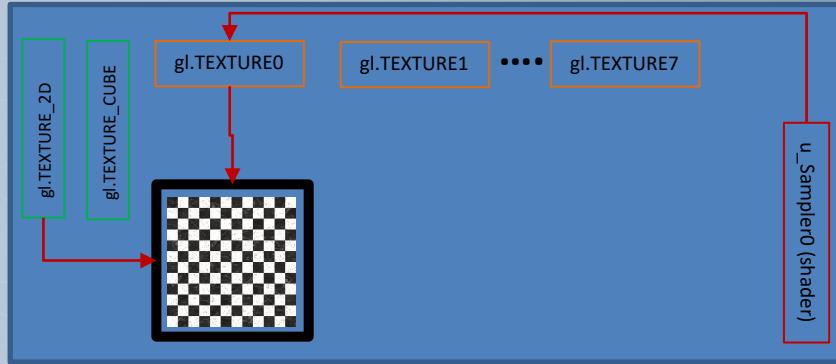
for( let i=0; i < objComponents.length; i ++ ){
    initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
    initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
    initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);
    gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
}
```

Example (Ex08-1)

- Shader in WebGL.js

```
var VSHADER_SOURCE = `attribute vec4 a_Position;  
attribute vec4 a_Normal;  
attribute vec2 a_TexCoord;  
uniform mat4 u_MvpMatrix;  
uniform mat4 u_modelMatrix;  
uniform mat4 u_normalMatrix;  
varying vec3 v_Normal;  
varying vec3 v_PositionInWorld;  
varying vec2 v_TexCoord;  
void main(){  
    gl_Position = u_MvpMatrix * a_Position;  
    v_PositionInWorld = (u_modelMatrix * a_Position).xyz;  
    v_Normal = normalize(vec3(u_normalMatrix * a_Normal));  
    v_TexCoord = a_TexCoord;  
}`;
```

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: `gl.createTexture()`
 - Bind texture to a target: `gl.bindTexture()`
 - Indicate how to interpret the image: `gl.pixelStorei()`
 - Texture parameters: `gl.texParameteri()`
 - Assign the texture image to the texture target: `gl.texImage2D()`
- Assign the texture to the shader
 - Active a texture unit: `gl.activeTexture()`
 - Bind texture to a target if necessary(): `gl.bindTexture()`
 - Pass the texture unit to a ‘sampler2D’ in shader: `gl.uniform1i()`
- Call shader to draw
 - Get a color from the texture in shader: `texture2D()`



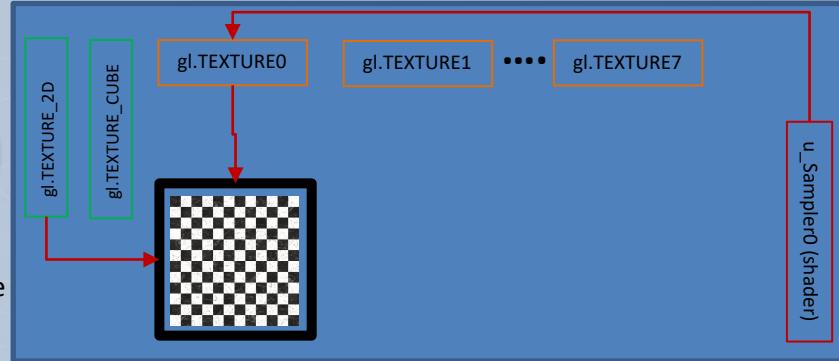
Texture coordinate of a fragment is calculated from the texture coordinates of vertices of the triangle by interpolation

```
var FSHADER_SOURCE = `precision mediump float;  
uniform vec3 u_LightPosition;  
uniform vec3 u_ViewPosition;  
uniform float u_Ka;  
uniform float u_Kd;  
uniform float u_Ks;  
uniform float u_shininess;  
uniform sampler2D u_Sampler0;  
varying vec3 v_Normal;  
varying vec3 v_PositionInWorld;  
varying vec2 v_TexCoord;  
void main(){  
    // let ambient and diffuse color are u_Color  
    // (you can also input them from outside and make them different)  
    vec3 texColor = texture2D( u_Sampler0, v_TexCoord ).rgb;  
    vec3 ambientLightColor = texColor;  
    vec3 diffuseLightColor = texColor;  
    // assume white specular light (you can also input it from outside)  
    vec3 specularLightColor = vec3(1.0, 1.0, 1.0);  
  
    vec3 ambient = ambientLightColor * u_Ka;
```

Example (Ex08-1)

- Shader in WebGL.js
 - sampler2D is the variable type to hold a 2D texture
 - Use “**texture2D()**” to get a texture color (RGBA) from a sampler2D variable
 - First parameter: the texture
 - Second parameter: a texture coordinate
 - Return: a RGBA color

- Load the texture image
- Prepare texture buffer
 - Create texture buffer: gl.createTexture()
 - Bind texture to a target: gl.bindTexture()
 - Indicate how to interpret the image: gl.pixelStorei()
 - Texture parameters: gl.texParameteri()
 - Assign the texture image to the texture target: gl.texImage2D()
- Assign the texture to the shader
 - Active a texture unit: gl.activeTexture()
 - Bind texture to a target if necessary(): gl.bindTexture()
 - Pass the texture unit to a ‘sampler2D’ in shader: gl.uniform1i()
- Call shader to draw
 - Get a color from the texture in shader: texture2D()



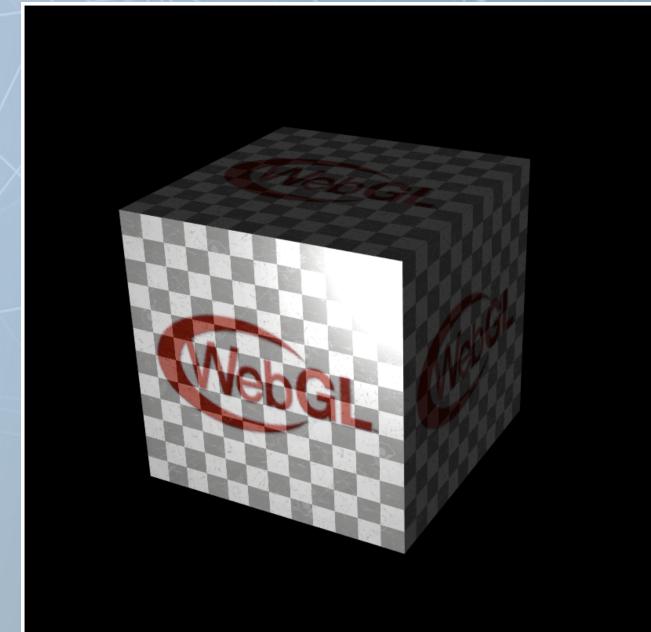
```
var FSHADER_SOURCE = `precision mediump float;
uniform vec3 u_LightPosition;
uniform vec3 u_ViewPosition;
uniform float u_Ka;
uniform float u_Kd;
uniform float u_Ks;
uniform float u_shininess;
uniform sampler2D u_Sampler0;
varying vec3 v_Normal;
varying vec3 v_PositionInWorld;
varying vec2 v_TexCoord;
void main(){
    // let ambient and diffuse color are u_Color
    // (you can also input them from outside and make them different)
    vec3 texColor = texture2D( u_Sampler0, v_TexCoord ).rgb;
    vec3 ambientLightColor = texColor;
    vec3 diffuseLightColor = texColor;
    // assume white specular light (you can also input it from outside)
    vec3 specularLightColor = vec3(1.0, 1.0, 1.0);
    vec3 ambient = ambientLightColor * u_Ka;
```

Let's try (5mins)

- Make sure you know the step to use a texture and how to pass it to shader
- Change `gl.uniform1i(program.u_Sampler0, 0)` to `gl.uniform1i(program.u_Sampler0, 2)`, does it work?
- What if you also modify `gl.activeTexture(gl.TEXTURE0)` to `gl.activeTexture(gl.TEXTURE2)`, does it work?
- Try any thing you want.

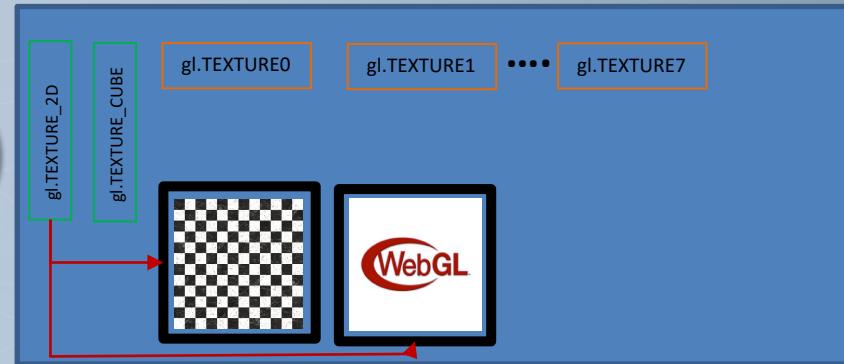
Example (Ex08-2)

- Mix multiple textures to color the cube
 - Use multiple textures
- Files
 - Index.html
 - WebGL.js
 - cuon-matrix.js
 - cube.obj
 - chess.jpg
 - webgllcon.jpg



Example (Ex08-2)

- main(): load one more texture
- Also call initTexture to initialize each texture
- Store each texture in a dictionary, “textures”



```
var textures = {};
var texCount = 0;
var numTextures = 2; //brick, grass

async function main(){
    canvas = document.getElementById('webgl');

    var imageChess = new Image();
    imageChess.onload = function(){initTexture(gl, imageChess, "chessTex");}
    imageChess.src = "chess.jpg";

    var imageBk = new Image();
    imageBk.onload = function(){initTexture(gl, imageBk, "webGLTex");}
    imageBk.src = "webglIcon.jpg";

    mvpMatrix = new Matrix4();
    modelMatrix = new Matrix4();
    normalMatrix = new Matrix4();

    gl.enable(gl.DEPTH_TEST);

    canvas.onmousedown = function(ev){mouseDown(ev)};
    canvas.onmousemove = function(ev){mouseMove(ev)};
    canvas.onmouseup = function(ev){mouseUp(ev)};
}
```

```
function initTexture(gl, img, texKey){
    var tex = gl.createTexture();
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);
    gl.bindTexture(gl.TEXTURE_2D, tex);

    // Set the parameters so we can render any size image.
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);

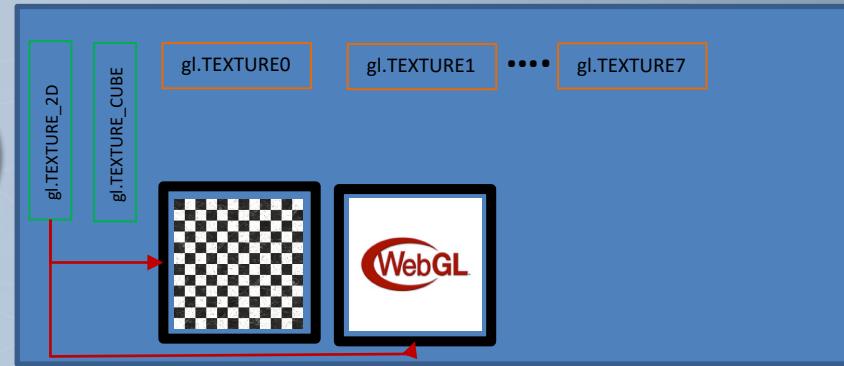
    // Upload the image into the texture.
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);

    textures[texKey] = tex;

    texCount++;
    if( texCount == numTextures)draw();
}
```

Example (Ex08-2)

- draw the first frame if all texture loading is done

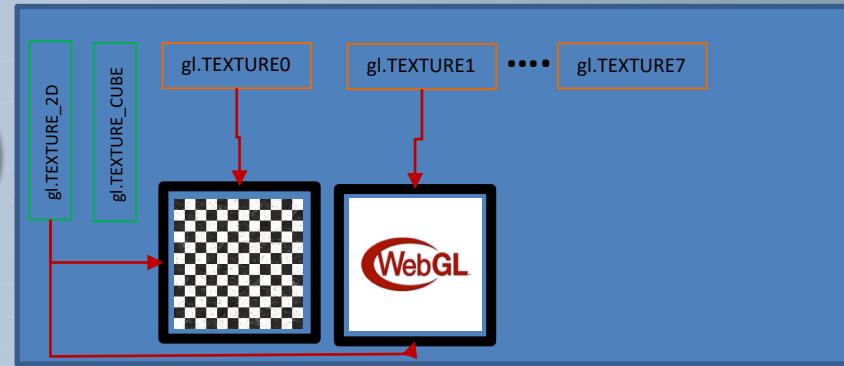


```
function initTexture(gl, img, texKey){  
    var tex = gl.createTexture();  
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, 1);  
    gl.bindTexture(gl.TEXTURE_2D, tex);  
  
    // Set the parameters so we can render any size image.  
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);  
  
    // Upload the image into the texture.  
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED_BYTE, img);  
  
    textures[texKey] = tex;  
  
    texCount++;  
    if( texCount == numTextures)draw();  
}
```

```
var textures = {};  
var texCount = 0;  
var numTextures = 2; //brick, grass  
  
▽ async function main(){  
    canvas = document.getElementById('webgl');  
  
    var imageChess = new Image();  
    imageChess.onload = function(){initTexture(gl, imageChess, "chessTex");};  
    imageChess.src = "chess.jpg";  
  
    var imageBk = new Image();  
    imageBk.onload = function(){initTexture(gl, imageBk, "webGLTex");};  
    imageBk.src = "webglIcon.jpg";  
  
    mvpMatrix = new Matrix4();  
    modelMatrix = new Matrix4();  
    normalMatrix = new Matrix4();  
  
    gl.enable(gl.DEPTH_TEST);  
  
    canvas.onmousedown = function(ev){mouseDown(ev)};  
    canvas.onmousemove = function(ev){mouseMove(ev)};  
    canvas.onmouseup = function(ev){mouseUp(ev)};  
}
```

Example (Ex08-2)

- Bind textures to texture units
- Assign texture units to sampler2D variables in shader



```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.scale(objScale, objScale, objScale);
    // modelMatrix.translate(0.0, 0.0, -1.0);
    // modelMatrix.scale(1.0, 0.5, 2.0);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();
```

```
gl.uniform3f(program.u_LightPosition, 0, 0, 3);
gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
gl.uniform1f(program.u_Ka, 0.2);
gl.uniform1f(program.u_Kd, 0.7);
gl.uniform1f(program.u_Ks, 1.0);
gl.uniform1f(program.u_shininess, 10.0);
gl.uniform1i(program.u_Sampler0, 0);
gl.uniform1i(program.u_Sampler1, 1);

gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

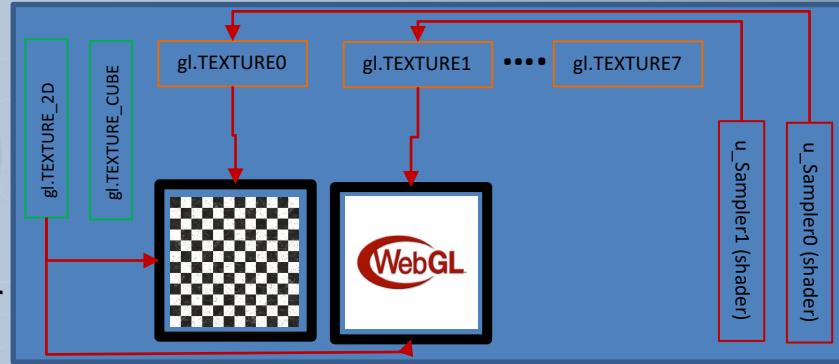
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textures["chessTex"]);

gl.activeTexture(gl.TEXTURE1);
gl.bindTexture(gl.TEXTURE_2D, textures["webGLTex"]);

for( let i=0; i < objComponents.length; i ++ ){
    initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
    initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
    initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);
    gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
}
```

Example (Ex08-2)

- Bind textures to texture units
- Assign texture units to sampler2D variables in shader



```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.scale(objScale, objScale, objScale);
    // modelMatrix.translate(0.0, 0.0, -1.0);
    // modelMatrix.scale(1.0, 0.5, 2.0);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();
```

```
gl.uniform3f(program.u_LightPosition, 0, 0, 3);
gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
gl.uniform1f(program.u_Ka, 0.2);
gl.uniform1f(program.u_Kd, 0.7);
gl.uniform1f(program.u_Ks, 1.0);
gl.uniform1f(program.u_shininess, 10.0);
gl.uniform1i(program.u_Sampler0, 0);
gl.uniform1i(program.u_Sampler1, 1);

gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, textures["chessTex"]);

gl.activeTexture(gl.TEXTURE1);
gl.bindTexture(gl.TEXTURE_2D, textures["webGLTex"]);

for( let i=0; i < objComponents.length; i ++ ){
    initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
    initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
    initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);
    gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
}
```

Example (Ex08-2)

- Fragment shader
 - Mix color from two textures

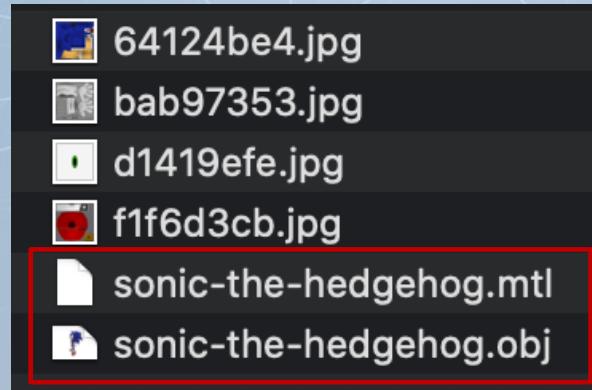
```
var FSHADER_SOURCE = `precision mediump float;
uniform vec3 u_LightPosition;
uniform vec3 u_ViewPosition;
uniform float u_Ka;
uniform float u_Kd;
uniform float u_Ks;
uniform float u_shininess;
uniform sampler2D u_Sampler0;
uniform sampler2D u_Sampler1;
varying vec3 v_Normal;
varying vec3 v_PositionInWorld;
varying vec2 v_TexCoord;
void main(){
    // let ambient and diffuse color are u_Color
    // (you can also input them from ouside and make them different)
    vec3 texColor0 = texture2D( u_Sampler0, v_TexCoord ).rgb;
    vec3 texColor1 = texture2D( u_Sampler1, v_TexCoord ).rgb;
    vec3 texColor = texColor0 * 0.4 + texColor1 * 0.6;
    vec3 ambientLightColor = texColor;
    vec3 diffuseLightColor = texColor;
    // assume white specular light (you can also input it from ouside)
    vec3 specularLightColor = vec3(1.0, 1.0, 1.0);`
```

Let's try (5mins)

- Try to modify this line, “`vec3 texColor = texColor0 *0.4 + texColor1 *0.6`;”, to `vec3 texColor = texColor0 *0.7 + texColor1 *0.3`
 - Change this line to other weight factors to mix them, make sure you understand it
- If you want to pass the third textures to shader, try to think what you should do?
- Try any thing you want

Color an Obj File Object by Texture Images

- The obj file from clara.io usually includes texture images for the object
- The metadata is in *.mtl file



Color an Obj File Object by Texture Images

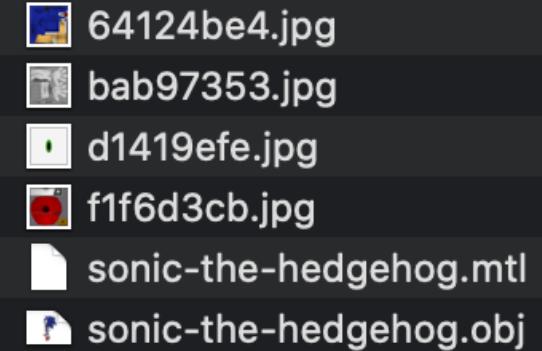
- Review the obj files
 - use sonic as an example (it consist of 13 components)

Tell you where the material metadata file is

```
# Blender v2.74 (sub 0) OBJ File: ''
# www.blender.org
mtllib sonic-the-hedgehog.mtl
o DrawCall_0277
v -1.861972 22.695541 2.067917
v -1.814528 22.398708 2.033974
v -1.594641 22.407452 2.108498
v -1.620953 22.704561 2.146652
v -1.897370 23.060692 2.074585
v -1.645572 23.072294 2.154800
v -1.923574 23.475042 2.060799
v -1.663958 23.490108 2.144958
v -1.939530 23.931679 2.025650
v -1.673995 23.948975 2.113655
v -1.674102 24.439888 2.057426
v -1.943755 24.423717 1.968201
v -1.659837 24.990534 1.969383
v -1.933084 24.975704 1.879944
v -1.627081 25.628675 1.842583
v -1.903709 25.612173 1.752319
v -1.545297 26.311020 1.632172
v -1.823511 26.294573 1.540985
v -1.384264 26.994339 1.293381
v -1.660048 26.984308 1.201553
v -1.183500 27.678129 0.890373
v -1.455058 27.677729 0.798042
v -0.911733 27.651968 0.975662
```

Color an Obj File Object by Texture Images

- sonic-the-hedgehog.mtl
- Although sonic consist of 13 components
- It only has 4 different materials
 - Each material has its own illumination parameters (ka, kd, ks...) and texture image



```
# Blender MTL File: 'None'  
# Material Count: 4
```

```
newmtl 64124be4_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd 64124be4.jpg
```

```
newmtl bab97353_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd bab97353.jpg
```

```
newmtl d1419efe_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd d1419efe.jpg
```

```
newmtl f1f6d3cb_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd f1f6d3cb.jpg
```

```
# Blender MTL File: 'None'  
# Material Count: 4  
  
newmtl 64124be4_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd 64124be4.jpg
```

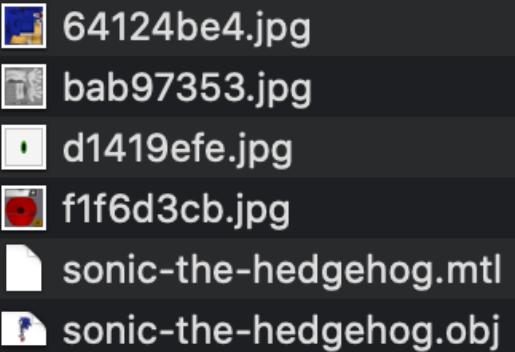
```
newmtl bab97353_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd bab97353.jpg
```

```
newmtl d1419efe_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd d1419efe.jpg
```

```
newmtl f1f6d3cb_dds  
Ns 96.078431  
Ka 0.000000 0.000000 0.000000  
Kd 0.700000 0.700000 0.700000  
Ks 0.010000 0.010000 0.010000  
Ni 1.000000  
d 1.000000  
illum 2  
map_Kd f1f6d3cb.jpg
```

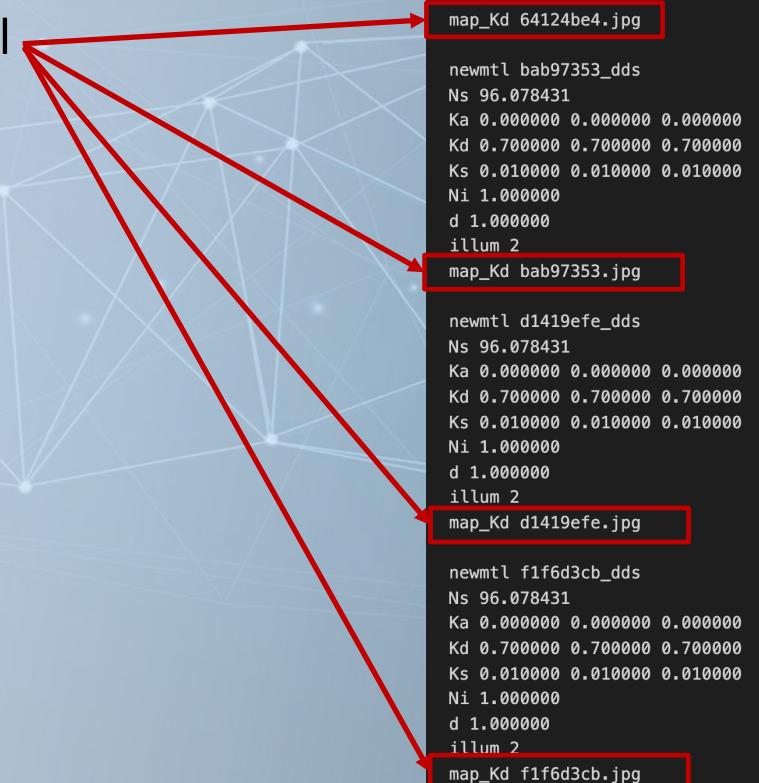
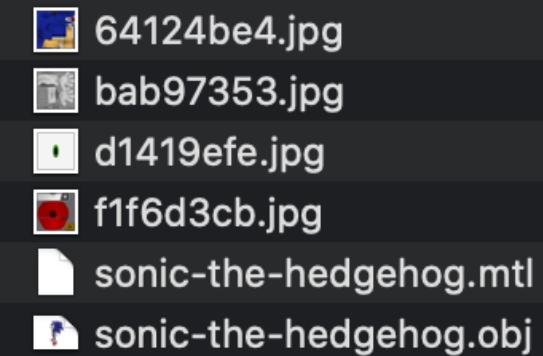
Color an Obj File Object by Texture Images

- Material IDs



Color an Obj File Object by Texture Images

- Texture image of each material



Color an Obj File Object by Texture Images

- Back to the obj file again
- If you search the keyword “usemtl”, you will realize that each component has one “usemtl” line (usually right before “face” section).
- It tells you what the material information is of this component
 - Ex: usemtl f1f6d3cb_dds (so the texture image for this component is “f1f6d3cb.jpg”)
 - So, you can find 13 usemtl lines in the sonic obj file

```
vn 0.740500 0.133300 -0.658700
vn 0.608300 0.625700 -0.488400
vn 0.545700 0.713700 -0.439300
vn 0.288100 0.932200 -0.219100
vn 0.053100 0.998100 -0.029700
vn 0.008200 1.000000 -0.002100
usemtl f1f6d3cb_dds
s 1
f 2032/1208/1607 2033/1209/1608 2034/1210/1609
f 2032/1208/1607 2034/1210/1609 2035/1211/1610
f 2032/1208/1607 2035/1211/1610 2036/1212/1611
f 2037/1213/1612 2032/1208/1607 2036/1212/1611
f 2037/1213/1612 2036/1212/1611 2038/1214/1613
```

```
# Blender MTL File: 'None'
# Material Count: 4

newmtl 64124be4_dds
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.700000 0.700000 0.700000
Ks 0.010000 0.010000 0.010000
Ni 1.000000
d 1.000000
illum 2
map_Kd 64124be4.jpg

newmtl bab97353_dds
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.700000 0.700000 0.700000
Ks 0.010000 0.010000 0.010000
Ni 1.000000
d 1.000000
illum 2
map_Kd bab97353.jpg

newmtl d1419efe_dds
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.700000 0.700000 0.700000
Ks 0.010000 0.010000 0.010000
Ni 1.000000
d 1.000000
illum 2
map_Kd d1419efe.jpg

newmtl f1f6d3cb_dds
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.700000 0.700000 0.700000
Ks 0.010000 0.010000 0.010000
Ni 1.000000
d 1.000000
illum 2
map_Kd f1f6d3cb.jpg
```

Example (Ex08-3)

- Use texture images to color the sonic
- Files

```
64124be4.jpg  
bab97353.jpg  
cuon-matrix.js  
d1419efe.jpg  
f1f6d3cb.jpg  
index.html  
sonic-the-hedgehog.mtl  
sonic.obj  
WebGL.js
```



Example (Ex08-3)

- main() in WebGL.js
 - I do not write a parser to parse the mtl file. So, I manually create and hardcode the mapping between object components and texture images
 - It is better if you write a parser for mtl file.

```
var cameraX = 3, cameraY = 3, cameraZ = 7;
var objScale = 0.1,
var objComponents = [];
var textures = {};
var imgNames = ["64124be4.jpg", "bab97353.jpg", "d1419efe.jpg", "f1f6d3cb.jpg"];
var objCompImgIndex = ["d1419efe.jpg", "64124be4.jpg", "64124be4.jpg", "f1f6d3cb.jpg",
    "bab97353.jpg", "64124be4.jpg", "64124be4.jpg", "64124be4.jpg",
    "f1f6d3cb.jpg", "64124be4.jpg", "bab97353.jpg", "f1f6d3cb.jpg",
    "d1419efe.jpg"];
var texCount = 0;
var numTextures = imgNames.length;

async function main(){
    canvas = document.getElementById('webgl');
    gl = canvas.getContext('webgl2');
    if(!gl){
        console.log('Failed to get the rendering context for WebGL');
        return ;
    }

    program = compileShader(gl, VSHADER_SOURCE, FSHADER_SOURCE);
    gl.useProgram(program);
}
```

```
program.a_Position = gl.getAttribLocation(program, 'a_Position');
program.a_TexCoord = gl.getAttribLocation(program, 'a_TexCoord');
program.a_Normal = gl.getAttribLocation(program, 'a_Normal');
program.u_MvpMatrix = gl.getUniformLocation(program, 'u_MvpMatrix');
program.u_modelMatrix = gl.getUniformLocation(program, 'u_modelMatrix');
program.u_normalMatrix = gl.getUniformLocation(program, 'u_normalMatrix');
program.u_LightPosition = gl.getUniformLocation(program, 'u_LightPosition');
program.u_ViewPosition = gl.getUniformLocation(program, 'u_ViewPosition');
program.u_Ka = gl.getUniformLocation(program, 'u_Ka');
program.u_Kd = gl.getUniformLocation(program, 'u_Kd');
program.u_Ks = gl.getUniformLocation(program, 'u_Ks');
program.u_shininess = gl.getUniformLocation(program, 'u_shininess');
program.u_Sampler = gl.getUniformLocation(program, "u_Sampler");

response = await fetch('sonic.obj');
text = await response.text();
obj = parseOBJ(text);

for( let i=0; i < obj.geometries.length; i ++ ){
    let o = initVertexBufferForLaterUse(gl,
        obj.geometries[i].data.position,
        obj.geometries[i].data.normal,
        obj.geometries[i].data.texcoord);
    objComponents.push(o);
}
load and setup components and texture images

for( let i=0; i < imgNames.length; i ++ ){
    let image = new Image();
    image.onload = function(){initTexture(gl, image, imgNames[i]);};
    image.src = imgNames[i];
}

mvpMatrix = new Matrix4();
modelMatrix = new Matrix4();
normalMatrix = new Matrix4();

gl.enable(gl.DEPTH_TEST);

canvas.onmousedown = function(ev){mouseDown(ev)};
canvas.onmousemove = function(ev){mouseMove(ev)};
canvas.onmouseup = function(ev){mouseUp(ev)};
```

Example (Ex08-3)

- draw() in WebGL.js

Before drawing a component bind the corresponding texture image to TEXTURE0, and tell u_Sampler to use the texture unit 0

```
function draw(){
    gl.clearColor(0,0,0,1);

    //model Matrix (part of the mvp matrix)
    modelMatrix.setRotate(angleY, 1, 0, 0); //for mouse rotation
    modelMatrix.rotate(angleX, 0, 1, 0); //for mouse rotation
    modelMatrix.translate(0, -1.2, 0);
    modelMatrix.scale(objScale, objScale, objScale);
    //mvp: projection * view * model matrix
    mvpMatrix.setPerspective(30, 1, 1, 100);
    mvpMatrix.lookAt(cameraX, cameraY, cameraZ, 0, 0, 0, 0, 1, 0);
    mvpMatrix.multiply(modelMatrix);

    //normal matrix
    normalMatrix.setInverseOf(modelMatrix);
    normalMatrix.transpose();

    gl.uniform3f(program.u_LightPosition, 0, 0, 3);
    gl.uniform3f(program.u_ViewPosition, cameraX, cameraY, cameraZ);
    gl.uniform1f(program.u_Ka, 0.2);
    gl.uniform1f(program.u_Kd, 0.7);
    gl.uniform1f(program.u_Ks, 1.0);
    gl.uniform1f(program.u_shininess, 10.0);

    gl.uniformMatrix4fv(program.u_MvpMatrix, false, mvpMatrix.elements);
    gl.uniformMatrix4fv(program.u_modelMatrix, false, modelMatrix.elements);
    gl.uniformMatrix4fv(program.u_normalMatrix, false, normalMatrix.elements);

    Draw components one by one
    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

    for( let i=0; i < objComponents.length; i ++ ){
        gl.activeTexture(gl.TEXTURE0);
        gl.bindTexture(gl.TEXTURE_2D, textures[objCompImgIndex[i]]);
        gl.uniform1i(program.u_Sampler, 0);

        initAttributeVariable(gl, program.a_Position, objComponents[i].vertexBuffer);
        initAttributeVariable(gl, program.a_TexCoord, objComponents[i].texCoordBuffer);
        initAttributeVariable(gl, program.a_Normal, objComponents[i].normalBuffer);

        gl.drawArrays(gl.TRIANGLES, 0, objComponents[i].numVertices);
    }
}
```

Let's try

- Make sure you know .mtl file and you know an object component should map to which texture image