Assignment 20: Using textures in shaders

In this assignment you have to write two complete shaders sets. The first will be applied to a Sphere, with different materials on its surface; The second to a scene where the diffuse illumination has been baked (recorded) into a texture. The first scene will use diffuse Lambert with specular Phong, and the second will use just an ambient term plus specular Blinn. The sphere will be lit with a spot light, will the second scene will use a directional light.

Both scenes will use the same vertex input layout:

- `location=0, vec3 inPosition` -> the position of the vertex in Local coordinates.
- `location=1, vec3 inNormal` -> the direction of the normal vector in Local space.
- `location=2, vec2 inTexCoord` -> the UV coordinates of the vertex.

They will also produce rendering result a single attachment, that will be stored into:

- `location=0, vec4 outColor` -> the final color of the fragment. Components `.rgb` contain the color. The fourth component, `.a` , which represents the transparency, is not used and should be set to `1.0f` for maximum compatibility.

Remember that the Vertex shader should put the computed clipping coordinates inside the global variable called `gl_Position` . The programmer must also decide which components should be passed from the Vertex to the Fragment shader to allow rendering. In particular, the texture coordinates are necessary to allow fetching the correct pixel from the images; the normal vector direction (in world coordinates) is required to correctly implement the Phong, Blinn and Lambert components of the BRDFs; and the pixel position in world coordinates is required to compute both the light direction for the spotlight model, and to compute the viewer direction using her position. The complete definition of the variables that are passed during the communication between vertex and fragment shader is part of this exercise, and you are free to choose the best option for your implementation.

Both scenes will receive a single set of uniforms (`set 0`), with four bindings. `bindings 0`, and `3` will be identical for both scenes. In particular, `binding 0` will be a uniform data block containing the matrices for transforming the position into clipping coordinates and world coordinates, plus the matrix for transforming the normal vector direction.

- `mat4 mvpMat` -> the World-View-Projection matrix to transform position in clipping space.
- `mat4 mMat` -> the World matrix to transform the position in World space.
- `mat4 nMat` -> the matrix to transform normal vector directions in World space.

Please note that, even if a `mat3 nMat` matrix would be enough for transforming the normal vector direction, here it is passed as a `mat4` matrix, where the last column and row are `0,0,0,1` .
`Binding 3` contains instead a 2D texture defining the specular color and powers. In particular, let as call `texel` the value sampled from the texture at the corresponding UV coordinates. Then:

- `texel.rgb` contains the specular color of the considered point.
- The corresponding specular power, used both in the Blinn or Phong models, can be computed as `200.0f * texel.a`.

For the first scene, `binding 1` will contain a combined image sampler with a 2D texture defining the *diffuse* color of the object, and `binding 2` will contain the following information in a uniform block to define the spot light and the position of the viewer:

- `vec3 lightDir` -> the direction of the light.
- `vec3 lightPos` -> the position of the light.
- `vec3 lightColor` -> the basic color of the light.
- `vec4 lightParams` -> the other parameters of the spot light.
- `vec3 eyePos` -> the position of the viewer.

The members of `lightParams` are the following:

- `lightParams.x` -> a `float` component containing the cosine of the inner angle.
- `lightParams.y` -> a `float` component containing the cosine of the outer angle.
- `lightParams.z` -> a `float` component containing the denominator exponent β.
- `lightParams.w` -> a `float` component containing the basic distance `g`.

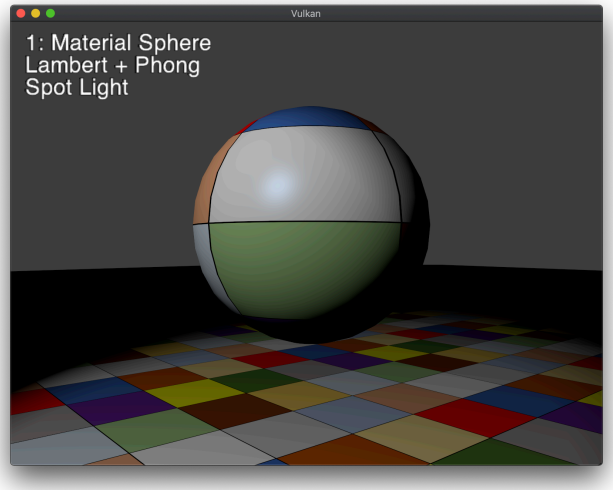
In the second scene, `binding 1` will contain a combined image sampler with a 2D texture defining the *ambient* color of the object, and the uniform block in `binding 2` will contain the following data to support a directional light and the position of the viewer:

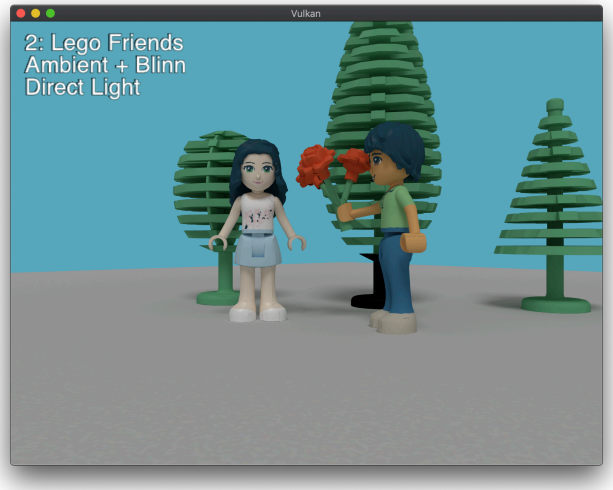
- `vec3 lightDir` -> the direction of the light.
- `vec3 lightColor` -> the basic color of the light.
- `vec3 eyePos` -> the position of the viewer.

The shaders for the two scene must be inserted inside the `/shaders/` folder, and must use the following names:

	Scene 1 (Sphere, with Lambert, Phong and a Spot light)	Scene 2 (characters, with Ambient, Blinn and a Directional light)
Vertex Shader	`BRDF1Vert.spv`	`BRDF2Vert.spv`
Fragment Shader	`BRDF1Frag.spv`	`BRDF2Frag.spv`

The expected results are the following:





Users can move the view using the same keys as in Assignment 0:

ESC – quit the application			SPACE BAR – move to the next scene			
Q: roll left	**W**: forward	**E**: roll right	**R**: up		↑: look up	
A: left	**S**: backward	**D**: right	**F**: down	←: look left	↓: look down	→: look right

Hints

To solve this exercise it is better to work into steps.

1 – Focus on the Vertex shader, and use a Fragment shader that returns a solid color.
2 – Focus on the implementation of the light models, and return a black and white value, corresponding to the quantity of light received in each point.
3 – Implement the BRDF, to see the effect of diffuse Lambert and Phong and Blinn specular models.
4 – Finally add the texture support to read the diffuse, ambient and specular color from.