LAB 5: Textures

Computer Graphics

2020-2021

1 Introduction

Go to Aula Global, **download** the framework (*CG2021_p5.zip*), **open** the solution file *visual.sln* which is inside the */visual* folder, **compile** it and **execute** it. You will also find a README.md that explain how to compile the code in case you are not using Windows.

1.1 About framework

We will use a new project, which uses the same framework as in the Assignment 4 but it contains an example about how to load and use a texture in the shader.

- There are two new textures, one with **COLOR_SPECULAR** and one with **NORMALS**.
- It also includes a shader that uses a texture.
- You can bring your illumination shaders from assignment 4.

1.2 Where to start

1. **Read the slides carefully**, using textures could be tricky
2. Bring the phong shader from assignment 4 to assignment 5.
3. **Load the textures** and pass them to the shader from the `Application class`.
4. Read the pixel from the **color texture (xyz)** and use it in your equation as the material ambient, diffuse and specular.
5. Read the alpha value from the **color texture (w)** and use it as the specular factor (not the shininess)
6. Once the illumination seems to work **compare it with the demo** supplied to be sure it looks correct.
7. Now read the pixel normal from the normal map and use it instead of the vertex normal (but remember that normals stored in colors are encoded from 0 to 1 and normals should go from -1 to 1, so you should do the conversion manually). Remember to normalize it afterwards and take into account the model transformation!.
8. Try to use the normal map and the color map in the assignment 3 plus (ignore the specular channel as assignment 3 does not support alpha for images).

2 Objectives

This assignment consists of understanding how the textures works in the GPU. At the end of this assignment you should have gained the following competences:

- How to read from a texture
- How to use the data in the light equation

By the end you should get some results comparable to:

Figure 1: Compare results with the online demo

3 Assignment

You must extend your shader to use textures now. First you must load textures into the GPU using the `Texture class` (it automatically uploads them to the VRAM). Then pass the texture reference to the shader using the uniforms. Check the slides about framework.

You lose 0.5 point if there are some of these errors:

- You remove to rotate the mesh using the key space as it is given.
- Using the coefficients in the wrong place.
- Not converting the normal by pixel correctly from color space to normal space.
- If normals do not take into account mesh rotation.
- Using the specular factor as specular gloss

Check also 4 Common errors section to avoid it.

3.1 Task 1: Use diffuse texture within Phong (2 points)

Now we will use the color texture to extract the material information of every pixel. We could assume that the color of the texture must affect all three components (p.e. it doesn't make sense that a red area of the face has a blue ambient). So we **read from the texture** and use the value to weight the ambient, diffuse and specular factors of the material.

(a) Diffuse texture. (b) Shaded model after applying the texture.

Figure 2: Task 1

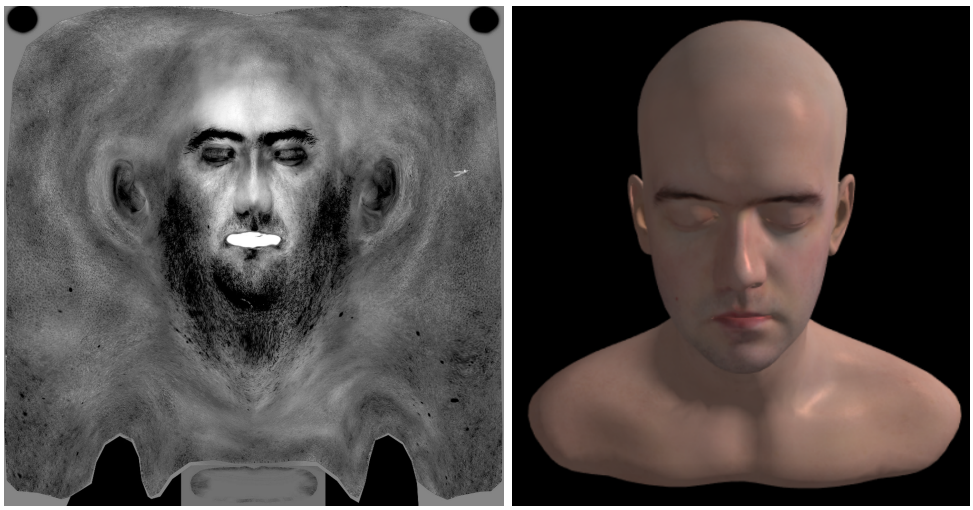
3.2 Task 2: Use specular texture within Phong (as specular factor) (2 points)

In the color texture we also have information about the specular coefficient (Ks), it is stored in the fourth channel of the diffuse texture.

Because **Ks** should be a color and we only can store one component, we can multiply this factor by the RGB color to get the amount of specular, as it makes sense that the hotspots are the same color as the surface.

$$\text{Ks} = \text{tex\_pixel.xyz} * \text{tex\_pixel.w};$$

Although for the skin sometimes it looks better if instead of using the color of the pixel, we use a constant white as reflections usually happen above the skin and not inside the skin.



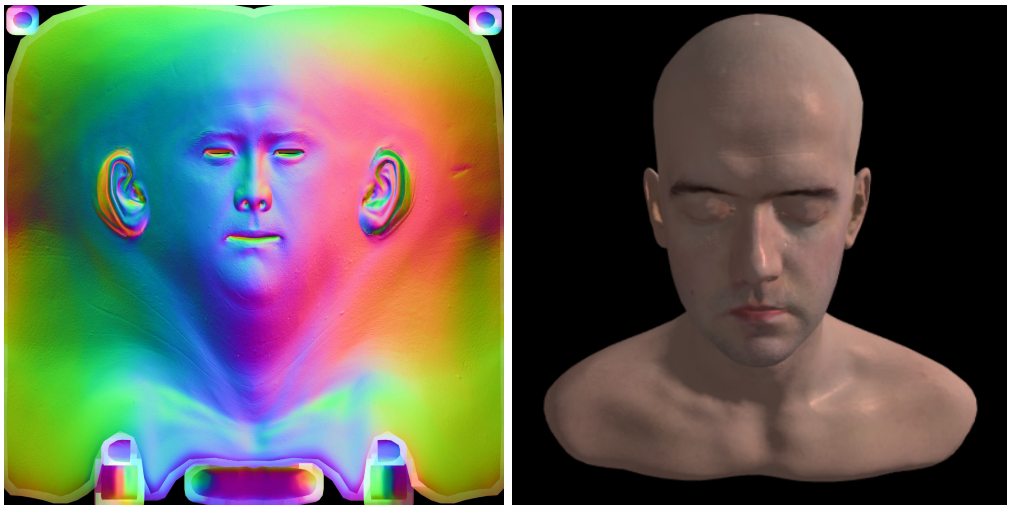
(a) Specular texture. (b) Shaded model after applying the texture.

Figure 3: Task 2

3.3 Task 3: Use normalmap texture within Phong (2 points)

In the project you will find another texture called ***lee_normal.tga***. This texture contains more precise normals (per pixel instead of per vertex). Use the normal from the normalmap texture in your equation to obtain more quality on the render, instead of using the normal per vertex that you had before. Remember that the normals are stored as colors, and colors can only go from 0 to 1, so to store normals (that go from -1 to 1) they were encoded by adapting the range -1..1 to 0..1. So -1 becomes 0, 0 becomes 0.5 and 1 becomes 1.

You must do the opposite conversion before applying any operation with the normals or the result will be wrong. Remember that normals are affected by the model matrix, because when an object rotates its normals rotates.



(a) Normalmap texture. (b) Shaded model after applying the texture.

Figure 4: Task 3

3.4 Task 4: Show several objects with several materials (2 points)

Take into account any necessary algorithms you may need in order to display multiple objects in the same scene rendered at the same time with different materials each one.

3.4.1 Modulating texture with uniforms

We could use the texture information as it is inside the illumination equation, but then we lose the freedom from code to change the material properties (p.e. to make the face more red or green).

So one trick is instead or using the texture directly, we can multiply it by the material properties. This way we have control from our application to change the color of the skin using the texture.

In this case be sure to change the material properties in your application to use (1,1,1) so they do not modify the texture by default.

3.5 Task 5: Use normal map and diffuse also in assignment 3 (2 points)

When porting this part to the Assignment 3 keep in mind:

- Now we need the ***worldPosition*** of the pixel, this information we did not have it before (we only had the position in screen space, we need to pass it)

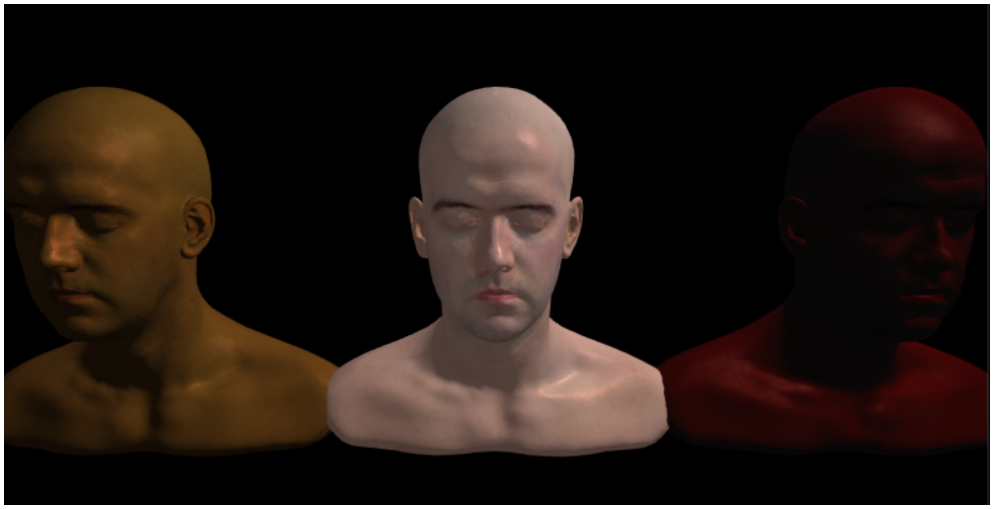


Figure 5: Showcasing different materials

- Pixels in class ***Image*** are stored using ***Color*** and the values go from 0 to 255, not from 0 to 1 as in the shader (and cannot store decimals!. We recommend to convert to 0..1 range and at the end convert back to 0..255 when writing to the image).
- The class ***Color*** can not store the alpha component so we can not read the specular factor, just ignore it for this part.
- Textures may have different size so careful when reusing the same UVs if they are not normalized.

4 Common errors

- Incorrect types: You cannot assign a vec4 to a vec3, or a float to a vec3. There are ways to convert types like: vec4 v = vec4(myvec3, 1.0)) or vec3 v = myvec4.xyz;
- Multiply a vec3 by a mat4: Remember to convert vec3 to vec4 adding a 1 in the fourth component (if you want to take into account the translation of the mat4) or a 0 (if you just want the rotation). To get the result as a vec3 you can do this: vec3 result = (M * vec4(V,1.0)).xyz
- Ambiguous types in functions: Some functions do not accept to have mixed types (like a float and an int). You must always pass the same type, like (float,float) or (int,int), when passing (float,int) the compiler doesn't know if you want to use the (int,int) version or the (float,float) version of that function y crashes.
- Inmutable varyings: You should not modify the varyings from the Fragment Shader. You must convert them to local variables first.
- Undeclared variable: You must declare all variables in the shader, no matter if you already did in the other shader.
- Error when reading gl_Vertex, gl_Normal or gl_Color on fragment shader: Those variable are restricted to the Vertex Shader, if you want them in the Fragment Shader you must pass them using varying
- Operating with numbers: Older versions of GLSL do not allow to operate between floats and ints, you must cast them. (0.0 is a float, 0 is integer, to cast you can type (float)my_int.

5 Delivery

You must send a ZIP file containing:

- A */src* folder with the source code (.h and .cpp)
- A */res/meshes* folder with the meshes (.obj) that you used.
- A */res/shaders* folder with the shaders used (.vs,.fs). item A */res/textures* folder with the textures.
- If you did the illumination and you applied textures in assignment 3, create an independent folder called P3extra and include the whole P3 project there with all the work.
- A README.txt with **NIAs**, **Names**, **Mails** and a description of how to use the application (which keys, which options, etc.).

Do not include strange files (.bin, .pch) that make the file too big, there is a size limit!

The ZIP must be delivered through the AulaGlobal on the specific task inside the course. The limit date is specified there, but it will be the day of the next session.