Computer Graphics (MIEIC)

Topic 5

Application of shaders

Objectives

- Learn basic shader concepts
- Use vertex shaders to manipulate object geometry
- Use fragment shaders to manipulate colors and textures on the scene

Practical work

The following points describe the topics covered during this practical class, as well as the tasks to be performed.

Some of the tasks are noted with the icon (image capture). At these points, you should capture and save an image of the application in the browser (e.g., using Alt-PrtScr on Windows or Cmd-Shift-3 on Mac OS X to capture to the clipboard and then save to file in an image management application of your choice). At the end of each class, students must rename the images to the format **"ex5-t<class> g<group>-n.png"**, where **class** corresponds to the class number (e.g., 2MIEIC01 is class '**01**'), and **group** corresponds to student group number (defined in the TP group file in Moodle), and **n** corresponds to the number provided in the exercise (e.g., **"ex5-t1g01-1.png"**).

In tasks marked with the icon (code), students must create a **.zip file from the folder that contains your code (typically in the 'ex5' folder, if you have code being used in other folders, include it too)**, and name it **"ex5-t<class>g<group>-n.zip"**, (with class, group and provided number identified as described above **"ex5-t1g01-1.zip"**).
At the end (or throughout the work), one of the elements must submit the files via Moodle, through the link provided for that purpose. Only one member of the group is required to submit the work.

Preparation of the Work Environment

Students should download the code available for this topic on Moodle, and place the ***ex5*** folder contained in the .zip file at the same folder as the previous lessons.

Shaders

Carefully study the additional slides provided with this practical lesson, and keep in mind the additional resources available on Moodle.

Experiences

The source code provided for this lesson, available on Moodle, displays a **teapot**, a mesh traditionally found in projects based on OpenGL. Observe carefully the various types of *shaders* available, and study the corresponding *vertex shader* and *fragment shader* code.

1. Selecting the *“Passing a scale as Uniform”* shader, change the *scaleFactor* value in the interface. Check what happens to the vertices of the teapot, alternating the “*Wireframe*” mode. Study the code to check how the *scaleFactor* value is passed.
2. Do the same with the *“Multiple textures in VS and FS”* shader.
3. Analyse the code for the first 6 sets of shaders.
4. At the ***ShaderScene*** class, observe the function ***update(t)***, a standard ***CGFScene*** function that receives the current time in milliseconds. Check how it is used in the vertex shader for the animation.
   Note that for this function to be invoked, the update period of the scene had to be defined in the ***init()*** function, using the ***setUpdatePeriod()*** method.
5. Observe the effect of the “*Sepia*” shader, and observe how the color changes in the fragment shader.
6. Observe the change in color of the texture applied to the object with “*Convolution*” shader, which implements an *edge detector* in the fragment shader.
   (https://en.wikipedia.org/wiki/Kernel_(image_processing)).

Exercises

Teapot Shaders

1. Create new vertex and fragment shader files to color the teapot according to the position occupied by the fragments in the screen - yellow in the upper half, blue in the lower half (see fig. 1).
   To do this, you must use the position of the vertices after the transformation (as stored in *gl_Position*). Create a *varying* variable in the vertex shader to store the position of the vertex, and to be passed to the fragment shader. There, retrieve this value and display yellow if y> 0.5 and blue if lower.

   (1 📷)

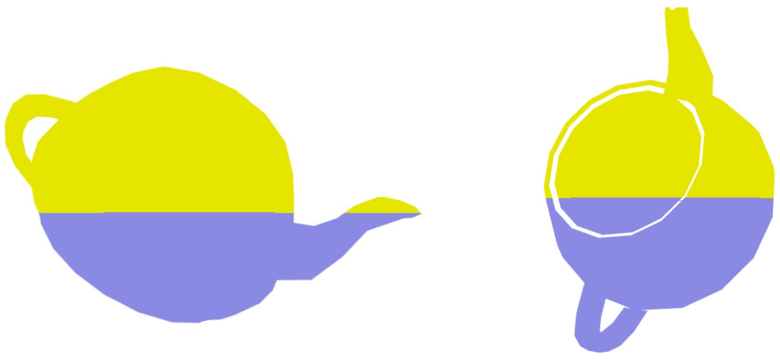

Figure 1: Exemplification of the teapot with color change according to the coordinates on the scene.

2. Change the animation shader files to create a back-and-forth translation effect on the X axis, following a sine wave. The translation effect should depend on the *scaleFactor* of the interface. Tip: add a new offset to *aVertexPosition* in the vertex shader.

3. Create a new fragment shader based on the "*Sepia*" shader that converts the texture color to shades of gray (Grayscale[1]). To do this, convert all RGB color components to L = 0.299R + 0.587G + 0.114B.

   (2)

Plane Shaders: Water Effect

1. Create two new shader files 'water.vert' and 'water.frag', based on 'texture2.vert' and 'texture2.frag'. Add the new files to the project and make them available on the interface. Select the ***Plane*** object, provided in the sample code, in the scene (using the dropdown menu in the interface).

2. Replace the textures on the scene with the images *waterTex.jpg* and *waterMap.jpg*. With the shaders created in the previous task, verify that you see a water texture with dark red spots on the Plane.

3. Change the vertex shader to use *waterMap.jpg* as a height map for the water texture (that is, each vertex must be shifted according to the value of one of the color components of the texture).

4. Animate the plane using the two new shaders, where the mapping of the texture coordinates to the vertices and fragments must vary over time, to obtain an effect similar to what can be seen in the example on the next page (link to video).

   (3) (1)

[1] https://en.wikipedia.org/wiki/Grayscale#Converting_color_to_grayscale

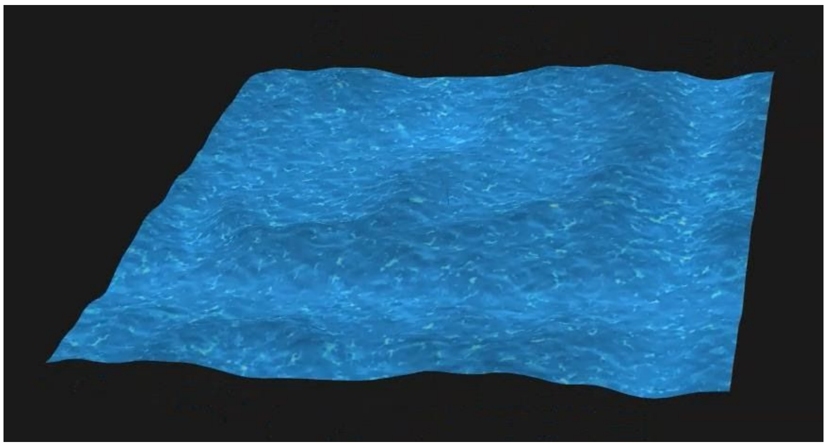

Figure 1: Water shader
Video: https://drive.google.com/open?id=1gSgOrhpVg10GxwIXMBcRewxV5dU8o1FH

Checklist

Until the end of the work you should save and later submit the following images and versions of the code via Moodle, **strictly respecting the naming rule**:

- **Images** (3): 1, 2, 3 **(named as "ex5-t<class>g<group>-n.png"**)
- **Code in zip file** (1): 1 (**named as "ex5-t<class>g<group>-n.zip "**)