Coursework commentary 2017–2018

CO3355 Advanced graphics and animation

Coursework assignment 1

General remarks

This coursework assignment consisted of two parts. Each part was divided into two questions. Part A focused on three dimensional geometric transformations and asked students to implement and apply them on a series of objects using:

- Processing.
- GLSL code. Part B asked students to devise and implement a mathematical function in order to generate a rectangular grid pattern and project it on an object using GLSL, and then to improvise and experiment with different shapes and effects.

Generally, there was a wide variance in the quality of the answers provided. Some students performed exceptionally well, but there were also some submissions that were too poor to receive a passing mark.

A large number of students (more than 30 per cent) chose to limit their submission to the first part only. Apart from missing an opportunity to experiment with an important aspect of the module's content such as procedural texturing, that also meant that they aimed at best for a mark of 60/100. Moreover, students this year appeared to be facing more difficulties with GLSL. Hence, while most performed very well in the first question of Part A, they failed to do so in the second question, which proved more challenging. As an effect, a high number of these submissions failed to attract a passing mark. Students are highly advised to attempt all parts asked for in an assignment.

The quality of reporting also varied substantially, with many of the reports being too thin or vague. In some cases, due to particularities in the configuration and/or missing files from submission, it is not possible for the examiners to execute the code submitted and reproduce results. This makes proper documentation of your attempt even more important. However, some of the reports failed to even clearly specify which questions had been attempted.

The examiners would like to stress the importance of following the submission instructions in terms of positioning and naming files and folders. Moreover, in order for a Processing program to run correctly, it needs to be placed in appropriate folders, with the folder name being identical to the name of the main Processing executable. Also, students should not forget to include all other necessary material (such as GLSL and OBJ files) in appropriate folders. Finally, they are asked to confirm that their program runs before they submit, by extracting the compressed file to a separate location and verifying that it works as expected.

Comments on specific questions

Part A

Question 1

This question asked students to write a Processing program that performs Translation, Rotation and Scaling on a given 3D object, complementing it with a simple Graphical User Interface to allow for user interaction. Students had to use their code on a minimum of three different objects, ranging from:

- Simple Processing primitives.
- Shapes generated via a script.
- Objects imported from a file.

Most students provided good answers here, using appropriate objects and implementing transformations correctly. However, some had problems with the sequence of transforms while others did not make provisions for all axes and/or directions in their implementation of rotation and translation. There were also some minor problems with respect to the sensitivity of the GUI controls, which was often too high, leading to disproportionate transformations that expanded outside the screen borders.

The external object selection could also prove problematic sometimes. If the object was of very high detail, it could entail a very large number of vertices and polygons. While this was not wrong per se, it could make calculations much heavier. To address this, some students chose to use Blender and other 3D modelling applications in order to modify the chosen model and reduce its size. Students are generally advised to carefully select the objects they experiment with, balancing between complexity and detail.

Question 2

In question 2 students were asked to implement the transformations of Question 1 but now using GLSL shaders, integrating GUI elements and experimenting with the same objects, and compare the results. The aim here was to implement the transformations using appropriate matrix multiplications in homogeneous coordinates, as is described/shown in the subject guide.

A challenge faced by most students was to stop GUI components from being transformed together with the scene objects. Most tackled this by resetting the shader right before drawing the GUI components. Another challenging aspect was applying different transformations to objects independently. In order to minimise CPU and GPU communication, processing groups as many shapes as possible to draw in a single call, using a common transformation matrix. This behaviour was prevented by disabling optimised stroke.

Despite these issues, the majority of students who attempted this question did well, with some exceptional attempts incorporating features such as lighting, which made the scenes very realistic, although it was outside the scope of this question.

Part B

Question 1

Here students were asked to write GLSL shaders that generate a rectangular grid pattern and project it onto a shape. The rectangle size should change dynamically, based on the mouse position.

A way to address this would be to capture the mouse position inside the Processing code and then pass the value to GLSL, letting the fragment shader

generate the grid pattern. Assessing whether a pixel would be drawn or not could be done using the mod() function. Unfortunately, this task proved difficult for many students.

Question 2

This question asked students to experiment with different effects, including time-varying shapes and colours.

This question, being an open-ended one, provided space for improvisation and allowed students to bring in more creative aspects. Even more than in the previous parts, there was a wide range of effort, from very rudimentary to extremely sophisticated.

Many students took the opportunity and produced really nice effects. Most began with code found in the assignment online references, utilising various random noise functions and experimenting with their parameters, modifying them using trigonometric or other mathematical functions. Others experimented with modifying the shapes of the object (i.e. working in the vertex shader), producing equally interesting and visually satisfying results. There were also some students who incorporated music and sound in their sketches, making the effects in shape or colour be dynamically generated based on variations in the accompanying audio, which was also very interesting.