Practical 2: Graphics implementation

CS4102 Computer Graphics

Due date: Wed 3 Apr (Week 11) 21:00
50% of the coursework grade
MMS is the definitive source for deadlines and weights

Aims

The aim of this practical is to gain experience in main tasks of computer graphics using WebGL. A successful submission will demonstrate the understanding of:

- how to model an object;
- how to transform coordinates using a vertex shader;
- how to texture and shade an object; and
- how to animate objects.

Task

You will implement a renderer in WebGL capable of showing a 3D object in a browser window. You are welcome to use any of the example code on studres and any of the examples provided as part of the module to help you get started, but the use of any external code beyond this must be clearly identified and referenced. You are not allowed to use 3rd party JavaScript libraries like `three.js`. The only exception are the two libraries we introduced in the labs that help with matrix manipulation: math.js and glmatrix. Your solution should consist of one HTML file for each of the four tasks listed below and it should be possible to serve a local webpage on a lab machine by running the following command, as discussed in lab sessions:

```
python3 -m http.server <port number>
```

The task is split into several parts, increasing in complexity. You are strongly encouraged to complete them in the stated order and not to move to more advanced tasks until you are happy with your solution to more basic tasks.

Part 1: Basic tasks

In this part, you should define a 3D model in terms of vertices, indices (of triangles), normals, and vertex colours. The object should not be one of the shapes we covered in labs (cube, cylinder, sphere, or cone), and should contain at least 8 individual faces (each of the faces may be represented by multiple triangles). A solution to this part will implement all of the following:

- one object as described above;

- one texture applied to the object;
- ambient shading (constant illumination for all points);
- simple animation, such as rotating the object around a point; and
- one camera, using oblique perspective.

Since oblique projection is used, you will need to scale the object down so that it comfortably fits within the clipping volume.

Part 2: Standard tasks

Only attempt this part once you have successfully completed Part 1. In this part, you will extend your solution to Part 1 to render a more realistic and engaging scene. Your solution will implement all of the following:

- perspective projection;
- three or more objects using more than one texture;
- flat shading with a coloured, directional light source; and
- animation: all objects should rotate around an arbitrary point (not the origin) at different speeds.

All parameters (direction and colour of the light source and the point around which the objects rotate) need to be defined in JavaScript and passed to the relevant shaders where appropriate. In your report, you should explain the animation parameters and camera parameters, and how you calculated illumination at each vertex.

Part 3: Advanced task

Only attempt this task once you have successfully completed Parts 1-2. This part extends your renderer with advanced techniques discussed in class (but not covered in labs). Your solution should implement **one** (and only one) of the following advanced tasks:

- Gouraud shading with a coloured spotlight, **or**
- reflection mapping (e.g., a skybox and a surface which reflects the skybox texture).

Note that this part requires a spotlight (not directional light from Part 2), and that any spotlight parameters should be passed as parameters from JavaScript. To best demonstrate Gouraud shading, your object should have a curved surface and appropriate normals similar to (but not identical) to examples we saw in labs. You are not expected to use parametric surfaces for this part, generating a surface algorithmically is fine.

Part 4: Champion level

These tasks are for those of you looking for a challenge. Only attempt this part once you are happy with your solution to Parts 1-3. It is perfectly fine not to attempt these. Attempt **one** (and only one) of the following very advanced tasks:

- a mirror surface showing other objects (two-pass rendering); **or**
- fractal-based generation of a terrain mesh.

These are open-ended challenges so partial implementations are fine, as long as they demonstrate advanced understanding of graphics concepts.

Submission

Hand in, via MMS, a ZIP file containing:

- the solution to each attempted part (as a separate html file or directory), and
- a brief report describing your solution to each attempted part.

Upload a single `.zip` file containing the report and code to MMS. The report does not have to be long. It should focus on which sub-tasks you completed, explain the approach you used to implement your solution (such as explaining the calculation needed for shading), and point out where in the code the computation happens. For example if you are describing terrain generation, you would explain your mesh structure and where in the code the perturbations happen.

Marking

This practical will be marked according to the guidelines at feedback section of the student handbook. Some examples of submissions in various bands are:

- A *poor implementation* **in the 7-10 grade band** will contain part of the solution that demonstrates some understanding of main graphics concepts but does not work reliably and misses much of the functionality needed for Part 1.
- A *basic implementation* **in the 11–13 grade band** will complete Part 1. The report should demonstrate understanding of basic graphics concepts. The lower parts of this mark range will correspond to submissions where there are weaknesses or the report demonstrates a lack of understanding.
- An implementation **in the 14–16 range** should complete Parts 1-2, including a clear description of what is done and why. A mark of 16 will correctly implement all requirements for Part 2 successfully, with understandable code and an insightful report.
- An implementation **in the 17–18 range** will complete Parts 1-3. This grade band requires an excellent implementation of Parts 1-2 and a solid, non-trivial progress on Part 3 including a report that demonstrates understanding of advanced graphics concepts covered in Part 3.
- A grade of **19 and higher** will require completion of Parts 1-4 and evidence of ability to implement complicated algorithms and incorporate knowledge from sources outside of the lecture materials.

Also note that:

- Standard lateness penalties apply as outlined in the assessment section of the student handbook.
- You must reference any external sources used. Guidelines for good academic practice are outlined in the GAP section of the student handbook.