Unit:	6G5Z2102 Computer Graphics
Assignment set by:	N. Costen
Assignment ID:	1CWK100
Assignment title:	Portfolio
Type:	Individual
Hand-in deadline:	Friday 26th February 2021, 21:00
Hand-in format and mechanism:	Via Unit area on Moodle
Support:	Via online appointments, timetabled sessions and documents.

Learning Outcomes Assessed: This assignment will assess your ability to:

- analyse, validate and apply the mathematics governing transforms, modelling, texturing and lighting in 2D and 3D graphics;
- develop interactive 2D and 3D graphics software using an industry-standard high-level programming language and graphics API;
- compare, evaluate and apply techniques for the digital representation of images;
- identify and describe the components of the graphics pipeline in the context of current computer gaming hardware.

Note: it is your responsibility to make sure that your work is complete and available for marking by the deadline. Make sure that you have followed the submission instructions carefully, and your work is submitted in the correct format, using the correct hand-in mechanism (e.g. Moodle upload). If submitting via Moodle, you are advised to check your work after upload, to make sure it has uploaded properly. Do not alter your work after the deadline. You should make at least one full backup copy of your work.

Penalties for late hand-in: see Regulations for Undergraduate Programmes of Study (`https://www.mmu.ac.uk/academic/casqe/regulations/assessment-regulations.php`). The timeliness of submissions is strictly monitored and enforced. All coursework has a late submission window of 5 working days, but any work submitted within the late window will be capped at 40%, unless you have an agreed extension. Work submitted after the 5-day window will be capped at zero, unless you have an agreed extension.

Please note that individual tutors are unable to grant extensions to coursework. Extensions can only be granted on the basis of a PLP, or approved Exceptional Factors (see below).

Exceptional Factors affecting your performance: see Regulations for Undergraduate Programmes of Study (`https://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf`). For advice relating to exceptional factors, please see the following website: `https://www2.mmu.ac.uk/student-case-management/guidance-for-students/exceptional-factors/` or visit a Student Hub for more information.

Plagiarism: is the unacknowledged representation of another persons work, or use of their ideas, as ones own. Manchester Metropolitan University takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook (`http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf` and Regulations for Undergraduate Programmes (`http://www.mmu.ac.uk/academic/casqe/regulations/assessment-regulations.php`). Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

As part of a plagiarism check, you may be asked to attend a meeting with the Unit Leader, or another member of the unit delivery team, where you will be asked to explain your work (e.g. explain the code in a programming assignment). If you are called to one of these meetings, it is very important that you attend.

Assessment Criteria:	Indicated in the attached assignment specification.
Formative feedback:	Written and spoken feedback will be provided as detailed in the attached assignment specification.
Summative feedback format:	Marking grid and written feedback (specified in a separate document) will be provided.

5Z2102 Computer Graphics - Portfolio

Assessment Summary: The assignment will have two parts, which have equal weightings. In the first, you will be required to develop up to three "demonstrations", each showing the possibilities of OpenGL in a different area. A video of the sets of code operating must also be supplied. In the second, you will be required to answer a selection of written questions, which will test your theoretical understanding of the content of the course.

Part I (code):

You will be required to design and develop a set of three "demonstrations", each showing up a different aspect of computer graphics in OpenGL. Each of your demonstrations will be worth a maximum of 33.3% of the marks available for the code component of the assignment.

The demonstrations must be developed in Microsoft Visual Studio v2019 and the C++ Programming language using OpenGL. You must ensure that the version of OpenGL that you use is compatible with that in the University laboratories. You may use the code given you for the laboratory exercises as a starting point, but please note that all of the files (and in particular the Shaders) will need to be edited to solve the problems.

The code should be presented as a VS "solution", with each demonstration operating as a separate "project" (please note that you should not present the solutions used for the laboratory exercises, but prepare your own). As well as the individual demonstrations and the `SOF` code, ideally you should present a library of your own classes which contain those pieces of code which are common to the different demonstrations. Base code for the VS solution and a number of videos on the preparation of the demonstrations will be made available via Moodle.

By default, the demonstrations should be drawn from the following list. It may be possible to develop other demonstrations, but any such task must be cleared with the unit leader before being attempted.

1. A C++ implementation of an abstract strategy game other than Noughts-and-Crosses. The particular choice of game is up to you; the web-page `https://en.wikipedia.org/wiki/Tic-tac-toe#Variations` has some options. The visual interface can use ASCII. A basic implementation will allow the two players to alternate goes, a more advanced version will allow the user to play the computer, while further marks will be gained for implementing automatic detection of success by either player. Particular attention will be paid to the object-oriented quality of the code when assigning marks to this demonstration.

2. A demonstration of the interactive rendering of coloured objects in OpenGL. It should be possible to add and subtract objects of different types; these should be drawn at random locations, scales and orientations relative to the viewer. The location and orientation of the view point should also be possible to move. Additional marks will be awarded for the use of the mouse to add and subtract objects, an increased range of objects and the addition of functionality such as the ability to swap between constant and interpolated colours.

3. A demonstration of the nature of articulated motion. As a minimum, this will display a composite object built of two components, a base and arm. It is acceptable that these be simple coloured boxes. At a minimum, the base should translate and both should both rotate, so that the arms is moved relative to the base. Additional marks will be awarded for the inclusion of additional components (so an upper and lower arm, with the upper arm connected to the lower arm with an "elbow"), and for the presence of interactive movement (preferably via the mouse).

4. A demonstration of collision detection and interactive object control via a small 3D dodging game. A "floor" should move towards the viewer, on which a player cube should be located. At random intervals, "obstacles" (also cubes) should appear from behind a back wall and move towards the viewer (as a consequence of the floor moving). The player is required to move forward and get to the back wall without hitting any obstacles (it is possible to dodge or jump them). Additional marks will be awarded for the inclusion of collision detection, randomization of the object shape, inclusion of a scoring mechanism (displayed via text) and inclusion of speed-ups.

5. A demonstration of the nature and possibilities associated with texture mapping. At least two objects will be rendered, textured with different images. The user must be able to interactively change between images and rendering options, through the keyboard. A range of features should be present (greater numbers will allow higher marks): multiple textures buffers allowing speedy changes between patterns, transparency of texture, mosaic-ed textures (so that different faces of a single objects have different patterns), different interpolation techniques, the use of user-determined patterns and the use of variable fonts to write text.
6. A demonstration of the nature of morphing. This will involve shader programming, so that the vertex-configurations are loaded into the renderer in advance, and only a parametric time value is required at a given frame. Extra marks will be awarded for the use of multiple key-frames and interpolation of texture vertexes. Note that this is a higher-level task, which requires a through understanding of shaders. Marks will be awarded accordingly.
7. A demonstration of path-finding algorithms. This should include allowing the user to choose between a range of different distance metrics and set movable start and end. In addition, higher marks will be awarded for the inclusion of high-cost cells, a complex maze-style environment which can be edited, and the use of a 3D scene.

All demonstrations must:

- Be developed and implemented via Microsoft Visual Studio and the C++ Programming language using modern OpenGL (v3.1 or higher) with shaders.
- Use the `SOF` OpenGL interface, but not edit it.
- Be playable on a PC.
- Be appropriately object-oriented - there should be a minimum of code and variables present in the “application” class, and the classes supporting the shapes to be rendered should avoid unnecessary duplication of code.
- Not infringe copyright for your graphics. All sources to third part materials that you have used must be referenced.
- Reference the original sources of code which is drawn from elsewhere, for example online tutorials. This be done in both your code and reports. Otherwise this could be construed as plagiarism. This is individual work; you should not share or write code for your colleagues. Please make sure you understand the University’s policy on collusion and its implications.

Part II (text):

Answer any four of the following questions. Each full question adds up to 20 points. To calculate the overall assignment mark, the number of points given for each question will be divided by 1.6. Thus if you got full marks for each question (80 points), 50 marks would be added to the overall assignment mark.

The weighting for each sub-question is given by the value in square brackets at the end of each sub-question. The amount of text required to answer each question and sub-question will necessarily vary, but slightly under one page of normal-size type-written text (this equates with 400 words) would be sufficient for each 20-point question. Note that this work-count should not include references, which are extra. You can expect to vary the amount of writing required for each sub-question, in accordance with the number of points possible.

Where appropriate, you should supply references to support your statements; it is not necessary to reference facts drawn from the course notes. If you do not use references, your text could be construed as exhibiting plagiarism. This is individual work; please make sure you understand the University’s policy on collusion and its implications. Credit will be given for evidence of wider reading around the area (i.e. the use of authoritative sources which have not been supplied to you in the course of the unit).

Question 1

1. Consider the following piece of C++ code. Explain what the four `cout` commands (labelled as outputs 1 to 4) will generate, and why this is the case. [6]

```
int f1(int a) {
        a = a + 1;
        return a;
}

int f2(int& a) {
        a = a + 1;
        return a;
}

int main()
{
        int xx = 0;
        cout << f1(xx) << "\n"; // output 1
        cout << xx << "\n";     // output 2
        int yy = 0;
        cout << f2(yy) << "\n"; // output 3
        cout << yy << "\n";     // output 4
}
```

2. Explain the meaning and importance of the C++ keywords `public`, `private` and `protected` in the context of classes and inheritance. [5]
3. Explain the meaning of the C++ keywords `abstract` and `virtual` in the context of inheritance. [4]
4. Explain the difference between the `stack` and the `heap` in C++ and why this is important to consider when designing classes. [5]

Question 2

1. Describe the process of modelling and displaying a red cube defined by 12 triangular polygons. [10]

   In answering, consider the following steps:

   (a) Data structure
   (b) Geometry and Topology
   (c) Polygon Facing
   (d) Colour
   (e) Projection
   (f) Display

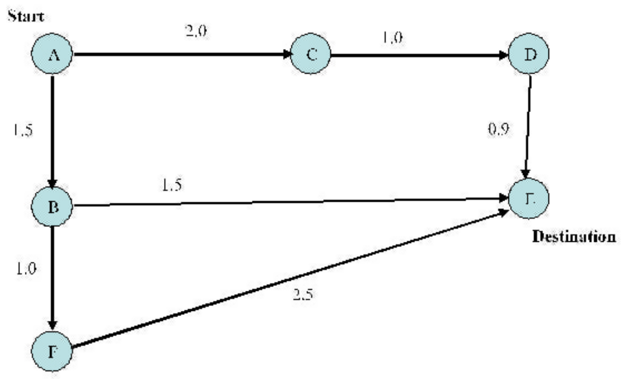
2. Explain why we use 4-valued homogenous coordinates to describe points and vectors in 3D space. [4]
3. Define the three basic elements required to allow geometric description of objects. [3]
4. Explain the meaning of the term `frame buffer` in computer graphics, and the meaning of `resolution` and `depth` in relation to the frame buffer. [3]

Question 3

1. What are the two main projection methods used in OpenGL? Describe their purpose, principles and differences. Draw a pair of pictures, which will illustrate the projection of an object using the two projection methods. [6]
2. Describe the Phong reflection model, explaining why it is necessary to include an ambient lighting term in the model. [6]
3. Explain the four basic types of light sources in OpenGL [4].
4. Describe the differences in purpose of the fragment-shader compared with the vertex-shader in OpenGL. [4]

Question 4

1. Write a short explanation of each of the following Path-finding terms [8]:
   (a) Node
   (b) Directed graph
   (c) Heuristic
   (d) Weighted graph
2. Consider the following graph which shows a collection of nodes through which a non-player character will pass from node A (the "start") to node E (the "destination"). Use a table to show the values of the following parameters for each iteration of Dijkstra's algorithm: 'status', 'cost-so-far' and 'parent'. Give these values for each node in the graph at each iteration. The 'status' parameter can take the following values: 'unvisited', 'open' and 'closed'. State the path the algorithm will generate. [12]



Question 5

1. Consider the following code example. Explain it, describing the language in which is written, its purpose within an OpenGL processing pipeline and the significance of each line. [10]

```
#version 330
in vec4 fragColor;
in vec2 fragTexCoord;
out vec4 finalColor;
uniform sampler2D diffuse;
uniform int useTexture = 1;
uniform float opacityLimit = 0.0f;
```

```
void main()
{
          if (useTexture == 1)
          {
                    vec4 col =  texture(diffuse, fragTexCoord);
                    if(col.a > opacityLimit)
                    {
                              finalColor.rgb = col.rgb;
                              finalColor.a = 1.0;
                    }
                    else
                              discard;
          }
          else
                    finalColor = fragColor;
}
```

2. Explain each of the following texture mapping concepts making clear how each is used [5]:
   (a) Texture tiling
   (b) Detail map
   (c) Mipmapping
3. Explain the differences between Dijkstra's pathfinding algorithm and the A* pathfinding algorithm. State with reasons which of these algorithm you would prefer to use in a computer game implementation. [5]

Question 6

1. Consider the following interpolation problem. An object starts centred at position $(1.0, 2.0, 1.0)$ and moves to position $(3.0, 2.0, 2.0)$ after 2 seconds and then to position $(5.0, 4.0, 3.0)$ after another 4 seconds. The movement is controlled by the following equations

$$\begin{aligned} v_f &= (1-t)v_i + tv_{i+1} \\ t &= \frac{f-k_i}{k_{i+1}-k_i} \end{aligned}$$

   where (v_i, k_i) are the key value and fractional time for the ith key-frame, and f is the fractional time. Draw a diagram showing its location against time, identify the type of interpolation used, and calculate its location 4 seconds after the start of the movement. [9]
2. Write short explanations the following terms in the context of computer graphics [6]:
   (a) Terrain heightmap
   (b) Multi-texturing
   (c) Fractal terrain
3. Briefly describe a use to which fractal geometry can be put in computer games. [3]

Assessment Criteria

You are required to design and develop the demonstrations, and describe them to your tutor. This will be achieved via a combination of the code itself and a video of the demonstration operation. You are also required to write text answers to the questions.

The final part of the portfolio submission requires you to submit the VS `solution`, composed of the final demonstration `projects` including source code and libraries, and also prepare the video of the operation of the demonstrations which needs to be uploaded into your YouTube channel. The video should cover all three of the demonstrations and need not last more than two minutes. It must be retained on the channel until 31st July 2021 at the earliest. The marking criteria are given below, note that the columns are independent of each other, and that the three demonstrations will be marked independently of each other. To advance down the columns all items must be completed in each block first i.e. all of the items in 40–49% must be fully working before items in 50–59% will be awarded marks.

With regard to the text, the questions will be marked by comparison with model answers. The levels below give an indication of the qualities expected for different marks, in parallel with the precise content.

Mark Range	Demonstrations (up to 3 items, each worth 12.5%)	Video (12.5%)	Text questions (up to 4 items, each worth 12.5%)
0%	No code submitted.	No video submitted.	No text submitted.
1–29%	No or limited implementation. Very primitive code; little or no evidence of design in construction of solution.	Major shortcomings in clarity and/or relevance. Impossible to discern the nature of the demonstrations submitted.	Very little information conveyed by answers; text and/or diagrams very difficult to understand.
30–39%	Partial completion of the specification requirements. Primitive code, showing a lack of thought in developing algorithmic solutions to problems.	Video lacks clarity and/or relevance. However, there is a little evidence that shows the nature of the demonstrations submitted.	Some appropriate information conveyed; text and/or diagrams difficult to understand. Few or no references.
40–49%	Completion of the basic elements of the specification requirements. Reasonably well organized code, with meaningful identifiers. Some evidence of planning in developing algorithmic solutions to problems.	Video is clear and relevant. There is some demonstration of the features of the demonstrations submitted. The goal of each demonstration is somewhat unclear.	A minimal level of correct information conveyed; text and/or diagrams comprehensible but simple. Some references to sources.
50–59%	Completion of some additional elements of the specification requirements. Well organized code, with meaningful identifiers at all levels. Some evidence of the development of objects. Well designed solutions to algorithmic problems.	There is a fair demonstration of major features of the demonstrations, some aspects lack clarity. The goal of each demonstration is demonstrated clearly.	A fair level of correct information conveyed, showing an understanding of the main points of the question. Text and/or diagrams clearly explain the main points. Clear references to sources, which can be followed up.
60–69%	Completion of several additional elements of the specification requirements. Well organized code, with meaningful identifiers at all levels. Evidence of the development of objects to solve problems. Well designed solutions to algorithmic problems.	Good presentation of main features of the demonstrations; straightforward to follow, good presentation of graphics possibilities.	A good level of correct information is conveyed using both text and diagrams as appropriate. The materials are clear and fully understandable, with appropriate and complete references. Some evidence of reading around the field.
70–79%	Completion of many additional elements of the specification requirements. Very well organized code, with meaningful identifiers at all levels. Evidence of the use of object-oriented methodologies to solve problems. Well designed solutions to algorithmic problems.	Good demonstration of all major features and some extra added features. Added subtitles about the author of the demonstrations and credentials.	A very good level of correct information is conveyed using both text and diagrams as appropriate. The materials are clear and fully understandable, with appropriate and complete references. Significant evidence of reading around the field.

Mark Range	Demonstrations (up 3 items, each worth 12.5%)	Video (12.5%)	Text questions (4 items, each worth 12.5%)
80–89%	Completion of all additional elements of the specification requirements. Very well organized code indeed, with meaningful identifiers at all levels. Copious evidence of the use of object-oriented methodologies (e.g. overloading, inheritance, templates, compilation to a library) to solve problems. Very well designed solutions to algorithmic problems.	Excellent demonstration of all features of the demonstrations. Strong evidence of prior planning. Demonstration video mixed with author's description (verbal or text) of the features.	A excellent level of correct information is conveyed using novel text and diagrams as appropriate. The materials are clear and fully understandable, with appropriate and complete references. Significant evidence of reading around the field and of insight into its issues.
90–100%	Completion of all additional elements of the specification requirements, other significant features also added. Excellent code indeed all levels. Copious evidence of the use of object-oriented methodologies (e.g. overloading, inheritance, templates, compilation to a library) to solve problems, including the use of design patterns. Very well designed solutions to algorithmic problems.	Outstanding demonstration of all features of the demonstrations. Compelling evidence of prior planning. Demonstration video mixed with author's clear description (verbal or text) of the demonstrations' features.	A truly excellent level of correct information is conveyed using novel text and diagrams as appropriate. The materials are clear and fully understandable, with appropriate and complete references. Significant evidence of reading around the field and of deep insight into its issues.

Submission

The following files should be submitted via Moodle or OneDrive.

1. A ZIP archive containing all C++ code, assets (image files, mesh files) and Visual Studio project and solution files required to build and run the programs.
2. A PDF file which describes the keyboard and mouse actions required to operate the demonstrations, and any extra features you have implemented.
3. A link allowing access to the video.
4. A PDF file giving your answers to the questions.

All PDF files should have your name and student ID at the head of each page; clearly show which demonstration or question you are attempting to answer.

OneDrive or Moodle submission:

- If the total size of the submission is less that 100MB, you can upload the zip file directly to the Moodle Inbox in this case you will not need to use OneDrive.
- If the submission is larger than 100MB, you should upload the submission to your University OneDrive account. Share the file with `n.costen@mmu.ac.uk`, which will send me an email with a link to your file. You should then submit a text file to the Moodle inbox which contains the path to the file. This path can be obtained by hovering the mouse over the file in OneDrive, and should look something like:

  `https://stummuac_my.sharepoint.com/personal/XXXXXXXX_stu__mmu_ac_uk/Documents/Filename.zip` (the text XXXXXXXX is your student ID).

Support and Feedback:

Verbal feedback will be provided on request via video conferencing (the tutor will in particular be available in the 14:00 Monday afternoon laboratory sessions) and through individual formative guidance via MS Teams. Contact details are available from the unit area on Moodle. Scaffolding code and videos on both setting up the demonstrations and also answering text questions will also be made available on Moodle. Marks and feedback on your final work will normally be provided within 20 working days of its submission deadline.