|  |  |  |
| --- | --- | --- |
|  | **FACULTY OF COMPUTING, ENGINEERING and SCIENCE** | Final mark awarded:\_\_\_\_\_ |

**Assessment Cover Sheet and Feedback Form 2015/16**

|  |  |  |  |
| --- | --- | --- | --- |
| Module Code:  CS3S603 | Module Title:  Real-Time Rendering Techniques | | Module Lecturer:  Paul Angel |
| Assessment Title and Tasks:  Procedural Terrain in GLSL | | | Assessment No.  1 of 2 |
| No. of pages submitted in total including this page: | | | Word Count of submission: N/A |
| Date Set:  27/10/15 | | Submission Date:  27/11/15 | Return Date:  15/01/16 |

|  |  |
| --- | --- |
| ***Part A: Record of Submission (to be completed by Student)*** | |
| **Extenuating Circumstances**  If there are any exceptional circumstances that may have affected your ability to undertake or submit this assignment, make sure you contact the Advice Centre on your campus prior to your submission deadline. | |
| **Fit to sit policy**:  The University operates a fit to sit policy whereby you, in submitting or presenting yourself for an assessment, are declaring that you are fit to sit the assessment. You cannot subsequently claim that your performance in this assessment was affected by extenuating factors. | |
| **Plagiarism and Unfair Practice Declaration:**  By submitting this assessment, you declare that it is your own work and that the sources of information and material you have used (including the internet) have been fully identified and properly acknowledged as required[[1]](#footnote-1). Additionally, the work presented has not been submitted for any other assessment. You also understand that the Faculty reserves the right to investigate allegations of plagiarism or unfair practice which, if proven, could result in a fail in this assessment and may affect your progress. | |
| **Details of Submission:**  Note that all work handed in after the submission date and within 5 working days will be capped at 40%[[2]](#footnote-2). No marks will be awarded if the assessment is submitted after the late submission date unless extenuating circumstances are applied for and accepted (Advice Centre to be consulted).  Work should be submitted as detailed in your student handbook. You are responsible for checking the method of submission. | |
| **You are required to acknowledge that you have read the above statements by writing your student number (s) in the box:** | Student Number:  13042467 |

**IT IS YOUR RESPONSIBILITY TO KEEP A RECORD OF ALL WORK SUBMITTED**

|  |
| --- |
| **Part B: Marking and Assessment**  **(to be completed by Module Lecturer)** |
| This assignment will be marked out of 100%  This assignment contributes to 40% of the total module marks.  This assignment is bonded |
| **Assessment Task:**  You are required to write an application that will render a 3D terrain using vertex and fragment shaders written in the OpenGL shading Language (GLSL). You can use any of the lecture demos or tutorial projects as a starting point.  You are required to complete the following tasks…   1. Create a 2D grid (lattice) of vertices that lies flat on the *xz* plane (so all *y* coordinates are equal to **0.0**). The size of the grid must be at least 32 **x** 32 points. The grid must be defined by at least one array storing the positions of each vertex and an index array containing the indices for each vertex that make up each triangle on the surface. Hint: The arrays can be setup as a static array or by dynamically allocating memory with malloc for example. 2. Setup the arrays you created above in Vertex Buffer Objects (VBOs) which are in turn encapsulated in a new Vertex Array Object (VAO). You can use the VAO/VBO examples given in the lecture demos as a guide. 3. Write a function drawSurface() to render the 2D grid stored in the VAO from the previous step. This can be called from the display() function in gldemo.cpp. 4. Implement a vertex shader that takes as input the per-vertex data from your VBOs created in step 2 above. Write a function in your vertex shader called calcOffset which is declared as…   vec4 calcOffset(vec4 pos)  …that takes as input a vertex position and outputs a new vertex position with an offset added accordingly. The offset can be calculated however you like, making good use of GLSL’s functions to create a 3D terrain-like surface. **Hint: More marks will be awarded for more complex functions. You can extend the function declaration to include additional parameters if you wish and you can add any per-vertex or uniform data you like as input to the vertex shader.**   1. If you connect each vertex in the grid in both the horizontal and vertical directions you would create a grid pattern as shown below. Implement a **fragment shader** that colours each fragment that lies on a grid line black and the other fragments white.      1. You will be required to explain your design and implementation in a short 5-10 minute code demo which will take place in the tutorial session after the assignment has been submitted. **The code demo is mandatory. The above sections will also be marked according to how well you demonstrate your understanding of them in the code demo.**   **Deliverables**   1. A zip containing the source code and executable of your implementation. This is to be submitted to UniLearn no later than the submission date shown on the assignment front sheet. Please name your zip file with your enrolment number (e.g. 12345678.zip). 2. A copy of this document is also to be included in your zip file, with your Student Enrolment Number filled in on the front sheet and the optional Reflection sheet (see Part C below) filled in accordingly. 3. A 5-10 minute code demo discussing your implementation, the results obtained and the problems you faced in implementing the assignment. |
| **Learning Outcomes to be assessed** (as specified in the validated module descriptor <http://icis.glam.ac.uk>):  LO1. Understanding of graphical modelling and rendering techniques.  LO2. Understanding of techniques used for real-time animation. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Marking Criteria** | **Mark Available** | **Mark Awarded** | | 1. Lattice / Grid setup | **15** |  | | 1. VAO and VBO setup | **10** |  | | 1. drawSurface function implementation | **10** |  | | 1. Vertex shader implementation | **30** |  | | 1. Fragment shader implementation | **15** |  | | 1. Code demo | **20** |  | |  |  |  | |  | **100** |  |   **Assessors Feedback** (linked to assessment criteria): |
| **Work on this module has been marked, double marked/moderated in**  **line with USW procedures.** |
| *Provisional mark only: subject to change and/or confirmation by the Assessment Board* |

|  |  |
| --- | --- |
| **Grading Criteria** | |
| **Performance Level** | **Criteria** |
| Fail  (< 40%) | The vertex and fragment shaders do not work correctly. The offset function and surface rendering are basic and little to no understanding of how shaders work is evident. |
| 3rd Class  (40% - 49%) | A correctly working vertex shader has been implemented but the fragment shader does not work correctly. A flat 2D mesh is sent to the GPU and modified correctly by the vertex shader but only a basic offset calculation is given. Only a rudimentary understanding of how shaders work is evident. |
| Lower 2nd Class  (50% - 59%) | A working vertex and fragment shader have been implemented but the implementation could be more efficient. A flat 2D mesh is sent to the GPU and correctly modified by the vertex shader but only a basic offset calculation is given. Some attempt at implementing a known terrain rendering technique has been made. A good understanding of how shaders work is evident. |
| Upper 2nd Class  (60% - 69%) | A correctly working vertex and fragment shader have been implemented. A flat 2D mesh is sent to the GPU and modified correctly by the vertex shader where a good offset calculation is given showing good use of GLSL’s language features and a good understanding of at least one terrain rendering technique. Good use of VBOs and VAOs is evident. A good understanding of how shaders work is also evident. |
| 1st Class  (70% +) | A correctly working vertex and fragment shader have been implemented. A flat 2D mesh is sent to the GPU and modified correctly by the vertex shader where an excellent offset calculation is given showing very good use of GLSL’s language features and a very good understanding of at least one terrain rendering technique. Good use of VBOs and VAOs is evident. An excellent understanding of how shaders work is also evident. |

|  |  |
| --- | --- |
| **Part C: Reflections on Assessment**  **(to be completed by student – optional)** | |
| **Use of previous feedback:**  In this assessment, I have taken/took note of the following points in feedback on previous work: | |
| **Please indicate which of the following you feel/felt applies/applied to your submitted work**   * A reasonable attempt. I could have developed some of the   sections further.   * A good attempt, displaying my understanding and learning, with   analysis in some parts.   * A very good attempt. The work demonstrates my clear   understanding of the learning supported by relevant literature and scholarly work with good analysis and evaluation.   * An excellent attempt, with clear application of literature and   scholarly work, demonstrating significant analysis and evaluation. | |
| **What I found most difficult about this assessment:** |  |
| **The areas where I would value/would have valued feedback:** |  |

1. University Academic Integrity Regulations [↑](#footnote-ref-1)
2. Information on exclusions to this rule is available from Campus Advice Centres [↑](#footnote-ref-2)