THE UNIVERSITY OF WESTERN AUSTRALIA

Department of Computer Science and Software Engineering

Mid-Semester Test 2018

UNIT CODE (CITS3003)
Computer Graphics and Animation

FAMILY NAME:_____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE: _____

This Paper Contains: **6** pages **(including title page)**
Time allowed: 45 **Minutes**

INSTRUCTIONS:

This paper contains 4 questions.

Candidates should attempt all questions.

TOTAL MARK: 30

Please provide your answer in the space provided for each question.

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Supervisors Only – Student left at:

This page has been left intentionally blank

Question 1. **(6 marks)**

What are attributes and primitives in OpenGL? Give two examples of each.

Answer

See Lecture 2 slides 8-10, and book sections 2.4 and 2.4.7.

Primitives: These are the simplest elements provided in a programming language e.g. OpenGL supports lines, points, loops, triangles, etc., denoted by GL_LINES, GL_POINTS, GL_STRIP, GL_LOOP, GL_TRIANGLES, GL_TRIANGLE_FAN, GL_TRIANGLE_STRIP.

Attributes: Properties that describe how an object should be rendered are called attributes

OR

Attributes are properties associated with primitives that give them their different appearances . E.g., Color, Size and Width, Stipple Pattern, Polygonmode etc.

Question 2. **(4 marks)**

(a) (2 marks) What is the difference between *request input mode* and *event input mode*?

Answer

See pages 10-13 of lecture 10.

A computer has a number of input devices such as keyboard and mouse. These devices contain a trigger which can be used to send a signal to the OS.

For the request input mode, the input is provided to the program only when the user triggers the device (e.g. entering a number then pressing <enter>). The program that requests this input comes to a halt until the input is entered.

Since most computer systems have more than one input device, each of these devices can be triggered at an arbitrary time by the user. Each trigger from these input devices generates an ”event” that is put in an ”event queue” which can be examined by the user program. For the event input mode, the program may be executing many other operations (i.e., it does not come to a halt) while waiting for an event to be triggered.

(b) (2 marks) Give any two common applications of vertex shaders in OpenGL.

Answer

See Page 3, lecture 6.

(i) Geometric transformations
- . change relative location, rotation, and scale of objects/camera
- . 3D perspective transformation - make far objects smaller

(ii) Moving vertices
- . morphing
- . wave motion and deformation due to physical forces
- . particle effects - fire, smoke, rain, ...

(iii) Lighting
- . calculate shading color using light and surface properties
- . special effect

Students are required to mention ANY TWO.

Question 3. **(10 marks)**

(a) (6 marks) Why do we need a frame of reference? Explain how points and vectors in 3D are represented in homogeneous coordinates. Why is the use of homogeneous coordinates important in computer graphics?

Answer

See Page 3–8, lecture 8.

We need a frame of reference to relate points and objects to our physical world, for example where is a point? We cannot answer this without a reference system.

In homogeneous coordinates points and vectors are denoted by 4 dimensional vectors

If we define $0.P = 0$ and $1.P = P$, then we can obtain the four dimensional homogeneous coordinate representation as follows:

$v = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & 0 \end{bmatrix}^T$

In vector the fourth dimension is zero, while for a point the 4th dimension is 1.

$P = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 & 1 \end{bmatrix}^T$

Importance: Homogeneous coordinates are key to all computer graphics systems. All standard transformations (rotation, translation, scaling) can be implemented with matrix multiplications using 4 x 4 matrices since hardware pipeline works with 4 dimensional representations.

(b) (4 marks) Explain what the following variables/functions do in an OpenGL program:

See Page 23 and 17, lecture 2.

i) `gl_FragColor;`

Each execution of fragment shader must output a color for the fragment. Used only on Mac. Value must be defined in the fragment shader.

ii) `glutInitWindowSize(175,450);`

Creates a window with width 175 pixels and height 450 pixels.

Question 4. **(10 marks)**

(a) (6 marks) What do the following functions do in OpenGL: 1) main(), 2) init() and 3) display() ?

Answer

See Page 16, lecture 2.

- **main()**: creates the window, calls the init() function, specifies callback functions (more about this in a later topic) relevant to the application, enters event loop (last executable statement)
- **init()**: defines the vertices, attributes, etc. of the objects to be rendered, specifies the shader programs (more about this later)
- **display()**: this is a callback function that defines what to draw whenever the window is refreshed.

(b) (4 marks) What is a *varying* variable? Describe when it would be suitable to use a varying variable.

Answer

See Page 13–14, lecture 5.

The qualifier varying is used to declare variables that are shared between the vertex shader and the fragment shader.

varying variables are used to store data calculated in the vertex shader and to pass down to the fragment shader. Again, because of the sharing of name space of the two shaders, varying variables must be declared identically in both shaders.

The varying qualifier can only be used with floating point scalar, floating point vectors and (floating point) matrices as well as arrays containing these types.

END OF PAPER