SOLUTIONS

Module:	3D Graphics Programming Tools		
Module Code	EBU5405	Paper	A
Time allowed	2hrs	Filename	Solutions_1516_EBU5405_A
Rubric	ANSWER ALL QUESTIONS		
Examiners	Dr. Marie-Luce Bourguet	Dr. Yizhe Song	

Question 1

a) Consider the following statement: "3D Graphics is created in real-time if the computer can process input as fast as or faster than the input is being supplied".

[6 marks]

i) What is the nature of the inputs in 3D Graphics?

(2 marks)

Solution: geometric primitives, 3D vertices

ii) What is the nature of the output?

(2 marks)

Solution: a raster image

iii) What is the architecture used for real-time 3D graphics?

(2 marks)

Solution: a rendering pipeline

b) Briefly explain how the following two techniques contribute to creating the illusion of depth in 3D Graphics.

[6 marks]

i) Perspective.

(3 marks)

Solution: The 3D clue provided by perspective is called foreshortening: nearby objects appear larger than distant objects. Perspective also refers to the angles between lines that lend the illusion of 3 dimensions.

ii) Hidden Surface Removal.

(3 marks)

Solution: This gives clues as to the true orientation of an object. We expect the front of an object to obscure its back from view. When hidden surface removal is used, back faces are not rendered.

c) 3D graphics viewport.

[7 marks]

i) Explain what a viewport is.

(2 marks)

Solution: A viewport is the region within a window that is used for drawing an object.

ii) Explain how a viewport can be used to enlarge or show a portion of an object inside a window.

(3 marks)

Solution: The image of the object is enlarged as the viewport area becomes larger. Only a portion of the object may be shown if the viewport is larger than the window.

iii) Modify the code of the function shown in Code box 1 in order to create two side by side viewports. The parameters w and h contain the width and the height of the window, in pixels.

(2 marks)

```
void myFunction (int w, int h)
{
    glViewport (0, 0, w, h);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
    glMatrixMode (GL_MODELVIEW);
}
```

Code box 1

Solution: glViewport (0, 0, sizex/2, sizey);
glViewport (sizex/2, 0, sizex/2, sizey);

d) For each of the following statements about the 3D graphics pipeline, state if it is true or false and justify your answer.

[6 marks]

i) The main purpose of the pipeline is to increase modelling efficiency.

(2 marks)

Solution: false, it is to increase rendering efficiency. Modelling does not belong to the pipeline.

ii) The pipeline architecture contains four transformation steps.

(2 marks)

Solution: false, it contains three transformations: modelling, viewing and projection transformations.

iii) The final step of the pipeline is removing parts of the model which are outside the camera view.

(2 marks)

Solution: false, the final step is rasterisation. Removing parts of the model is called clipping and is done before rasterisation.

Question 2

a) Consider the display callback function shown in Code box 2.

[8 marks]

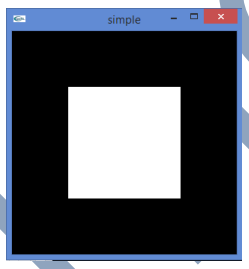
```
void mydisplay(){
     glClear(GL_COLOR_BUFFER_BIT);
      glBegin(GL_POLYGON);
            glVertex2f(-0.5, -0.5);
            glVertex2f(-0.5, 0.5);
            glVertex2f(0.5, 0.5);
            glVertex2f(0.5, -0.5);
      glEnd();
      glFlush();
}
```

Code box 2

i) Assuming that the OpenGL state machine is in its default state, sketch the 2D graphics output inside its window (draw the window as well).

(2 marks)

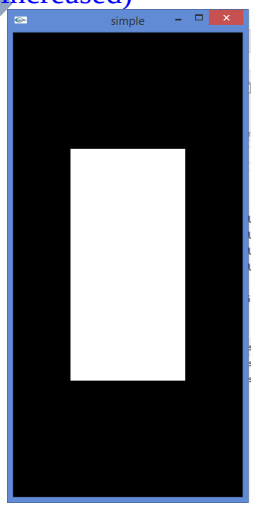
Solution:



ii) Sketch the new graphics output after the user has enlarged the window to be twice as tall as it used to be, while still having its original width.

(2 marks)

Solution: (vertical margins should have increased)



iii) Write the reshape callback function which ensures that the graphics output conserves its initial aspect ratio when the window's height is increased.

(4 marks)

Solution:

```
void myReshape(int w, int h)
{
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-1.0, 1.0, -1.0 * (GLfloat) h / (GLfloat) w, 1.0 * (GLfloat) h / (GLfloat) w);
}
```

b) The modelview duality.

[6 marks]

i) Give two examples of OpenGL modelling transformation function.

(2 marks)

Possible solutions: glTranslated; glRotatef; glScaled

ii) Give one example of OpenGL viewing transformation function.

(1 mark)

Solution: gluLookAt

iii) Explain why OpenGL uses only one matrix for modelling and viewing transformations.

(3 marks)

Solution: These two types of transformation are equivalent: there is no visual difference between moving an object backward and moving the reference system forward. So both viewing and modelling transformations are conbined in the ModelView matrix.

c) Homogenous coordinates.

[5 marks]

i) Explain what a homogenous coordinate system is.

(2 marks)

Solution: A homogenous coordinate system adds one value w to any point or vector, for example, a 3D point will be defined with 4 values in homogenous coordinates.

ii) Why are homogenous coordinates used in 3D Graphics?

(2 marks)

Solution: They are used to allow the representation of translation transformations in matrix form, because translations are not linear transformations.

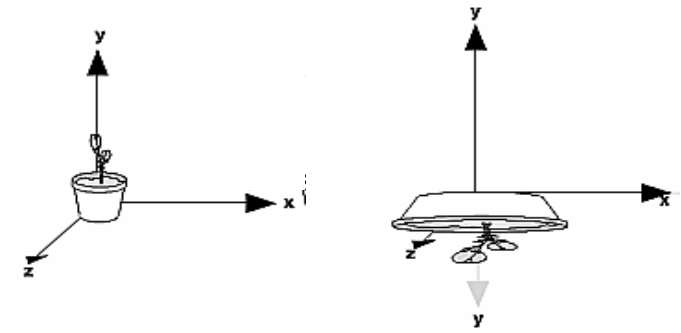
iii) Give valid homogenous coordinates for the following 3D point: A(2,3,1)

(1 mark)

Possible solution: (2,3,1,1)

d) Starting from the object model shown on Figure 1 a), find the single modelling transformation that can create the object instance shown in Figure 1. b). Give the corresponding OpenGL command with appropriate arguments. Justify your choice of transformation and arguments.

[6 marks]



Solution: this transformation is a non uniform scaling transformation that includes a mirror about X axis.

OpenGL function: glScalef(3.0, -1.0, 1.0).

Question 3

a) Briefly describe and contrast the two following modelling techniques:

[6 marks]

i) Surface modelling (polygon mesh).

(4 marks)

Solution: A polygon mesh is a collection of polygons (faces), which are together connected to form the skin of the object. All the geometry must be declared using a list of vertices and a list of faces that refers to the vertices.

ii) Iterative modelling (fractal object such as the Sierpinski gasket).

(2 marks)

Solution: The Sierpinski gasket is an example of iterative modelling where we start with a simple primitive (e.g. a tetrahedron) and iteratively subdivide the faces to obtain a more complex object.

b) Consider the display callback function code shown in Code box 3. Would you describe this code as linear or hierarchical modelling? Justify your answer by commenting on any dependency or lack of dependency between the various parts.

[6 marks]

```
void display(){
  gluLookAt(1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glPushMatrix();
  glRotatef(angle1, 0.0, 1.0, 0.0);
  part1();
  glTranslatef(0.0, PART1_HEIGHT, 0.0);
  glRotatef(angle2, 0.0, 0.0, 1.0);
  part2();
  glPopMatrix();
  glPushMatrix();
  glRotatef(angle1, 0.0, 1.0, 0.0);
  glTranslatef(0.0, PART1_HEIGHT, 0.0);
  glRotatef(angle2, 0.0, 0.0, 1.0);
  glTranslatef(0.0, PART2_HEIGHT, 0.0);
  glRotatef(angle3, 0.0, 0.0, 1.0);
  part3();}
```

Code box 3

Solution: It is hierarchical because the rotation transformation applied to part1 is propagated to part2. However, the transformations (two rotations and one translation) are not propagated to part3 because the current transformation matrix is reset by a call to the glPopMatrix function. So parts 1 and 2 are dependent, but part 3 is independent.

c) Suppose that you want to apply three modelling transformations (rotation R, scaling S and translation T) to a GLU cylinder which is aligned with the z axis and has its base in the plane z=0.

[6 marks]

i) With such a starting point, in which order should you apply the three transformations?

(2 marks)

Solution: First scaling (S), then rotation (R), then translation (T)

ii) In an OpenGL program, in which order should the transformation statements be specified?

(2 marks)

Solution: Reverse order of above.

iii) How is the current transformation matrix calculated?

(2 marks)

Solution: CTM = I x T x R x S

d) Consider the image of a cube shown in Figure 2.

[7 marks]

i) What type of projection transformation has been applied to the cube? Justify your answer.

(4 marks)

Solution: Perspective projection, we can clearly see three vanishing points where lines seem to converge.

ii) With respect to the two shading modes in OpenGL, what type of shading has been applied to the cube? Justify your answer.

(3 marks)

Solution: Flat shading as each face appears to be made of one single solid colour. By default, OpenGL does smooth shading.

Figure 2

Question 4

a) What shading technique can be used to obtain the object shown in Figure 3 while declaring only one shade of grey? How does it work in OpenGL?

[6 marks]

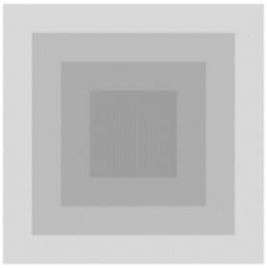


Figure 3

(3 marks)

Solution: Blending. The colour (or grey level) must be declared with a degree of transparency using the alpha channel or A value of the RGBA declaration. Blending must be enabled otherwise OpenGL will ignore all the alpha channels, and the blending function must be specified to tell OpenGL how to blend the source and destination colours.

b) In the OpenGL lighting model, there are different kinds of light.

[6 marks]

i) What kind of OpenGL light approximates best a red laser beam in a lab that produces a very bright spot where it strikes an object? Justify your answer.

(3 marks)

Solution: Specular light because bright spots are created by specular reflection, i.e. light reflected in one direction only.

ii) What OpenGL light source property can best approximate sun light? Justify your answer.

(3 marks)

Solution: Distant light, i.e. a light source that is positioned at infinity and produces light rays that are all parallel.

c) Consider the object shown in Figure 4.

[5 marks]

i) What kind of OpenGL light is being used to illuminate the object of Figure 4?

(1 mark)

Solution: Ambient light.

ii) What are the properties of this kind of light and what effect does it have on the surface of an object?

(4 marks)

Solution: Ambient light doesn't come from any direction. Objects illuminated by ambient light are evenly lit on all surfaces in all directions. Ambient light approximates scattered light in the environment.

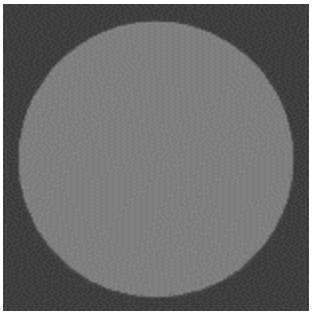


Figure 4

d) Explain the following statement by providing an example: "In OpenGL, the material colour components actually determine the percentage of incident light that is reflected".

[5 marks]

Solution: The colour of a surface is calculated by multiplying each of the light source terms by each of the material property terms. For example, if a grey light source with RGB values of (0.5, 0.5, 0.5) is used to illuminate an object with reflective properties specified in RGB terms of (0.5, 1.0, 0.5), the RGB components of the resulting colour will be: (0.5 * 0.5, 0.5 * 1.0, 0.5 * 0.5) = (0.25, 0.5, 0.25), i.e. unsaturated green.

e) In an OpenGL program, how can you make sure a light source tracks the changing position of an object that is animated?

[3 marks]

Solution: The position of a light source is declared using a statement such as:
glLightfv(GL_LIGHT0, GL_POSITION, light_pos);
This statement must appear in the program after the modelling transformation statements so they will affect both the object and the light source.