NUI Galway
OÉ Gaillimh

Semester I Examinations 2011/ 2012

Exam Code(s)	4IF, 4BP, 3BA, 1SD
Exam(s)	4th Year B.Sc. (Information Technology) 4th Year B.E. (Electronic & Computer Engineering) 3rd Year B.A. (Information Technology) Higher Diploma in Applied Science (Software Design & Development)
Module Code(s)	CT404, CT336
Module(s)	GRAPHICS AND IMAGE PROCESSING
Paper No.	I
Repeat Paper	
External Examiner(s)	Prof. M. O'Boyle
Internal Examiner(s)	Prof. G. Lyons Dr. J. Duggan * Dr. S. Redfern
Instructions:	Answer any three questions. All questions carry equal marks.
Duration	2 hours
No. of Pages	7
Department(s)	Information Technology
Course Co-ordinator(s)	

Requirements:

MCQ	Release to Library:	Yes	☐	No ☐
Handout				
Statistical/ Log Tables				
Cambridge Tables				
Graph Paper	Required			
Log Graph Paper				
Other Materials				

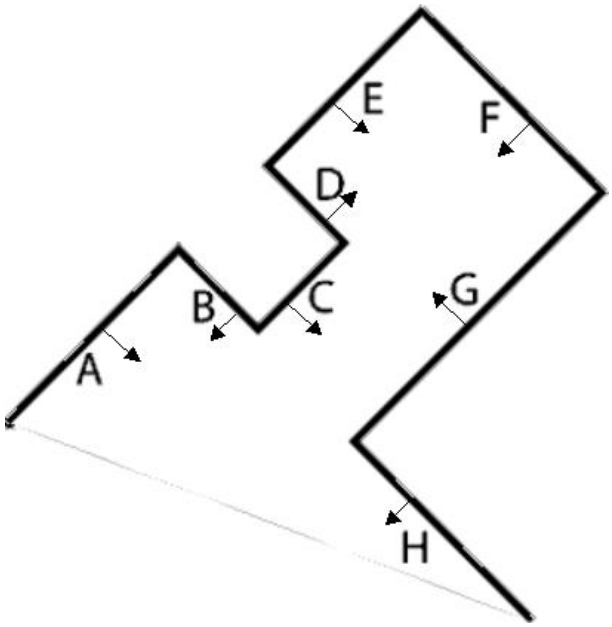
Q.1.

(a) Consider the display of a realistic forest in an interactive 3D graphics environment: the key trade-off is frame-rate versus polygon count. In this context, discuss the use of textures, visibility culling, levels of detail (LODs), mipmaps, bumpmaps and billboards in order to obtain a maximal frame-rate. Illustrate your answer with diagrams. (**10 marks**)

(b) The Binary Space Partitioning (BSP) algorithm is widely used in modern graphics programming.

(i) Indicate a situation where the BSP approach is very useful, and another situation where it is of no use (**2 marks**)

(ii) Consider the diagram below, which depicts a simple 2D scene involving 8 polygons. The polygons are labeled A through H and the arrows indicate their *surface normals*. Construct a BSP tree for this scene, and briefly explain your steps in constructing it. (**8 marks**)



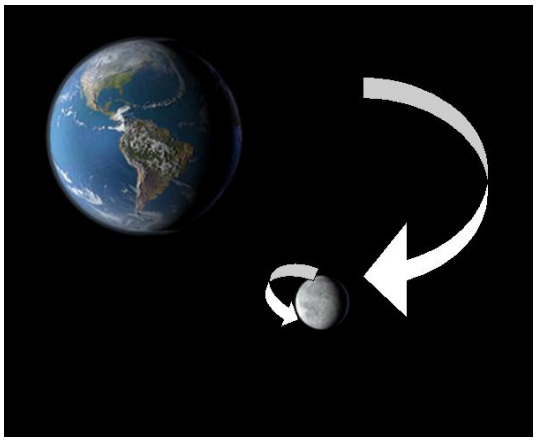
Q.2.

(a) *Antialiasing* is an approach in 2D raster graphics, which uses colour (depth) as a means to simulate an increase in resolution. With reference to the 'G' figures illustrated below, discuss the *antialisasing* technique, and in particular the concept of sub-pixel accuracy (**5 marks**).

(b)

Write X3D code to produce an animation of a moon moving around a static earth. You should assume that two jpeg files "earth.jpg" and "moon.jpg" have been provided for you to texture map onto two spheres.

The moon should rotate on its own axis as well as around the earth. Note that some useful X3D nodes are summarised on the final page of this exam paper. (**7 marks**).



(c)

Write X3D code to produce an extruded model, similar to the semi-transparent (glass) chess piece illustrated.

Note that some useful X3D nodes are summarised on the final page of this exam paper. (**8 marks**)

Q.3.

The C++ code on this page provides a skeleton 3D OpenGL/GLUT program.

(a) Using only polygons (rather than GLUT or GLU primitives), extend the display( ) function in order to display a solid cube at the centre of the world (**6 marks**)

(b) Extend the init( ) function to define a light, and a red material for the cube (**6 marks**)

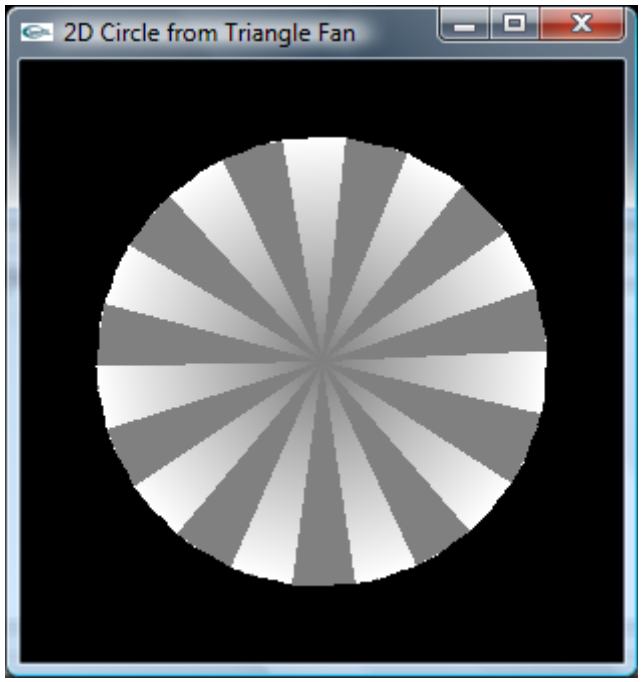
(c) Further extend the display( ) function to animate the cube so that it rotates around its own centre (**8 marks**)

Note that some useful OpenGL functions, and other standard general-purpose C functions, are listed at the end of this exam paper.

```
#include <GL/glut.h>


void init(void) {
  glClearColor (0.0, 0.0, 0.0, 0.0);
  glShadeModel (GL_SMOOTH);
  glEnable(GL_DEPTH_TEST);

   // lights and materials code needed here -------


}

void display(void) {
  // rendering code needed here -------------------

}

void reshape (int w, int h) {
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
         1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
   else
      glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
         1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("OpenGL");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

Q.4.

(a) Write the `display()` function for an *OpenGL* program which renders the following 2D circle using the `GL_TRIANGLE_FAN` primitive. (Note the use of colour to differentiate the triangles in the fan) (**8 marks**):



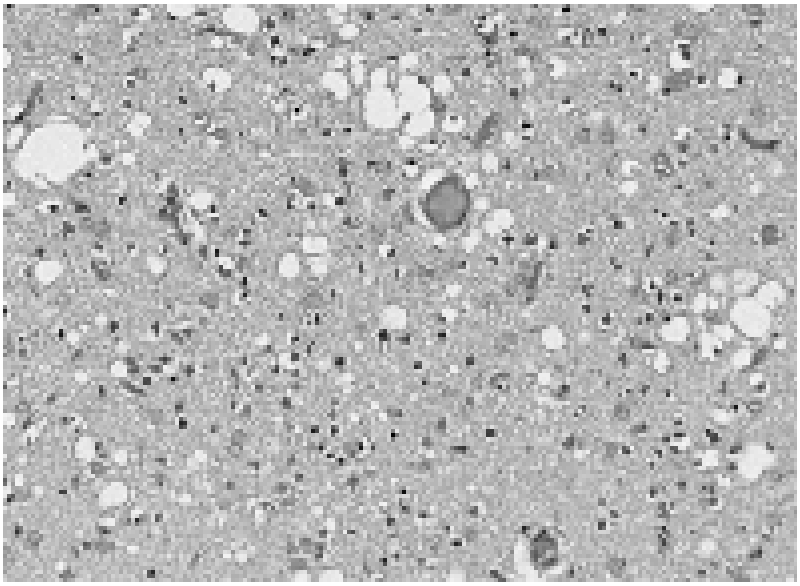
(b) Explain the *keyframe* approach to animation in computer graphics, and illustrate with code its use in *X3D*, referring to the *TimeSensor*, *Transform*, *OrientationInterpolator* and *PositionInterpolator* nodes in your answer. (**5 marks**)

(c) Why are nested co-ordinate systems useful for 3D graphics/animation programming? In your answer, explain and provide code samples illustrating the use of nested co-ordinate systems in both OpenGL and X3D. (**7 marks**).

Q.5.

(a) Describe the *mathematical morphology* approach to image processing. Outline some typical circumstances in which this approach is useful (**8 marks**).

(b) The image below is taken from tissue sample of a human brain affected by neurological damage. Of interest are white areas that are at least 5 pixels in diameter. **Outline** and **defend** a morphology-based algorithm for automatic isolation of areas matching this specification (**12 marks**).



Some useful X3D nodes:

Node	Important Fields and Nested Nodes
Shape	Nested Nodes: Appearance, Geometry Nodes (Box, Sphere, Cone, Cylinder, Text, Extrusion, etc.)
Appearance	Nested Nodes: Material, ImageTexture
Material	Fields: diffuseColor, specularColor, emissiveColor, ambientIntensity, transparency, shininess
ImageTexture	Fields: url
Transform	Fields: translation, rotation, scale, center. Nested Nodes: Other Transforms, Shapes, Sensors
TimeSensor	Fields: enabled, startTime, stopTime, cycleInterval, loop
PositionInterpolator	Fields: key, keyValue
OrientationInterpolator	Fields: key, keyValue
Extrusion	Fields: crossSection, spine, scale, orientation, beginCap, endCap, creaseAngle
Box	Fields: size
Sphere	Fields: radius
Cylinder	Fields: radius, height, side, top, bottom
Cone	Fields: height, bottomRadius, side, bottom
PointLight	Fields: on, location, radius, intensity, ambientIntensity, color, attenuation
ROUTE	Fields: fromNode, fromField, toNode, toField

Some useful OpenGL and related functions:

Function	Comments
glBegin(type), glEnd()	Together these delimit a set of vertices. 'Type' defines how to interpret them (GL_POLYGON, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS)
glVertex2f(x,y), glVertex3f(x,y,z)	Define a vertex
glColor3f(r,g,b)	Define the current colour state of OpenGL (red, green, blue)
glMatrixMode(matrix)	Apply subsequent transformations to the specified matrix (GL_MODELVIEW or GL_PROJECTION)
glLoadIdentity()	Remove any existing transforms on the current matrix
gluOrtho2D(left, right, bottom, top)	Define a 2D orthographic projection with specified world coordinate viewport
gluLookAt(eyex,eyey,eyez, atx,aty,atz, upx,upy,upz)	Position the camera (*eye* coords), target it (*at* coords) and define its 'up' vector (*up* coords)
gluPerspective(fov, aspect, near, far)	Define a 3D perspective projection with field-of-view (fov), aspect ratio, near and far clipping planes
glTranslatef(x,y,z)	Translate the coordinate system of the current matrix
glRotatef(a,x,y,z)	Rotate the coordinate system of the current matrix
glScalef(x,y,z)	Scale the coordinate system of the current matrix along each of its principal axes
glutDisplayFunc(display)	Register your display (render) function with glut
glutIdleFunc(idle)	Register your idle function with glut
(float)rand()/RAND_MAX	Produces a random float between 0.0 and 1.0