

Name	
Student Id	
Signature	

The University of New South Wales
November 2005
Final Examination

COMP3421/COMP9415
Computer Graphics

- Time allowed: three hours
- Number of Questions: 4
- Answer **All** Questions
- Total number of marks: 100
- Calculators permitted.
- You may keep this paper.
- Answer each question in a separate booklet.
- Write your answers using ink.

Question 1 (20 Marks)

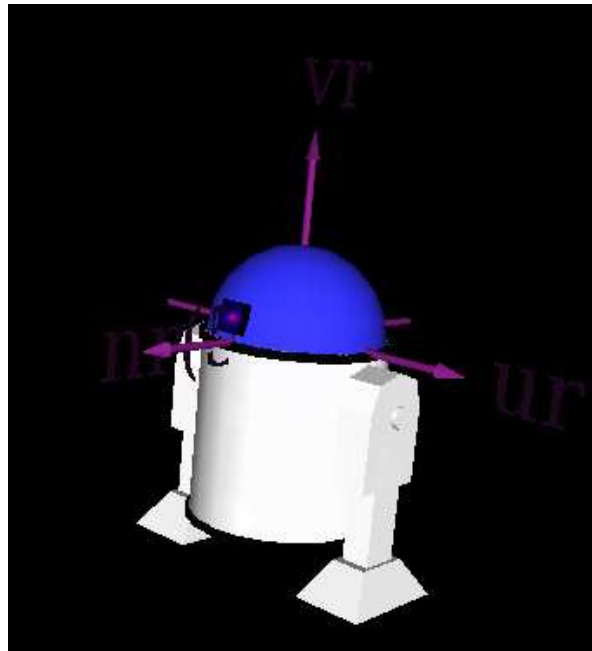
Give brief (no more than two sentences) definitions of the following

- (a) vertex shader
- (b) surface normal
- (c) trilinear filtering
- (d) bowtie polygon
- (e) device-independent colour

Question 2 (25 Marks)

- (a) (12 marks) Work out the the transformation matrix to place R2D2 into the world lying on his front. R2D2 is defined in a local coordinate system where the y axis is the centre of the cylinder that makes up his body. The z axis points in the direction he is facing and the origin is the top of the cylinder and the centre of the hemisphere that is his head. The radius of the cylinder is 2 units, so the front of his body where the cylinder meets the hemisphere (call this point F) is the point $(0,0,2)$.

Work out the local-to-world transformation that has him scaled to half size and lying face down on the ground (the XZ plane, $y=0$) so that he is lying on the Z axis with point F at $(0, 0, 5)$ is world co-ordinates.

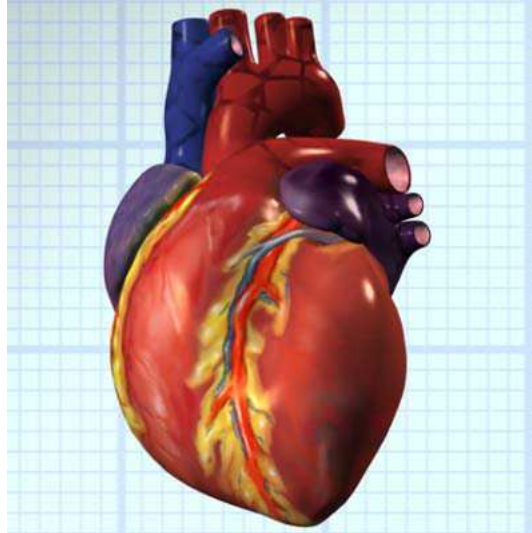


- (b) (13 marks) A triangle with corners at $A(0,0,0)$, $B(1,1,0)$ and $C(3,5,0)$ is being used to approximate a surface with normals of $(1,0,0)$ at A , $(0,1,0)$ at B , and $(0,0,1)$ at C . It has diffuse reflection coefficients (RGB) of $(1,0,0.1)$, and no specular reflection. It is illuminated by a directional light sources (no light attenuation with distance), with direction to the light of $(1,1,1)$ and colour $(1,0,0)$. The viewer (eye position) is at $(-25,-25,25)$. What colour will be used at the midpoint of AB if the triangle is
- i) flat shaded (ignoring vertex normals)
 - ii) Gourard shaded
 - iii) Phong shaded

Question 3 (25 Marks)

You have been asked to design a computer system to help train doctors for heart surgery. The idea is they can practice with the computer system instead of cutting up real hearts. Your system should enable them to cut into a virtual heart and see the results.

To give you some ideas, here is a computer model of a heart:



And here is a PHANTOM haptic device. It can report position and orientation of the stylus as well as providing force feedback so that you can feel as if you are touching a virtual object.



Tell me (using diagrams where appropriate)

- what hardware your system would require,
- what software and algorithms your system would use,
- how you would store the information needed,
- and how a user would interact with your system.

Question 4 (30 Marks)

In this question you will modify my solution to the assignments to add traffic lights to the simulation.

At the end of the paper you will find a listing of part of my solution. When you want to change one of my files you can indicate it by saying something like “Delete line 63 of DSCanvas.java and replace with the following code.”

If you want to cut and paste code from my solution, don’t copy it out again, just write something like “Insert lines 12–16 from Tree.java”.

Each of the parts of this question can be done independently of each other.

- (a) Write a `paint` method for a `TrafficLight` class. Traffic lights are defined like signs with two control points giving the left and right sides. However, they are drawn with three coloured dots, red at the left side, yellow in the centre and green at the right side. Each dot is three pixels in size. (See picture below.)



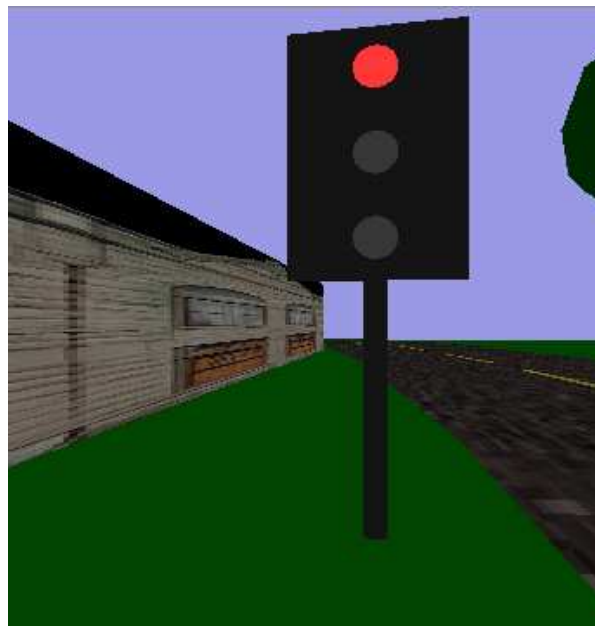
- (b) Write a `TrafficLightTool` class that lets you interactively add traffic lights with `DSEdit`.
(c) Write a `light` method for the your `TrafficLight` class with type

```
public void disc(GL gl, float[] colour);
```

It should draw one of the lights in the traffic light. Draw it as a filled circle with centre (0,0,0) and radius 0.7 in the YZ plane. (So all the x coordinates are 0). Since it glows by itself rather than reflecting from the light source you have to specify the colour like this:

```
gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, black); //  
gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, colour); //set emissive colour
```

- (d) Add a `render3d` method to your `TrafficLight` Class so that it can display the traffic light in `DSView`. The traffic light is just a sign with the red, yellow and green lights added. Of course, only one of the lights is on at any one time, so just draw the red light as red and set the colour of the other two lights to black like in the picture below. The centre of the red light is at height 11, the yellow at 9, and the green at 7.



- (e) Modify your solution so that the lights switch on and off. The red light is on for 20 seconds, then the green light is on for 16 seconds and then the yellow light for four seconds.

You can assume that `DSView` has a static method `time()` that tells you how many seconds have elapsed since the program started.

Some opengl functions you might find useful:

```
gl.glPointSize(p); // point size for GL_POINTS in pixels
GLUT glut = new GLUT();
glut.glutSolidCube(gl,1); //side length 1, centre (0,0,0)
gl.glTranslated(tx,ty,tz);
gl.glRotated(angle,ax,ay,az); //(ax,ay,az) is axis of rotation
gl.glScaled(sx,sy,sz);
float[] red = {1.0f, 0.0f, 0.0f, 1.0f}; // the colour red
gl.glColor4fv(red); //set colour to red
gl.glMaterialfv(GL.GL_FRONT,GL.GL_EMISSION,red); //set emissive colour to red
```

Light.java

```
1  /** This class represents a light */
2  import java.awt.*;
3  import java.util.*;
4  import net.java.games.jogl.*;
5
6  public class Light extends DSPolygon {
7
8      private double radius = 6; //radius of circle
9      /** name of the extra parameter
10       * @returns name of extra parameter (eg "width")
11       */
12
13     boolean isOn = true;
14
15     public boolean isOn(){
16         return isOn;
17     }
18
19     public void toggle(){
20         isOn = !isOn;
21     }
22
23     public float[] getPosition(){
24         float[] pos = new float[3];
25         Point2D p0 = pts2d.get(0);
26         pos[0] = (float) p0.x;
27         pos[2] = (float) p0.y;
28         pos[1] = (float) extra;
29         return pos;
30     }
31
32
33     public float lineWidth() {
34         return 2.0f;
35     }
36
37     /** add a control point */
38     public void addPoint(Point p) {
39         if (pts2d.size() < 1) {
40             super.addPoint(p);
41         }
42     }
43
44
45     /** paint this light into g.*/
46     public void paint(GL gl, GLU glu, GLDrawable glc){
47         if (fill != null){
48             setColor(gl,fill);
```

```

49         gl.glDisable( GL.GL_TEXTURE_2D );
50     }
51     gl.glLineWidth(lineWidth());
52     gl.glBegin( GL.GL_POLYGON); //draw the light
53
54     Point2D centre = (Point2D)(pts2d.get(0));
55
56     for (int i = 0; i< 40; i++){
57         double angle = (i/40.0)*2*Math.PI;
58         gl.glVertex2d(centre.x+radius*Math.cos(angle),
59                     centre.y+radius*Math.sin(angle));
60     }
61     gl.glEnd();
62 }
63
64 public boolean contains(int x,int y){
65     Point2D centre = (Point2D)(pts2d.get(0));
66     double dx = x - centre.x;
67     double dy = y - centre.y;
68     return (Math.sqrt(dx*dx + dy*dy) < radius);
69 }
70
71 }

```

Sign.java

```

1  /** This class represents a window */
2  import java.awt.*;
3  import java.util.*;
4  import net.java.games.jogl.*;
5  import net.java.games.jogl.util.*;
6
7  public class Sign extends Wall {
8
9      double signBot = 6;
10     public float lineWidth() {
11         return 2.0f;
12     }
13
14     public void render3D(GL gl, GLU glu, GLDrawable glc){
15         if (fill != null || texture != null){
16             if (texture == null) {
17                 setColor(gl,fill);
18                 gl.glDisable( GL.GL_TEXTURE_2D );
19             } else {
20                 setColor(gl, Color.white);
21                 gl.glEnable( GL.GL_TEXTURE_2D );
22                 gl.glTexEnvf( GL.GL_TEXTURE_ENV, GL.GL_TEXTURE_ENV_MODE,
23                             GL.GL_MODULATE );
24                 gl.glBindTexture( GL.GL_TEXTURE_2D, texture.name(gl, glu, glc) );
25                 gl.glTexParameteri( GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MAG_FILTER,
26                                     GL.GL_NEAREST);
27                 gl.glTexParameteri( GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER,
28                                     GL.GL_NEAREST);
29             }
30             gl.glPushMatrix();
31             Point2D start = pts2d.get(0);
32             Point2D end = pts2d.get(1);
33             double wallLength = Math.sqrt((start.x-end.x)*(start.x-end.x)
34                                         + (start.y-end.y)*(start.y-end.y));
35             double wallHeight = wallLength;
36             gl.glBegin(GL.GL_POLYGON);
37             gl.glNormal3d((end.y-start.y)/wallLength, 0, -(end.x-start.x)/wallLength);
38             gl.glTexCoord2d(0, 0);
39             gl.glVertex3d(start.x, signBot, start.y);
40             gl.glTexCoord2d(0, scale);
41             gl.glVertex3d(start.x, extra, start.y);
42             gl.glTexCoord2d(scale, scale);
43             gl.glVertex3d(end.x, extra, end.y);
44             gl.glTexCoord2d(scale, 0);

```

```

45         gl.glVertex3d(end.x, signBot, end.y);
46         gl.glEnd();
47
48         setColor(gl,fill);
49         gl.glDisable( GL.GL_TEXTURE_2D );
50         Point2D c = centre();
51         gl.glTranslated(c.x, signBot/2, c.y);
52         gl.glScaled(0.5,signBot,0.5);
53         GLUT glut = new GLUT();
54         glut.glutSolidCube(gl,1);
55
56
57         gl.glPopMatrix();
58     }
59 }
60 }

```

SignTool.java

```

1  import java.awt.*;
2  import java.awt.event.*;
3  import java.util.*;
4
5  /** Polygon tool for graphical editor
6  @author Tim Lambert
7  */
8
9  public class SignTool extends WallTool{
10
11
12     public SignTool(DSEdit editor) {
13         super(editor);
14     }
15
16     public String getName() {
17         return "Sign";
18     }
19
20     public String getMessage() {
21         return "Click and drag to create a road sign";
22     }
23
24
25     public Wall newWall(){
26         return new Sign();
27     }
28
29 }

```

Wall.java

```

1  /** This class represents a wall */
2  import java.awt.*;
3  import java.util.*;
4  import net.java.games.jogl.*;
5
6  public class Wall extends DSPolygon {
7
8     public float lineWidth() {
9         return 4.0f;
10    }
11
12    /** paint this curve into g.*/
13    public void paint(GL gl, GLU glu, GLDrawable glc){
14
15        if (fill != null){
16            setColor(gl,fill);
17            gl.glDisable( GL.GL_TEXTURE_2D );
18        }

```

```

19         gl.glLineWidth(lineWidth());
20         gl.glBegin( GL.GL_LINES ); //draw the wall
21         for(int i = 0; i < 2; i++) {
22             Point2D p = (Point2D)(pts2d.get(i));
23             gl.glVertex2d(p.x, p.y);
24         }
25         gl.glEnd();
26     }
27
28
29     /** add a control point */
30     public void addPoint(Point p) {
31         if (pts2d.size() < 2) {
32             super.addPoint(p);
33         }
34     }
35
36
37     /** remove specified control point */
38     public void removePoint() {
39         return;
40     }
41
42     public boolean contains(int x,int y){
43         Edge2D e = new Edge2D((Point2D)pts2d.get(0),
44                               (Point2D)pts2d.get(1));
45         Point2D pe = e.toLineSpace(new Point(x,y));
46         return (pe.getX() > 0 && pe.getX() < 1 && Math.abs(pe.getY()) < EPSILON);
47     }
48
49 }

```

Local illumination equation

$$I = I_a k_a + \sum_{k=1 \dots n} f_{\text{att}}(d_k) I_k (k_d \bar{N} \cdot \bar{L}_k + k_s (\bar{V} \cdot \bar{R}_k)^n)$$

where

- k_a = ambient reflection coefficient
- k_d = diffuse reflection coefficient
- k_s = specular reflection coefficient
- \bar{N} = surface normal
- I_a = Ambient light intensity
- I_k = Intensity of light source k
- \bar{L}_k = Direction to light source k
- \bar{V} = Direction of viewer
- $\bar{R}_k = 2\bar{N}(\bar{N} \cdot \bar{L}_k) - \bar{L}_k$
- n = Phong exponent
- d_k = distance to light source k
- $f_{\text{att}}()$ = light attenuation function

All the vectors (N, L_k, V) should be normalized.

Bezier curve

$$B(t) = \sum_{i=0 \dots 3} b_i(t) p_i$$

where

- $b_0(t) = (1-t)^3$
- $b_1(t) = 3t(1-t)^2$
- $b_2(t) = 3t^2(1-t)$
- $b_3(t) = t^3$
- p_i = Control point i

B spline curve

$$B_j(t) = \sum_{i=0 \dots 3} b_i(t) p_{i+j}$$

where

- $b_0(t) = (-t^3 + 3t^2 - 3t + 1)/6$
- $b_1(t) = (3t^3 - 6t^2 + 4)/6$
- $b_2(t) = (-3t^3 + 3t^2 + 3t + 1)/6$
- $b_3(t) = t^3/6$
- p_i = Control point i

2D Transformations

Translation (t_x, t_y)	Scaling (s_x, s_y)
$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Rotation by θ	Window-Viewport
$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \frac{v_w}{w_w} & 0 & v_x - \frac{v_w}{w_w} w_x \\ 0 & -\frac{v_h}{w_h} & v_y + v_h + \frac{v_h}{w_h} w_y \\ 0 & 0 & 1 \end{bmatrix}$

Vector operations

$$\begin{aligned}
 \hat{a} &= (a_x, a_y, a_z) \\
 \hat{a} \cdot \hat{b} &= a_x b_x + a_y b_y + a_z b_z \\
 \|\hat{a}\| &= \sqrt{a_x^2 + a_y^2 + a_z^2} \\
 \hat{a} \times \hat{b} &= (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_x)
 \end{aligned}$$

3D Transformations

$$\begin{array}{cc}
 T(t_x, t_y, t_z) & S(s_x, s_y, s_z) \\
 \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \\
 R_X(\theta) & R_Y(\theta) \\
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
 \\
 R_Z(\theta) & \text{perspective} \\
 \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 1 \end{bmatrix}
 \end{array}$$

Viewing Transformation

$$\begin{aligned}
 \bar{n} &= \overline{VPN} / \|\overline{VPN}\| \\
 \bar{u} &= \overline{Vup} \times \overline{VPN} / \|\overline{Vup} \times \overline{VPN}\| \\
 \bar{v} &= \bar{n} \times \bar{u} \\
 M_{XYZ \rightarrow UVN} &= \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} T(-o_x, -o_y, -o_z)
 \end{aligned}$$

Linear Interpolation

Parametric equation of line from \bar{a} to \bar{b} is

$$L(t) = \bar{a} + (\bar{b} - \bar{a})t, \quad 0 \leq t \leq 1$$

This is also the formula for linear interpolation.