

Name	
Student Id	
Signature	

The University of New South Wales
November 2004
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.

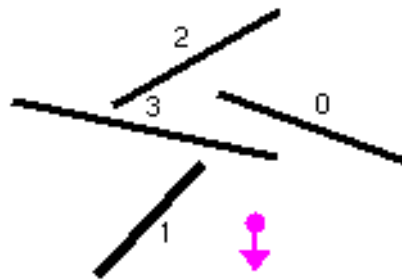
Question 1 (20 Marks)

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

- (a) BSP tree
- (b) perspective projection
- (c) clipping
- (d) homogenous co-ordinates
- (e) diffuse reflection

Question 2 (25 Marks)

- (a) (9 marks) I have a unit square with corners $(0,0)$ and $(1,1)$. It is scaled by a factor of two, rotated by 45 degrees about $(0,0)$ and then translated by $(1,0)$. It is then rendered using a window with corners $(0,0)$ and $(3,2)$ into a viewport with corners $(0,0)$ and $(150,100)$. Work out the transformation matrix that maps the square to screen space. Sketch the resulting picture, labelling the c-ordinates of the corners.
- (b) (8 marks) Insert the objects shown below into a BSP tree in the order indicated and draw the resulting tree. If you have to split any objects, show the splits and clearly label each piece on the diagram below and the tree.

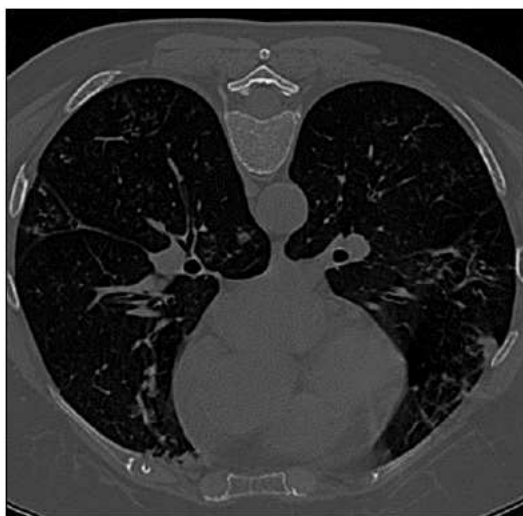


What order will the objects be drawn in if the viewpoint is as in the diagram above?

- (c) (8 marks) A B spline has control points $(0,0)$, $(1,0)$, $(1,1)$ and $(0,1)$. Calculate the co-ordinates of the midpoint of this curve.

Question 3 (25 Marks)

You are to design a computer system to display CT (Computer tomography) data to doctors. CT uses X-rays to calculate the density of body tissue for each voxel in a region of interest. The picture below shows one slice through the lungs of a person. Typically there will be dozens of slices to give a complete 3d picture of the lungs.



A image processing program takes these slices and computes the boundary of the lungs as a polygon (shown in white below).



You are to take the slice data (each slice is a 2d image) and the lung boundary data (a set of polygons for each image) and display it to doctors. Obviously you could just show them a sequence of pictures like the one above, but you need to be able to show more than one slice at a time, and to display them as a 3d model.

Tell me (using diagrams where appropriate)

- (a) what hardware your system would require,
- (b) what software and algorithms your system would use,
- (c) how you would store the information needed,
- (d) 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 a car tool that lets you add computer controlled cars to the simulation.

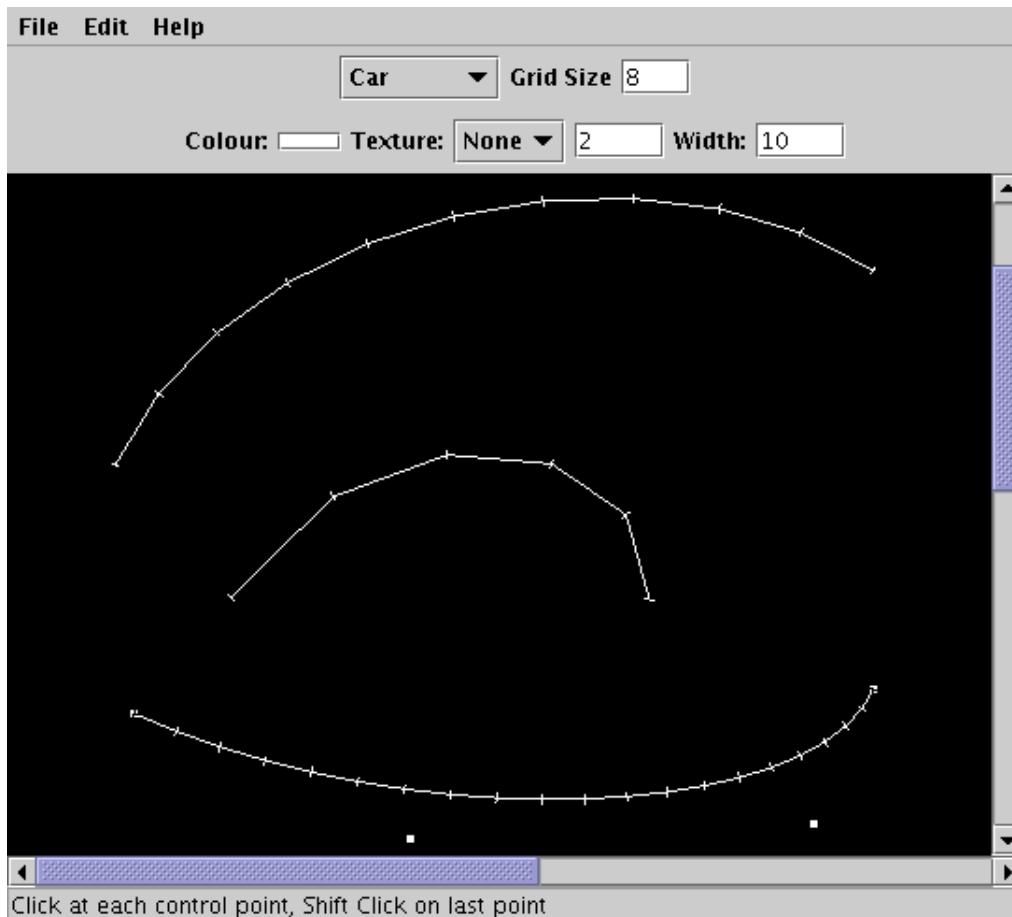
The computer controlled cars follow a path defined by a Bezier curve. The scale value of DSShape determines the speed of the car. Specifically, a scale factor of 1 means the car takes one second to go along one segment (part of curve defined by four control points) of the curve, a scale factor of 2 means that it will take two seconds and so on.

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.”

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

- (a) Write a Car class suitable for use in DSEdit. DSEdit should display the path that the car will follow and allow that path to be altered by moving the control points. The path should be drawn as a line showing the path that the car will follow with a tic mark showing where it will be after zero, 0.1, 0.2, 0.3, etc seconds.

The picture below shows three cars with scales of (from top to bottom) 1, 0.5 and 2.



The tic marks above are 5 pixels long.

Hint: I drew the path as a single `GL_LINE_STRIP`, including the tic marks. But you can draw it differently if you wish.

- (b) Write a CarTool class to allow the creation of Cars.
- (c) Add a `render3d` method to your Car Class so that it can animate a car along the path defined by the Bezier curve. The car should be rendered as a rectangular prism 20 units long, 10 units

high and 10 units wide (use a `glutSolidCube` appropriately scaled). It should be moved along the path and rotated so that the front of the car points in the direction that it is moving.

You can assume that your `render3d` method will be called ten times a second.

When the car goes off the end of the last segment of the curve it should continue again from the beginning.

Some `opengl` functions you might find useful:

```
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);
```

Road.java

```
1  import java.awt.*;
2  import java.util.*;
3  import net.java.games.jogl.*;
4
5  public class Road extends DSPolygon {
6      ArrayList pol = new ArrayList(); //Polygonal outline of the road
7      double width = 10; //road width
8      double length; //cumulated length along road
9      Point2D lastV; //last vertex on road spine
10
11     // the basis function for a Bezier spline
12     static float b(int i, float t) {
13         switch (i) {
14             case 0:
15                 return (1-t)*(1-t)*(1-t);
16             case 1:
17                 return 3*t*(1-t)*(1-t);
18             case 2:
19                 return 3*t*t*(1-t);
20             case 3:
21                 return t*t*t;
22         }
23         return 0; //we only get here if an invalid i is specified
24     }
25
26     // derivative of the basis function for a Bezier spline
27     static float bdash(int i, float t) {
28         switch (i) {
29             case 0:
30                 return (-3*t + 6)*t - 3;
31             case 1:
32                 return (9*t - 12)*t + 3;
33             case 2:
34                 return (-9*t +6)*t;
35             case 3:
36                 return 3*t*t;
37         }
38         return 0; //we only get here if an invalid i is specified
39     }
40
41     //evaluate a point on the Bezier
42     Point2D p(int i, float t) {
43         float px=0;
44         float py=0;
45         for (int j = 0; j<=3; j++){
46             Point2D p = (Point2D)(pts2d.get(i+j));
47             px += b(j,t)*p.x;
48             py += b(j,t)*p.y;
49         }
50         return new Point2D(px,py);
51     }
52 }
```

```

53 //evaluate a derivative on the Bezier
54 Point2D pdash(int i, float t) {
55     float px=0;
56     float py=0;
57     for (int j = 0; j<=3; j++){
58         Point2D p = (Point2D)(pts2d.get(i+j));
59         px += bdash(j,t)*p.x;
60         py += bdash(j,t)*p.y;
61     }
62     return new Point2D(px,py);
63 }
64
65 final int STEPS = 12;
66
67 private void addVertex(GL gl, int i, float j){
68     Point2D q = p(i,j);
69     Point2D qdash = pdash(i,j);
70     double len = Math.sqrt(qdash.x*qdash.x + qdash.y*qdash.y);
71     if (lastV != null){
72         double dx = q.x - lastV.x;
73         double dy = q.y - lastV.y;
74         length += Math.sqrt(dx*dx + dy*dy);
75     }
76
77     if (len == 0) {
78         len = 1;
79     }
80     qdash.x = qdash.x*(width/2)/len;
81     qdash.y = qdash.y*(width/2)/len;
82     Point2D p1 = new Point2D(q.x-qdash.y, q.y+qdash.x);
83     Point2D p2 = new Point2D(q.x+qdash.y, q.y-qdash.x);
84     pol.add(p1);
85     pol.add(0,p2);
86
87
88     gl.glTexCoord2d(length*scale/100, 0);
89     gl.glVertex2d(p1.x, p1.y);
90     gl.glTexCoord2d(length*scale/100, 1);
91     gl.glVertex2d(p2.x, p2.y);
92
93     lastV = q;
94 }
95
96
97 /** paint this curve into g.*/
98 public void paint(GL gl, GLU glu, GLDrawable glc){
99
100     if (fill != null || texture != null){
101         if (texture == null) {
102             setColor(gl,fill);
103             gl.glDisable( GL.GL_TEXTURE_2D );
104         } else {
105             gl.glEnable( GL.GL_TEXTURE_2D );
106             gl.glTexEnvf( GL.GL_TEXTURE_ENV, GL.GL_TEXTURE_ENV_MODE,
107                 GL.GL_REPLACE );
108             gl.glBindTexture( GL.GL_TEXTURE_2D, texture.name(gl, glu, glc) );
109         }
110         if(pts2d.size() >= 4){
111             length = 0.0;
112             lastV = null;
113             pol.clear();
114             gl.glBegin( GL.GL_TRIANGLE_STRIP ); //draw the Bezier
115             addVertex(gl,0,0);
116             for (int i = 0; i < pts2d.size()-3; i+=3) {
117                 for (int j = 1; j <= STEPS; j++) {
118                     addVertex(gl,i,j/(float)STEPS);
119                 }
120             }
121             gl.glEnd();
122         }
123     }

```

```

124
125     }
126 }
127
128 /** get the width
129     @returns the width
130 */
131
132 public double getRoadWidth(){
133     return width;
134 }
135 /** Set the road width
136     @param width of road
137 */
138
139 public void setWidth(double width){
140     this.width = width;
141 }
142
143 public String toString() {
144     StringBuffer result = new StringBuffer(super.toString());
145     result.append(" " + width);
146     return result.toString();
147 }
148
149 // Parse the coordinates
150 public void fromTokens(StringTokenizer st) throws MalformedURLException {
151     super.fromTokens(st);
152     try {
153         width = Double.parseDouble(st.nextToken());
154     }
155     catch (NoSuchElementException e) {
156         throw new MalformedURLException(e.getMessage());
157     } catch (NumberFormatException e) {
158         throw new MalformedURLException(e.getMessage());
159     }
160 }
161
162 public boolean contains(int x,int y){
163     return toPolygon(pol).contains(x,y);
164 }
165
166 // 3D Version of add vertex. Does not have to handle any of the interactive
167 // aspects of the problem.
168 private void addVertex3D(GL gl, int i, float j){
169     Point2D q = p(i,j);
170     Point2D qdash = pdash(i,j);
171     double len = Math.sqrt(qdash.x*qdash.x + qdash.y*qdash.y);
172     if (lastV != null){
173         double dx = q.x - lastV.x;
174         double dy = q.y - lastV.y;
175         length += Math.sqrt(dx*dx + dy*dy);
176     }
177
178     if (len == 0) {
179         len = 1;
180     }
181     qdash.x = qdash.x*(width/2)/len;
182     qdash.y = qdash.y*(width/2)/len;
183     Point2D p1 = new Point2D(q.x-qdash.y, q.y+qdash.x);
184     Point2D p2 = new Point2D(q.x+qdash.y, q.y-qdash.x);
185     gl.glTexCoord2d(length*scale/100, 0);
186     gl.glVertex3d(p1.x, 1, p1.y);
187     gl.glTexCoord2d(length*scale/100, 1);
188     gl.glVertex3d(p2.x, 1, p2.y);
189     lastV = q;
190 }
191
192
193 public void render3D(GL gl, GLU glu, GLDrawable glc){
194

```

```

195     if (fill != null || texture != null){
196         if (texture == null) {
197             setColor(gl,fill);
198             gl.glDisable( GL.GL_TEXTURE_2D );
199         } else {
200             gl.glEnable( GL.GL_TEXTURE_2D );
201             gl.glTexEnvf( GL.GL_TEXTURE_ENV, GL.GL_TEXTURE_ENV_MODE,
202                 GL.GL_MODULATE);
203             gl.glBindTexture( GL.GL_TEXTURE_2D, texture.name(gl, glu, glc) );
204             gl.glTexParameteri( GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MAG_FILTER,
205                 GL.GL_NEAREST);
206             gl.glTexParameteri( GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER,
207                 GL.GL_NEAREST);
208         }
209         if(pts2d.size() >= 4){
210             length = 0.0;
211             lastV = null;
212             gl.glBegin( GL.GL_TRIANGLE_STRIP ); //draw the Bezier
213             gl.glNormal3d(0,-1,0);
214             addVertex3D(gl,0,0);
215             for (int i = 0; i < pts2d.size()-3; i+=3) {
216                 for (int j = 1; j <= STEPS; j++) {
217                     addVertex3D(gl,i,j/(float)STEPS);
218                 }
219             }
220             gl.glEnd();
221         }
222     }
223 }
224 }
225 }
226 }
227 }

```

RoadTool.java

```

1  import java.awt.*;
2  import java.awt.event.*;
3  import java.util.*;
4
5  /** Road tool for graphical editor
6   @author Tim Lambert
7   */
8
9  public class RoadTool extends PolygonTool{
10
11
12     public RoadTool(DSEdit editor) {
13         super(editor);
14     }
15
16     public String getName() {
17         return "Road";
18     }
19
20     public String getMessage() {
21         return "Click at each control point, Shift Click on last point";
22     }
23
24     // create the Road
25     public void mouseClicked(MouseEvent e) {
26         if (p==null){
27             p = new Road();
28             editor.getProperties(p);
29             canvas.getShapes().add(p);
30             canvas.select(p);
31             p.addPoint(gridPoint(e));
32         }
33         if (e.isShiftDown()){
34             p = null;

```

```

35     } else {
36         p.addPoint(gridPoint(e));
37     }
38     canvas.repaint();
39 }
40
41
42 }

```

Tree.java

```

1  /** This class represents a tree */
2  import java.awt.*;
3  import java.util.*;
4  import net.java.games.jogl.*;
5
6  public class Tree extends Wall {
7
8      public float lineWidth() {
9          return 2.0f;
10     }
11
12     /** paint this tree 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_LINE_LOOP); //draw the tree
21
22         Point2D centre = (Point2D)(pts2d.get(0));
23         Point2D rad = (Point2D)(pts2d.get(1));
24         double dx = rad.x - centre.x;
25         double dy = rad.y - centre.y;
26         double radius = Math.sqrt(dx*dx + dy*dy);
27         double theta = Math.atan2(dy,dx);
28
29         for (int i = 0; i < 40; i++){
30             double angle = theta + (i/40.0)*2*Math.PI;
31             gl.glVertex2d(centre.x+radius*Math.cos(angle),
32                         centre.y+radius*Math.sin(angle));
33         }
34         gl.glEnd();
35     }
36
37     public boolean contains(int x,int y){
38         Point2D centre = (Point2D)(pts2d.get(0));
39         Point2D rad = (Point2D)(pts2d.get(1));
40         double dx = rad.x - centre.x;
41         double dy = rad.y - centre.y;
42         double radius = Math.sqrt(dx*dx + dy*dy);
43         dx = x - centre.x;
44         dy = y - centre.y;
45         return (Math.sqrt(dx*dx + dy*dy) < radius);
46     }
47
48 }

```

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

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

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.