CompSci.372.FC 2002
Computer Graphics

Mid Term Test 30th April 2002, 6.30 pm – 7.30 pm

Surname (Family Name):	
First Name(s):	
ID Number:	
Login Name (UPI):	

Instructions:

1. Attempt **ALL** questions.
2. The test is for one (1) hour.
3. This is a closed book test.
4. Calculators are **NOT** permitted.
5. Write your answers in the spaces provided. There is space at the back for answers that overflow the allotted space.
6. **Questions total 50 Marks**.
7. This test is worth 10% of your final marks for CompSci372FC

Section	Marks	Maximum Marks
Q.1		15
Q.2		6
Q.3		6
Q.4		11
Q.5		12
Total		50

Question 1 – Short answer test [15 marks]

Complete each of the following statements by filling in the underlined blank spaces. Each blank space is worth 1 mark.

[15 marks]

(a) The name of the 3D graphics API used in this course is **OpenGL**. The API comes together with a utility toolkit named **GLU**, which amongst others provides a function to define the viewpoint and the view direction. We use an additional toolkit named **GLUT**, which contains high-level functions for window creation, event handling and the drawing of solid objects.

(b) If $\mathbf{v} = (3,-1,0)$ and $\mathbf{u} = (1,-1,4)$ then the dot product is $\mathbf{v}.\mathbf{u} = \mathbf{4}$.

(c) Let α be the angle between two normalised vectors **u** and **v**.

Then $cos\ \alpha =$ **u.v** and $sin\ \alpha =$ **|u×v|**.

(d) Given a parametric surface $\mathbf{p}(s,t)$ with $0 \le s,t \le 1$ the normal **n** at a point can be calculated by $\mathbf{n}(s,t)=\frac{\partial \mathbf{p}}{\partial s}\times\frac{\partial \mathbf{p}}{\partial t}$.

(e) Given three different points A, B and C of a polygon, its normal **n** can be calculated by **n=(A-B)×(C-B)**. This calculation will fail to yield a useful result if **all three points lie on a line (or if two of the points are almost the same).**

(f) In order to map an image onto a polygon mesh we have to define a **texture coordinate** for each vertex.

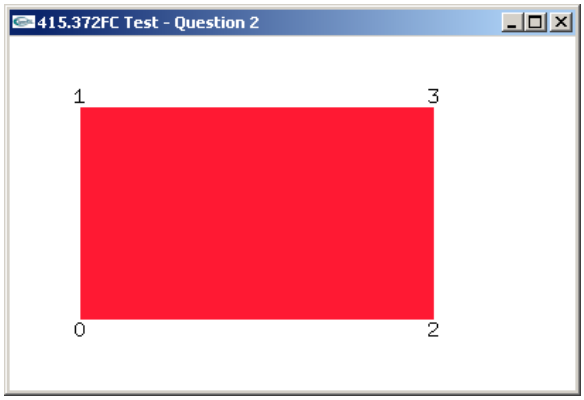
(g) Texture mapping doesn't look right for uneven surfaces like orange skin. The illusion of an uneven surface is obtained by using **bump mapping**, which modulates the **surface normal** before applying illumination calculations.

(h) In order to display a 3D scene the scene components must be projected onto a view plane. A parallel orthographic projection is a projection where the projectors through all points are parallel and **perpendicular (=orthogonal)** to the view plane. For a **perspective** projection all projectors emanate from a single eye point (centre of projection).

(i) We can zoom into a scene by reducing the value of the parameter **fov** in the command gluPerspective(float fov, float aspect, float near, float far).

Question 2 – OpenGL [6 marks]

In this question you have to draw a 2D rectangle as shown on the right. The vertex with the index zero has the coordinates (50, 50) and the vertex with the index 3 has the coordinates (300, 200).



A. Complete the code fragment below so that it defines a global array containing the four 2D vertices of the above rectangle [2 marks]:

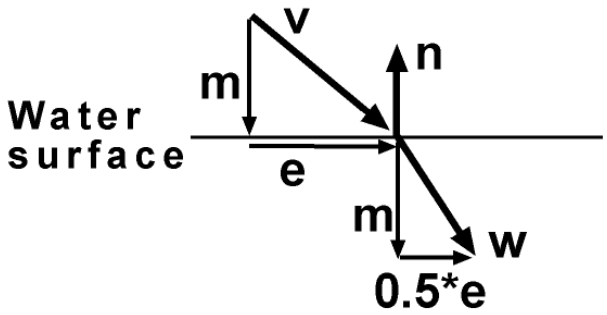
```
const int numVertices=4;
const float v[numVertices][2] = {{50,50}, {50,200}, {300,50}, {300, 200}};
```

B. Complete the display function below so that it draws the convex polygon defined by these four vertices using the **GL_TRIANGLES** mode and the **glVertex2fv** command. The vertex numbers in the resulting image (shown above) have been inserted afterwards for clarity and you don't have to draw them. [4 marks]

```
void display(void)
{
    // clear all pixels in frame buffer
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 0.1, 0.2);    // reddish colour
    glBegin(GL_TRIANGLES);
    glVertex2fv(v[0]);
    glVertex2fv(v[2]);
    glVertex2fv(v[1]);
    glVertex2fv(v[1]);
    glVertex2fv(v[2]);
    glVertex2fv(v[3]);
    // note that there are many other orders in which the vertices can be listed
    glEnd();
    glFlush();
}
```

Question 3 – 3D Geometry [6 marks]

A. A light vector **v** passing from air into water is refracted on the water surface. Given v and the **unit** normal **n** of the water surface, derive a formula for the resulting vector **w**. Assume that the projections of **v** and **w** onto **n** have the same length and that the projection of **w** onto the water surface has half the length of the projection of **v** onto the surface.
Your formula may only contain vector operations and the dot product (scalar product). You are allowed to draw on the illustration below in order to clarify the derivation of your formula [6 marks].

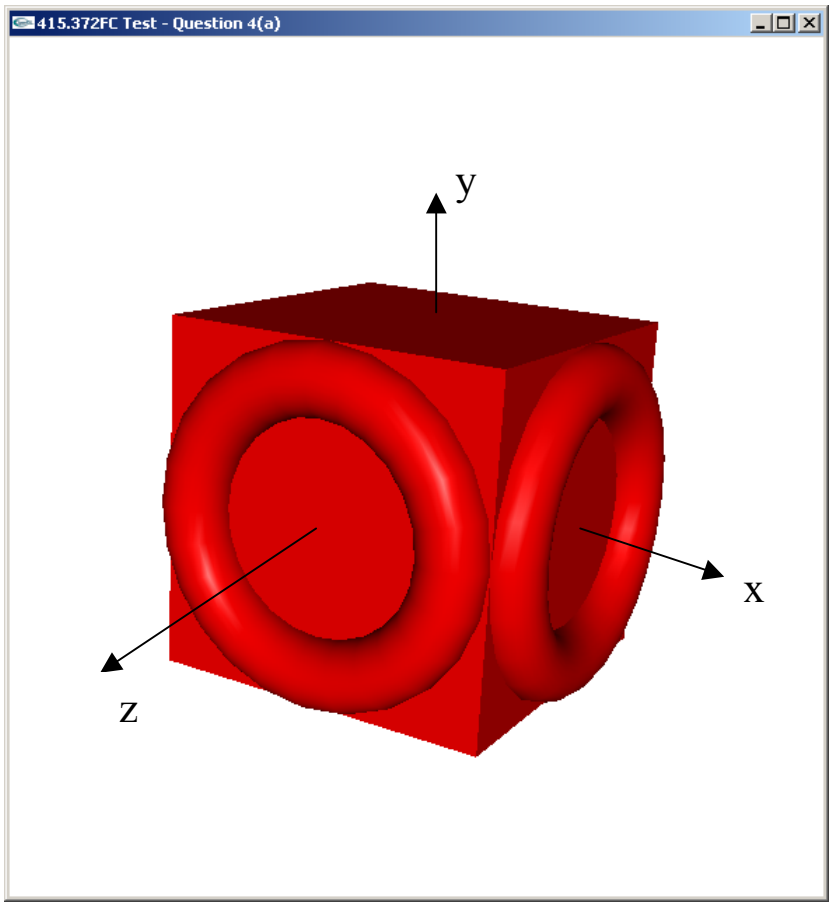


m=(**v**.**n**)**n**
e=**v**-**m**
w=**m**+0.5***e**=0.5*(**v**+**m**)=0.5*(**v**+(**v**.**n**)**n**)

// Note: Except of the last line this calculation is the same as for the reflection vector
// (see chapter 4, slide 12 of the handouts)

Question 4 – Transformations [11 marks]

A. Complete the display method on the next page so that it draws the scene shown in the image below. The axes and their labels have been added for clarity and you don't have to draw them.



The scene consists of a cube centred at the origin with a side length of 2 units. Inserted into the front face and right hand side face of the cube is a torus with an inner radius of 0.2 units and an outer radius of 0.8 units such that exactly half of each torus lies outside the cube.

You may use the following functions:

```
void myCube();
```

draws a cube centred at the origin with a side length of `2`.

```
void myTorus();
```

draws a torus centred at the origin with an inner radius of `0.2`, an outer radius of `0.8` and its axis aligned with the z-axis [7 marks].

```
void display(void)
{
    glMatrixMode( GL_MODELVIEW );  // Set the view matrix ...
    glLoadIdentity();              // ... to identity.
    gluLookAt(0,0,8, 0,0,0, 0,1,0);  // camera is on the z-axis
    trackball.tbMatrix();          // rotate the object using the trackball ...

    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // set material properties
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_ambient_and_diffuse);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    // Draw the cube and the tori
```

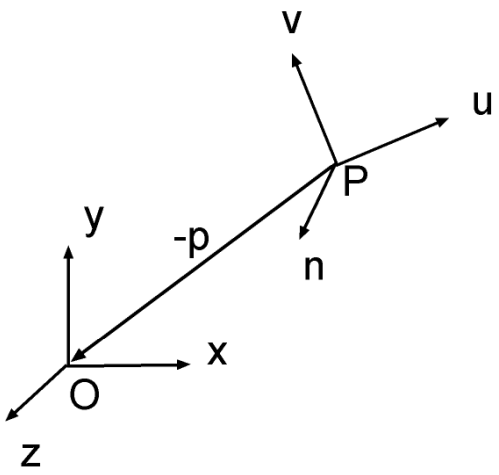
```
    myCube();
    glPushMatrix();
    glTranslatef(0,0,1);
    myTorus();
    glPopMatrix();
    glTranslatef(1,0,0);
    glRotatef(90,0,1,0);
    myTorus();
    // it's good style to put the code fragment above between glPushMatrix() and glPopMatrix()
    // so that no subsequent drawing commands are affected by the transformations.
    // Not necessary in this question since there are no subsequent drawing commands.
```

NOTE: If you don't use glPushMatrix() as done above then the translation for the second torus has to take into account the translation for the first torus, ie.

```
    myCube();
    glTranslatef(0,0,1);
    myTorus();
    glTranslatef(1,0,-1);
    glRotatef(90,0,1,0);
    myTorus();
```

```
    glFlush ();
    glutSwapBuffers();
}
```

B. Given is a world coordinate system with the origin O=(0,0,0) and a second coordinate system with the origin P=(p_x, p_y, p_z) in world coordinates. The basis vectors **u**, **v** and **n** of the second coordinate system are normalised and orthogonal to each other. What are the *uvn*-coordinates of the world coordinate system origin O? [4 marks]



The vector from the origin of the *uvn*-coordinate system to the point O is –**p**. The *uvn*-coordinates of O are just the lengths of the projections of this vector onto the **u**, **v** and **n** coordinate axis, i.e. the *uvn*-coordinates are (-**p**.**u**, -**p**.**v**, -**p**.**n**).

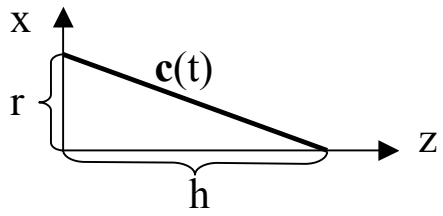
Note 1: This is just a very simple case of the view transformation where P is the eye point (please read chapter 8, slide 7 of the handouts)

Note 2: You will get the same result if you multiply the point O with the view transformation matrix.

Question 5 – Modelling [12 marks]

In this question you will develop a function for drawing a cone whose centre axis is the z-axis and which has a height of `h` and a circular base with radius `r` centred at the origin.

A. What is the parametric equation of the profile curve $\mathbf{c}(t)=(x(t),0,z(t))$ of this cone? Note that the profile curve is a simple line segment as shown below. Define the line segment $\mathbf{c}(t)$ such that the parameter t lies within the interval [0,1] [2 mark].



$\mathbf{c}(t)$ is a line segment from the point $(r,0,0)$ to the point $(0,0,h)$, ie.

$$\mathbf{c}(t)=\begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix}+t\left(\begin{pmatrix} 0 \\ 0 \\ h \end{pmatrix}-\begin{pmatrix} r \\ 0 \\ 0 \end{pmatrix}\right)=\begin{pmatrix} r(1-t) \\ 0 \\ th \end{pmatrix}$$

B. What is the parametric equation of the surface of revolution $\mathbf{p}(s,t)=(x(s,t),y(s,t),z(s,t))$ formed by rotating the line segment $\mathbf{c}(t)$ around the z-axis. Define the equation such that the parameter t lies within the interval [0,1] and the parameter s lies within the interval $[0,2\pi]$ [2 mark].

$$\mathbf{p}(s,t)=\begin{pmatrix} r(1-t)\cos s \\ r(1-t)\sin s \\ th \end{pmatrix}$$

C. Implement the function

```
mySolidCone(float r, float h, int nStacks, int nSlices)
```

which draws a cone as described in the previous parts of this question. The surface of revolution $\mathbf{p}(s,t)$ representing the cone should be subdivided into `nSlices` segments in circumferential direction and into `nStacks` segments along its axis.
Given is a function

```
CVec3df point(float s, float t, float r, float h)
```

which returns the point $\mathbf{p}(s,t)$ on the surface of the cone and a function

```
CVec3df normal(float s, float t, float r, float h)
```

which returns the unit normal $\mathbf{n}(s,t)$ at that point.

Please complete the following code. The class `CVec3df` and the constant `Pi` are included by `Geometry.h` and are the same as in the lecture and the assignment [8 marks].

```
#include <math.h>
#include "Geometry.h"

CVec3df point(float s, float t, float r, float h){
    // returns the point p(s,t) of cone with radius r and height h
}

CVec3df normal(float s, float t, float r, float h){
    // returns the unit normal n(s,t) of a cone with radius r and height h
}

void mySolidCone(float r, float h, int nSlices, int nStacks)
{
    float s,t;
```

```
    CVec3df n,p;
    for (int j=0; j< nStacks; j++)
    {
        glBegin(GL_QUAD_STRIP);
        for (int i=0; i<= nSlices; i++)
        {
            t=(float) j/(float) nStacks;
            s=2.0f*Pi*(float) i/(float) nSlices;
            n=normal(s,t,r,h);
            glNormal3f(n[0], n[1], n[2]);
            p=point(s,t,r,h);
            glVertex3f(p[0], p[1], p[2]);
            t=(float) (j+1)/(float) nStacks;
            p=point(s,t,r,h);
            glVertex3f(p[0], p[1], p[2]);
        }
        glEnd();
    }
```

```
}
```

This page left blank for formatting purposes, and any questions that overflow.