NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2, 2011/2012

CS3241 - COMPUTER GRAPHICS

May 2011 Time Allowed: 2 Hours

INSTRUCTION TO CANDIDATES

1. This is an OPEN book examination.
2. This examination paper contains **SEVEN (7)** questions and comprises **EIGHT (8)** printed pages.
3. Answer *ALL* questions within the spaces provided in this booklet.
4. You are allowed to use the back of the paper but please remember to state "P.T.O."
5. *Cross out any draft* or otherwise we will mark the poorer answers.
6. No calculator should be needed in this exam. You can leave your answer in surd form, namely, you can write $\sqrt{2}$ instead of 1.4142...
7. **In the programming questions, you can assume some basic normal vector arithmetic functions are provided for the 2D or 3D vector class `Vector2D (Vector3D)`, even a function `normalize(Vector2D)` to compute a normalized vector.**
8. Please write your matriculation number below, but NOT your name.

TIDINESS COUNTS!

We will deduct marks if your writing is too messy.

MATRICULATION NUMBER:_____

(this portion is for examiner's use only)

Question	Max. Marks	Score	Check
Q1	10		
Q2	8		
Q3	6		
Q4	6		
Q5	10		
Q6	4		
Q7	6		
Total	50		

Question 1 [10 marks]

You are given 10 statements below. State if each statement is True (T) or False (F) by writing your answer in the box provided. A correct answer will give you 1 mark. An empty answer will give you zero, but ***a wrong answer will result in -1 mark***. (The lowest mark for this question is ZERO)

☐ If n is the number of objects in a scene, the worst time complexity for ray tracing is $O(n^3)$ for a certain fixed screen size.

☐ There is a maximum resolution of human eyes. If we increase the resolution of a display device to be double or even triple of that resolution, we can solve any aliasing problem.

☐ In 3D movies, we can see the characters "pop out" from the screen is because each of our eyes receives a different image.

☐ We cannot apply Phong shading in ray tracing.

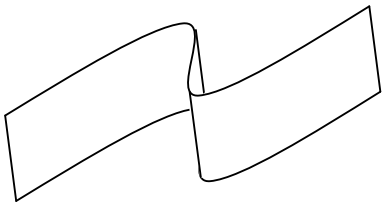
☐ Red + blue = magenta in the subtractive color model.

☐ If we want to model a burger with sesames on top of it, it is always the best idea to model each piece of sesame as a small separate object with different translation and rotation.

☐ Dithering can be applied to convert a picture to black and white only.

☐ To compute the intensity of a surface point in Gouraud shading, we need to map the point in the screen coordinates back to the world coordinates.

☐ We can model the ribbon-like surface on the right by a Bezier patch, in which one side of it is a straight line.



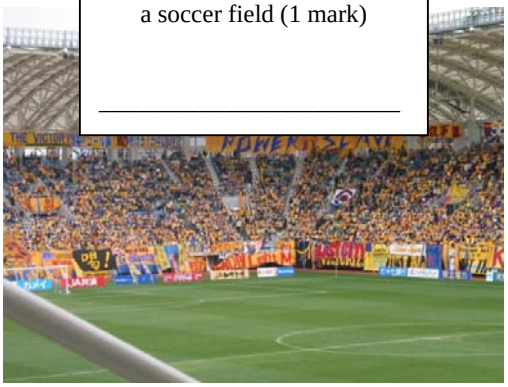
☐ The dimension of a fractal curve is always more than 1, if we do not allow a single straight line to be a generator.

Question 2 [8 marks]

Describe the best techniques taught in our course to draw the following scenes in *real time rendering*. For each picture, give ONE most important technique. (Except for the "hair" and "leather", please give two techniques.)

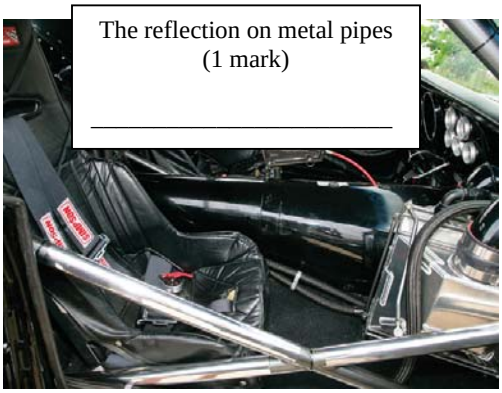
The thousands of audience on a soccer field (1 mark)

_____



The reflection on metal pipes (1 mark)

_____



Straight hairs (2 marks)

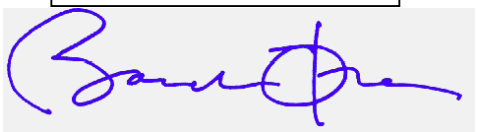
_____

Leather with wrinkles (2 marks)

_____

Signiture (1 mark)

_____



Flags flying in the wind (1 mark)

_____

Question 3 [6 marks]

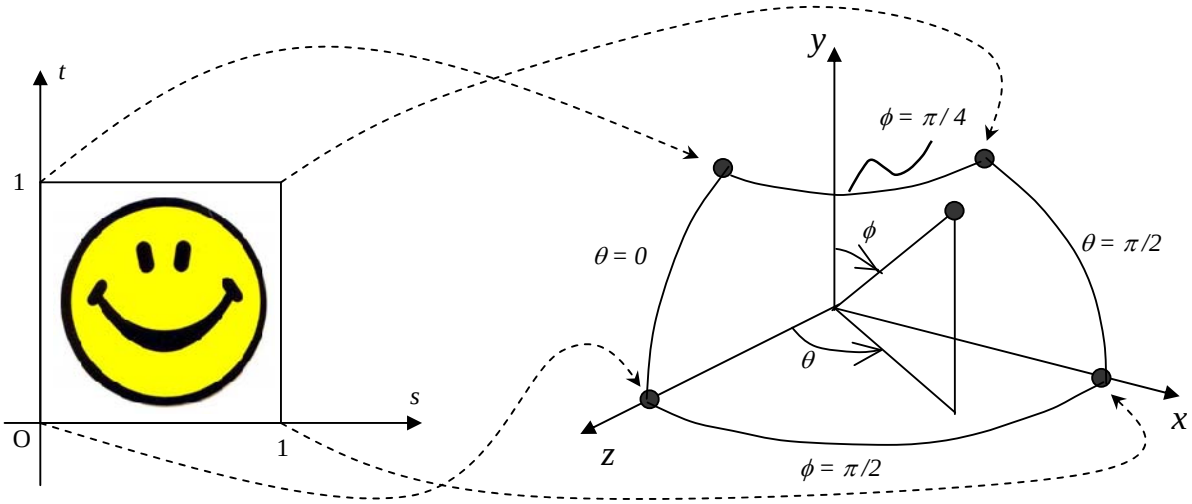
We can specify a sphere by

$$p(\theta,\phi) = (\sin\theta \sin\phi, \cos\phi, \cos\theta \sin\phi).$$

However, we only decide to draw a portion of the sphere by limiting θ and ϕ to be

$$\pi/2 \geq \phi \geq \pi/4, \qquad \pi/2 \geq \theta \geq 0.$$

The sphere patch is subdivided by *n* times *n* small rectangles. The texture on the left is going to be mapped onto this patch by the four corners shown below. Fill in the code to specify the texture coordinates according to the following **orientation**. You can assume the texture is already loaded, set up and ready to use.




```
void glDrawSphereVertex(double theta, double phi)
{
  double x,y,z
  x = sin(theta)*sin(phi);  y = cos(phi); z = cos(theta)*sin(phi);
  glNormal3f(x,y,z);   glVertex3f(x,y,z);
}
void drawSpherePatch(int n) // n = number of subdivisions
{
  glEnable(GL_TEXTURE_2D);
  for (i=0;i<n;i++)
    for(j=0;j<n;j++)
    {


      glBegin(GL_POLYGON);


        glDrawSphereVertex( iπ/2n, π/4 + jπ/4n );


        glDrawSphereVertex( (i+1)π/2n, π/4 + jπ/4n );


        glDrawSphereVertex( (i+1)π/2n, π/4 + (j+1)π/4n );


        glDrawSphereVertex( iπ/2n, π/4 + (j+1)π/4n );



      glEnd();
    }
}
```

Question 4 [6 marks]

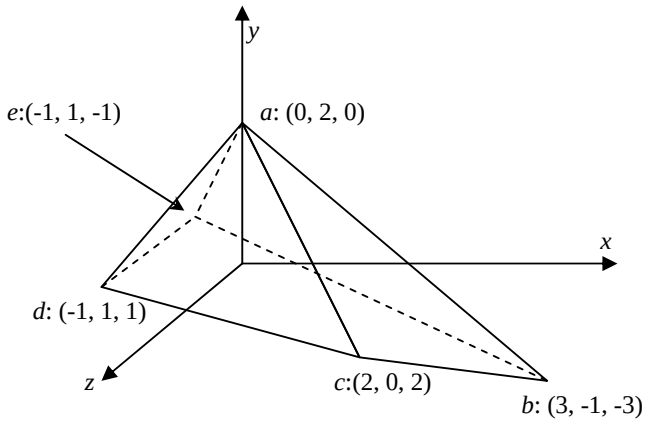
a. The following table is an example of such data with sizes 5 x 5. Compute its sum table onto the right empty table according to the lecture note. (2 marks)

10	0	0	0	10
0	10	0	10	0
0	0	10	0	0
0	10	0	10	0
10	0	0	0	10

b. Provided that a gray scale *N* by *M* texture data is given as an array double `myTexture[N][M]`. Write an algorithm to compute the sum table and store them into the array double `mySumTable[N][M]` in O(*NM*) time. (Hint: You can assume that `myTexture[i][j]` and `mySumTable[i][j]` return zeros if any `i` or `j` is less than zero.) (4 marks)

Question 5 [10 marks]

Here are 4 triangles (*abc*, *acd*, *ade* and *aeb*) on a mesh:



a) You are given that the four normal vectors of the four triangles are (12,12,0), (0,4,4), (-2,2,0) and (0,6,-6) for triangles *abc*, *acd*, *ade* and *aeb* respectively. Calculate the averaged normal at the point *a* (2 marks)

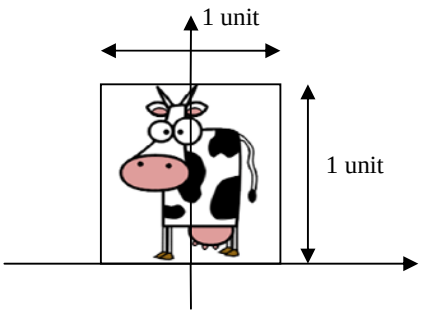
b) Let the projective matrix be $M = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and it takes the vertices from the world coordinates to the screen coordinates. Calculate the screen coordinate of the vertices of triangle *adc*. (1 marks)

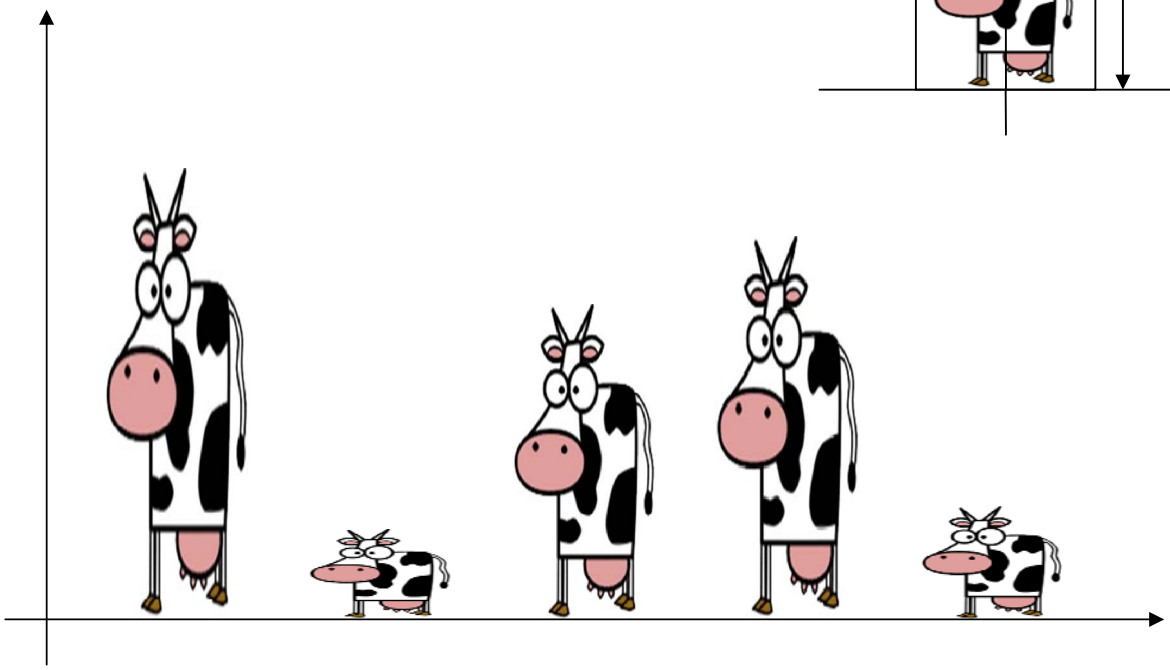
c) Let $y = 3$ be the current scanline when *acd* is being drawn. Compute the end points of the intersections of triangle *acd* on the screen coordinates. Calculate the normal vectors on these two endpoints by Phong shading method, assuming the vertex normal vector at d and c are (-1,0,0) and (1,0,0) after the averaging the normals of neighboring polygons (5 marks)

d) Let the current pixel on this scanline is at (0,3), compute its normal vector by Phong shading method. (You may leave this final normal vector WITHOUT normalized to unit length) (2 marks)

Question 6 [4 marks]

Given a function **drawACow()** that draws a cow standing at the origin with height and width of 1 units as the right side. Write a program to display a chart to show the population of cows in a farm like the following diagram.





The width of each cow in the chart is 8 units. And the height of the cow is 8 units times the number of cows that year, which is given by a function **cowPopulation(int year)**. The "origin" of the graph is at position (20,20) on the screen coordinates. And each cow is 10 units rightwards from the origin. Write a function to display the chart for 10 years starting from year 2001. Note that, after exiting your function, you should keep the model view matrix stack the same as the time when your function starts. (You don't need to draw the two axis arrows.)

Question 7 [6 marks]

a. Write pseudo code to compute a point p(t) on a quartic (degree 4) Bezier curve, with control points p[i] (i=0 to 4) in the two dimensional space. (2 marks)
b. Write pseudo code to compute the velocity p_dash(t) at *t*. (2 marks)
c. Write OpenGL code to draw the quartic Bezier curve with 10 subdivisions as shown below. Also draw the "hairy" unit normal vectors (namely, the length is 1 unit). You can assume OpenGL can take in the Vector2D class, e.g. `Vector2D p; glVertex2fv(p);` . (2 marks)

```
Vector2D p(t, Vector2D p[5])
{









}

Vector2D p_dash(t, Vector2D p[5])
{









}

drawHairyBezierCurve(Vector2D p[5])
{









}
```

--- END OF PAPER ---