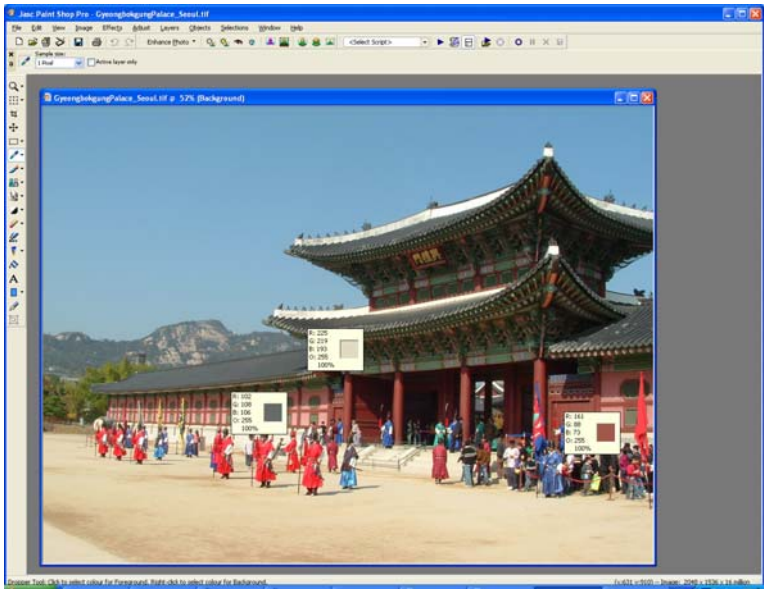
Computer Science

COMPSCI 372 S1 C - Assignment 1 Sample Solution

1. *Introduction to Computer Graphics* *(10 Marks)*

(a) [1 Marks] Open the image "GyeongbokgungPalace_Seoul.tif"with a drawing program and find out the RGB values of the pixel in the **top left corner** of the image. If you use PaintShop then you can use the "DropperTool" (see below). Create a Word document `Ass1Answers.doc` and write your answer into it.



Solution:

```
RGB = (112, 161 193)

NOTE: If you use the dropper tool it easier to pick the correct (top-right) pixel if you
first enlarge the image. Alternatively you can save the image in a suitable
text format (e.g. ".ppm?) - the top-left pixel is the first pixel in the file).
```

(b) [1 Marks] What was the name of the NZ developed software which was used for the crowd animation scenes in "Lord of the Rings"? Give a link to a video on the web giving an example how the software works. Write your answer into the document `Ass1Answers.doc`. [HINT: Check out the company website or YouTube]

Solution:

```
The company is "Massive Software" (http://www.massivesoftware.com/

There are lots of good examples illustrating how the software works on YouTube
and the company website, e.g. under "Special Features":
http://www.massivesoftware.com/special-features/
More informative examples are given in the SIGGRAPH course notes, which however
are not publicly available.
```

(c) [1 Marks] Computer animations are usually recorded with 20-30 frames/second. ***South Korea*** uses the Television standard "NTSC M" – how many images per second does this TV standard utilise? Write your answer into the document `Ass1Answers.doc`.
[Hint: Search the web with Google]

Solution:

The "NTSC M" standard was initially designed for Black and White television using 30 frames/second and only required a narrow bandwidth. However, after colour TV was invented it turned out that encoding the colour signal interfered with the audio signal. With some clever engineering the developers managed to adapt the standard to colour TV by dropping the frame rate to 29.97 frames/second.

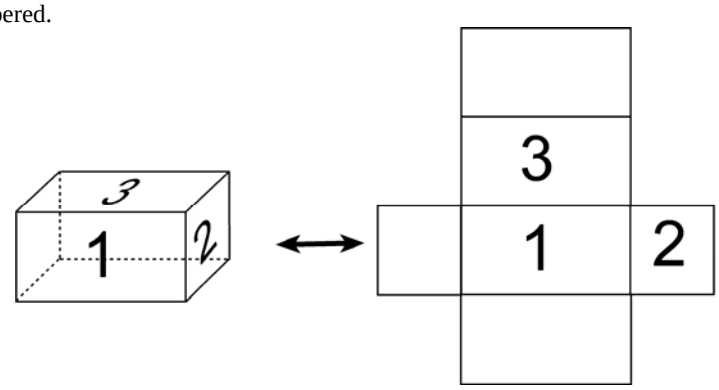
More information on "NTSC M": http://en.wikipedia.org/wiki/NTSC
Detailed explanation why the frame rate is 29.97:
http://www.videointerchange.com/pal_secam_conversions.htm

(d) [2 Marks] Despite huge progress in the past few years there are still no completely realistic computer generated facial animations available. Explain why it is so difficult to model and animate a realistic looking human face.

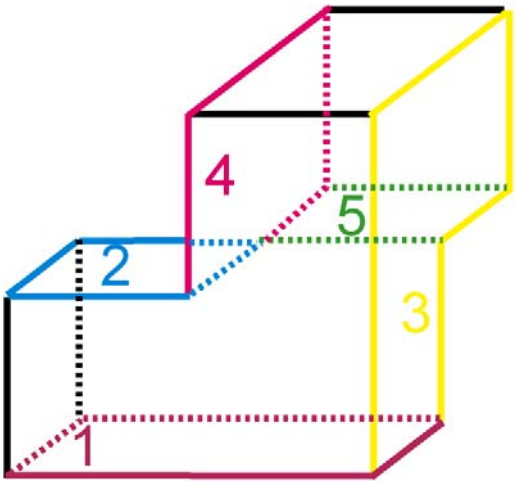
Solution:

Humans are very familiar with human faces because we see them everyday. Hence small abnormalities in facial texture (e.g. too smooth), skin illumination or facial expressions are immediately registered by the human brain.

(e) [5 Marks] The picture below shows on the left a 3D object with 6 faces and on the right a 2D shape which when folded appropriately represents the 3D object on the left. In order to illustrate the correspondence between the two representations three of the faces have been numbered.



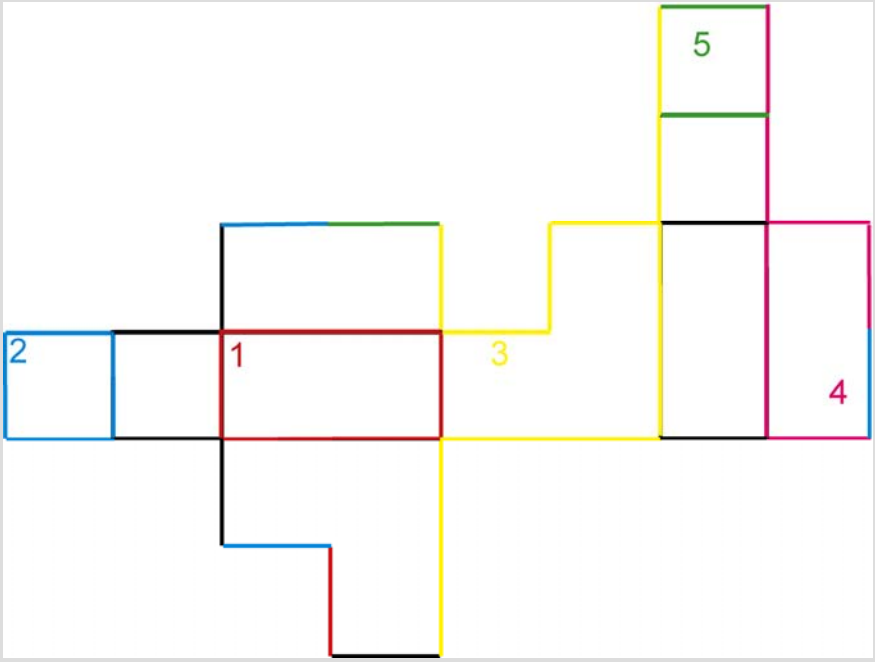
The image below shows a more complex 3D object with 10 faces. Draw a 2D shape which, when folded appropriately, results in this object. Note that there are multiple correct solutions. Make sure that your 2D shape has 10 faces and that it's connected (i.e. it must be one single shape). Indicate which of the faces of your 2D shape correspond to the faces 1-6 in the image below and colour the edges as in the image below.



You can draw your solution using a drawing package such as Freehand or you can draw it on a piece if paper and then either scan it in the lab or make a digital photograph of it. Please add the image of your solution to the document `Ass1Answers.doc`.

Solution:

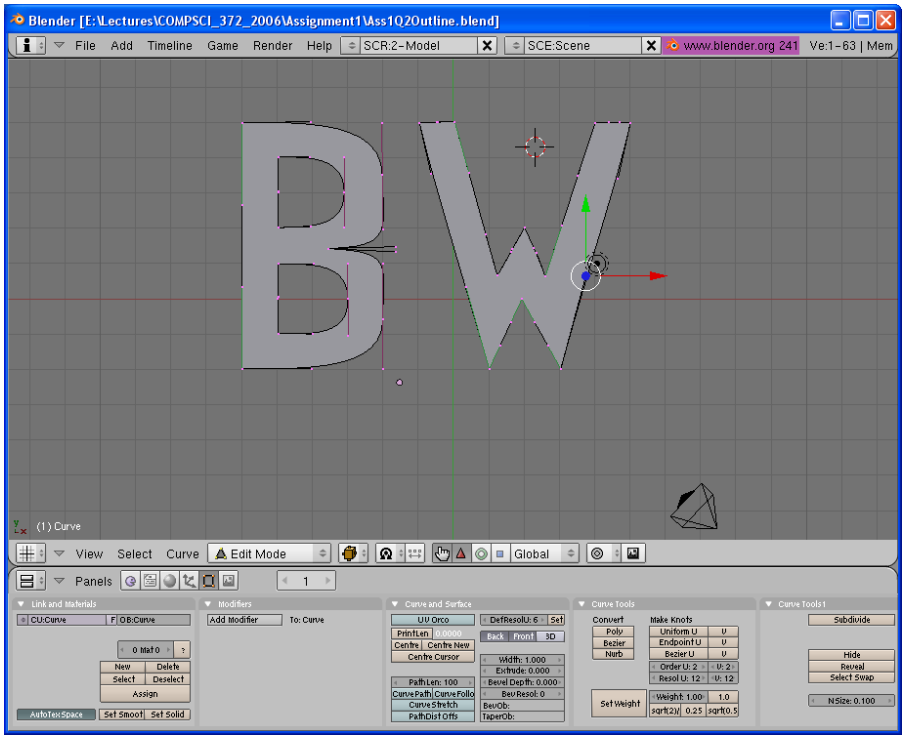
There are many different correct answers for this question. Below is **only one of the possibilities.**



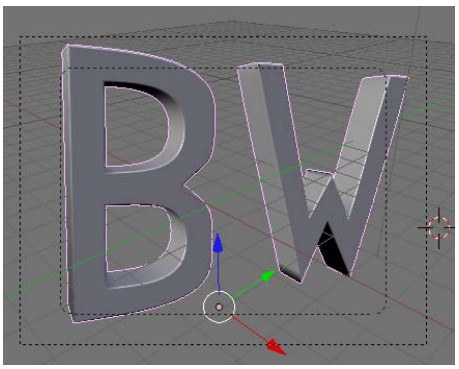
2. *Modelling with "Blender"* *(10 Marks)*

In this question you will design a rounded 3D model of the first two letters of your UPI (i.e. your initials). Please use **capital** letters. For example, my UPI is "bwue001"and hence I create a model of "B W".
Please make absolutely sure that you model the first two letters of **your** UPI – modeling different letters will result in zero marks.

(a) [6 Marks] Model the outline of the 2D shape of your initials using Bezier curves as described in the Blender tutorial (http://www.cs.auckland.ac.nz/~jli023/opengl/blender3dtutorial.htm , section "curves"). At the end your model should look similar to the picture below. Save your model as a file `Ass1Q2Outline.blend`.

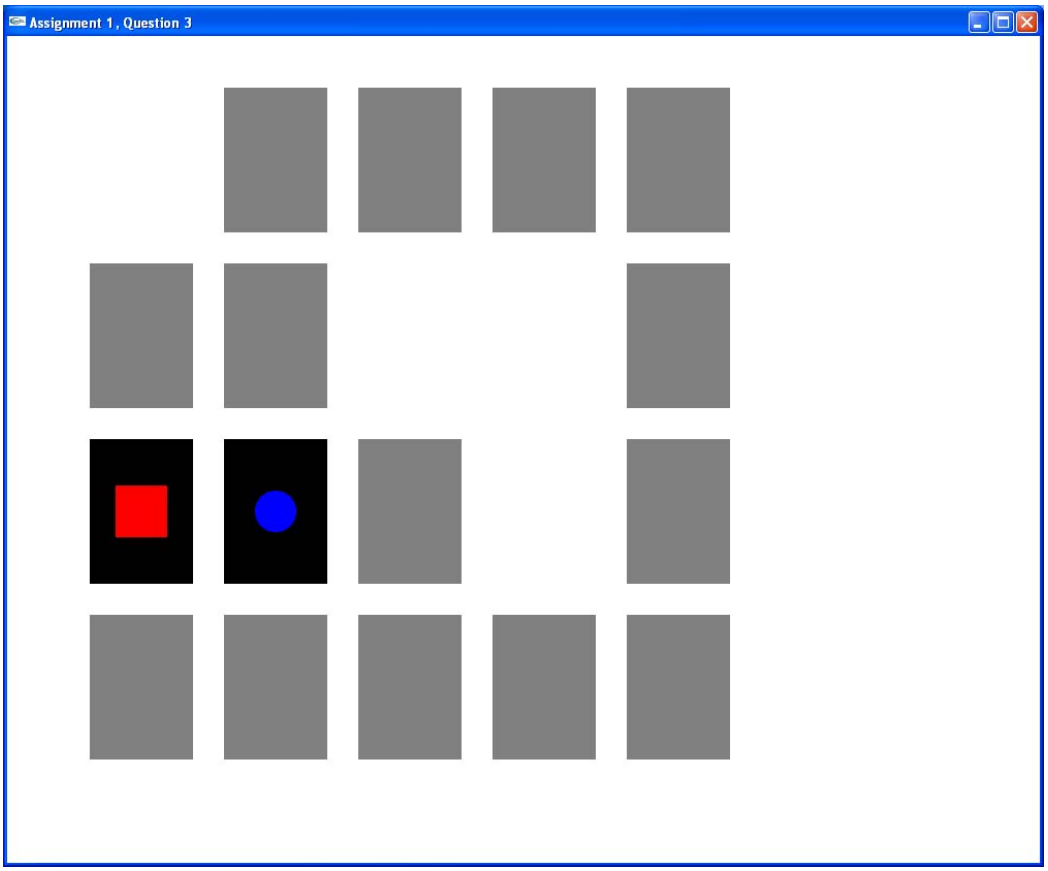


(b) [4 Marks] Extrude the 2D shape into a 3D model and create nice smoothly rounded edges using the beveling tool as described in the tutorial. Save your model as a file `Ass1Q2Final.blend`, position it appropriately with respect to the camera and create a screenshot and save it as `Ass1Q2Final.jpg`. The images below give an example.



3. Introduction to C/C++ and OpenGL (10 Marks)

In this question you will program a simple memory game. The game uses 10 pairs of identical cards which lie on the table face down. You can click 2 cards which are put face up – if they are the same they are removed, otherwise they are put face down again. The aim is to find all pairs of matching cards as quick as possible with as few clicks as possible. The picture below shows an example.



Download the file Ass1Q3.zip from the assignment web page and unzip it. The file contains a .NET solution with the program skeleton for this memory game. The files `MyMath.h` and `MyMath.cpp` contain the class `CVec2df` which implements a two-dimensional floating point vector. The x-coordinate of an object `v` of type `CVec2df` is accessed with v[X] (or v[0]) and the y-coordinate with v[Y] (or v[1]). The class implements typical arithmetic operations on vectors such as vector addition and multiplication with a scalar.
The file `ass1Q3.cpp` contains a function `geometryLibraryTest()` with some examples, e.g.

```
CVec2df v1;                          // v1 = (0, 0)

v1[X]=3.0f; v1[Y]=1.0f;              // v1 = (3, 1)

CVec2df v2(1.0, 2.0);                // v2 = (1, 2)
```

`v2*=5.0f;` // $\mathbf{v}_2 = 5 * \mathbf{v}_2 = \begin{pmatrix} 5 \\ 10 \end{pmatrix}$

CVec2df v=v1+v2; // $\mathbf{v} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \begin{pmatrix} 5 \\ 10 \end{pmatrix} = \begin{pmatrix} 8 \\ 11 \end{pmatrix}$

(a) [2 Marks] Implement the operator '`*`' for the `CVec2df` class which takes as input two vectors and returns the dot product (type float) of these vectors. The dot product of two vectors **u** and **v** is defined as: $\begin{pmatrix} u_x \\ u_y \end{pmatrix} \bullet \begin{pmatrix} v_x \\ v_y \end{pmatrix} = u_x v_x + u_y v_y$.

(b) [2 Marks] Implement the output operator '`<<`' for the `CVec2df` class. A vector is output with its coordinates between brackets, i.e. "(*x-coordinate*, *y-coordinate*)". In the example above **v** should be output as "(8, 11)". You can test your solutions for (a) and (b) by uncommenting the last two lines of the function `geometryLibraryTest()`.

Solution (for (a) and (b)):

```
// header file
class CVec2df
{
public:
   // rest of class omitted
   // dot product
   friend float operator*(const CVec2df& v1, const CVec2df& v2);
   // output operation
   friend ostream& operator<<(ostream& s, const CVec2df& v);
 private:
   float *v;
 };
```

```
// source file
float operator*(const CVec2df& v1, const CVec2df& v2)
{
   return v1[X]*v2[X]+v1[Y]*v2[Y];
}

ostream& operator<<(ostream& s, const CVec2df& v)
{
   s << "(" << v[X] << ", " << v[Y] << ")";
   return s;
}
```

An OpenGL scene is drawn by defining a display method which is given as an argument to `glutDisplayFunc()`. In order to make your code more readable it's a good idea to use an object-oriented design and to define various classes for geometric objects, which all inherit a `draw` method from a common super class.

(c) [3 Marks] A convex polygon is specified by a list of vertices. Implement the draw method of the class `CConvexPolygon2df` such that it draws a filled polygon using the RGB colours defined in its superclass.

Solution:

```
void CConvexPolygon2df::draw(){
       glColor3f(r,g,b);
       glBegin(GL_POLYGON);
       for(int i=0;i<numVertices;i++)
              glVertex2fv(vertices[i].getV());
       glEnd();
}
```

(d) [3 Marks] Define a class `CEllipsoid2df` in the files `Object2df.h` and `Object2df.cpp` which is a subclass of `CConvexPolygon2df`. An ellipsoid has a center (type `CVec2df`), a major radius *a* and a minor radius *b* (type `float`) and a colour (specified by three `float` variables `r`,`g`,`b`). Note that in the constructor you have to define: "`type=ELLIPSOID;` "

The ellipsoid class should have the same methods as the convex polygon class, i.e.

- a default constructor
- a constructor taking the center and the major and minor radius as arguments
- a method `setColour`
- a method `draw`
- a destructor

Note that in order to get full marks you should inherit methods where appropriate and only write new code where necessary. The ellipsoid **must** be represented by a convex polygon with n=32 vertices $\mathbf{v}_i$ around its circumflex. If the ellipsoid has the centre (x_c,y_c) and the major radius a and the minor radius b than these points are defined by

$$\mathbf{v}_i = \begin{pmatrix} x_c + a\cos\left(2\pi\frac{i}{n}\right) \\ y_c + b\sin\left(2\pi\frac{i}{n}\right) \end{pmatrix}$$

After you have implemented the class, please uncomment the line `#define ELLIPSOID_IMPLEMENTED` at the top of the file `ass1Q3.cpp`.

Solution:

```
// Header file
class CEllipsoid2df:public CConvexPolygon2df
{
public:
        CEllipsoid2df();
        CEllipsoid2df(const CVec2df& c, const float aa, const float bb);
        virtual ~CEllipsoid2df();
private:
        CVec2df centre;               // Centre
        float a,b;
};
```

```
// Source file
CEllipsoid2df::CEllipsoid2df():CConvexPolygon2df(){ type=ELLIPSOID; }

CEllipsoid2df::CEllipsoid2df(const CVec2df& c, const float aa,
                             const float bb):centre(c),a(aa),b(bb)
{
   type=ELLIPSOID;
   int numCircumferentialSteps = 32;
   numVertices=numCircumferentialSteps;
   vertices=new CVec2df[numVertices];
   for(int i=0;i<numVertices;i++){
        vertices[i][X] = centre[X] + a*cos(2*PI*((float)i/(float)numVertices));
        vertices[i][Y] = centre[Y] + b*sin(2*PI*((float)i/(float)numVertices));
   }
}

CEllipsoid2df::~CEllipsoid2df(){}
```