Computer Science

COMPSCI 373 S1 C - Assignment 2

Due Date: Monday 13th May 2013, 9pm

This assignment is worth 8.3333% of your final grade.

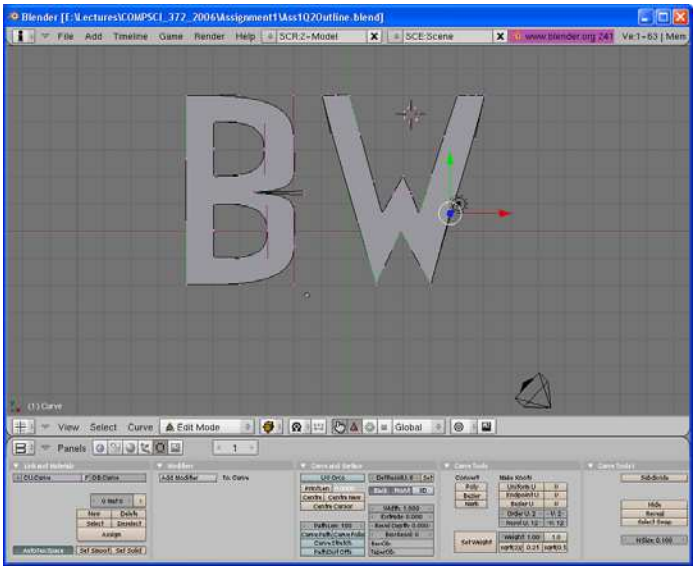
1. Modelling with "Blender" *(10 Marks)*

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.
NOTE: If your name is written with non-Latin characters (e.g. Chinese or Korean characters) then please feel free to model instead your name using your native language. The level of complexity must be at least the same. E.g. for a Chinese name model at least one character (e.g. 陳) and for a Korean name at least two characters (e.g. 은진).

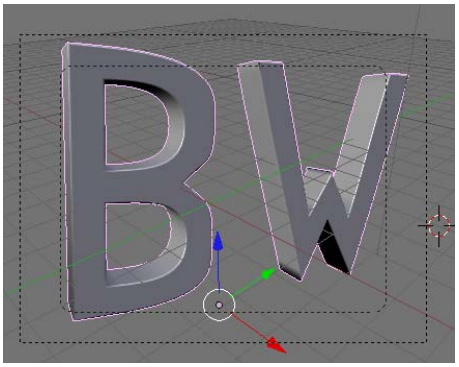
(a) [6 Marks] Model the outline of the 2D shape of your initials using Bezier curves as described in the Blender tutorial (http://www.cs.auckland.ac.nz/~jli023/opengl/blender3dtutorial.htm , section "curves").

NOTE: The interface of the latest version of Blender looks very different from the tutorial. However, most of the commands are the same. One difference I found is that you need to press "ALT+c" to close a Bezier curve. In order to get a 2D shape formed by two nested curves go to the "Bezier curve" tab on the right of the interface and press "Shapes -> 2D". TIP: make regular backups of your model.

At the end your model should look similar to the picture below. Save your model as a file `Ass2Q1Outline.blend`.



(b) [4 Marks] Extrude the 2D shape into a 3D model and create smoothly rounded edges using the beveling tool. Save your model as a file `Ass1Q1Final.blend`, position it appropriately with respect to the camera and create a screenshot and save it as `Ass2Q1Final.jpg`. The images below give an example.



2. 3D Modelling and Texture Mapping with OpenGL (17 Marks)

In this question you will create a 3D scene containing a plane with grass, 12 trees, a large mirror, two bouncing balls, a Greek monument, and 20 flowers. Most of the code is already completed and you have to implement only a few methods containing concepts related to our lecture.

Download the file Ass2Modelling.zip which contains a .NET solution including all necessary source files. Unzip the files and compile and run them. You should see a grass covered plane, 2 bouncing balls, trees, and a reflecting mirror.

NOTE: In order to make debugging easier and improve the understanding of texture maps, all texture images are stored in Portable Pixel Map format (ppm). This format is very inefficient to load and your program might take a long time to load when run in DEBUG mode. You can avoid this by compiling your program in RELEASE mode.

(a) [6 Marks] Complete the function `draw` of the class `CFlower` which draws a flower at position (x, -1, z).
The stem of a flower is represented by a cone oriented in y-direction (i.e. upwards) with a radius of `stemRadius` and a height of `stemHeight`.
The flower has two leaves, each of which is an ellipsoid with major axis `leafLength/2` and a medium and minor axis which are 40% and 10%, respectively, of these values. The leaves are attached to the stem at the heights `leafHeight1` and `leafHeight2` and have an angle with respect to the stem of `leafAngle` and -`leafAngle` degrees. The leaves of each flower lie in a plane which has an angle of `angleWithXAxis` degrees with the x-axis.
Centred on the top of the stem is a blossom which is an ellipsoid with the major and medium radius `blossomCentreRadius` and a minor radius of 0.5 times the major radius (i.e. it's a squashed sphere with a height of 50% of its diameter).
The blossom has `numPetals` each of which is a longitudinal ellipsoid with a major radius of `petalLength/2` and a medium and minor radius of 0.5 times and 0.2 times the major radius, respectively. The petals are distributed over the top half of the blossom using a spiral point function which is defined as follows:

A spiral point set of size N, distributed over the surface of a sphere, is defined by[1]

$$\theta_k = \cos^{-1} h_k \ , \quad h_k = -1 + \frac{2(k-1)}{(N-1)} \ , 1 \le k \le N$$

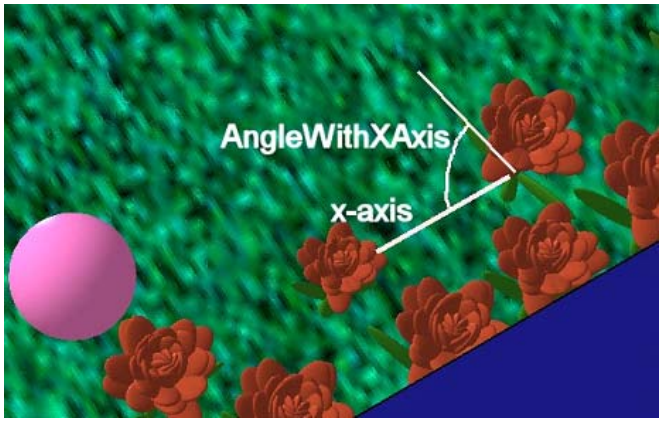
$$\phi_k = \left(\phi_{k-1} + \frac{3.6}{\sqrt{N}} \frac{1}{\sqrt{1-h_k^2}} \right) \left(\mathrm{mod}\, 2\pi \right)$$

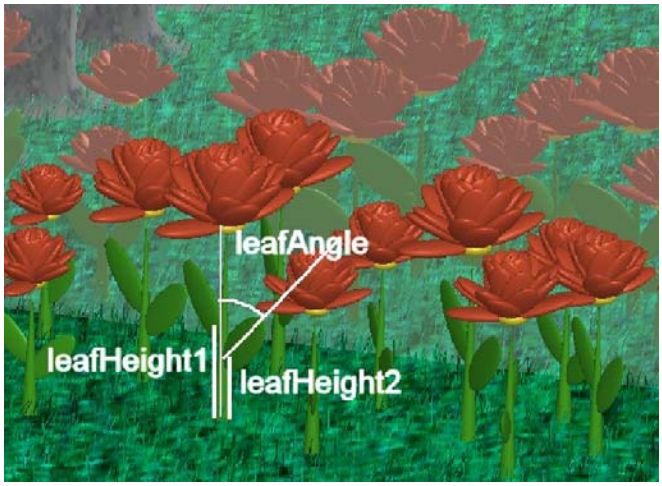
where (ϕ, θ), $0 \le \theta \le \pi$, $0 \le \phi \le 2\pi$ are the spherical coordinates of the points.

You can use these coordinates to rotate the petals into the correct positions. At the end you should obtain a solution similar to the images below. Note that you can use different methods to generate the flower petals as long as the results look similar.

Please use the materials defined in the files Flower.h and Flower.cpp
For the stem and leaves use "`mat_stem`" (light green)
For the blossom centre use ""`mat_blossom_centre`" (yellow)
For the blossom petals use ""`mat_petal`" (red)

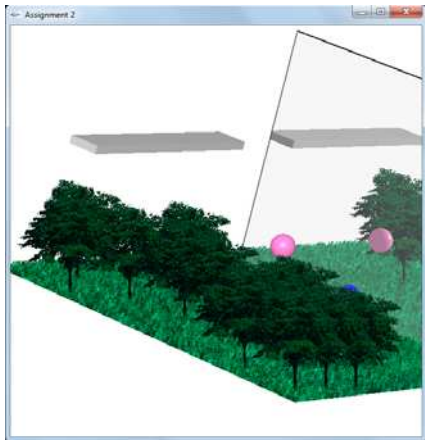
Your final result should look similar to the images below.





[1] E. A. Rakhmanov, E. B. Saff, and Y. M. Zhou. Minimal discrete energy on the sphere. Mathematical Research Letters, 1(6):647-662, November 1994.

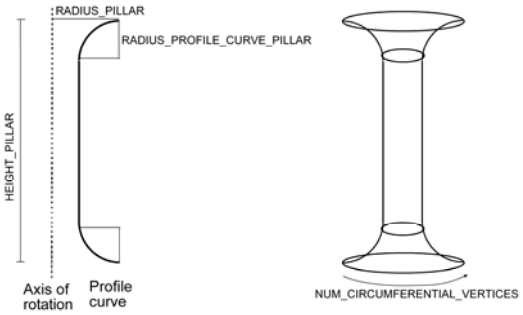
(b) [6 Marks] Complete the class `CMonument` so that it draws a Greek monument. Currently only the top slap of the monument is defined and the displayed scene will look as indicated on the right.

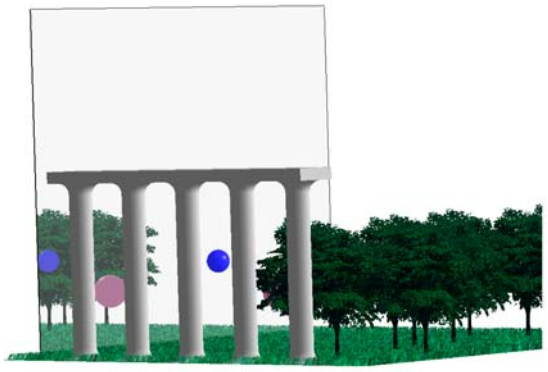


You should modify the class such that it draws the five pillars of the monument. Define a pillar as a **surface of revolution** (with correct surface normals!). Then draw this surface (using quad strips) five times in the draw method and perform appropriate translations such that all pillars are correctly positioned under the top slap.

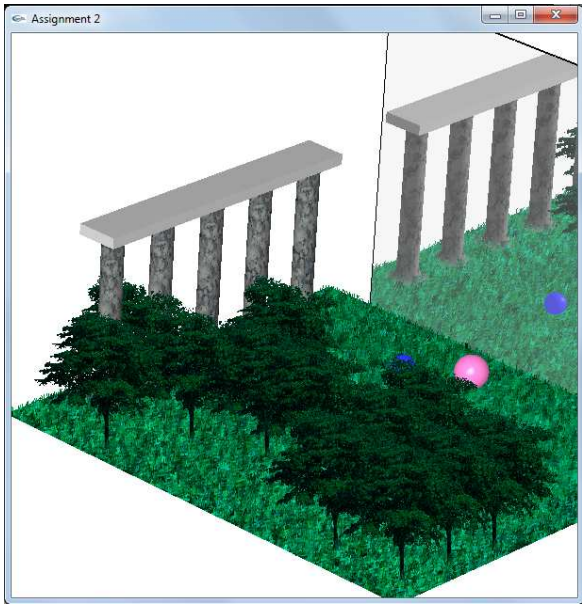
The surface of revolution should be subdivided in circumferential direction using `NUM_CIRCUMFERENTIAL_VERTICES` vertices and in horizontal direction using `NUM_HORIZONTAL_VERTICES` vertices evenly distributed over the two quarter circles of the profile curve.

The resulting scene should look as indicated in the image below on the right.





(c) [5 Marks] Map a stone texture onto the pillars. You may use any image of a stone texture, but remember that the dimensions of a texture must be a power of 2. You are allowed to use the image "stone.ppm" shown below on the left. The result should look similar to the image on the right.



In order to complete this exercise you have to compute the texture coordinates in the constructor of the `CMonument` class (approximately 2 lines of code) and you have to bind and apply the texture in the `draw` method of that class (approximately 6 lines of code). If you want to use a different texture please change the file name in the constructor and don't forget to submit the ppm-file with that texture.

Note: When rendering the texture image use **texture coordinates and vertex normals** and use

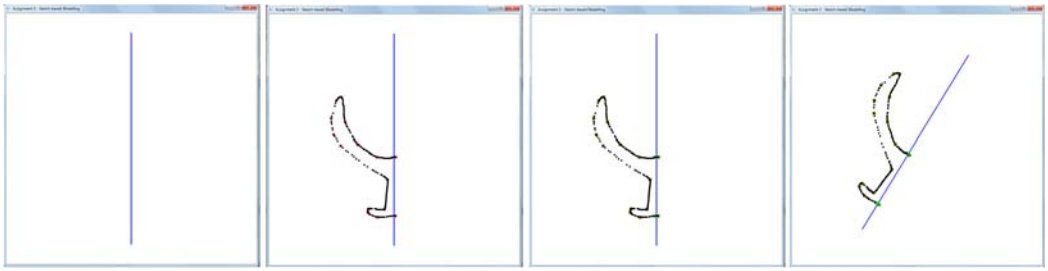
```
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

This way the texture is modulated with the shaded polygons, which gives an illuminated texture.

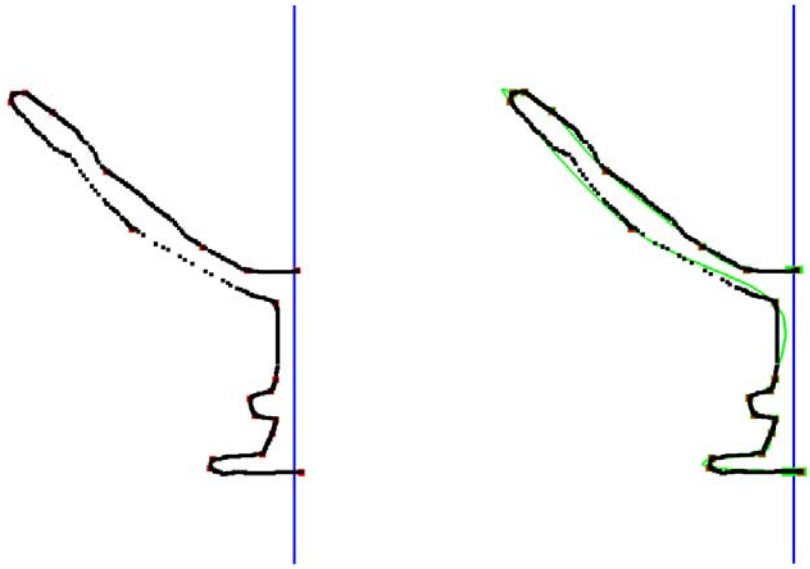
3. Sketch-based Modelling (11 Marks)

In this question you will create a sketch-based modelling tool to create a surface-of-revolution from a sketched curve.

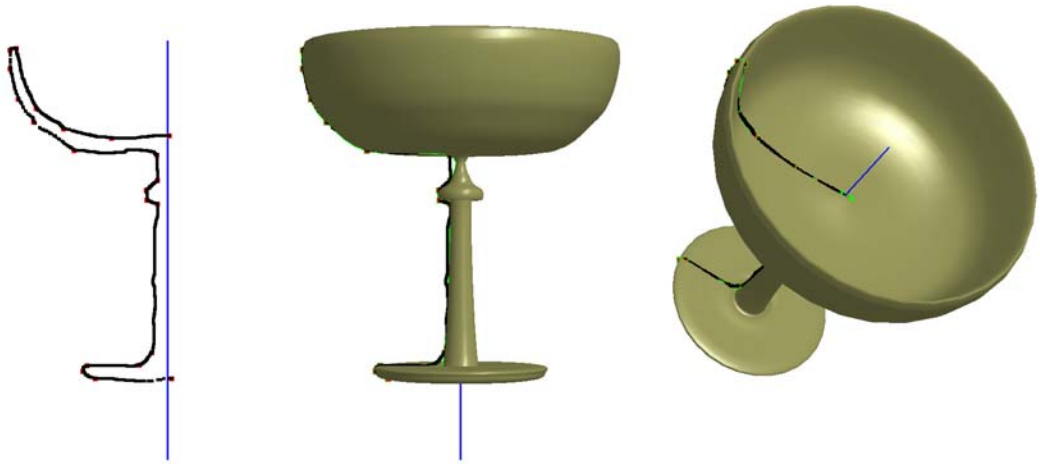
Download the file Ass2Sketch.zip which contains a .NET solution including all necessary source files. Unzip the files and compile and run them. You should see the picture on the left. Please sketch a profile curve by pressing the **right mouse button** and moving the mouse. The curve needs to lie on the left side of the blue rotation axis (as indicated in the second image from the left). If you press "s" the surface-of-revolution is computed. The program skeleton has not yet code for constructing the curve and surface and you will hence only see the control points of the Catmull-Rom spline curve (in green). You need to define the actual curve and surface in the next two steps of this question.
Note that after pressing the 's' key you get into "surface mode" and you can rotate the surface by pressing the **left mouse button** and moving the mouse. You can get back to the sketch mode by pressing "r" (reset) two times.



(a) [5 Marks] Insert into the file CatmullRom.cpp the missing code for computing and displaying the Catmull-Rom spline curve defined by the `n` control points stored in the array `ctrlPoints`. After completing this part you should be see a green curve smoothly interpolating the red points on your sketch.



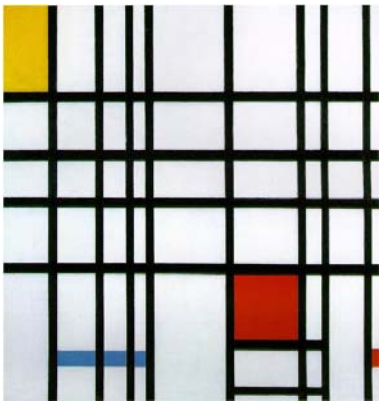
(b) [6 Marks] Insert into the file CatmullRom.cpp the missing code for computing and displaying the surface-of-revolution defined by the Catmull-Rom spline profile curve defined in the previous step. After completing this part you should be able to create nicely shaded 3D models as indicated in the picture below.

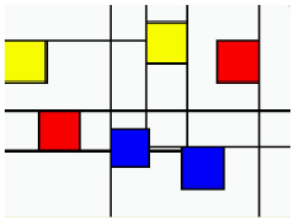


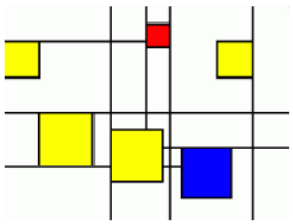
Bonus Question (OPTIONAL!!) – Informative Art (2% of the final grade)

At a conference some years ago a Swedish group (Lars Erik Holmquist and Tobias Skog, Future Application Lab, Viktoria Institute, Göteburg, Sweden) presented a paper titled 'Informative Art: Information Visualization in Everyday Environments'. The idea behind this paper was to combine a dynamically updated information display with the decorative role of visual art, such as posters and paintings.
In this question you should develop an animation displaying 'Informative Art'. You are marked according to creativity (i.e. select a suitable piece or art to display the information and create an animation inspired by it), artistic skills (does your animation look like 'art'?) and technical skills (the quality of your animation).

Here are two examples from the original paper:
Example 1: The image below on the left is a painting from Piet Mondrian (Composition with Red, Yellow, Blue - 1921). The authors create an interactive display similar to this painting consisting of 6 squares indicating the weather in 6 cities. The size of a square indicates temperature and its colour indicates sunshine, cloudy weather or rain. The two images on the right show two instances of the resulting interactive display.







Example 2: The image below on the left is a work from the American 'Pop" artist Andy Warhole (100 Campell's Soup Cans). The authors create an interactive display similar to this work representing a count-down clock or "egg-timer". The interactive display uses asparagus soup (yellow) and tomato soup (red) cans. When the countdown starts all cans are asparagus soup cans and at each time step one asparagus soup can is replaced with a tomato soup can. The images on the right show two instances of the resulting animation.

You can find more examples on the "Image of the Week" section of the Graphics Group website: http://www.cs.auckland.ac.nz/research/groups/GG/weeklyimages/

Your task: Create an "informative art" animation. Do **NOT** copy the above ideas or ideas developed by somebody else. Look for a suitable data set you want to visualize and find a suitable art piece you want to use for this visualization. Submit a Word document `InformativeArt.doc` which explains the data set you visualize, the art piece you chose and how the art piece is modified to represent this information. Submit a program with the main method `InformativeArt.cpp` which implements your idea. If you submit several files please create a zip-file `InformativeArt.zip`. **Please note that this bonus questions will only be marked if you provide a working implementation compileable with the .NET IDE!**

☺ *Enjoy!*

Assignment submission– NOTE: Due date is Monday the 13th May 2013, 9pm

All files must include your name, UPI and ID in a comment at the beginning of the file. All your files must be able to be compiled in the stage 3 lab without requiring any editing – any file which doesn't compile results in zero marks. I strongly recommend testing your submission in the lab before submitting, i.e. download the program skeletons, add your modified files, and run the application.
In particular any file with OpenGL commands should include the corresponding OpenGL libraries as:

```
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>
```

All your files should include adequate documentation.

The assignment due date is Monday, the 13th May 2013 (time: 9pm). Late assignments are not accepted.

Please submit the assignment using the **Assignment Drop Box** (https://adb.auckland.ac.nz/).
If you are a CS771 student please use the CS771 assignment drop box.

NOTE: Please start early. For example, if you get sick a few days before the deadline we expect you to have part of the assignment finished so that we can mark you on this. If you are sick for an extended period of time we can consider an extension or reweighting your other assignments.
If you have any problems please contact me as early as possible to discuss the best option to deal with this :-)

Please submit to the assignment drop box:

- the files for question 1, i.e. `Ass1Q1Outline.blend`, `Ass1Q1Final.blend`, and `Ass1Q1Final.jpg`.
- the file `Flower.cpp` with the missing code completed.
- the file `Monument.cpp` with the missing code completed.
- the file `CatmullRom.cpp` with the missing code completed.
- any other files you have changed (NOTE: The best solution does not require changes to any other files).

Bonus question submission – NOTE: Due date is Monday the 13th May 2013, 9pm

The bonus questions will only be marked if you provide a working implementation using OpenGL which is compileable with the .NET IDE in the stage 3 labs. If you want to use libraries not installed in the labs (e.g. for user interfaces or sound) please talk to me.
The BONUS QUESTION DUE DATE is Monday, the 13th May 2013, 9pm.

Please put all files related to the bonus question into a zip-file with the name "Ass2Bonus_<your UPI>" (e.g. "Ass2Bonus_bwue001.zip") and email it to me: burkhard4u@gmail.com (please use this email address only for the bonus question). Please note that the marking of the bonus question is separate from the rest of the assignment and will not be included in the email you get from the markers.