**Assignment due date: Thursday, Oct. 27, 9am**

**Hand-in to be submitted at the start of tutorial,
code to be submitted on the mathlab server by the above due date**

**Student Name (last, first):**

***Student number:***

***Student UtorID:***

***I hereby affirm that all the solutions I provide, both in writing and in code, for this assignment are my own. I have properly cited and noted any reference material I used to arrive at my solution, and have not shared my work with anyone else.***

______________________________
***Signature***

***(note: -3 marks penalty for not completing properly the above section)***

This assignment has 2 main goals: First, to let you practice and improve your understanding of surfaces, normals, gradients, and the different ways to represent them. Secondly, you will implement hierarchical objects (plants in this case) that will help you improve your understanding of coordinate frames and transformations. You will create simple animations and gain insight into how terrain and terrain features are created algorithmically.

***Learning Objectives:***

You will apply your understanding of surfaces, normals, and tangents to the task of solving geometric problems. This will help you practice and improve your knowledge of these mathematical entities. You will also learn to transform between world and camera coordinate systems.

You will explore the use of binary space-partitioning trees for visibility estimation.

You will understand how plants can be generated algorithmically using L-systems, and how the L-systems can be tweaked to generate different plant shapes.

You will learn to generate and render surface from Mathematical functions to simulate terrain.

***Skills Developed:***

Manipulating 3D surfaces algebraically

Working with hierarchical sequences of transformations in OpenGL.

Handling illumination, def ning normal vectors and material properties.

Creating more complex animations.

***Reference material:***

The Lecture notes up to this point, found on the course website.

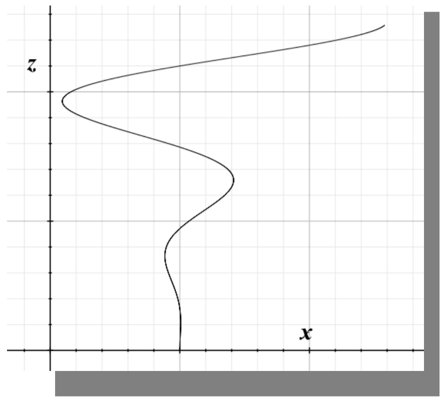
Detailed comments in ***PlantLife.cpp***. The comments describe in detail what it is that you have to implement, and they will tell you how the plant generation process works.

Your OpenGL reference text. On-line material on illumination and texture mapping inOpenGL.

***Part 1 – Pen and paper questions***

1- Surfaces of revolution

A simple way to create a vase shape is to start with a parametric curve arc as shown below:



Then revolve that arc around the ***y*** axis. This parametric curve can be derived using an instance of the parametric equation (i.e. for some choice of scalars *a, b*):

$$\bar{p}(t) = \left(a + \ t^2 \cos t\,, 0, \frac{t}{b}\right), 0 \ \leq t \ \leq 2\pi$$

a) (5 marks) Give a parametric equation and appropriate parameter bounds for the resulting surface of revolution *p(u, v)* in terms of the surface parameters *u* and *v*.

b) (5 marks) Give an equation for the tangent plane at a point *p(u, v)* in terms of the surface parameters *u* and *v*.

c) (4 marks) Give an equation for the outward-facing unit normal vector at point *p(u, v)* in terms of the surface parameters *u* and *v*.

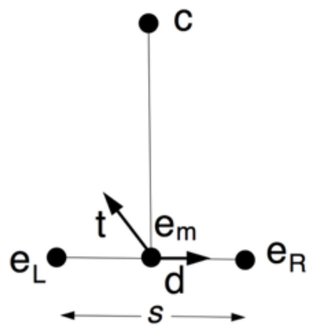
***Part 1 – Pen and paper questions***

2- Camera Transformations

(16 marks) In stereo rendering, two cameras are needed with slightly different vantage points and view directions. Each is used to render an image for the corresponding eye. This can be specified with the following parameters:

- $\boldsymbol{c}$: The center of interest, a point in world space that lies along the optical axis of both cameras.
- $\boldsymbol{e}_m$: The midpoint between the eye positions of each camera.
- $\boldsymbol{t}$: An "up" vector that allows us to specify a tilt rotation about the axis passing through $\boldsymbol{c}$ and $\boldsymbol{e}_m$. The z-axis is a special case.
- $s$: The distance between the two eyes.

All answers should be given in terms of the above stereo parameters, as well as any intermediate quantities you specify.



a) (2 marks) Give an expression for the unit vector $\boldsymbol{d}$ that is perpendicular to both $\boldsymbol{t}$ and $\boldsymbol{c} - \boldsymbol{e}_m$ such that $(\boldsymbol{c} - \boldsymbol{e}_m, \boldsymbol{t}, \boldsymbol{d})$ is a right–handed coordinate frame.

b) (3 marks) Give expressions for $\boldsymbol{e}_L$ and $\boldsymbol{e}_R$ the eye positions of each camera.

c) (3 marks) Give expressions for the basis vectors that make up the two cameras' coordinate frames: $(\boldsymbol{u}_L, \boldsymbol{v}_L, \boldsymbol{w}_L)$, and $(\boldsymbol{u}_R, \boldsymbol{v}_R, \boldsymbol{w}_R)$.

d) (5 marks) Give an expression for the 4x4 homogeneous transformation matrix that transforms a point in the camera space of the left eye $(\boldsymbol{p}_L)$ to the camera space of the right eye $(\boldsymbol{p}_R)$. In other words, what is MLR such that $\boldsymbol{p}_R = \boldsymbol{M}_{LR}\boldsymbol{p}_L$.

e) (3 marks) Give a test that determines whether or not a polygon face with normal $\boldsymbol{n}$ is a backface that can be culled when rendering from both cameras.

***Part 1 – Pen and paper questions***

3- Camera coordinates and coordinate conversion

a) (7 marks) Give the ***world-to-camera*** 3D homogeneous transformation matrix for a camera with:

$$\vec{e}_{wc} = (1,2,5)^T$$

$$\vec{t} = (1,1,0)^T$$

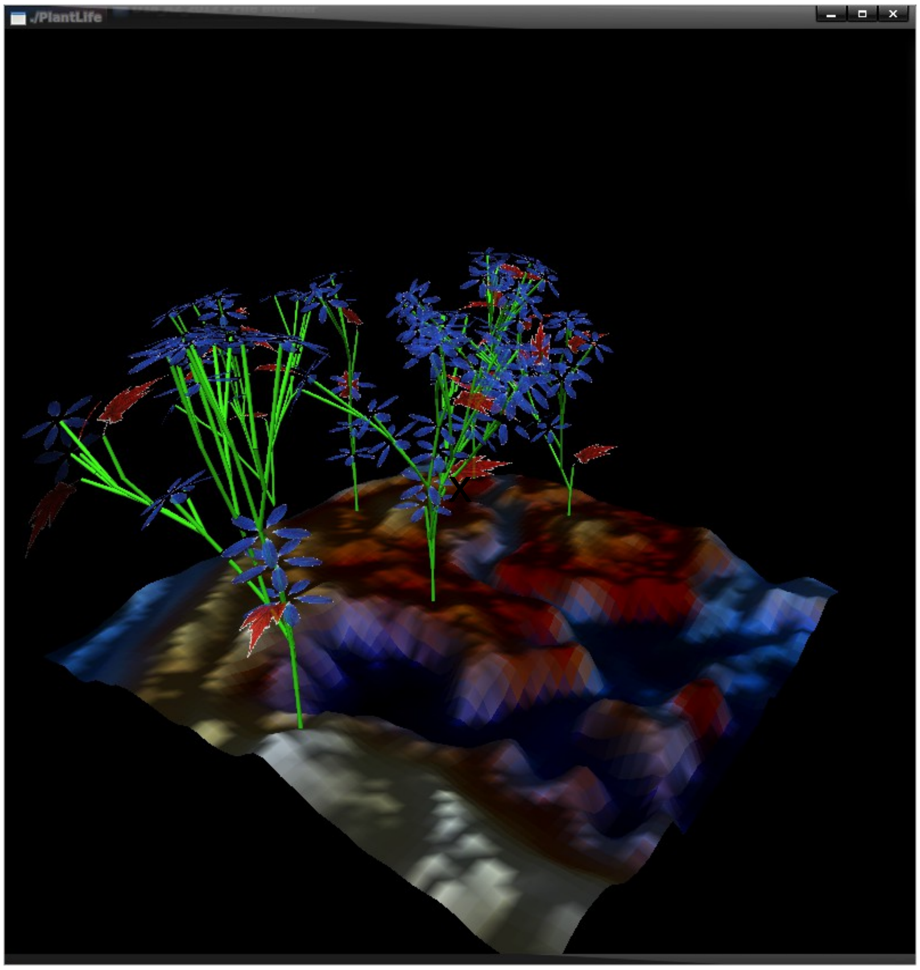
looking at point

$$\vec{p}_{wc} = (-1,-2,0)^T$$

Give the derivation and values for each of the vectors that def ine the camera's coordinate frame, as well as the f inal 4x4 transformation matrix.

b) (3 marks) Using the above, show the ***camera-to-world*** 3D homogeneous transformation matrix.

## *Part 2 – Plant Life 2016*



***Screenshot of my solution for A2***

Download and unzip the starter code provided with this assignment into a suitable (empty) directory on your account in the mathlab server (mathlab.utsc.utoronto.ca). Compile the starter code by typing '***make***'. The resulting executable '***PlantLife***' opens an empty window and creates a bare-bones GUI with a '***quit***' button. Initially the graphics window is empty.

Your task for this assignment will be the creation of a forest of virtual plants. Each virtual plant will be generated using a stochastic L-system. An L-system is a simple, rule-based algorithm for generating complex structures. In particular, the L-system involves a set of symbols (in our case, 'a','b','c','d') representing plant structures, and a set of rules for expanding these symbols on successive levels of the plant so as to generate a plant-like structure.

## *Part 2 – Plant Life 2016*

For example, suppose we had only two symbols: 'a', and 'b', together with the substitution rules a ab, and b a. A plant grown with such a system would look like this:

| | |
|---|---|
| Level 0 (root) | a |
| Level 1 | ab |
| Level 2 | ab a |
| Level 3 | ab a ab |
| Level 4 | ab a ab ab a |

You could then draw a plant by starting at level 0, and drawing a stem wherever you find an 'a', and a leaf wherever you find a 'b'.

The file PlantLife.cpp contains a function MakePlant(), that defines the symbols used, and the substitution rules for each symbol. Note that we use a stochastic L-system, i.e. one in which there are multiple expansion rules for each symbol, and at each level one of these rules is chosen randomly with specified probabilities.

Your task consists of:

a) Completing the implementation of the stochastic L-system. Currently, only the substitution rules for 'b' are implemented. Note that I provide already a tree-structure to hold the L-system, and that a recursive function already exists for expanding each level of the tree.
You have to complete the part that generates and inserts a new level of nodes for the tree in the case of 'a' type nodes.

b) Draw the plant structure. Note that the plant is a hierarchical structure, so you will have to create a suitable function to draw each node using transformations relative to the parent. This will take the form of a tree traversal. Note also that each node contains information about the rotation angles w.r.t. the parent's x and z coordinate axes as well as a local scaling factor.

You will need to create functions that draw flowers, and leafs. Note you're not allowed to use glut objects for leafs and flowers. I want you to define your own using polygons, and remember you must provide for each vertex a corresponding normal vector so that illumination can be properly handled.

c) Once you have completed the L-system and the basic rendering of the plant, you will provide simple GUI controls to enable the user to rotate the forest around th Z coordinate, and to zoom-in or zoom-out.

d) Create a surface on which to set the plants. To do this, you will use some interesting equation to determine the height of a surface on a grid of points . You are free to choose how to set the height, but it should not be random, and it should not be a simple sphere, parabola, or such regular quadratic forms.

## *Part 2 – Plant Life 2016*

Read the starter code for specific details and instructions about what you need to do. There are suggested extensions you can complete for extra credit. Any extra credit beyond the 50 points the programming part is worth, will be applied to your mark for the written part of the assignment.

In addition to completing your programming task, you must complete the CHECKLIST included with the starter code. This checklist also details the marking scheme for the programming part.

### ***Turning in your Solution to Part B***

All your code should remain in the directory 'a2/PlantLife/ '. In addition to your code, you must complete the CHECKLIST. Failure to complete the checklist will result in *__zero*__ marks on your assignment.

Your code should be **well commented** if you want to receive full (or even partial) credit for it. To pack and submit your solution, execute the following commands from just outside the ***a2/PlantLife*** directory:

```
tar -cvfz a2_solution_studentNo.tgz  ./a2
Submit -c cscd18f16 -a A2  -f solution_studentNo.tgz
```

Failure to comply with the above naming convention **<u>will cost you 10%</u>** of your mark for the programming part!

Compatibility

All of your assignments must compile and run on mathlab.utsc.utoronto.ca (Linux). I strongly encourage you to work directly on the lab machines at IC 406. Remote work will be difficult due to the slowness of displaying graphics over ssh.