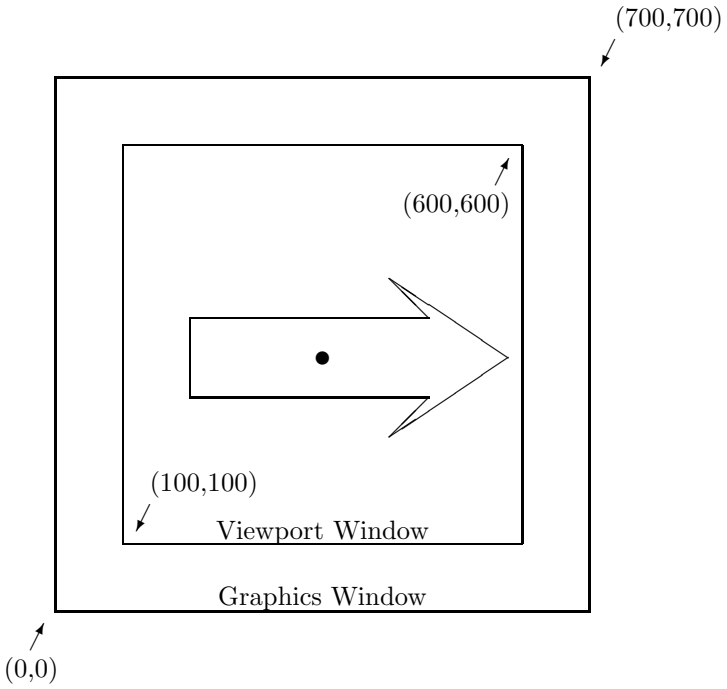
CSC 315 – Introduction to Computer Graphics

Fall 2007

Programming Assignment 2
Due by Midnight, October 10, 2007

This assignment will be comprised of two C or C++ programs utilizing the OpenGL API. The credit awarded for each program is noted at the end of each description.

Program 1 - 2D Transformations and Animation You are to write a program which will define a viewport within a graphics screen and then draw the figure shown in the diagram below. You may omit the central dot which represents the point around and through which rotations and reflections will occur. The graphics screen and viewport will be defined as follows:



In the first program the arrow should be drawn as an outlined, not filled, object. You may use any of the `glVertex` commands to draw your pixels. You may, altrnatively, use one of the following: `GL_LINES`, `GL_LINE_LOOP`, or `GL_POLYGON`.

Here are the particular things I want this program to do.

- Draw the initial graphics and viewport windows and set the background to black. Using some form of the `glRect` command, set the background of the viewport to white.
- The functions of the mouse and keyboard should be as follows:
  - If the mouse is within the viewport, clicking the left button should start the arrow rotating counterclockwise around the central point. Each additional click of the left mouse button will increase the speed of rotation with a maximum of 10° per iteration. Note that clicking the left mouse button will start an *animation*. The arrows should continue to rotate until another event occurs. In other words, once a rotation has started, further clicks of the left mouse button increment the angle by which the figure is rotate during each animation step.
  - If the mouse is within the viewport, clicking the right mouse button should decrement the angle by which the arrow is rotated with the minimum being -10° per iteration.[1]
  - If the mouse is outside the viewport, clicking the left mouse button after a line has been completed should increase the scale of the arrow by 5%.

[1] Once the value of the rotation angle becomes negative, the arrow will be rotating clockwise.

- If the mouse is outside the viewport, clicking the right mouse button after a line has been completed should decrease the scale of the arrow by 5%.
- Hitting "r" or "R" on the keyboard will cause the current image to be *reflected* about the vertical axis passing through the central point. If an animation is in progress, the animation will continue with the reflected image.
- Hitting "s" or "S" on the keyboard should stop any animation.
- Hitting "q" or "Q" on the keyboard should exit the program.

Now, here is the hardest part. If, during the progress of your program, the arrow goes outside the viewport, it needs to be clipped using the **Sutherland-Hodgman polygon-clipping algorithm**. Also, you can write your animation code by following a draw-delete-draw mentality, it will be more efficient for you to implement something called **double buffering**. Double buffering should dramatically increase the efficiency of the animations. Finally, for this program you will need to construct all of the transformation matrices and write the code to execute them correctly and expediently. *(40 pts.)*

Program 2 - Cranking up the Power of OpenGL: The object of this exercise is to give you a taste of some of the available features of OpenGL. I want to you take the program you wrote in part one and incorporate as many features of OpenGL as you can find to make it more efficient. I also want you to fill the arrow with the color of your choice.[2] If possible, in this program, find a way to implement commands like `glRotate`, `glScale`, etc.[3] The grade for this program will be based on the number of new OpenGL commands used. I'm serious, try to have some fun with this. It will require that you spend some time on the `WWW` looking at the programming guide for OpenGL. *(10 pts.)*

Assignment Submission Procedures

You should have a directory named `assign2`. Under this directory I would like you to have four other directories `prog1`–`prog2`. Each directory should contain the corresponding program code and makefile. Send me, via electronic mail, a compressed tar file of the `assign2` directory and all the subdirectories. Use your last name as the name of tar file. If your last name were Doofus, you would `cd` to the directory containing the `assign2` directory and type:

```
tar -cvf - assign2 | gzip --best >Doofus.tgz
```

Mail this file to `pounds_aj@mercer.edu` by the time noted.

[2] If you use `GL_POLYGON`, this should be easy.
[3] You may have to redefine how you represent your segments and you may have to use different transformation matrices.