MAT 2170: Laboratory 4

Key Concepts

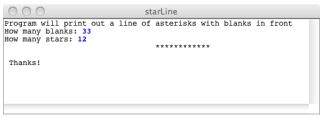
1. `for`, `while` loops
2. integer arithmetic
3. graphics

Introduction

- The main purpose of this lab is to give you practice using loops. In all the programs, you must consider the **stopping criteria** for each loop. Examples worked out with pencil and paper may help you discover what is needed for ***initialization***, ***update***, and ***termination*** in the loops.
- Complete Worksheet #4 before the beginning of Lab 4.
- Create the Lab exercises in the directory `mat2170/lab4` under your user account (not in any other directory). This can be done at the time you create each project.
- Remember: in graphics programs, store the window's width (`getWidth()`) and height (`getHeight()`) as floating point values.
- You can get the slider value in a `SliderProgram` using the method `getN()`, which returns an integer.
- **Warning** regarding graphics programs: once your program appears to produce the correct results, enlarge the graphics window (by dragging its bottom–right corner down and to the right) after the graphics objects have been drawn in order to verify that your program isn't drawing anything outside the observable part of the window.

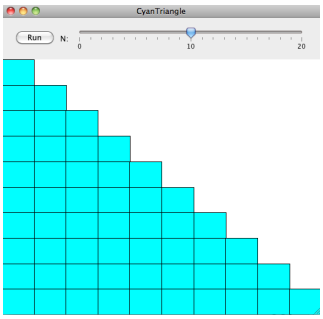
Exercises

1. (Line of Stars) Write a console program, `starLine`, that asks the user for the number of blanks in the offset, and the number of asterisks on the rest of the line. Then display an offset of that many blanks, followed by the number of asterisks they desire, all on one line.



2. (Exercise 6, page 128) Write a dialog program, `Reverse`, which **displays** the digits of a non–negative integer in reverse order. As an example, if the user enters `1234`, your program should output `4321` with a nice message explaining what it represents. Integer arithmetic and a `while` loop should be used to accomplish this.

3. (Lower-left Triangle) Write a `SliderProgram`, `LowerTriangle`, that will produce a series of identical blocks which fill the lower-left triangle of the graphics window. A sample output is shown at right, with the slider set to 10. Copy the `LowerTriangle` java skeleton program from our web site, and place it in an empty java file in the `src` directory of your `LowerTriangle` project. Modify the `run()` method, which is invoked each time the run button is pushed while the program executes.
   The `init()` method initializes the program, setting the size of the window (during testing) and the range of the slider. When publishing your program, set the window size to 500 × 500.

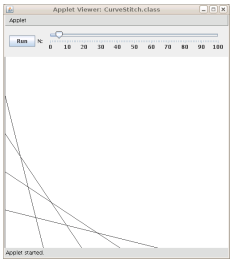


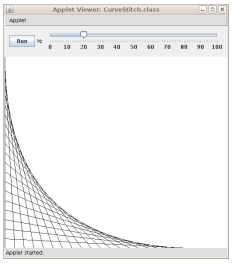
Nested `for` loops are required (basically the loops from the Triangle Number Table program that displayed and summed the numbers from 1 to `N`).

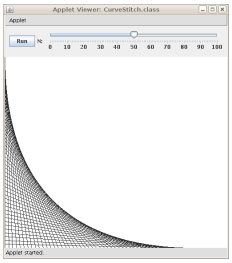
To obtain the width and height of the window, use the messages `getWidth()` and `getHeight()` (do **not** use hard–coded values, aka magic numbers).

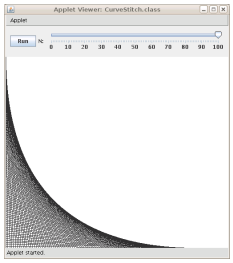
Do not create a block before the nested loops, and then use the `move()` message to reposition it since that will **erase** the block each time before redrawing it. Instead, before entering the loops, determine a block's width and height, and the position of the first block. Then, create a `GRect` object inside the inner loop, based on your pre–calculated values. After displaying each block, modify the position values as necessary using the block's width and height.

4. (Curve–stitch design[1]) Write a `SliderProgram`, `CurveStitch`, which will produce a "curve–stitch" parabola. A single `for` loop is sufficient. Your program should produce a graphics window that is 600 by 600 pixels and the slider should range between the values 3 and 100. When publishing your applet, select the window size of 600 by 600 pixels. Within a graphics or slider program, the methods `getWidth()` and `getHeight()` can be used to determine the width and height of the graphing window in pixels.


slider = 6

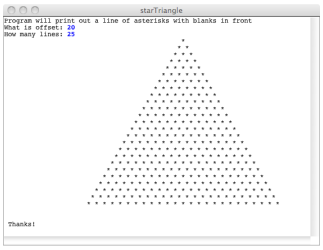

slider = 21


slider = 50


slider = 100

Begin by determining the endpoints for the vertical line along the left side of the graphics window (utilize `getWidth()` and `getHeight()`), as well as the `deltaX` and `deltaY` values, which will step the endpoints down the left–hand side and across the bottom of the window. Once these values are known, begin a loop which will create a `GLine` in the initial position (along left side of the window), then update the endpoints for the next line to be drawn. The last line segment drawn should be the horizontal line along the bottom of the window.

5. (Group Project, Triangle of Stars) Write a console program which asks the user for an offset (number of blanks in left margin of output) and for the number of rows in the triangle of stars which is desired. Then display a triangle (with offset margin) which is centered, begins with a single asterisk, the second line will have 2 asterisks, the third line will have 3, and so on. Each asterisk in a row is separated from the next by a single blank.



Wrap-up

When you have completed the lab:

1. Publish each project to your web site & submit lab 4electronically
2. Be sure I've checked your Worksheet #4 before leaving lab
3. Staple together the program printouts from this lab, in the order assigned, and hand in at the beginning of Lab 5.

[1] `www.deimel.org/rec_math/circular_3.htm`