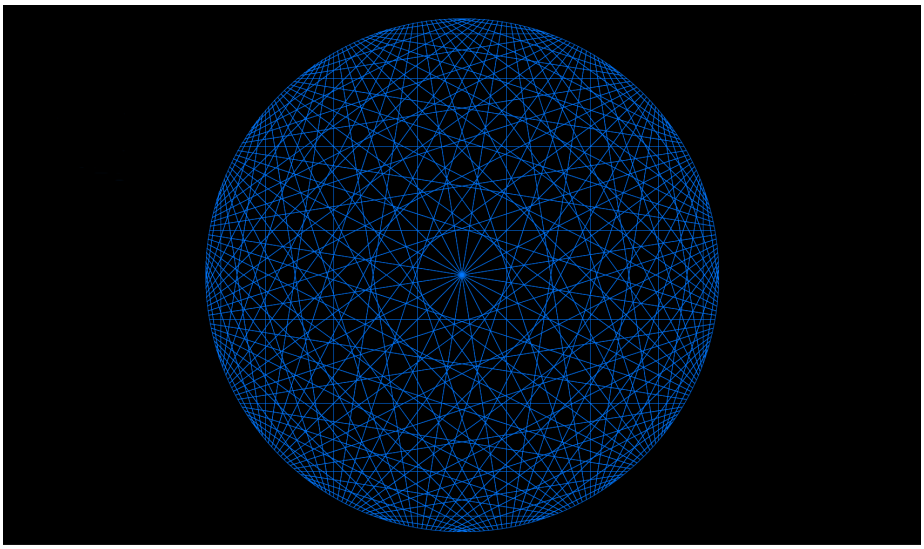
CS 351
Project 1
Modulo Times Tables Visualization
Due 9-3-23

Joseph Haugh



In this warmup project you will be making a mathematical visualization to make sure you have your development environment setup. This project is meant to teach you the basics of JavaFX through the GUI, your IDE/text editor by simply completing the project and git because you will need to keep track of your code through version control.

Project Description

This project is based on this: `https://www.youtube.com/watch?v=qhbuKbxJsk8&vl=en` Mathologer video. This video explains and illustrates the strange pictures that are produced by visualizing the times tables as a circle. The core of the project is producing what the video starts to show at 7:30. Namely incrementing shown times table by a given increment.

Project Requirements

- Handle times tables of numbers up to 360
- Handle up to 360 points around the circle
- Button to pause and start the visualization at will
- Control to change the increment size of each time step
- Control to change how fast the visualization runs (frames per second)
- Control to jump to any given times table number and number of points
- Ability to cycle between at least 10 different colors as visualization runs

Setting Up Your Project

These instructions assume you have already installed the JDK, IntelliJ, and git on your working machine. (The CS machines already have git installed, so you can just use it immediately.)

1. Create a new project on lobogit.unm.edu
   (a) Plus sign menu at top of page → New project
   (b) Give project a name
   (c) Leave the visibility level private
   (d) Create project
2. You now have an empty project on the server. Notice the command line instructions on the project page. These instructions will disappear once you add a file, so do not add a file directly in GitLab at this point.
3. Now set up a git repository on your working machine
   (a) Open a terminal/shell
   (b) If this the first time setting up git on your working machine, follow the "git globalsetup" instructions to set your name and email. (You will need to do this on every machine you work on, but once it is set up, you don't need to do it again for your next project.)
   (c) Follow the instructions for "Create a new repository" to create a new local repository with a README.md file and and push the update to the server. The "touch README.md" command creates an empty README.md file in your working directory. I suggest you just open README.md in a text editor and actually add some content now instead of starting with an empty file here. You can then proceed with "git add README.md" and the rest of the instructions as given.
4. Set up an IntelliJ project in the local directory.
   (a) Open IntelliJ.
   (b) Create New Project.
   (c) Leave selection as "Java" for which sort of project.
   (d) Select Java 17 for your project SDK. If you don't see the appropriate option in the drop down box, select "New" and browse to where the JDK is installed on your system.
   (e) Next, don't create project from template, Next
   (f) For project location, select your git repository directory. This will also update the project name , but you can change them separately if you want to for some reason.
   (g) Finish.

(h) You now have an empty IntelliJ project with some automatically generated configuration files. Do not add them to your git repository.

5. Set up a .gitignore file to make sure you don't accidentally add configuration and/or class files to the repository.

   (a) Create a file named.gitignore in the top level of your project directory. You can do this with any text editor, but since you already have IntelliJ open, you might as well use that. In the Project tool window, right-click on the top level of the project → New → File
   The file name must be .gitignore(note the dot at the beginning). Go ahead and add this new file to git. (or you can add it manually later when you commit)

   (b) Add files and directories to the .gitignore file.
   out
   .idea
   myproject.iml

   (c) Commit this new file. Ctrl-K will bring up the commit dialog. Make sure .gitignore is selected. Add a commit message. Click "Commit" or drop down to "Commit and Push" in one step.

6. Work on your project, putting source code in the src directory. When you create a new file, you may choose to add it to git at that point. This doesn't commit the file yet, so don't forget that step once you have done enough work to commit the next chunk of your project.

7. Make sure you push to the server when you are done with your current work session. (Or push every few commits, depending on how you prefer to work.)

Submitting Your Project

Submit the link to your Lobo git project to UNM canvas before the deadline. Make sure your project follows my submission guidelines as given on the course website.