Final Project Proposal

Paul Plew

April 2, 2021

Summary

For the final project I have decided to make a program that generates mazes procedurally. There will be two different ways to use the program. The first being a visual demonstration of a maze generating algorithm where the user can choose the number of cells on the width and the height. The program will then generate a maze using one of a few algorithms. Right now I have decided to demonstrate maze generation using the `depth-first`, and the `Kruskal's` algorithms. The second way the user can use this program is to create printable mazes that can be solved by hand. My main goal in this project is to make the program as memory and CPU efficient as possible, so it can be run on potato browsers.

I am also planning to add a maze solving mode that implements an AI that can solve mazes. Like the generation of Mazes I would like to demonstrate two different algorighms. One such algorithm is the `Wall follower` algorithm that will stick on either the right or left wall and turn when that wall turns. A second algorim candidate is `Trémaux's algorithm` which has a few rules. In this algorithm the paths are marked once each time they are passed, and if the path has been marked twice that is not a solution and the solver will move on.

Plans

Plan 1: Murphy's Law Plan

April 9: Create Canvas and Start page with buttons
April 16: Start depth-first maze generator
April 23: Finish depth-first maze generator
April 28: Start Kruskal's maze generator
April 30: Finish Kruskal's maze generator

Plan 2: Blue Sky Plan

April 9: Finish Start page and depth-first generator
April 16: Finish Kruskal's maze generator
April 23: Make Printable Maze page & Playable Maze page
April 25: Finish Maze solving AI
April 30: Finalize, fix bugs, Add color selection and size selection

Software Specification

Technical Information

This program will use a cell class to contain a cell with walls. The walls will be removed depending on how the algorithm progresses through them. Each maze will be an array of these cells behind the scenes and as the mazes are generated the array of cells will be updated, and iterated over multiple times.

The program will have two different 'modes' where the user can either watch mazes generate over and over until they find one they want to print, or a mode where the user can solve an algorithmically generated maze (either in the application itself or printed out and solve it by hand).

The javascript will be written using the `p5.js` libraries and will contain two top classes. The first class will represent cells and the second will represents the world. The world class will be extended by multiple subclasses, each representing a state of the world. These classes will be named like: `start generate`, `play`, and `print`. The draw function will then ask the world object what the current state is and will show the appropriate state

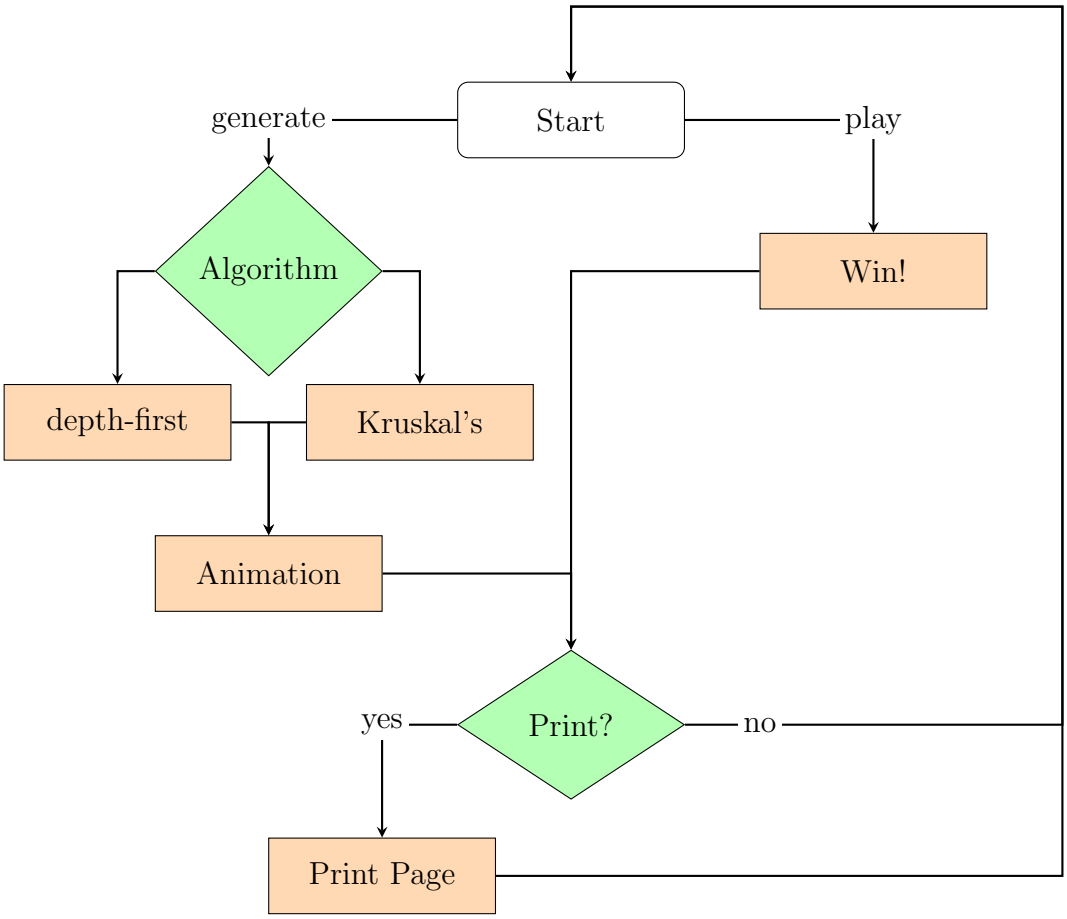
User Story Pseudo Code

`user`: "As someone who LOVES mazes I want to learn more about how computers generate mazes and I want to play some at the same time! Oh how I wish I had a way to do that. Every day I go without playing mazes and learning I get more grumpy and my family is starting to complain :("

```
Enter Maze Generator
```

`user`: "OMG!! All of my problems are solved and I am no longer grumpy I love seeing these maze generation algorithms and the mazes they generate are so fun to solve. I can even print them out and my kids can solve them. This is truly fun for the whole family."

Flow Chart Pseudo Code



Component	Excellent (9-10)	Competent (4-8)	Not yet Competent (0-3)
Objectives	All major and minor objectives are met including those that are not required.	All major and minor objectives are met, but not exceeded.	Major and/or minor objectives are not met.
Design	Application looks clean and polished. The student obviously put a lot of time and effort into how it is presented.	The application is clean, but not as polished as it could be. The student put some time and effort into how it is presented.	There are obvious visual elements missing the student had trouble creating and placing different elements.
Programming	Student shows advanced knowledge of programming principles such as: Scope, Classes, Inheritance, etc.	Student shows knowledge of programming principles but misses a few.	Student shows lack of understanding regarding some programming principles.
Comments/Readability	Student put effort into making their code readable and added comments for readers to better understand what is occurring.	Student put some effort into comments and code readability, but there could be some improvement	Student missed out on many opportunities to explain aspects of the application.