**Pathogen**

Genetic Pathfinding

**Team**

Ben Scuron

Ryan Hodge

Kai Iwugo

Nick Coffin

**Repository URL** <https://github.com/cis3296f22/Pathogen>

**Project Board URL** <https://github.com/orgs/cis3296f22/projects/112>

Table of Contents

[Project Proposal 3](#_Toc118244195)

[Project Abstract 3](#_Toc118244196)

[High Level Requirement 3](#_Toc118244197)

[Conceptual Design 3](#_Toc118244198)

[Proof of Concept 3](#_Toc118244199)

[Background 4](#_Toc118244200)

[Required Resources 4](#_Toc118244201)

[Project Design 5](#_Toc118244202)

[Vision 5](#_Toc118244203)

[Persona Adam, Computer Science Student 5](#_Toc118244204)

[Persona Jessie, Software Engineer Manager 5](#_Toc118244205)

[Class Diagram (11/01/2022) 7](#_Toc118244206)

[Project Progress 8](#_Toc118244207)

[Week 2 Progress 8](#_Toc118244208)

## Project Proposal

### Project Abstract

This document proposes a browser-based application of a genetic algorithm that has the capability to path find through a maze. There will be a population of agents whose job will be to path find through the maze. There will be different options for maze generation, such as: maze generation algorithms, maps, and user-drawn mazes. The agents will learn to move through the maze over the course of multiple generations. Each generation will go through an evolutionary process: selection, mating, crossover, and mutation. Over time, a subsequent population will converge to the optimal path. An explanation of the algorithm, describing how it works will be present in the application to serve as a learning resource for the user.

### High Level Requirement

Pathogen visualize agents converging on an optimal path through a maze via a genetic algorithm. Users begin the simulation by setting configuration parameters–dimensions of the maze, the mutation rate of the agents, and other parameters, and then observe the results.



### Conceptual Design

The application will be developed using HTML5, CSS, TypeScript, and React. In addition to this vanilla software, it will be using the p5.js JavaScript graphics library. The p5.js library exposes an application programming interface (API) that makes drawing to an HTML5 canvas trivial. The application will be run in the browser and will be supported by any web browser that supports HTML5 canvas. The bulk of the code will be completed in TypeScript, a strongly typed version of JavaScript. Object-oriented programming in TypeScript will be utilized to develop the application. HTML will be used for the main structure of the website and CSS will be utilized to make the appearance of the application more streamlined.

### Proof of Concept

Link to repository: <https://github.com/bscuron/Life>

Ben Scuron implemented Conway's Game of Life in HTML, CSS, and JavaScript. Also used the same p5.js JavaScript library that is being proposed to use in this project. The link to the repository points to the code itself. In the README.md file there is a link that says, 'Click here!'. If you click on that link, you will see a live demo. There is no compilation needed since the programming language used was JavaScript, which is an interpreted language.

### Background

The browser-based application will be a visualization of a path finding genetic algorithm. Agents will attempt to 'learn' to find the optimal path throughout multiple generations. After all agents in a generation die, a new generation is spawned. Before spawning the new generation, the best agents are selected from the previous generation, replicating the survival of the fittest. To calculate which agents are the best, a fitness function is used to determine how 'desirable' each agent is in a process called 'selection'. The fitness function is entirely dependent on the use-case of the genetic algorithm. Since the application will use a genetic algorithm to find the optimal path from one point to another, the fitness function will consider the amount of time it took for the agent to arrive at the destination or if it even did, the length of the path from start to end, and other factors that are pertinent to the use case. Agents that have a higher fitness function are more likely to have an offspring in the next generation. To generate the next generation, two parents are drawn from the mating pool of agents. Higher fitness agents from the last generation have a better chance of being chosen. Once two parents are chosen, their 'DNA' are partitioned to create a child for the new generation. The partitioning of two parents' 'DNA' to pass onto the child is called crossover. This process of two parents being chosen from the previous generation, partitioning their 'DNA' and creating a child is repeated until the new population size is equal to the old population size. One important process that happens when a child is created is 'mutation'. This is a key part in the evolutionary process that allows for change in the population. After each child receives its 'DNA' from its parents, each gene in the 'DNA' has a random probability of 'mutating'. For instance, if the 'DNA' of the child is 00101, there is a chance that gene 1 could flip to a 1. Now, the child's 'DNA' would be 1001. The processes of 'selection', 'mating', 'crossover', and 'mutation' make up the evolutionary process that allows for the population to converge to the desired path. An example of a genetic algorithm would be 'Smart Rockets' by the Coding Train. In his demo, he uses a genetic algorithm for rockets to reach a target, while avoiding obstacles. The Coding Train's example is different than mine in that the user is not able to draw in his example, and the time it takes an agent to find the target is not considered. This can lead to a non-optimal solution to reaching the target. In this specific way, my fitness function would be almost entirely different. The Coding Train produced a YouTube video documenting his creation of this project: <https://www.youtube.com/watch?v=bGz7mv2vD6g> . In addition to the video, the Coding Train uploaded his source code to GitHub: <https://github.com/CodingTrain/Coding-Challenges/tree/main/029_SmartRockets> . I do not intend on using the source code at all for this project, as I would like to implement my own solution that can hopefully perform in a more optimal manner.

### Required Resources

The only resources that will be needed to acquire for the project is a deeper understanding of genetic algorithms. The description I gave in the background section of this document provides a good level of detail, however there is still more to learn. For example, there are multiple ways to perform the 'crossover' process. One way would be to split the parents' 'DNA' at the mid-point and distribute the two halves to the child agent. Another way would be to select a random point in the parent's 'DNA' and partition there. I would need to research which 'crossover' method leads to the most optimal results the fastest. As for hardware and software requirements for this project, everything can be run in a browser. Having the p5.js JavaScript library downloaded with a browser that supports HTML5 canvas is all that is needed to complete this project. This is another impressive feature of genetic algorithms. They do not require data to start learning. The downside of genetic algorithms is that they can often be slow. However, for my use case a genetic algorithm will be plenty fast.

## Project Design

### Vision

FOR teachers and students WHO want a visual representation of how genetics evolve and reward positive outcomes over the course of multiple generations. Pathogen is a pathfinding algorithm THAT trains a population of agents to make their way through a maze without colliding with its walls. The most successful agents will reproduce to create their offspring. The newly created child is then randomly mutated. The evolutionary process of: 'selection', 'mating', 'crossover', and 'mutation' demonstrates how species evolve over time.

UNLIKE other pathfinding algorithms that are used in a commercial setting such as: video game algorithms, package delivery system algorithms, mail delivery algorithms, OUR product allows teachers to show students how agents with genes that produce successful behavior reproduce with other successful agents and how random mutation helps the population overcome hurdles over multiple generations. The focus of the other commercial application is entertainment and profit.

### Persona Adam, Computer Science Student

Adam, age 21, is a senior computer science student at Drexel University. He has one sister, two brothers, and a golden retriever. In his free time, he loves going on long walks down the Schuylkill River with dog. After college, Adam wants to work at Amazon helping them find the best routes for their vans to deliver packages. Adam is interested in all things artificial intelligence and is eager to learn as much as he can. He came across a YouTube video explaining the idea of genetic algorithms. In the video, the concept of genetic algorithms was explained, but Adam had trouble following along. Adam learns best when being able to play around with different parameters to see how they may affect the larger system. He might use Pathogen to gain a deeper understanding of how genetic algorithms work, how variables/parameters effect the learning rate of the agents, and how natural selection can be simulated algorithmically.

### Persona Jessie, Software Engineer Manager

Jessie, age 28, works as a software engineer manager at a growing package delivery company in Miami. She loves crocheting and watching old movies with her roommate after work, and during the weekend she spends most of her time working on unfinished personal projects. Jessie dreams of moving up the ladder at work and securing a higher paid managerial position. Jessie has an idea to implement a new algorithm at work which will determine the fastest routes to take when delivering packages but has very little background with machine learning which would be used in her design. To learn more, she starts to research machine learning and comes across path-finding algorithms. She can understand the concept, but still does not know how it could be applied to her specific algorithm. Jessie may use Pathogen for a proof of concept of path-finding algorithms. It may help her understand what machine learning can provide to pathfinding, how this can be compared to her own algorithm, and what initial steps to take to provide a starting point for the algorithm.

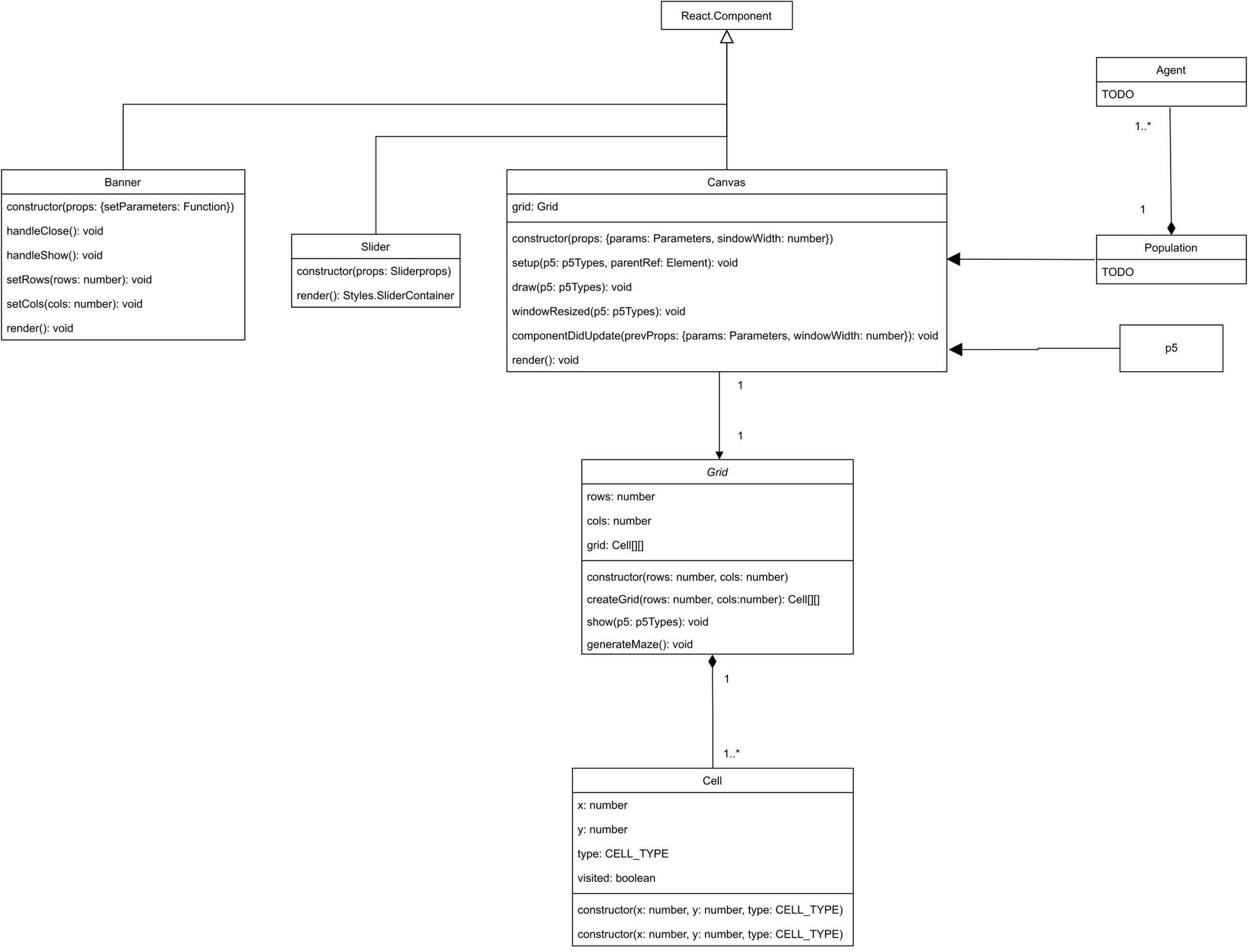
**Persona Mrs. Eastbrooke, Middle School Science Teacher**

Mrs. Eastbrooke, age 62, is a middle school science teacher at a school just outside of Philadelphia. Mrs. Eastbrooke has two daughters and enjoys taking them on educational field trips. She spent much of her early career as a biologist in the field, but as she approaches retirement has moved to a role in the classroom. Mrs. Eastbrooke’s background provides her with an excellent understanding of the topic she teaches, but in her older age she often struggles to find modern tools to teach her lessons. She recently encountered this issue when teaching a unit on evolution. She wanted to provide her students with a visual representation of how a population might converge on similar, desirable traits over generations. Mrs. Eastbrooke found the Pathogen webpage and believes the simulation it provides will be a great visual example of how a species might evolve over time. She will be incorporating it into her future lessons.

**Persona Jack, Graduate Neurobiology Student**

Jack, age 23, is a grad student at Harvard University studying neurobiology. He was born and raised in Philadelphia where both his parents were high school teachers, so he feels like educating others is in his blood. After getting his bachelor’s degree in biology from Temple University he decided to take his education to the next level. He’s had multiple internships, and has amassed tons of knowledge at this point, so he wants to share it with others just like his parents did with him. His outlet for this is tutoring undergraduate students at Harvard, specifically in evolutionary biology. Jack is by nature a visual learner and believes that it’s easier to learn something when you have a visual representation of the concepts. Normally, biology applications that demonstrate evolutionary steps take time to learn and aren’t beginner friendly. That’s why Jack prefers to use Pathogen, because the user only needs to learn a few buttons, and it demonstrates the concept of survival of the fittest and evolution. This is what Jack uses to teach as a tutor and lets students play around with it in a fun, interactive teaching environment.

### Class Diagram (11/01/2022)



We’re using a JavaScript library called React to create this application. The three components which the customer will see and use are Canvas, Banner, and Slider. They’re all components of the React library, which is why they all extend the React.Component class, hence the clear arrows with the solid line. Canvas is where the action takes place and where the cells and agents are located which ultimately make up the grid and population respectively. The canvas contains a grid object which will perform the functions that keep track of the rows and columns of each cell. This is shown with the solid arrow pointing from Canvas to Grid. The Grid class contains a two-dimensional array of cells, which makes up Grid. Since Grid is entirely made up of cells, we use a one-to-many association to represent that relationship. The Cell class keeps track of the individual cell’s coordinates, and which type of cell it is; whether it’s an agent, maze wall, or empty. The Canvas class also has a population object, which is the population of agents that will evolve to solve the maze. The population is entirely made up of agents, which is why we use a one-to-many relationship.

## Project Progress

### Week 2 Progress

**Sprint Goal:** Foundation of maze generation and UI: Generate maze grid, generate maze using depth-first search (DFS), create UI to communicate with maze (change number of rows and cols of maze).

**Sprint 1 Task List**

* User visible features worked on during sprint 1

|  |  |  |
| --- | --- | --- |
| Tasks in Sprint | Task Status at end of Sprint | Assigned To |
| Create Maze Grid | Complete | Scuron |
| Generate Maze | Complete | Scuron |
| Generate Start Node | Complete | Scuron |
| Setup communication interface between p5 classes and UI | Complete | Hodge |
| Settings Menu | Complete | Hodge |
| Diagrams | Complete | Coffin |
| Documentation | Complete | Iwugo |

**Sprint 2 Backlog**

* User visible features to be worked on during sprint 2

|  |  |  |
| --- | --- | --- |
| Tasks in Sprint | Task Status at end of Sprint | Assigned To |
| FEATURE - Change mutation rate |  |  |
| FEATURE - Generate population |  |  |
| Evolution process |  |  |
| FEATURE - Skip learning/evolution process |  |  |
| Agent movement based on genes |  |  |
| Agent dies |  |  |
| Initiate evolution process |  |  |
| Generate finish/end node |  |  |
| Create agents |  |  |
| Create population of agents |  |  |
| Add settings option to control population size |  |  |