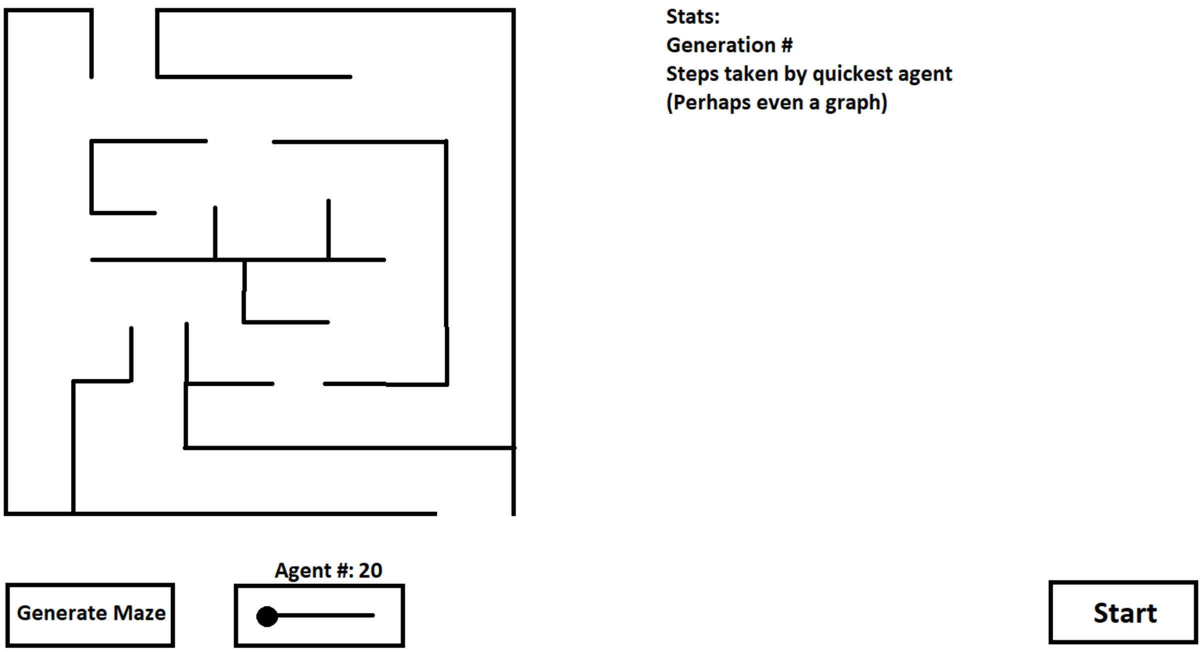
Teddy, Jason, Matthew, John
Professor Jenkins
CSC 470
6 November 2022

High and Low-Level Design

High Level Design:

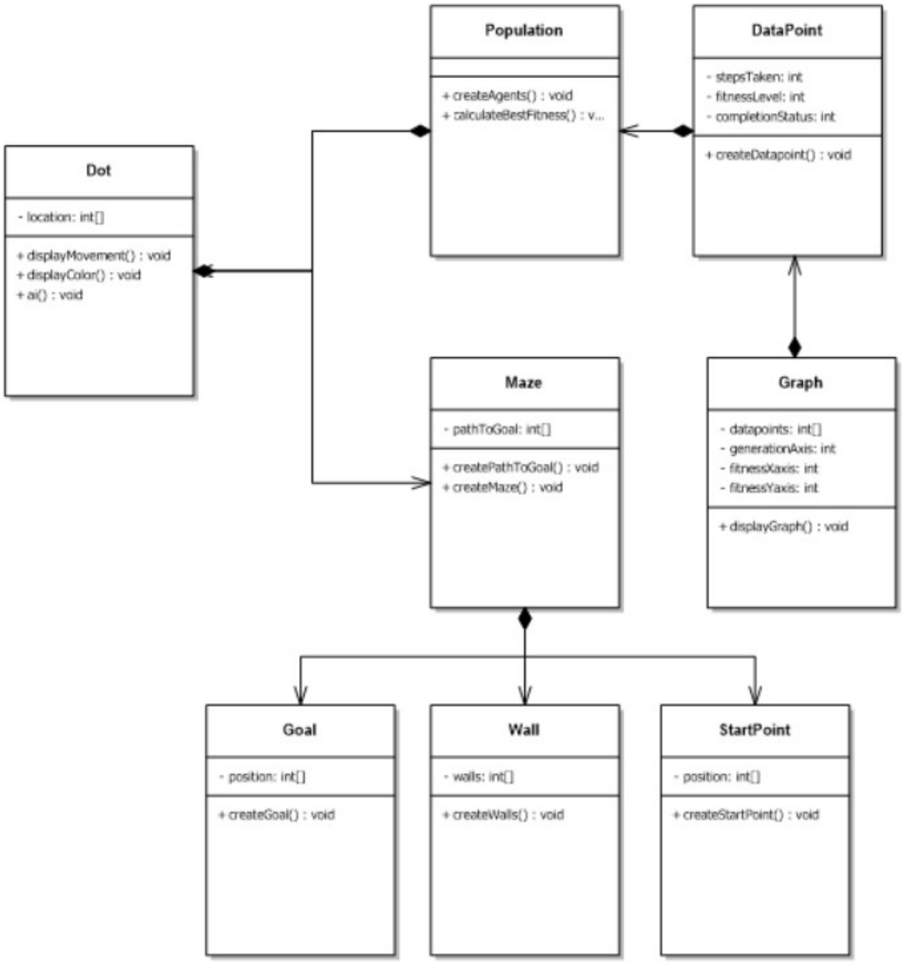
- Description of architecture chosen
    - We will be using a component-based architecture
- How the major pieces of the program will fit together
    - Our program will be an object-oriented approach in python. There will be a maze that is generated, then dots created after that with an AI that learns between generations.
- Describe hardware and software the program will run on
    - The only real requirement for this software is that the hardware is able to run the Python language.
    - This will be running primarily on home computers or laptops
- Security - No security will be required since this project does not have any sensitive data and does not handle any sensitive data
- Reports about program - this will generate a graph about the performance of the ai in the maze and display that to the user
- User Interface - will feature buttons to generate a new maze and start the AI navigation, a slider to choose the number of dots moving in the maze, and will display the maze with moving agents along with statistics



Teddy, Jason, Matthew, John
Professor Jenkins
CSC 470
6 November 2022

Low Level Design:

- No need for database or use of 1NF, 2NF, 3NF, or Normalization
    - This is due to our program not requiring the use of a database in general.
- Describe reports
    - Our program will have live statistics that are updated as the program runs off to the side of the maze. A few ideas for statistics include, but are not limited to: Generation number, steps taken by the quickest agent, and a graph that plots how many steps it takes per generation in total.
- UML diagram



Teddy, Jason, Matthew, John
Professor Jenkins
CSC 470
6 November 2022

Below is what each class of the UML diagram will need to accomplish

- Dot
    - Display movement of the dots
    - Display color to follow different dots
    - Keep track of the location of the dots
    - AI
- Population
    - Using agent number UI slider, chooses amount of agents to generate
    - Which generated dots are staying based on fitness
- Maze
    - Walls - all the walls in the maze
    - Path to goal - the path to the goal from the start that is used to generate the maze
    - Goal - the endpoint of the maze
    - Startpoint - the startpoint of the maze
    - Generate maze - the method that generates the maze based on some given parameters like the startpoint and endpoint position
    - Create wall - method that is part of maze generation, creates walls with 2 anchor points
    - Display maze - method that colors the grid tiles for the goal and startpoint, puts lines between the anchor points of walls
- Wall
    - Anchorpoint 1 - one end of the wall
    - Anchorpoint 2 - the other end of the wall
- Goal
    - Position - a grid square that wins the maze if it is entered
- Startpoint
    - Position - the grid square that every dot will enter the maze from
- Graph
    - Datapoints - all the datapoints to be used on the graph
    - Generation axis maximum - the x axis maximum value
    - Fitness level maximum - the y axis maximum value
    - Fitness level minimum - the y axis minimum value
    - Display graph - function that displays the graph on the screen, places the axis and datapoints and colors the datapoints based on their maze completion status
- Datapoint

- Fitness level - the average fitness level of the ai when they died/finished
- Steps taken
- Maze completion status - used to show on the graph what amount of ai finished the maze in this generation
- Create datapoint - function used to input the values of the datapoint once the ai finishes the maze