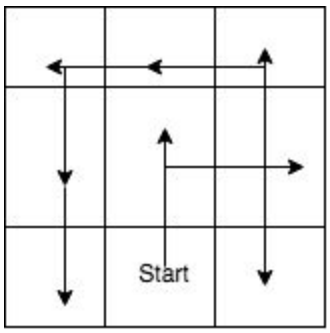
Name: Thiago Santos
Class: AI
Final Project Report - Maze Game

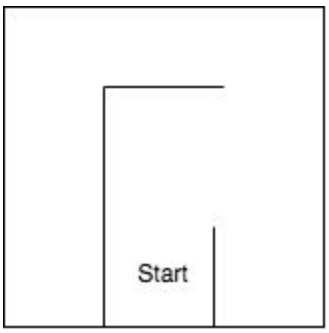
In this project, the goal was to solve 3 problems, related with a Maze Game. In order to solve both, a variety of AI search algorithms and Logic was used. The main 3 problems was:

1) **Generate a MAZE, with multiple paths from any point X to another point Y.**

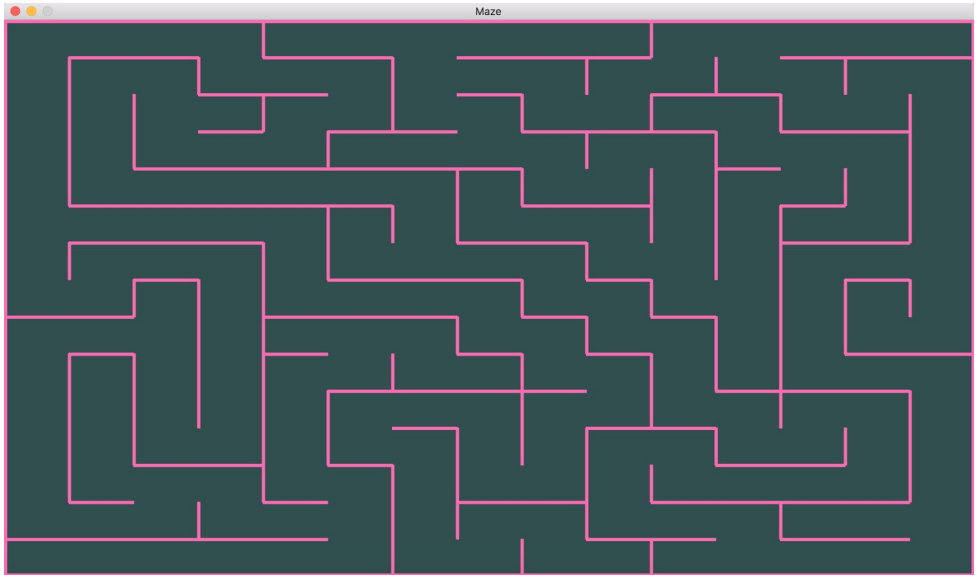
At first, I decided to generate the MAZE by running DFS. Since DFS keeps going on the depth of a node, instead of looking to all nodes at Depth i, it seems a good idea to use it to generate the MAZE. The idea was to choose a random node to go next, and the direction you choose to go to that random node, you then remove the wall. See the example below:



Remove the wall that the arrow went

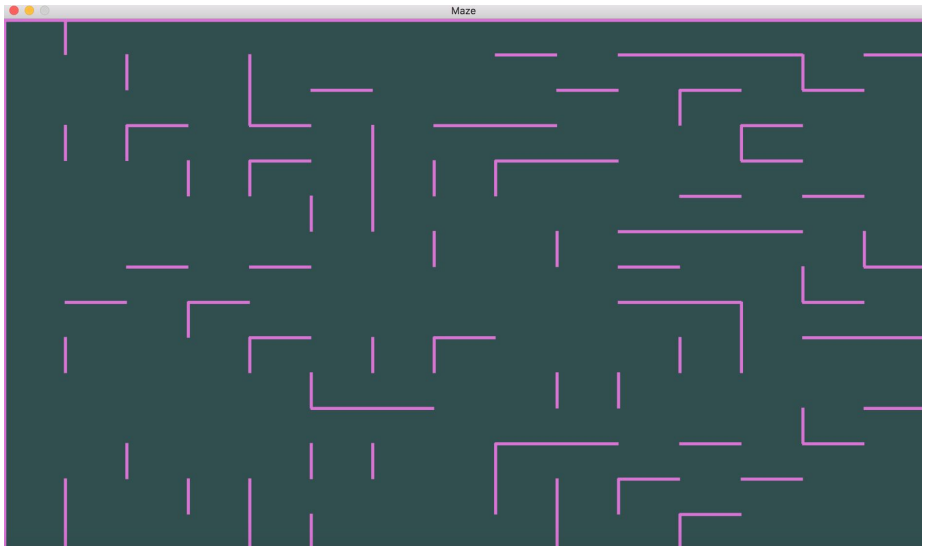


See an example from the game:



The problem with this approach is that since we are running DFS(BFS would do the same) we are calculating a path from a node to another. Thus, there's 1 and only 1 path from node X to node Y. This became a problem later, because the agent's goal is to avoid the wall and at same time avoid ghost. However, if there's only one path, he cannot avoid the ghost, he would be in a dead end, every time.

How to solve this problem?
The idea to solve was to randomly remove some of the walls. Each tile holds its own walls, and the maxim of wall a tile can have is 3 and the minim is 0. I go then through all tiles and choose just those one with more than 1 wall. Then, I randomly choose to remove one of the walls or to keep them. This approach made the maze way more opened, having multiple paths from X to Y. See the result below



It became a much easier maze, but for the game proposal it's actually a very good one.

2) **Robot/Agent has to find the shortest path from his position to a goal (Fruit), but has to avoid the blocked walls and the ghosts at the same time**

To find the shortest path to a goal, avoiding walls and ghosts, robot was implemented in A* algorithm. I also tried DFS and BFS, but they were a little slower when executed with a maze of size > 50. The idea here is simple, instead of keeping running A* before every mode, what cause the game to become slow(low efficient), the robot calculate the path once, and then start moving. However, before move, he checks his new 2 targets, and double check if there's any ghost surrounding those 2 goal targets. If there is, he calculate a new path, avoiding those 2 tiles and of course the wall and any target that currently have a ghost. Otherwise, he just move. This idea still not the best approach, but it did quite worked well, and saved a lot of calculation.

3) **Ghosts/Monsters has to find a path from his position to the Robot. May be the shortest one or not.**

The ghost's goal is to catch the robot. In order to do that, they were implemented in 2 algorithms, DFS and DumbDFS. By DFS, the ghost finds the shortest path to robot, and keep moving, trying to catch him. However, most of the ghost are implemented in DumbDFS, that means random walk through the make. However, it's not actually 100% random, since it uses the idea of DFS. The ghost walks random, but around the Robot, keeping close. This make the world a little more challenging for the robot.

How to play the game

Some rules of the game:

- Robot's goal is to collect the bonus points(fruits) and to avoid walls and the ghosts
- Robot and Ghosts cannot move if there is no bonus fruits on screen - This is to make game mode better to play and to visualize
- Robot has health, like human, not lives. In another words, when caught or pass through an ghost, it loses some health. As longer you stay in touch with ghost, more damage you get. When health gets to zero, you are dead
- Each time the robot collects a fruit, it gets 50 points.

There are two ways of playing the game

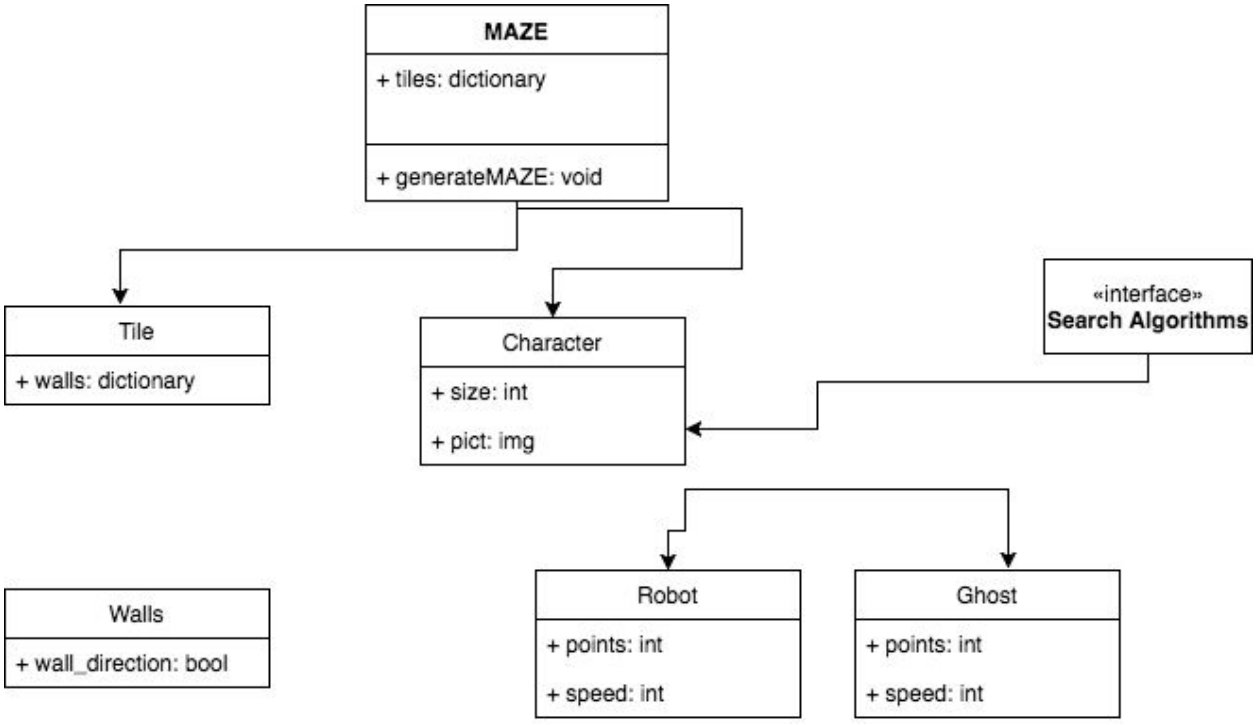
- AI x AI → Computer plays both modes, as robot and as ghosts. This is a very game to watch. Sometimes the robot gets confuse, but most of the times it can avoid the ghosts and get the fruit by following the shortest path
- Player x AI → A human can play as the robot, and can use the arrows to move the robot around. In order to play this mode, you must type "Y" or "Yes" for the question prompted " Do you wanna to play as the robot(Y/N)"

Other features

- You can press G, if you wanna change the maze configuration
- You can press Space, at any time, to change the color of the maze, for what better fits your needs.

Object-Oriented Orientation

The game is well structured and designed, with object-oriented orientation. See the UML below:



The idea is that The maze has tiles and some characters. Each Tile has its own walls. A character implements a variety of search algorithms, and each Robot and Ghost are a character. This idea made the game very simple to handle changes or update something.

How to execute the game

The game is implemented in Python 3. The graphic of the game is implemented using Pygame Library. Thus, you need both in your computer. They are already installed in the CS server, however, when login in the server, you must login with graphic privilegios. To do that, you must login by **ssh -XY username@cs.indstate.edu**

Then, you just have to go inside of the folder Classes, and execute the class main.py by running: **python3 main.py**

It will ask you in the command line the size you wanna for your maze. Just type, and have fun playing the game. See and example below.

REFERENCES

http://www.astrolog.org/labyrnth/algrithm.htm
https://puzzling.stackexchange.com/questions/38/how-can-i-generate-mazes-that-often-have-multiple-forks-or-choices