Snake with AI

Group Members

Adrial Armijo

Douglas Galm

Adrian Salazar

Chris Villanueva

Class

CPSC 481-03

T/TH 5:30-6:45

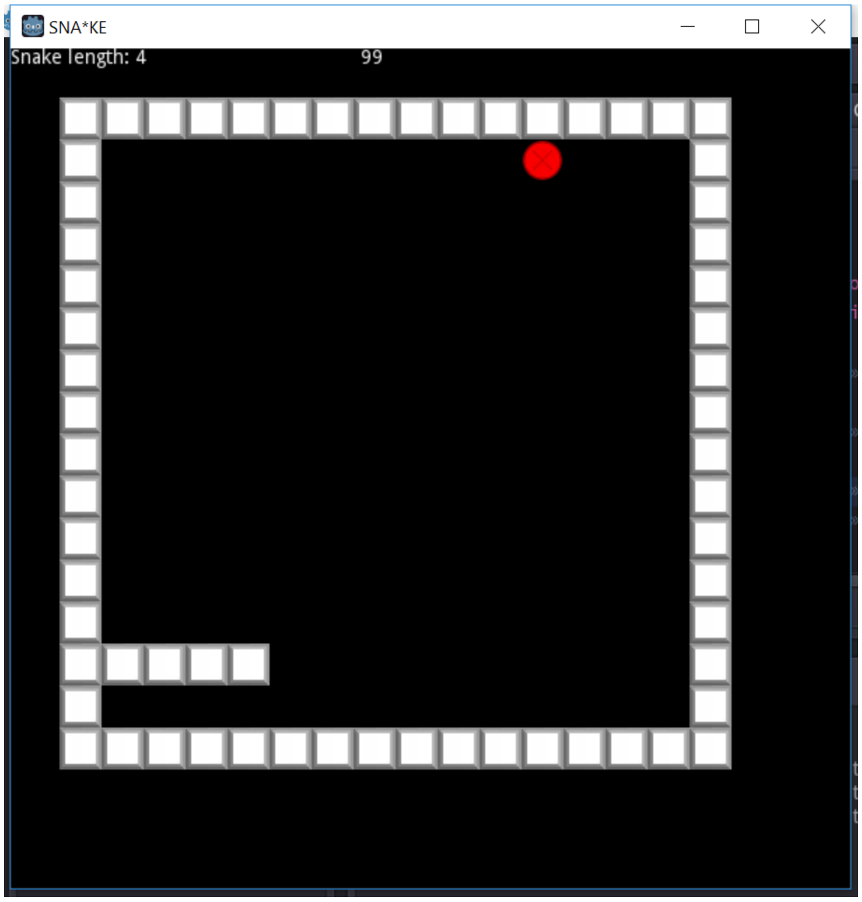
Professor

Wenlin Han

Overview

This is our final working game of snake with AI implementation. The game field is a 14x14 cell board, and a player starts with controlling one block, which is termed the “snake”. There are boundaries on the edges of the game and an item the size of one block is placed randomly on the board. The objective is for the player to move the snake in binary directions to collect the other item on the board. Once the item is collected, the snake will increase in size by one block, then another item is placed randomly on the board. The game continues with the same process of the snake trying to acquire the block and increasing its size. The size increase will make it more difficult because the snake cannot cross over its own body or collide with the boundaries. The game is over when the snake either collides with its own body or the boundaries on the edge of the board.

Snake Algorithm Analysis



Algorithm Implementation

Our algorithm that we implemented on our player (snake) is to find the shortest path to the object by:

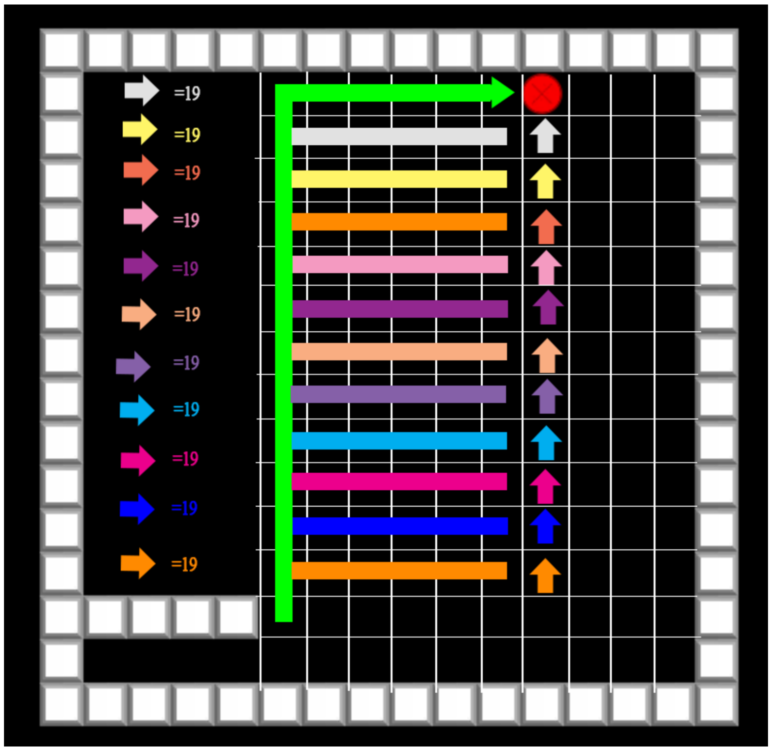
1. Determine if the object is above or below, and move up or down depending
2. Then once it intercepts the objects X axis, it determines if the object is on the left or right
3. Then it heads towards the determined direction and obtains the object

Ideally, the best algorithm would find the shortest path between the source (snake head) and the destination (food) and use this shortest path to consume the food and increase the player's score. In our implementation we use compared distances in relation to the snakes current location while constantly updating the parameters of the algorithm as time goes by. There is no time constraint for the snake to get to the food, however, we would like this time to be as short as possible to increase efficiency.

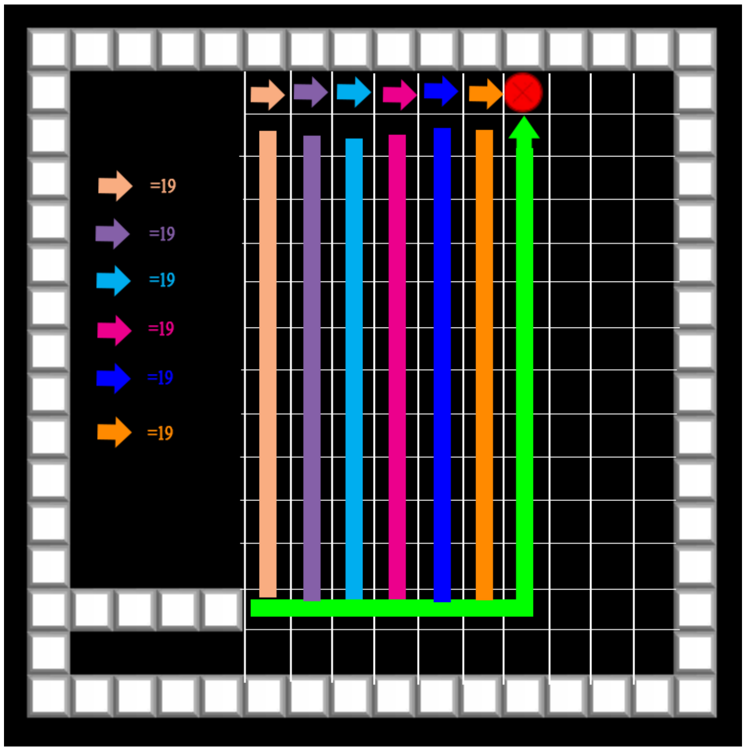
Algorithm Explanation

You can see the algorithm being implemented on the next image, highlighted in green. However, we still wanted to explore other options, but we found that the **binary** paths will give the same value. In other words, the other **direct** routes would be the same distance.

We explore other options, by initially having the object go up first (towards the object), then take a different path other than the green. As shown, the paths all take 19 steps to get to the object. The green path takes 19 steps and is the implementation of our main algorithm.



The next diagram shows an algorithm, but the steps switched. The player (snake) determines if the object is on the left or right, then it determines if it's up or down. However, as shown, even with this other implementation, the results are the same as the previous implementation.



Deciding on an Algorithm

Therefore, we have determined that while A* is a more complex solution to the problem, the binary solution works just as well. We have decided to implement the binary solution as it runs at a much faster time than the A* implementation. Our program still uses a form of AI in determining the fastest route to from the source (snake head) to the destination (food).

Future Improvements

There are a couple of updates that we could make to the program to make it run better. First of all, some code optimizations would increase the frame rate of the game to make it feel smoother. This would also help to reduce the amount of input lag that is experienced by the player. Another improvement would be to create more dynamic algorithms for handling food generation and optimize to reduce space and time complexity. Lastly, we would like to add different animations and improve upon the existing UI.