Jiunn Siow

Project 3 Report

Board

1. I used the following function members for the board class
   1. const Game& m\_game; //this is the reference of the game object
   2. int\* m\_nShips; //the number of ships that are placed on the board
   3. int hit; //how many hits a ship has sustained
   4. char\*\* board; //two dimensional array that reflects the board
2. The 2 dimensional array reflects each movie a players makes per row and column. The hits record the number of times a ship is hit, and when that number reaches the length of the ship, the ship is then removed from the board.

Game

1. I used the following function members for the game class
   1. int m\_nRows;
   2. int m\_nCols;
   3. int m\_nShips;
   4. vector<Ship> ships;
   5. struct Ship

{

int m\_length;

string m\_name;

char m\_symbol;

};

1. A vector of ships allow us to store the ship in an array, and then add and remove it whenever we wish more easily. This allows us to also keep track of the number of ships on the board.

Player

1. I used the following function members for the GoodPlayer class
   1. char\*\* attacked; //two dimensional array that reflects the board
   2. int dir; //NSWE direction
   3. int step; //acts as the step for AI to choose direction
   4. int unitNumbers; //distance from the original points
   5. vector<int> shipPoints; //records all ship lengths
   6. vector<Point> centers; //records all center points
   7. vector<Point> hit; //records the hit points
2. The GoodPlayer acts is two states
   1. In state 1, the goodplayer hits randomly on the board in order to first determine where it will proceed next in its attack.
   2. In state two, the goodplayer decides which direction it will take so that it may reflect the best chance to sink an opponent’s ship. If the goodplayer hits the ship of an opponent, it will proceed to discover the trend in which the ship’s alignment with the board may be.

Pseudocode

bool MediocrePlayer::placeShipsHelp(Board& b, int numShips)

{

If numships < 0

Return false

If numships is equalt to the number of ships on the board

Return true

While r is less than the number of rows on the board

While c is less than the number of columns on the board

While I is less than the number of ships on the board

If placing the ship horizontally is good

if moving in any direction will indicate the presence of the ship

Return true

If placing the ship vertically is good

if moving in any direction will indicate the presence of the ship

Return true

Return false

}

Point GoodPlayer::recommendAttack()

{

If the direction is invalid

For loop and iterate until 200

Generate random row number for r

Generate random column number for c

If the point attacked is just a “.”

return to original point

decrement step

return function

if dir is not equal to 0

center is equal to the previous element in the vector

if dir equals 1

r is equal to the center’s r coordinate minus the unitnumbers

c id equals to the center’s c coordinate

if dir equals 2

r is equal to the center’s r coordinate plus the unitnumbers

c is equals to the center’s c coordinate

if dir equals 3

r is equals to the center’s r coordinate

c is equal to the center’s c coordinate minus the unitnumbers

if dir equals 4

r equals the center’s r coordinate

c is equal to the center’s c coordinate plus the unitnumbers

if the c coordinate, r coordinate exceeds board limits or the coordinate is not “.”

Dir equals search directions with previous element

Unitnumbers equals 1

If dir equals 0

Pop the center vector

If center vector is empty

Dir equals 0

If center vector is not empty

Dir equals 4

Return recommdattack()

Return original point

}