Design Project Name: *Soliton*

Semester Term: *Spring 2014*

Team Members: *Joshua Trahan*

Project Overview:

Soliton is a simple, text-based, turn-based, top-down perspective stealth game inspired by Metal Gear Solid. Levels are loaded directly from text files. The guards move one tile every time the player moves one tile, and have a view extending four tiles diagonally 90 degrees from each other, including the tiles in between. If the player crosses into that range, the game ends and the player loses. If the player gets to the end of the level without being spotted, he wins.

Project Scope: *Solition is a single player only game.*

Project Specifics: *The goal of the game is for the player to get from the beginning of the level to the end of the level without being spotted by any guards. Guards move one tile in their pre-set route when the player moves one tile. If the player ends up up in the view of the guards, they lose and the game ends. Players and guards can NOT move diagonally. If there is a wall between the guard and the player, the guard can NOT see the player.*

Levels will be built 1:1 from text files containing the characters that represent the level (outlined in the User View section). A line containing 'G' will designate the end of the level drawing, and user-defined guard paths will start on the line after that. To move the guard in a direction, the user will enter 'm' followed by the direction for the guard to move (ex. "M^" for move up). To make a guard simply face a direction, the user will enter 'f' followed by the direction for the guard to move. To make a guard stay, the user will simply enter 's'.

User View: *The user sees a top-down view of the entire level, including current position and heading of the guards, represented by ASCII characters. Floor tiles will be represented by the '.' character, wall tiles will be represented by '|', '-', and '+' characters, the player will be represented by an '@' character, and guards will be represented according to their heading: A guard facing left will be '<', right will be '>', up will be '^', and down will be 'v'. The end of the level will be represented by a 'E' tile.*

Project Details: *The board will be drawn in the beginning. Everything will be represented on the screen except the paths guards WILL take.*

Input: *The input needed from the player will be, at the main menu, a choice of 1 - 3. Choice 1 will be to play a prebuilt level, Choice 2 will be to play a user-created level, and Choice 3 will be to exit the game. If 1 is chosen, the game will display a list of prebuilt levels to load in. If 2 is chosen, the game will prompt the user for the name of the level file he wants to load in. The user will type either the full path of the level file, or they will type just the name of the file if the file is in the same directory as the game executable.*

Once the level is loaded, input will simply consist of the characters 'w', 'a', 's', 'd', and 'l'. 'w' will move their character up, 's' will move their character down, 'a' will move left, and 'd' will move right. 'l' will leave the game and quit to the main menu.

Enemy movement will be pre-programmed into the map files below the map that will be drawn out with ASCII. Enemies will be numbered, starting from 0, by the order their starting positions appear in the map (left to right, then up to down). This number will be followed by a ':', then a series of two-character instructions separated by spaces will define their paths. The two character instructions will be an action, then a direction, as follows:

First character: ['m', 'l']. 'm' stands for move, and will make the enemies move in the direction specified by the second character. 'l' stands for look, and will make the enemies rotate to look in the direction specified by the second character.

Second character: ['u', 'd', 'l', 'r']. Self-explanatory; 'u' is up, 'd' is down, 'l' is left, and 'r' is right.

Data needed: *I will need to use a two-dimensional character array to store the map layout. There will be a level class, a player class, and an enemy class, each with variables that will store its location on the map with x and y coordinates, and, in the case of enemies, a variable storing the direction they are looking and a one-dimensional array holding the paths they are going to take. The player object is going to simply be a member variable of the level class, and the enemy objects are going to be stored in an array within the level class.*

Project Flow (flowchart or pseudo-code):

1. *Ask the user to input the filename of a level that they want to load. Load that level and guard paths from text file.*
2. *Draw level into console, placing player and guards into positions designated by the level file.*
3. *Ask user to input direction he will move.*
4. *Move user in their specified direction.*
5. *Move all guards in the direction specified by the paths loaded from level file.*
6. *If the tile the player is on is the same tile the end of the level is on, continue to 8. Else, go to 3.*

7. *Check if the player is within the field of view of any guards. If the user is, draw a screen to let him know that he has lost, and continue to 8.*
8. *Draw a congratulatory screen to let the user know that he has won the game. Go to 10.*
9. *Draw a screen to let the user know that he has lost the game. Go to 10.*
10. *Ask the user if they want to load a new level, repeat that level, or end the game.*
11. *If they want to load a new level, go to 1. If they want to repeat the level, go to 2. If they want to end the game, continue to 12.*
12. *Close the program.*