**Spike:** 1

**Title:** Simple Game Loop

**Author:** Steven Efthimiadis, 1627406

**Goals / deliverables:**

The goal is to create a simple game loop that can check input, update the data and render the map. The input must check what available moves you have and check if you have valid input or it will send an error message. The update must check your selected move is valid or it won't update the players position. Also inside the update is if a player wants the game you are able to. The render must be able to print the map and the player position. If you land on a death tile you lose and you win if you land on the goal tile.

To create this spike, you require:

- Basic flow chart of the game loop. See Apendix
- Basic understanding of c++
- Code. See Appendix

**Technologies, Tools, and Resources used:**

List of information needed by someone trying to reproduce this work

- Visual Studio 2015 or equivalent IDE
- If you're doing sequence programming all methods must be in order
- Multidimensional arrays in c++ are [Y][X] and not [X][Y] like other languages.
- Update the header file with the required libraries before programming unless you discover you need a different one.

**Tasks undertaken:**

Before programming, you must:

- Create a flow chart
    - Start with input
        - Can we move?
        - Move input
    - Successful or invalid move?
    - Check Position
    - What is the game state now?
        - Win?
        - Lose?
        - Continue?
    - Should I keep looping through the program?

- Programming
    - What do I need in the header file?
    - Create your variables
        - Am I using a string or char array?
            - Am I using the right header?
            - Do I need pointers or global variables?
    - Create your input function
        - Correct input check?
        - Available moves check?
    - Create your output function
        - Am I able to move location?
        - Did I select to quit?
    - Create your render function
        - Print the array
        - Where am I?
    - Check the game state
        - Did I win The Game?
            - Should I keep looping through the game?
        - Did I lose The Game?
            - Should I keep looping through the game?
        - Continue?

**What we found out:**
The flowchart helps you look at the program from IPO (input, process and output) side of the program and you are able to work out how the game should loop.

The array was a difficult part because you need to set it up correctly. Using a char is good idea but in the long run a string would be a better option because a string is more flexible in the terms of using a switch statements.

With the input, getting it to see what moves are available was relatively easy. The main problem was getting to read in data for multiple available paths. Checking the input was a simple switch statement.

The output was difficult because if you don't understand where the x and y values go you will have off input and your game won't run properly.

The rendering of the map was easy. You cout the map and where your positioned you would replace the empty space with a P to show the player where you are.

When you reach an end point the game will finish but you must be able to read the position where if you land on a “D” tile you must lose and win when you land on a “G” tile.

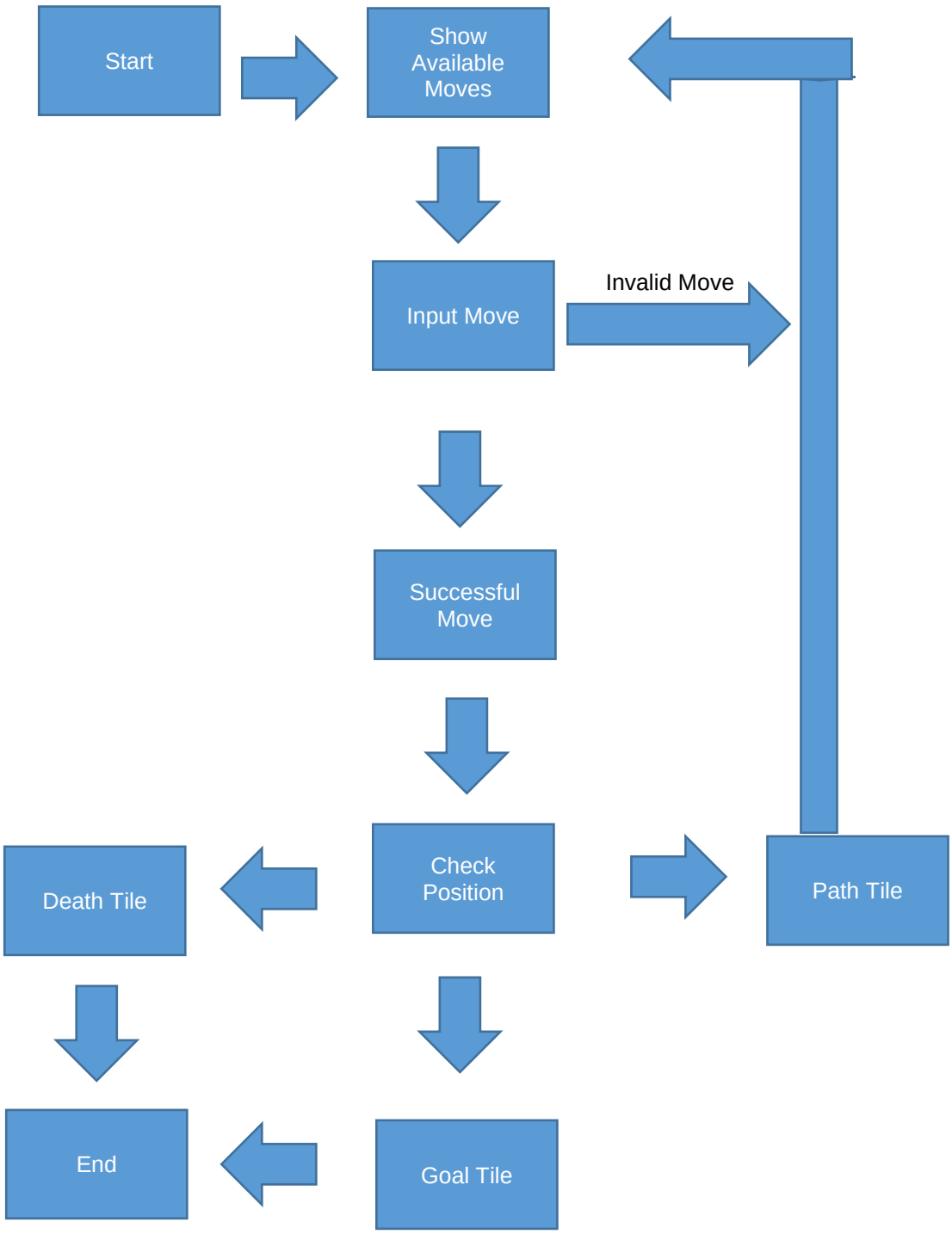
If you quit the game you must be notified or you won’t understand what’s going on.

**Open issues/risks:**

- Player able to move out of the map by one tile
    - Add another layer of # on the outside and only print out the first 8 rows so the outside layer is hidden.

# Appendix

## Flowchart



## Code

*Stdafx.h*

```
#pragma once

#include "targetver.h"

#include <stdio.h>
#include <iostream>
#include <string>
#include <algorithm>
#include <tchar.h>
```

*Spike1.cpp*

```
#include "stdafx.h"
using namespace std;
		string _map[9][8] = {
{"#","#","#","#","#","#","#","#",},
{"#","G"," ","D","#","D"," ","#",},
{"#"," "," "," ","#"," "," ","#",},
{"#","#","#"," ","#"," ","D","#",},
{"#"," "," "," ","#"," "," ","#",},
{"#"," ","#","#","#","#"," ","#",},
{"#"," "," "," "," "," "," ","#",},
{"#","#","S","#","#","#","#","#",},
{"#","#","#","#","#","#","#","#",}
};


int PlayerPosX = 2;
int PlayerPosY = 7;
string playermove;


bool GameState = true;

string checkmove() {
	string moves = "";
	bool checkfirstmove = false;
	if (_map[PlayerPosY - 1][PlayerPosX] != "#" && PlayerPosY > 0) {
		moves =  "N";
		checkfirstmove = true;
	}

	if (_map[PlayerPosY + 1][PlayerPosX] != "#" && PlayerPosY < 7) {
		if (checkfirstmove) {
			moves += ", ";
		}
		moves += "S";
		checkfirstmove = true;
	}
	if (_map[PlayerPosY][PlayerPosX - 1] != "#" && PlayerPosX > 0) {
		if (checkfirstmove) {
			moves += ", ";
		}
		moves += "W";
		checkfirstmove = true;
	}
	if (_map[PlayerPosY][PlayerPosX + 1] != "#" && PlayerPosX < 7) {
```

```
		if (checkfirstmove) {
			moves += ", ";
		}
		moves += "E";
		checkfirstmove = true;
	}
	return moves;
}

void input() {

	bool correctmove = false;
	while (correctmove == false) {
		cout << "Select Move  from " << checkmove() << endl;
		cin >> playermove;
		switch (tolower(playermove[0])) {
		case 'n':case 's':case 'e':case 'w': case 'q':
			correctmove = true;
		default:
			if (correctmove == false) {
				cout << "invalid move" << endl;
			}
		}
	}
}

void update() {
	if (playermove == "w") {
		if (_map[PlayerPosY][PlayerPosX - 1] != "#" && PlayerPosX >= 0) {
			PlayerPosX--;
		}
	}
	else if (playermove == "e") {
		if (_map[PlayerPosY][PlayerPosX + 1] != "#" && PlayerPosX <= 7) {
			PlayerPosX++;
		}
	}
	else if (playermove == "n") {
		if (_map[PlayerPosY - 1][PlayerPosX] != "#" && PlayerPosY >= 0) {
			PlayerPosY--;
		}
	}
	else if (playermove == "s") {
		if (_map[PlayerPosY + 1][PlayerPosX] != "#" && PlayerPosY <= 7) {
			PlayerPosY++;
		}
	}
	else if (playermove == "q") {
		cout << "Why you quit this awesome GAME?" << endl;
		cout << "The Game has ended. Enter any key to exit";
		cin >> playermove;
		return exit(0);
	}
}


void render() {
	cout << endl;

	for (int i = 0; i < 8; i++) {
		for (int j = 0; j < 8; j++) {
			if (i == PlayerPosY && j == PlayerPosX) {
```

```
                        cout << "P";
                }
                else {
                        cout << _map[i][j];
                }

        }
        cout << endl;
    }
    cout << endl;
}

int main()
{
    cout << "Welcome to GridWorld!" << endl;
    cout << "Valid Commands: N,S,E and W for Directions and Q to quit" << endl;
    while (GameState) {
        input();
        update();
        render();

        if (_map[PlayerPosY][PlayerPosX] == "G") {
            cout << "Congrates you won the GAME!" << endl;
            cout << "The Game has ended. Enter any key to exit";
            cin >> playermove;
            exit(0);
        }
        if (_map[PlayerPosY][PlayerPosX] == "D") {
            cout << "Sorry you lost the GAME!:(" << endl;
            cout << "The Game has ended. Enter any key to exit";
            cin >> playermove;
            exit(0);
        }
    }
  return 0;
}
```