CMP5344 Discrete Mathematics and Declarative Programming

Logbook resubmission

Ahmed Mohamed

ahmed.mohamed3@mail.bcu.ac.uk

18124225

Word count:

3200

Table of Contents

Members of the group

- Ahmed Mohamed 18124225
- Daniel Lockyer 17105158
- Arslan Badar 18117505

What is the main purpose of the product?

This product will allow the user to play a game of Pacman, the main objective of Pacman is to collect coins which increase the score of the player. This is done while evading ghosts which patrol the maze, if Pacman encounters these ghosts when they are not in frightened mode; Pacman would lose a life this incentivises the player to plan out routs that do not collide with ghosts while they're not in frightened mode.

Pacman starts off with three lives, each time he collides with a ghost he loses one of them. When all lives are lost the game ends. However, Pacman can eat power ups that are placed around the maze, once Pacman eats these power ups the ghosts change colour and change state. In this state they are frightened of Pacman if he collides with them, they are eating and the player receives 200 points. Once the ghosts are eaten, they return back to the starting point which is in the middle of the map. Where they revert back to their normal state and start patrolling the maze again (Wu, B, 2012).

Pacman

What is Pacman

Pacman is an arcade game, it was developed and released in 1980 by a Japanese company called Namco. The game was designed and created to appeal to both male and female players, as most games in the markets were male focused.

Pacman became a commercial success shortly after being released in the American market. Now Pacman is considered one of the greatest games of all time, this is because it has had a big influence on game development and gaming culture.

This can be seen in the marketing and gameplay of Pacman. Games that were released before 1980, did not have a mascot or a playable character. Pacman popularise the concept of having a play a character. This later influence games such Mario, Lara Croft, Master Chief.

Pacman also popularise the use of cut scenes between level changes, in the original Pacman after completing the first level. The screen goes black and an animation starts playing where Pacman is Being chased by one of the ghosts (Blinky). Both go out of view, Blinky returns on screen now in the frightened mode being chased by Pacman. This allowed the second level to load in the background, while at the same time keeping the player interested the game even though no game place is available at the time. (Klinefelter, 2018)

Who is the user?

The user group for this game is very wide, between the ages of five and 18. The user will know will need to know basic knowledge of how to operate a computer and it's peripheral such as the keyboard. The user will be prompted with instructions on how to play the game while on the menu screen. Which means they are not required to have previous knowledge of how the game works.

Why would the user want to use this product?

The user would want to use this product because it is a game, they are designed to help bring joy and entertainment to people's lives, for the older users they might want to play this game because it may bring back memories due to the original game "Pacman" being from the 1980's.

What type of software is it?

This specific software is a video game, it goes by the name of Pacman

What is required for the product to be deemed as a success?

For this product to be deemed a success, the user should be able to start the game and have all required elements load. The player should be able to move Pacman, using the keyboard. The players should be able to move Pacman around the map and collect dots which should increase his score by 10. If Pacman in counters a ghost he should lose a life if they are not in the frightened mode. If Pacman eats a power up all ghost on the map should change colour and change their state into frightened mode. If Pacman eats the ghost whilst they are in frightened mode, the player should receive a bonus 200 points. The ghosts should then return to the starting point which is in the centre of the maze, the game should either end when Pacman loses all his lives or all the dots located around the maze are eating.

What are the requirements for the product?

Create a program which allows the user to play a simple game of Pacman this game should be programmed in a Declarative style and demonstrate appropriate use of Functional Programming. (Cooper, n.d.)

- Some appropriate features to include should you choose to implement a Pacman
- Game:
    - At least a one-level game of Pacman with a single maze level
    - Power pellet collection should trigger Pacman "powering up"
    - Ghosts to chase Pacman around the maze
    - A way of keeping track and presenting high scores for players from a menu interface

User Story Specifications

Story 1

Title	Player wants to move Pacman
Narrative	- As the player - I want to move Pacman around the maze - So that he can consume all dots on the maze and complete the level.
Scenario 1:	- Given the level has fully loaded - And the pressed key is part of the control scheme - When the player presses the up, down, left or right on the keyboard - Then the Pacman should move in the corresponding direction
Scenario 2:	- Given Pacman is by a wall - And the player presses a key that is part of the control scheme - When the player tries to move Pacman in the direction of an obstructing wall - Then Pacman should not move due to the obstruction

Table 1 (User Story Specifications)

Story 2

Title	The Players score increases
Narrative	- As the player - I want gain points certain events - So that I can get a high score
Scenario 1	- Given Pacman is near dots - And Pacman goes over it - When Pacman eats the dots - And the dots disappear - Then the player's score increases by 10
Scenario 2	- Given Pacman has activated a powerup - And the ghosts are in frightened mode - When Pacman eats the frightened ghost - Then the player's score should increase by 200

Table 2 (User Story Specifications)

Story 3

Title	Player moves Pacman over consumables
Narrative	- As the player - I want Pacman to eat the dots - when he reaches them - So that Pacman can complete the level
Scenario 1	- Given Player presses a valid movement key in a chosen direction - And that there are dots in that path of movement - And there are no obstructions - When Pacman reaches the dots - Then the food dot should disappear
Scenario 2:	- Given the player presses a valid movement key in a chosen direction - And that there is a power up in that path of movement - And there are no obstructions - When Pacman reaches the powerup - Then all the ghosts on screen should turn blue - And change their state to frightened so they are editable for Pacman

Table 3 (User Story Specifications)

Story 4

Title	Pacman loses lives
Narrative:	- As the player - I want Pacman to lose a life when colliding with non-frightened ghosts - So that the game ends when Pacman loses all lives
Scenario 1	- Given that a ghost is next to Pacman - When the ghost hits Pacman - Then Pacman should lose a life - And start flashing
Scenario 2	- Given that Pacman has only 1 life left - And a ghost is approaching Pacman - When the ghost hits Pacman - Then Pacman should lose a life - And the game should end

Table 4 (User Story Specifications)

Data Models

Sprites

Pacman is made up of both visual and audible data. The game consists of many sprites, which are two dimensional images/animations that are present in a scene. Examples of sprites in Pacman are:

- Ghosts
- Ghost eyes
- The maze
- Powerups
- Dot
- Pacman

These sprites will be stored in the program code in Base64 Format, this means that the program does not need any external files to run. Therefore, allowing it to use less resources when rendering images, it also means that when the program is used on different machines, less files will be needed allowing the overall file size to be smaller.

Keyboard inputs

The system will require keyboard inputs to operate. The player will need to input a valid key, this can be any of the up, down, left and right arrows on the keyboard.

Once the player enters the key inputs into the system, the player will receive visual output. For example, Pacman will move in the direction the entered input corresponds to. Each key on the keyboard corresponds to a KeyEvent code in JavaScript, which the system can use to give the correct output; in this case visual outputs. (Karl-Bridge-Microsoft, 2018)

Input	KeyEvent Codes	output
Up arrow	VK_UP 0x26	Pacman moves upwards
Down arrow	VK_DOWN 0x28	Pacman moves downwards
Left arrow	VK_LEFT 0x25	Pacman moves left
Right arrow	VK_RIGHT 0x27	Pacman moves right

Table 5 (Keyboard input and outputs)

My Tasks

Sprite Animations

At the start of this project we weren't 100% Sure how the animation of the Pacman, ghosts and power ups would be inserted into the program. I first started out by creating flash animation of the Pacman movement. To allow Pacman to moving up, down, left and right. Also, for when Pacman is eating forwards, backwards, up and down.

Adobe animation 2020 was used to create the animations. The sprite images were placed on a timeline then the timeline was merged so it created an animation. Further along our project we realise that our program would not take in gift animations. We also then realise that the way the animations would be inserted into the code would be in relation to the position of the Pacman. When the player moved Pacman up the animation would change corresponding to where the Pacman is on the way in X,Y coordinates. In this case it would make more sense encode sprite images into base64.

How to Move Pacman

The way the keyboard inputs are recognised by the program, stored and used is facilitated by javascripts event keycodes.

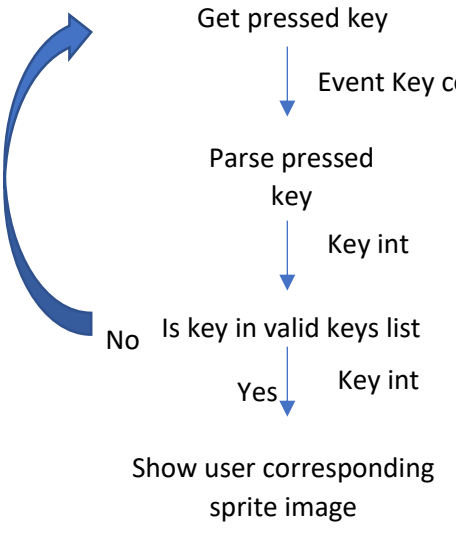
The program starts off by creating an empty set which is like a list that can store multiple items. When a player presses a certain key, that key code is stored inside of the empty set.

A JavaScript event key code is given to each key on the keyboard. Only one key is stored at a time in the list. Once the key is pressed the key is added to the list and once a new key is pressed that old key code is removed from the list. (Oracle, n.d.)

he assigned key codes correspond to a Pacman sprite image. Once that key is present in the list that sprite image is sent out as an output to the player.

pseudo code
- Create variable to sort empty list - Create variable to store press key - Create variable to store valid key codes - Create function to reset list to empty - Check if pressed key matches valid key code - If valid key is pressed add to empty list - If new valid key code is pressed reset list to empty - And new key code to empty list

Table 6 (Keyboard input handling pseudo code)



Flowchart 1(Keyboard input handling pseudo co)

Generating Ghost movement

Original Pacman ghost AI

The original Pacman use sprites and tile sets to distinguish the player from their environment. Pacman was originally created for a screen resolution of 224×228 and a tile set that was 8×8. The Screen would be split into a grid of 28×36 tiles. Not all of these tiles can be accessed by the player, some tiles were dedicated for the walls of the maze and others were dedicated for the score. (Birch, 2010)

The ghost and the player character interact with the tales differently. Pacman is allowed to change directions at any time if there is no wall obstructing that path. However, ghost cannot change directions until they reach the middle of that tile, this gives the player an advantage as they are not required to be exactly in the centre of each tile to make a turn. This is because humans do not possess the reaction time required to do this perfectly every time.

The ghost in the original Pacman had three different states. These were chase mode, scatter mode and frightened mode.

Mode	Result
Scatter	Each ghost would find it way to one of the corners of the maze
Chase	Each ghost uses its AI to find Pacman in the maze and chase him
frightened	Each ghost turns blue and roams the maze randomly

Table 7 (Pacman ghost states)

Each ghost has its own personality meaning it chases the player in different ways. There are three ghost Pinky, Blinky, Inky and Clyde. The colours are as followed pink, red, blue and orange. All three ghosts use the same AI but utilise it differently to chase Pacman. However, what differentiates them from each other is, each ghost targets are different tile set. For example, Blinky who is the most aggressive ghost, targets the tile that Pacman is currently on. This updates every frame of the game.

Figure 1 (Pacman ghost chase AI)

Once Blinky enters Chase mode, it locates the exact position of Pacman, then proceeds to calculate the shortest path to that tile. In this mode ghosts are not allowed to turn around and they cannot turn to a wall which is obstructing that direction. The diagram below illustrates how the AI chases and catches Pacman.

Once pinky enters chase mode, it's a locates the exact position of Pacman and then targets four tiles in front of Pacman and not the exact tile Pacman is on.

For inky, it's locates the exact tile Blinky is on. Then calculates the distance between Pacman and Blinky, the line created between pinky and Pacman is rotated 180°. This is inkys target tile.

Clyde is very different to how he chases Pacman, his target location when in chase mode is the exact tile that Pacman is on. This is only the case when Clyde is eight tiles away from Pacman, however if Clyde is closer than eight tiles his target location changes to one of the corners of the maze. (Birch, 2010)

Flood Fill algorithm

The original algorithm for Pacman is called Dijkstras's algorithm. (Dao, 2017) This algorithm is used to find the shortest path between a source node and a target node, in this case Pacman and the ghosts. (Mukherjee, 2012) I decided not to use this type of algorithm, because it was very difficult to find research/documentation on have such an algorithm would be implemented in F#.

Instead I decided to use a flood Fill algorithm. This type of algorithm is most commonly used in image editing software such as photo shop/paint. This algorithm determines the areas that are connected to a node in a multidimensional array and floods that Area with the desired colour. As can be seen in the figure below. (Techie Delight, 2016)

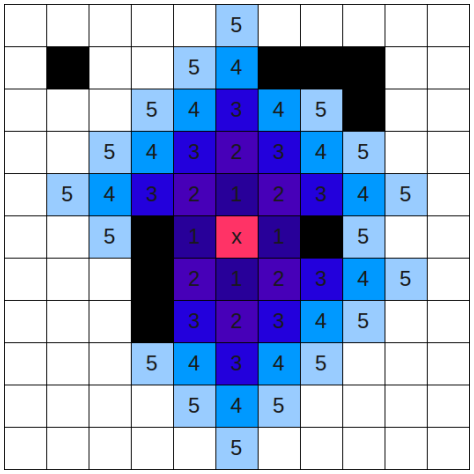


Figure 2 (Flood fill algorithm) (Will, 2014)

The field of robotics also uses this type of algorithm for robot maze problems, where robots need to make a decision on how to complete a maze. In most of these cases the target note is static, however for Pacman the target note needs to be dynamic. This is because Pacman is the target and it is constantly moving, however the Target the note in a maze does not change. (Tjiharjadi, Wijaya and Setiawan, 2017)

The algorithm will start off by giving Pacman a value of 0, every tile set directly connecting to Pacman will be given a value of 1. The algorithm then will continue to fill all the tiles that are not obstructed by a wall with an incremental value. As can be seen in the diagram below.

This algorithm updates every time the player moves Pacman, giving the ghosts a new path to follow. This means no matter which tile set the ghost is on its can find it is way to Pacman, all it has to do is move to the tile with the lowest value (NiteHackr, 2016)

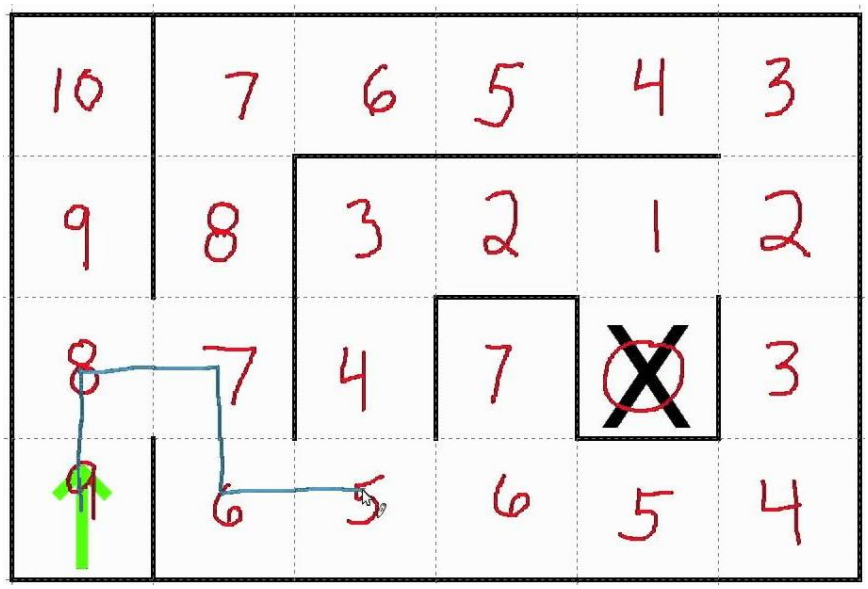


Figure 3 (Pacman ghost chase AI) (Backus, 2015)

Code implementation

Dependencies That are required:

- Webpack
- Node.js
- .net
- Fable
- Fable.Core.JsInterop

Fable

Fable is a compiler that is powered by Babel design to convert F# code into JavaScript. (semuserable, 2020)

Webpack

Webpage is a module bundler; it creates a single file output which can easily be added to webpages. Webpack uses modules from both fable files and other JavaScript libraries to generate the single file. (Cieslak, 2016)

Node.js

Node.js is an open source JavaScript runtime environment that can runs JavaScript code on multiple platforms.

How to install these dependencies?

Webdev:

- npm install webpack-dev-server --save-dev

Node.js:

- We went to nodejs.org and downloaded the latest version of nodejs. We downloaded version 14.3.0.

.net:

- This was installed when we install visual studio.

Fable:

- npm install --save fable-core

Keyboard inputs

```
module Keyboard =

    let mutable keysPressed = Set.empty
//creates an empty set to store the pressed keys.

    let resetSet () = keysPressed <- Set.empty
// reset allows the set to be emptied.
    let isPressed keyCode = Set.contains keyCode keysPressed
// checks if the pressed keys stored in the set matches the keycode.


    let update (e : KeyboardEvent, pressed) =
// Event is triggered when the keycode is pressed
      let keyCode = int e.keyCode
//The keycode is stored as an int
      let When =  if pressed then Set.add else Set.remove
//if the corresponding keycode is pressed.
      keysPressed <- When keyCode keysPressed

//Store key code in keyspressed set.
```

Flood fill algorithm

```
/// Recursive flood fill function
/// This algorithm is used to fill the whole of the maze.
let flood Fill fill (x,y) =
  let rec f n = function
    | [] -> ()
    | ps ->
        let ps = ps |> List.filter (fun (x,y) -> Fill (x,y))
        ps |> List.iter (fun (x,y) -> fill (x,y,n))
        ps |> List.collect (fun (x,y) ->
            [(x-1,y);(x+1,y);(x,y-1);(x,y+1)]) |> f (n+1)
 f 0 [(x,y)]
```

(Trelford, 2012)
(Sweigart, 2011)

Reflection

At the start of this assessment my group wasn't 100% sure on how we should go about dividing the tasks between us. The added difficulty came from the fact that we did not follow the planning guide Provided to us from the start. We presumed the tasks that we had to complete were obvious, however further down the planning stages we realise that the tasks were a lot more complicated that than we originally thought. I believe this is one of the reasons why we were not able to create a fully functioning program.

We presumed that once everyone in the group had completed all the tasks it would be easy to merge the code together. We seem to realise that fable requires a certain file structure to work. This file structure we seem to realise that fable requires a certain file structure to work. We did not implement this file structure from the start, which meant we had to spend more time editing our current files to fit Fables file structure.

After we had split the tasks between everyone in the group, I felt happy that I could complete the tasks that I chose. I then realise that the tasks that I chose were a lot more complicated and a lot more complicated than I previously thought. Unfortunately, at this time everyone in the group had multiple modules that they had to work on, and it was hard to find a time when everyone was free. This limited the amount of time we were able to spend together on the project.

By The time we agreed on set days of the week that we can meet up, the country went into lockdown because of COVID-19. This meant I and other people in the group had to move back to our parents houses which also meant that I could not bring my desktop pc with me. This slowed our progress even more. Once I was set up with all the software and hardware that I needed to complete the project, I started doing research on how to complete the tasks assigned to me.

The things that were good and worked well was the fact that everyone in the group was patient with each other as we all had assignments due, this allowed us to compromise and agree upon set days of the week that we were all available.

I believe if we did more research on each task before agreeing upon who did what, would have saved us a great deal of time. As some tasks that were given to people in the group, we later found out was not needed for example creating sprite animations.

In conclusion I learnt that when in a group randomly assigning tasks is not the best practice, we should have researched every task before assigning it to someone so we knew that that tasks were necessary and that it could be done. I also learned that sometimes we have to challenge decisions made in the group to ensure that we are not agreeing just because of groupthink

Next time I'm working in a group I will encourage other members of the group to state their strengths and weaknesses, as this will make it easier when dividing up tasks. And that also when a task is assigned to someone, we should research the task as a group before someone agrees to complete it, in which case other members of the group can provide assistance.

Referencing

Backus, M. (2015). Dynamic Programming / Flood Fill Algorithm. YouTube. Available from: https://www.youtube.com/watch?v=Zwh-QNlsurI [Accessed 30 Jul. 2020].

Birch, C. (2010). GameInternals - Understanding Pac-Man Ghost Behavior. [online] gameinternals.com. Available from: https://gameinternals.com/understanding-pac-man-ghost-behavior [Accessed 30 Jul. 2020].

Cieslak, K. (2016). Getting started with Fable and Webpack. [online] kcieslak.io. Available from: http://kcieslak.io/Getting-Started-with-Fable-and-Webpack [Accessed 30 Jul. 2020].

Cooper, E. (n.d.). Tasks 2020. [online] login.microsoftonline.com. Available from: https://moodle.bcu.ac.uk/pluginfile.php/6995739/mod_resource/content/0/Tasks%202020.pdf.

Dao, S. (2017). Dijkstra's Algorithm for Pacman. [online] iamstevendao.github.io. Available from: https://iamstevendao.github.io/blog/posts/algorithm/2017/09/17/dijkstra-Pacman.html [Accessed 30 Jul. 2020].

Karl-Bridge-Microsoft (2018). Virtual-Key Codes (Winuser.h) - Win32 apps. [online] Microsoft.com. Available from: https://docs.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes.

Klinefelter, D. (2018). 5 Ways 'Pac-Man' Influenced Modern Video Games. [online] Fandom. Available from: https://www.fandom.com/articles/pac-man-influence-video-games [Accessed 30 Jul. 2020].

Mukherjee, S. (2012). Dijkstra's Algorithm for Solving the Shortest Path Problem on Networks Under Intuitionistic Fuzzy Environment. Journal of Mathematical Modelling and Algorithms, 11(4), pp.345–359.

NiteHackr (2016). Flood Fill path finding. [online] www.allegro.cc. Available from: https://www.allegro.cc/forums/thread/616037 [Accessed 30 Jul. 2020].

Oracle (n.d.). KeyEvent (Java Platform SE 7). [online] docs.oracle.com. Available from: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html [Accessed 30 Jul. 2020].

semuserable (2020). Starting with Fable (F#). [online] DEV Community. Available from: https://dev.to/semuserable/starting-with-fable-f-kbi [Accessed 30 Jul. 2020].

Sweigart, A. (2011). Recursion Explained with the Flood Fill Algorithm (and Zombies and Cats) - The Invent with Python Blog. [online] inventwithpython.com. Available from: https://inventwithpython.com/blog/2011/08/11/recursion-explained-with-the-flood-fill-algorithm-and-zombies-and-cats/ [Accessed 30 Jul. 2020].

Techie Delight (2016). Flood Fill Algorithm. [online] Techie Delight. Available from: https://www.techiedelight.com/flood-fill-algorithm/ [Accessed 30 Jul. 2020].

Tjiharjadi, S., Wijaya, M.C. and Setiawan, E. (2017). Optimization Maze Robot Using A* and Flood Fill Algorithm. International Journal of Mechanical Engineering and Robotics Research, pp.366–372.

Trelford, P. (2012). Phillip Trelford's Array | Pacman Tiles. [online] Trelford.com. Available from: http://trelford.com/blog/post/PacTile.aspx [Accessed 30 Jul. 2020].

Will (2014). algorithm - How many moves to reach a destination? Efficient flood filling. [online] Stack Overflow. Available from: https://stackoverflow.com/questions/8120179/how-many-moves-to-reach-a-destination-efficient-flood-filling [Accessed 30 Jul. 2020].

Wu, B. (2012). The Computational Intelligence of the Game Pac-Man. Internet of Things, pp.646–651.