CMP5344 Discrete Mathematics and Declarative Programming

logbook

Ahmed Mohamed

18124225

## 24th February

Group made, members:

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

## 2nd March

Decided to make Pacman

Working on sprites and graphics

GitHub repo set up, able to share between everyone

## 3rd March

Made a task table and divided tasks between members

I chose to do the following:

* Html Aspects of the Pacman game
* Sounds and dots
* Code to check for walls
* Background rendering

We all decided that everyone should help and do the following:

* Keyboard input
* Rendering everything in the game
* Testing and Debugging

5TH MARCH

One of my tasks was to create the animations for the Pacman, the ghost and the power ups. 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 Pac man is on the way in X,Y coordinates.

## Planning

## Initial Plans:

Below is a breakdown of the tasks we will need to complete and who is completing what

|  |  |  |
| --- | --- | --- |
| Name | Task | Completed |
| Dan | Html web pages | completed |
| Dan/Ahmed | Loading maze and graphics | Somewhat compoleted |
| Ahmed | Sprite Animations | completed |
| Dan | Checking for walls |  |
| Dan | Background rendering |  |
| Ahmed | Keyboard input | Somewhat completed |
| Ahmed | Generating Ghost movement |  |
| Ahmed | Choosing Pacman image | completed |
| Dan | Sounds and dots |  |
| Dan | Create canvas |  |
| Ahmed | Create background |  |
| Ahmed | Create Ghosts |  |
| Arslan | Move Pacman |  |
| Arslan | Define the Pacman state |  |
| Arslan | Detect pills and collisions |  |
| All | Rendering everything in the game |  |
| All | Testing/Debugging |  |

## What have we decided to do?

We have chosen to develop a Pacman game with multiple levels.

## 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?

The game must function, with no crashes and for the user to be able to collect all of the “dots” allowing them to pass onto the next level. Upon continuing to the next level, the user’s score must be brought across between levels allowing the user to potentially achieve a new high score.

## Who is the user?

The User is someone aged between 5 - 80. The user will need some basic knowledge of how to operate a computer and run applications, so they can actually run the application.

## 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 “Pac-man” being from the 1980’s.

## What is the main purpose of the product?

The object of the game is to collect coins while also evading the ghosts to achieve the highest score possible before you run out of lives

## What are the requirements for the product? (As Stated in the Tasks List)

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.

* This should not be a simple port of your previous PacMan game, you should effectively design and implement this program from the beginning so that it leverages appropriate techniques from the Declarative Programming paradigm.
* 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
* Some additional features you could explore when aiming for higher marks:
  + An expansion of the game to allow for network capabilities that keep track of player scores via an online leader board

## Data Models

We will need to represent a lot of data within the Pac-man game. All the data must be of a certain type and are represented in some form.

## What data must be represented?

Pac-man 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 Pac-man are:

* Ghosts
* Ghost eyes
* The maze
* Powerups
* Food aka “The Dots”
* Pac-man

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.

Other data that will be included in the game are sounds such as:

-          Pac man eating food capsules around the maze

-          Pac man eating power ups

-          Ghost eyes dashing around the maze

-          Game start

-          Game over

These sound files in their raw format are of file type mp3. However, for the Pac man program to read in and understand the sound files, we must be able to specify the file paths of these sound files. The file paths themselves are a form of data which we will need to deal with and are of type string. The sound files must also be a .Wav type, in order to play.

## How is keyboard input data recognised?

## 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. JavaScript event key code is used to give each key on the keyboard an identifying number. For example, the up arrow has an event key code of 38 the and down arrow has an event key code of 40. Only one key is stored at a time. 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.

## Dependencies we will need:

* WebDev
* Node.js
* .net
* Fable
* Fable.Core.JsInterop

## How did we 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

## User Story Specifications

**Story**: Player wants to move Pac-man

**Narrative**:

  As a Pac-man player

  I want to move Pac-man around the maze

  So that he can consume all dots on the maze and complete the level.

**Scenario 1**: Player presses a valid key that is part of a given control scheme

   Given the keyboard is functioning

  And the pressed key is part of a control scheme

   When the player presses the movement key

   Then the Pac-man should move in the corresponding direction

**Scenario 2**: Player presses key to move Pac-man in the direction of a wall

   Given Pac-man is by a wall

  And the player presses a key that is part of a valid control scheme

   When the player tries to move Pac-man in the direction of an obstructing wall

   Then Pac-man should not move due to the obstruction

**Story**: Player moves Pac-man over consumables

**Narrative**:

  As a Pac-man player

  I want Pac-man to eat food dots when he reaches them

  So that Pac-man can complete the level

**Scenario 1**: Player presses movement key such that Pac-man reaches food

   Given Pac-man is near a food dot

  And the player presses a key that moves Pac-man towards the food

   When Pac-man reaches the food dot

   Then the food dot should disappear

**Scenario 2**: Player moves Pac-man over power ups

   Given Pac-man is next to a powerup consumable

  And the player presses a key that moves Pac-man towards the powerup

   When Pac-man reaches the powerup

   Then all the ghosts on screen should turn blue and become edible for Pac-man

**Story**: Player scoring

**Narrative**:

   As a Pac-man player

   I want the score to increase in certain events

   So that I can achieve a high score

**Scenario 1**: Pac-man reaches food dots

   Given Pac-man is near food

   When Pac-man eats the food

   Then the player’s score increases by 10

**Scenario 2**: Pac-man eats a ‘frightened’ ghost

   Given Pac-man has activated a powerup

    And the powerup is still active

   When Pac-man eats the frightened ghost

   Then the player’s score should increase by 200

**Story**: Pac-man loses lives

**Narrative**:

   As a Pac-man player

   I want Pac-man to lose a life when approached by a ghost

   So that the game ends when Pac-man loses all lives

**Scenario 1**: Pac-man gets hit by a ghost

   Given that a ghost is next to Pac-man

   When the ghost hits Pac-man

   Then the player shall lose a life

**Scenario 2:** Player runs out of lives

   Given that the player has only 1 life left

  And a ghost is approaching Pac-man

   When the ghost hits Pac-man

   Then the game should end