1/5/2020

Maurice Thompson-Hamilton

Student no: 018016836

PacMan

A description of PacMan development

Table of Contents

[Glossary 1](#_Toc21681296)

[Key Words 1](#_Toc21681297)

[Introduction 2](#_Toc21681298)

[Aims 2](#_Toc21681299)

[Objectives 2](#_Toc21681300)

[Literature Review 3](#_Toc21681301)

[Findings and Discussions/Results 4](#_Toc21681302)

[Conclusion 5](#_Toc21681303)

[Future Work 6](#_Toc21681304)

[Bibliography 7](#_Toc21681305)

[Appendix 8](#_Toc21681306)

# Glossary

GTA - Grand Theft Auto

VB – Visual Basic

GEC – Games Engine Creation

# Key Words

Games Industry, Employability, Computers, Game Mechanics, Industry Standards, Professionalism

# Introduction

The clone of the classic PacMan game developed using S2D framework.

# Aims

The aim is to explore the development of a classic game such as PacMan.

# Objectives

Choose a development environment by experimenting with a variety available to students.

Select and apply game and graphical principles to create images, models and game systems.

Implement algorithms, data structures and file handling using high-level programming language and API to support games and graphics development.

# Controls

Once the game has started, the following keys will move PacMan around the level.

|  |  |  |
| --- | --- | --- |
| DIRECTION | KEY | ALTERNATIVE KEY |
| UP | W | Up Arrow |
| DOWN | S | Down Arrow |
| LEFT | A | Left Arrow |
| RIGHT | D | Right Arrow |
| EXIT | ESCAPE | ESCAPE |

# Findings and Discussions

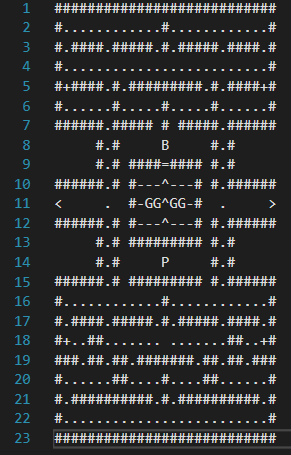


Figure : level\_0.txt - file describing initial level layout.

While developing this PacMan clone, there are various decisions to be made.

The one issue relates to the amount of screen to use. The original dimensions of 1024 x 768 pixels was divided up into 32 x 32pixel segments giving a grid to work with. This grid formed the basis for the whole game where the placement of all entities, such as walls and munchies, could adhere to. In addition, loading a text file that described what entity should be placed where, simplified the initial placement of all entities. As Figure 1 shows, each character and symbol represent a 32 x 32 pixel tile of an entity in the game. For instance, each “#” represents a wall tile and each “.” represents a munchie.

Start

Can TARGET

perform

CurrentMove?

Can TARGET

perform NextMove?

Is CurrentMove =

STOP?

Set CurrentMove =

STOP!

Move TARGET to

CurrentMove

STORE:

NextMove

Yes

End

RETRIEVE:

NextMove

SUBROUTINE:

Check for walls

while

preforming

NextMove

No

No

Yes

Yes

No

No

RETRIEVE:

CurrentMove

SUBROUTINE:

Check for walls

while

Performing

CurrentMove

OVERWRITE:

CurrentMove

with

NextMove

OVERWRITE:

LastMove

with

CurrentMove

RETRIEVE:

LastMove

SUBROUTINE:

Check for walls

while

Performing

LastMove

Can TARGET

perform LastMove?

OVERWRITE:

CurrentMove

With

LastMove

Yes

Figure : Movement flow diagram

A game play issue arose where the player would have to accurately time their keypresses to successfully negotiate turns down different paths. This was due to the 32 x 32pixel grid system in that there was no ley-way, either you turned at the exact moment or you didn’t! To account for this, the movement control system used two flags called nextMove and currMove and the held directional information for the player. As shown in Figure 2, by storing the player’s next move, the game would proceed to move the player in their current direction until the next move was a valid move. However, this had to be performed when the player was near enough to the center of their current tile, otherwise they would collide and even pass through walls which is not intended.

Figure : Flow diagram of Movement

There are multiple methods for detecting collisions used within the game namely, Bounding Box, Circle Range and Tile Array.

The Bounding Box collision compares two boxes to see if they intersect at all. Since the ghosts are square, this type of collision is well suited for any collisions with ghosts. The wall tiles could also use this type of collision, however, continual repeated checks on entities that never move was deemed in-efficient and an alternative method was used.

The Circle Range collision compares the distance between the centres of two circles with the sum of the radi of the two circles. This type of collision provided a high level of accuracy for circular entities such as PacMan and the munchies.

The Tile Array collision takes the coordinates of an entity and compares the corresponding tile in the map array built from the file shown in Figure 1. This method of collision is efficient for entities that never move since their position could easily be found by multiplying the map coordinates by 32 to give the screen position in game. This means that anytime a tile from the map needed to be checked, as simple and low computational cost would provide the required result. The alternative would require the calculation of the tile’s position on screen for every tile and then compare each tile to another entity like PacMan of a ghost.

The ghosts used in the game have a variety of features. Firstly, the array used to contain the ghosts can be any integer. This means that by changing one line of code, any number of ghosts can exist on the screen during game-play.

The ghosts also have a persona. Each persona dictates the colour and the target that the ghosts will try to reach as shown in Figure 3.

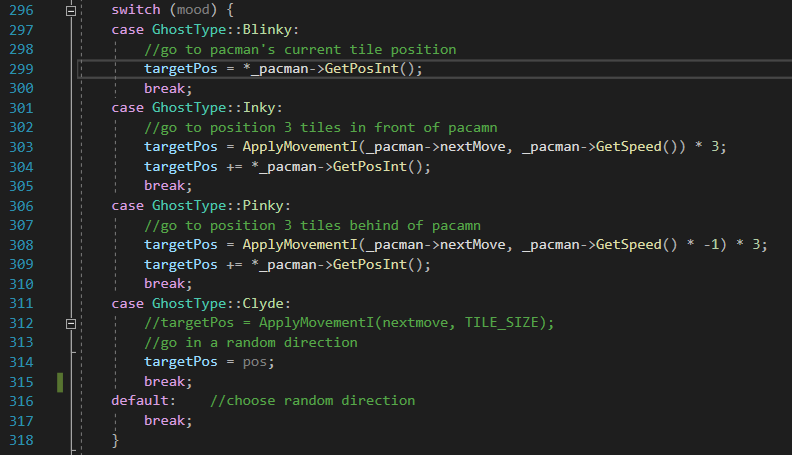


Figure : Figure 4: Code listing showing the targets of the four types of ghosts

The four personas where name Blinky, Inky, Pinky and Clyde as they were in the original PacMan game. Blinky would target the location that the player was at. Inky would target a location 3 tiles in front of the player. Pinky would target a location 3 tiles behind the player. Clyde would look to move in a random direction.

# Conclusion

Development of PacMan was an enjoyable experience in general. Using various principles helped in providing experience with the development environment, deeper knowledge of the programming language and an understanding of how to use third-party API’s. Developing a PacMan clone proved to be useful by providing an opportunity to experiment with a well-known game concept.

In future, API’s with documentation will be easier to work with as more examples will be available to compare and contrast against developed code. Certain principles require further testing and experimentation as future projects will require different approaches. In addition, only a small selection of API’s was used to develop this version of PacMan and future projects will use different libraries and API’s, which will provide different functionality and have a different focus. For example, the API’s used in this project was aimed at 2D graphical games and interfaces, whereas future API’s will be geared towards 3D graphics and Virtual Reality to name a few.