JNU

School of Engineering

PAC-MAN

by

Jatin Verma (14000151)

R. Shravan (21701620)

Under the Guidance of

Dr. Mukesh Giluka

Asst. Prof. , SoE, JNU

Acknowledgment

First, we would like to thank School of Engineering for implementing Object Oriented Programming as a subject for our course.

We would also like to thank our Object Oriented Programming Assistant Professor Dr. Mukesh Giluka Mukherjee for giving us an opportunity to do this project on Pac-man.

We would also like to thank our friends who were our backbone and helped us in collecting the requirements for our project.

At last we would like to thank all the other people who were involved directly or indirectly in collecting the requirements.

- Jatin Verma & R. Shravan

-

Contents

1 Introduction

Pac-Man is a maze arcade game developed and released by Namco in 1980. Pac-Man was a widespread critical and commercial success, and it has an enduring commercial and cultural legacy. The game is considered important and influential, and it is commonly listed as one of the greatest video games of all time.

Pac-Man was originally written in a scripting language, and later refactored and written in C, to give it a significant speed boost.

1.1 Project Aim

The aim of the project is to recreate the all-time classic game Pac-man in C++ using our concepts of Object Oriented Programming.

1.2 About the Game

Pac-Man is a maze chase video game where the player controls the eponymous character through an enclosed maze.

Objectives and Rules of the Game:

- The objective of the game is to eat all of the dots placed in the maze while avoiding four coloured ghosts — *Blinky* (red), *Pinky* (pink), *Inky* (cyan), and *Clyde* (orange) — that pursue him.
- When all of the dots are eaten, the player advances to the next level.

- If Pac-Man makes contact with a ghost, he will lose a life and the game ends when all lives are lost.
- Each of the four ghosts have their own unique, distinct artificial intelligence (A.I.), or "personalities":
    - Blinky gives direct chase to Pac-Man
    - Pinky and Inky try to position themselves in front of Pac-Man, usually by cornering him
    - and Clyde will switch between chasing Pac-Man and fleeing from him.
- Placed at the four corners of the maze are large flashing "energizers", or "power pellets". Eating these will cause the ghosts to turn blue with a dizzied expression and reverse direction.
- Pac-Man can eat blue (*frightened*) ghosts for bonus points. When eaten, their eyes make their way back to the centre box in the maze, where the ghosts are "regenerated" and resume their normal activity.
- Eating multiple blue ghosts in succession increases their point value.
- After a certain amount of time, blue-coloured ghosts will flash white before turning back into their normal, lethal form.
- Eating a certain number of dots in a level will cause a bonus item, usually in the form of a fruit, to appear underneath the centre box, which can be eaten for bonus points.

1.3 Motivation

Video games have always been an integral part of our childhood and Pac-Man is one of the earliest games we have ever played when we were young.

This project of recreating the all-time classic game Pac-man would give us the opportunity to learn more about how games are developed by recreating the game ourselves. The game would also help us to increase our knowledge of *Object Oriented Programming* and enhance our grasp in it.

1.4 Challenges

- To recreate the game in C++ using Object Oriented Programming.
- To implement graphics to the game.
- To implement the game according to the rules and objectives of the game

2 Game Design

2.1 Features

1. Classes and Inheritance
2. User-defined Header Files
3. Polymorphism
    A. Compile Time
    - Function Overloading
    - Default Parameter

    B. Run Time
    - Abstract Class
    - Function Overriding

4. Data Abstraction: Data Hiding and Encapsulation

2.2 Classes and Inheritance

Using the concept of inheritance, the classes used have a Hierarchical type of inheritance.

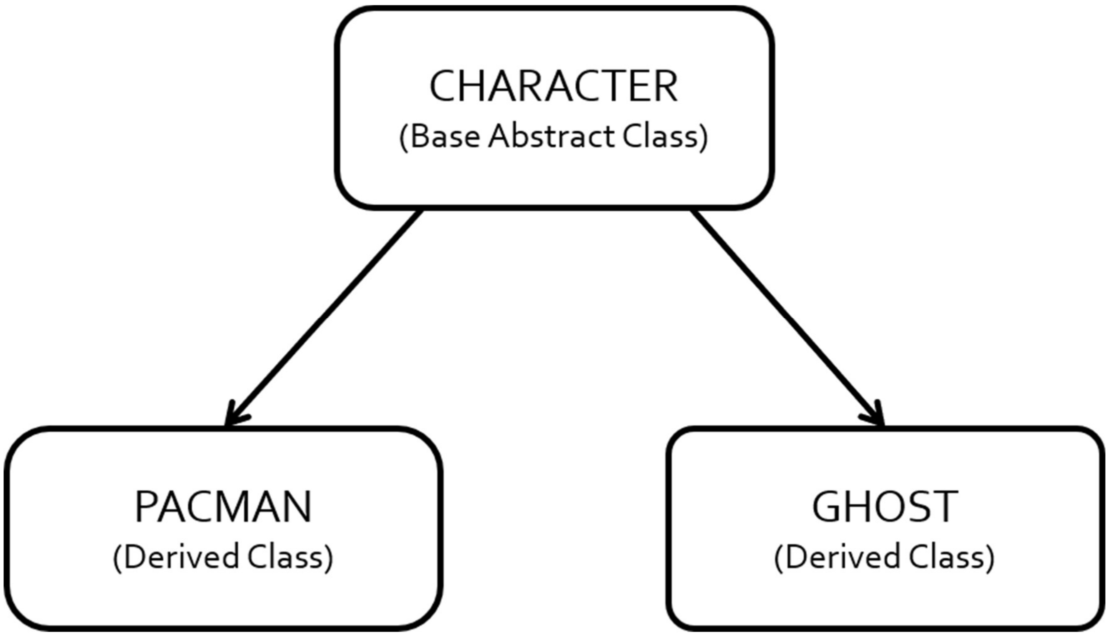

Fig.1. Hierarchical Inheritance

2.2.1 Character class

- This is the *base class* from which both Pacman and Ghost have been derived.
- This is also an *Abstract Class*.
- It contains data members such as:
    - X *(x co-ordinate)*
    - Y *(y co-ordinate)*
    - currentDir *(current facing direction)*

- It contains members functions like:
    - get_currentDir *(to get current direction)*
    - isWall *(to check if wall is present ahead)*
    - isAtCenter *(check if at the centre of tile)*
    - getTile *(get which type tile it is)*
    - getFollowingTile *(get which type of tile is ahead)*
- It contains pure virtual member functions draw, move and reset.

2.2.2 Pacman class

- This class is *publicly* derived from Character class.
- It contains data members such as:
    - dirStore *(direction inputted by user)*
    - angle *(angle made with the x-axis)*
- It contains members functions like:
    - get_angle *(to get angle made with x-axis)*
    - move *(function for Pac-man to move)*
    - draw *(to draw/display Pac-man on-screen)*
    - reset *(to reset Pac-man back to the starting position)*
    - death *(applying animations and translating Pac-man back to original positions).*

2.2.3 Ghost class

- This class is *publicly* derived from Character class.
- It contains data members such as:
    - ghostColour *(colour of ghost)*
    - moveType *(type of moment)*
    - speed *(speed of ghost)*
    - eaten *(check: whether ghost is alive or not)*

- It contains members functions like:
  - set_moveType *(to set the move type: CHASE/ SCATTER/ FRIGHTEN)*
  - move *(function for ghost to move)*
  - draw *(to draw/display ghost on-screen)*
  - reset *(to reset ghost back to the starting position)*
  - set_Frighten *(apply movetype FRIGHTEN to ghost).*

2.3 User-defined Header Files

2.3.1 base.h

This header file contains a class 'Character' which has some functions which are common to both classes Pac-man and ghost. So, this class has been inherited by those classes and those functions are overridden.

2.3.2 pacman.h

This header file contains a class 'Pacman' which has the features of Pac-man and some functionalities which uses those features like in which direction Pac-man should move, how much points it should get for eating any particular pill or ghost or any fruit and display Pac-man.

2.3.3 ghost.h

This header file contains a class 'Ghost' which has the features of ghosts and functionalities based on those features like when should a ghost leave the Ghost pen, in which direction a particular ghost should move to catch Pac-man as soon as possible, when to acquire which mode and moveType, how to save themselves from Pac-man while in frighten mode and display the ghosts.

2.3.4 maze.h

This header file contains a maze matrix i.e., a matrix where ghosts chase Pac-man *(Different types of tiles are defined here like wall, portal etc)*, display the maze and the points rewarded to Pac-man on the maze after it eats a pill, fruit or a ghost, where to put fruit in the maze and more. everything related to maze is defined here.

2.3.5 interface.h

This header file contains some functions which take care of the things like highest score till date, current score of the player, Number of lives each player have, updation of the high score if previous high score is broken and more.

2.3.6 png_load.h

This header file has functions defined which loads the png images from the image directory to an image buffer so that images can be further binded to a tex handle.

2.3.7 textures.h

This header file contains textures/images of each entity in the game i.e., Pac-man and ghosts are created and blend with the images loaded inside the main memory.

2.3.8 load_an_bind_texture.h

This header file contains a function that request one texture handle and binds texture object from the image buffer, blends it with the environment using GPU and frees the image buffer.

2.4 Polymorphism

2.4.1 Compile Time

2.4.1.1 Function Overloading

Functions with the same name are used with different functionalities but differ in the type of parameters or number of parameters.

For e.g.: In the code for this game, *penMove(int)* is a member function for the ghost to move up and down in the pen area and *penMove()* is a member function for the dead ghost to go back to pen area.

2.4.1.2 Default Parameter

Functions with default values assign to function parameters

2.4.2. Run Time

2.4.2.1 Abstract Class

Class having atleast one pure virtual member function.

For e.g.: Class Character is an abstract class with virtual member functions draw, move and reset.

2.4.2.2 Function Overriding

When a base class and derived class have a member function with the same name and during run time, function overriding leads to execution of function of derived class.

For e.g.: For an object of class Pacman when *draw()* function is called, the *draw()* function of Pacman is called over that of base class Character.

2.5 Data Abstraction: Encapsulation and Data Hiding

Encapsulation is ensured due to use of classes and data hiding is ensured due to private, protected and public access in classes.

3 Setup

3.1 Operating Environment

- Operating System : Linux
- GPU
- OpenGL

3.2 Compilation Instructions

In order to compile execute the solution some commands must be executed in the terminal. First, to set up compilation on Linux use:

```
$ ln−fs Makefile.linuxMakefile
```

The solution can then be made using:

```
$ make pacman −B
```

3.3 Execution Instructions

In order to execute the solution use:

```
$ ./pacman
```

Unless changes are being made to the code, compilation does not need to be run every time before execution

4 Game Controls

Movement:

- Arrow Keys - Left, Up, Right, Down

Game:

- P - Pause/ Unpause the game
- Q - Quit the game
- R – Start a new game *(Only when "Game Over" is showing)*

5 Screenshots

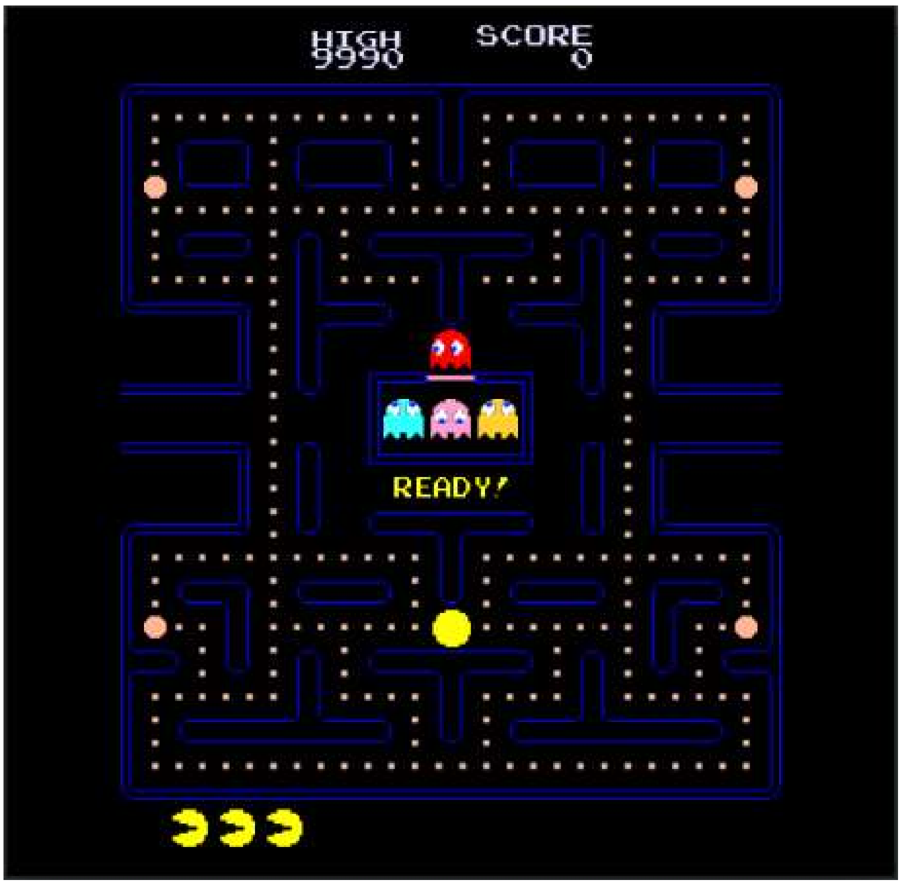

Fig.2. Start Screen

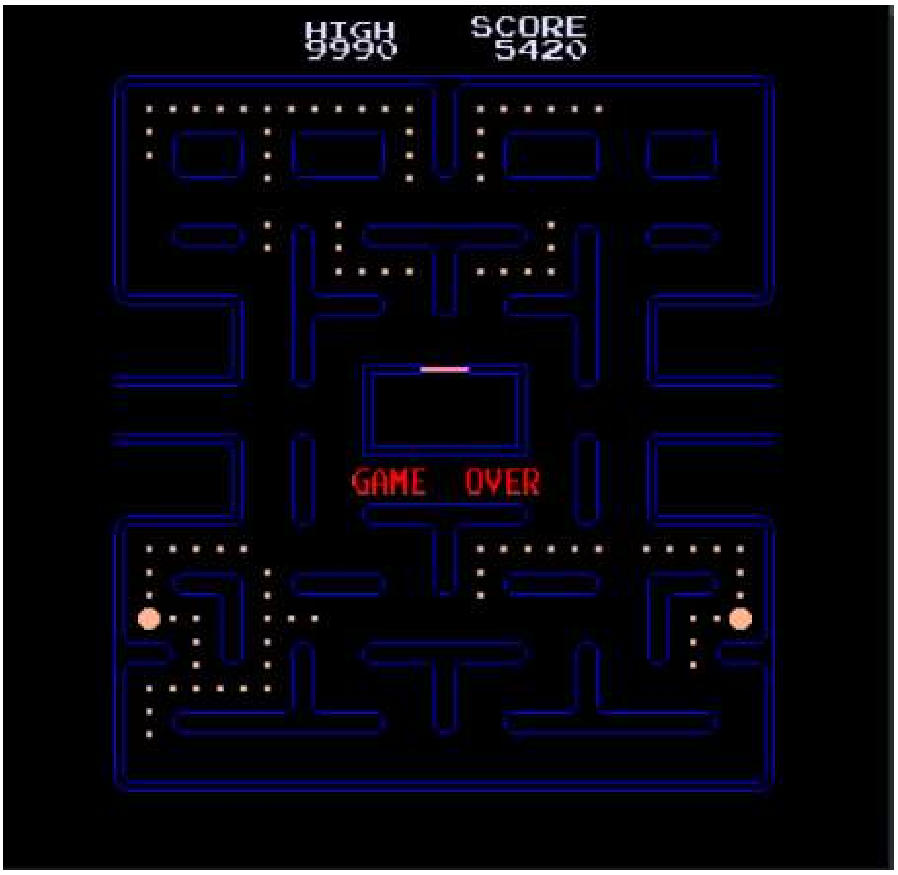

Fig.3. Game Over screen

6 Future Developments

Future developments included are as follows:

- Since now only one level is repeated with the same difficulty, future developments include different levels layouts to be added and also different difficulty modes to be added.
- Multiplayer Option to be added.
- Player vs Player mode to also be added.

7 References

1. https://en.wikipedia.org/wiki/Pac-Man
2. https://engineering.purdue.edu/OOSD/F2008/assignment/assignment4.html