Requirements Specification

For

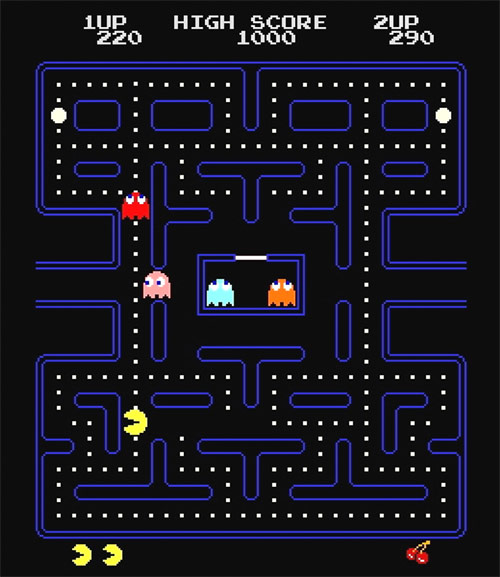
Capstone: PAC-MAN

**Prepared by: Jacob Stewart**

**COP-2936-86403**

**Instructor: Dr. Banisakher**

**Date: 02/17/2024**



[This Photo](https://gamersup.blogspot.com/2011/08/pacman-arcade.html) by Unknown Author is licensed under [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/)

Table of Contents

[1. Introduction 2](#_Toc159063605)

[1.1 Purpose 2](#_Toc159063606)

[1.2 Intended Audience 2](#_Toc159063607)

[2. Overall Description 2](#_Toc159063608)

[2.1 Product Overview 2](#_Toc159063609)

[2.2 User Classes and Actors 2](#_Toc159063610)

[2.2.1 User Classes: 2](#_Toc159063611)

[2.2.2 Actors: 2](#_Toc159063612)

[2.3 Program Environment 3](#_Toc159063613)

[2.4 Use Case 3](#_Toc159063614)

[U.C.1 Move Pac-Man 3](#_Toc159063615)

[U.C.2 Eat Pellet 3](#_Toc159063616)

[U.C.3 Eat Power Pellet 4](#_Toc159063617)

[U.C.4 Death 4](#_Toc159063618)

[U.C.5 Move Ghosts 4](#_Toc159063619)

[U.C.6 Kill Pac-Man 5](#_Toc159063620)

[3. Domain Requirements 5](#_Toc159063621)

[3.1 System and Platform Requirements 5](#_Toc159063622)

[3.1.1 Gameplay 5](#_Toc159063623)

[3.1.2 Level Design: 5](#_Toc159063624)

[3.1.3 User Interface: 5](#_Toc159063625)

[3.1.4 Audio-Visual Elements: 5](#_Toc159063626)

[3.1.5 Game Progression: 6](#_Toc159063627)

[3.1.6 Scoring System: 6](#_Toc159063628)

[3.1.7 Performance: 6](#_Toc159063629)

[3.1.8 Compatibility: 6](#_Toc159063630)

[3.2 Accessibility and Usability Requirements 6](#_Toc159063631)

[3.3 Safety and Security Requirements 6](#_Toc159063632)

[3.4 Maintenance and Reliability Requirements 6](#_Toc159063633)

[4. Future Requirements 6](#_Toc159063634)

[Appendix A: Example User Interface 7](#_Toc159063635)

# Introduction

## Purpose

This document aims to report the features and intended behavior of the application and describe the functional and non-functional requirements. In short, this document will explain what the application will do and how it will perform.

## Intended Audience

The intended audience for this document includes but is not limited to:

**Developers**: Those responsible for creating the game, this document will serve as an outline of how the game is to be built.

**Designers**: Those responsible for creating the visual and audio components of the game.

**Testers**: Those responsible for ensuring the game meets the specified requirements.

**Project Managers**: Those responsible for overseeing the development of the game.

# Overall Description

## Product Overview

This program is designed to be a Java application capable of running on any system that can access Java such as Windows, Mac, and Linux. This application is meant to replicate the Pac-Man created in 1980 and originally released by NAMCO.

## User Classes and Actors

For this application, the following classes described below will represent the User Classes and Actors

### User Classes:

**Player**: The external user who interacts directly with the game through keyboard inputs to control the PAC-MAN character.

**Ghosts**: Non-player characters who act as obstacles, not directly controlled by the player, but instead provide a challenge for the Player to overcome.

### Actors:

**Game Panel**: Class responsible for managing the game’s logic, user inputs, game states, and rendering. Also known as the engine.

**Game Map**: Class that represents the maze that Pan-Man and the ghosts move around, this class will define the boundaries as well as the location of orbs, walls, and tunnels.

**Level Manager**: Class responsible for tracking the game progression, including loading the next level and handling events such as loading the level when Pacman dies.

**Score Manager**: Class responsible for tracking the player's score and ensuring it is appropriately updated while also maintaining a record of the high score.

## Program Environment

The program environment is designed to operate as a Java Application and interactive video game. The application will run in a windowed setting at a specified resolution that is expected to fit all screen sizes. The program will be created in the Eclipse IDE, without the expectation of using external packages, relying on those provided by the Java JDK.

## Use Case

Below is a set of use cases to represent the actions and resolutions to actions taken in the application.

### U.C.1 Move Pac-Man

Description: Allows the player to move Pac-Man through the maze.

Primary Actor: Player

Preconditions: Pac-Man is alive and the game is in progress.

Postconditions: Pac-Man has moved to a new position in the maze.

#### U.C.1.1 Game Flow:

The player inputs a direction (up, down, left, right) for Pac-Man to move.

The game checks if the movement is valid (not blocked by a wall).

If the movement is valid, Pac-Man moves to the new position.

If Pac-Man encounters a pellet, the pellet is eaten, and the player's score is increased.

### U.C.2 Eat Pellet

Description: Allows Pac-Man to eat pellets in the maze.

Primary Actor: Player

Preconditions: Pac-Man is alive and positioned on a tile with a pellet.

Postconditions: The pellet is removed from the maze, and the player's score is increased.

#### U.C.2.1 Game Flow:

Pac-Man moves onto a tile containing a pellet.

The pellet is removed from the maze.

The player's score is increased by a certain amount (e.g., 10 points per pellet).

### U.C.3 Eat Power Pellet

Description: Allows Pac-Man to eat power pellets in the maze.

Primary Actor: Player

Preconditions: Pac-Man is alive and positioned on a tile with a power pellet.

Postconditions: The power pellet is removed from the maze, ghosts become vulnerable, and the player's score is increased.

#### U.C.3.1 Game Flow:

Pac-Man moves onto a tile containing a power pellet.

The power pellet is removed from the maze.

Ghosts change to a vulnerable state, allowing Pac-Man to eat them for points.

The player's score is increased by a certain amount (e.g., 50 points per power pellet).

### U.C.4 Death

Description: Handles the event of Pac-Man colliding with a ghost.

Primary Actor: Player

Preconditions: Pac-Man is alive and collides with a ghost.

Postconditions: Pac-Man loses a life, and the game state is updated.

#### U.C.4.1 Game Flow:

Pac-Man collides with a ghost.

Pac-Man loses a life.

If Pac-Man has remaining lives, the game continues. Otherwise, the game is over.

### U.C.5 Move Ghosts

Description: Controls the movement of ghosts in the maze.

Primary Actor: Game Engine

Preconditions: The game is in progress.

Postconditions: Ghosts have moved to new positions in the maze.

#### U.C.5.1 Game Flow:

For each ghost:

Determine the next move based on the ghost's behavior (chase, scatter, or flee).

Check if the move is valid (not blocked by a wall or another ghost).

Move the ghost to the new position.

### U.C.6 Kill Pac-Man

Description: Handles the event of a ghost colliding with Pac-Man.

Primary Actor: Ghost

Preconditions: A ghost collides with Pac-Man.

Postconditions: Pac-Man loses a life, and the game state is updated.

#### U.C.6.1 Game Flow:

A ghost collides with Pac-Man.

Pac-Man loses a life.

If Pac-Man has remaining lives, the game continues. Otherwise, the game is over.

# Domain Requirements

These domain requirements are essential for defining the core functionality and user experience of Pac-Man within its specific domain of arcade-style maze games.

## System and Platform Requirements

### 3.1.1 Gameplay

* Pac-Man must move smoothly through the maze, avoiding collisions with walls and ghosts.
* Ghosts must follow specific movement patterns, such as chasing Pac-Man or moving randomly.
* Eating pellets should increase the player's score.
* Eating a power pellet should temporarily allow Pac-Man to eat ghosts.

### 3.1.2 Level Design:

* Each level should have a unique maze layout with walls, pellets, power pellets, and tunnels.
* Subsequent levels may increase in difficulty, such as by adding more ghosts or reducing the duration of power pellets.

### 3.1.3 User Interface:

* The user interface should include elements such as a score display, lives remaining, and level indicators.
* Controls should be intuitive and responsive, allowing players to easily navigate the maze.

### 3.1.4 Audio-Visual Elements:

* The game should include appropriate graphics for Pac-Man, ghosts, pellets, and the maze.
* Sound effects should enhance the gameplay experience, such as eating pellets or colliding with ghosts.

### 3.1.5 Game Progression:

* The game should progress through levels as the player eats all the pellets in a maze.
* Game-over conditions should be triggered when Pac-Man loses all lives.

### 3.1.6 Scoring System:

* Scoring should be based on various actions, such as eating pellets, eating ghosts, and completing levels.
* Different actions may yield different amounts of points.

### 3.1.7 Performance:

* The game should run smoothly and without lag on a variety of hardware configurations.
* Load times should be minimal to maintain player engagement.

### 3.1.8 Compatibility:

* The game should be compatible with common platforms and input devices.
* The game should have a common screen resolution and aspect ratio to match most systems.

## Accessibility and Usability Requirements

The window will support movement but will not support resizing.

The application will be obtained from the internet; however, internet access is not required once the game is downloaded to the user’s chosen area of memory.

The application will come attached with a User Manual describing how the game is played as well as containing the inputs necessary to move the player character.

### Safety and Security Requirements

This application will not collect any personal data from the user. The application will only create a single save file on the user’s computer to document the high score achieved. Steps will be taken to ensure all data streams and input streams are closed correctly to mitigate data leaks to the user’s system.

### Maintenance and Reliability Requirements

The selected frameworks and APIs selected (Java, AWT) is expected to be fully maintained and updated for the foreseeable future.

# Future Requirements

The potential features that may be added include but are not limited to:

Map color changes as the player progresses through the levels.

Fruit spawning acts as an incentive for the player to leave the orb path for extra points.

The potential addition of non-standard map layouts.

Power-ups that increase the user's speed for a short time.

Resolution changing, allowing full-screen gameplay and scaling for different monitors.

# Appendix A: Example User Interface

A screenshot of a video game

Description automatically generated

A video game with a black background

Description automatically generatedA video game screen with a square pattern

Description automatically generatedA video game screen with a square pattern

Description automatically generatedA screen shot of a video game

Description automatically generated