**2805 ICT**Systems and Software Design  
**Assessment | Milestone 1  
Student Name:** Mitchell Roles **Student ID:** s5132278  
**Student Email:** [mitchell.roles@griffithuni.edu.au](mailto:mitchell.roles@griffithuni.edu.au)

# Table of Contents

**Contents**

[1. Table of Contents 2](#_Toc46472818)

[2. Project Requirements 3](#_Toc46472819)

[2.1 Functional Requirements 3](#_Toc46472820)

[2.2 Use Cases 4](#_Toc46472821)

[2.2.1 Actor Description 4](#_Toc46472822)

[2.2.2 Documented Template 4](#_Toc46472823)

[2.2.3 Activity Diagram 4](#_Toc46472824)

[2.3 Use Case Diagram 4](#_Toc46472825)

[2.4 Non-Functional Requirements 4](#_Toc46472826)

[2.5 Constraints 4](#_Toc46472827)

[3. Project Risk 5](#_Toc46472828)

[4. Project Feasibility 6](#_Toc46472829)

[5. Prototype Demonstration 7](#_Toc46472830)

[6. Conceptual Design 8](#_Toc46472831)

[6.1. Automatic Document generation 8](#_Toc46472832)

# Project Requirements

## Functional Requirements

Functional Requirements are, as the name implies, the needed functionality of a system or what that system needs to achieve with legal/expected input from a user. These functional requirements will detail what precisely is the desired behaviour of the product. For this specific project the goal is to create a version of the game Pac-Man so the functional requirements will reflect the needs to meet this end.

|  |  |  |
| --- | --- | --- |
| **ID:** FR-1 | **Name:** Broad GUI Requirements | |
| Functional Requirement | | Rationale |
| The system will need to offer a user an interface to which they can interact with the game in an intuitive and simple way, allowing them to select their desired settings and attempt to beat the game. | | Having a functional and intuitive GUI will allow for an enjoyable user experience, one which they can access what they want quickly and efficiently. |
| **ID:** FR-2 | **Name:** Functional Tutorial | |
| Functional Requirement | | Rationale |
| The system will require a dedicated section of itself to educate the user on how the core gameplay aspect functions as well as the win and lose conditions. | | The tutorial will allow a user to smoothly be introduced to the gameplay aspect of the system, avoiding any possible frustrations without some amount of guidance |
| **ID:** FR-3 | **Name:** Game Score | |
| Functional Requirement | | Rationale |
| The system should display some information, ascertaining to how well a user is doing in the game. “Eating” pellets, power-pellets or fruit all should increase the score. | | Having some form of score will allow a user to, at a glance, determine how well they are currently doing within a play state. |
| **ID:** FR-4 | **Name:** Win Condition | |
| Functional Requirement | | Rationale |
| The system needs to recognise when a user has completed a level. This will be done when the final pellet has been “eaten” | | Having a win condition will give a user a visible goal to move towards. This will also allow the game to finish and not continue forever. |
| **ID:** FR-5 | **Name:** Lose Condition | |
| Functional Requirement | | Rationale |
| The system will need to recognise when a user has lost the level. This will be when a ghost has touched “Pac-Man” and they are not under the effect of a power-pellet. | | This will give the user something to avoid and add some challenge to the experience, to prevent it from becoming boring. |
| **ID:** FR-6 | **Name:** Power Pellet | |
| Functional Requirement | | Rationale |
| The game aspect of the system will need to recognise when the user has “eaten” a power-pellet and have them enter a power up state, where the ghost will scatter and can be eaten themselves. | | This will give the player more choice as to how they wish to complete the level, going for the power up early or waiting until there are little pellets left. |

This might need to change.

## Use Cases

### Actor Description

|  |  |  |
| --- | --- | --- |
| **Actor Name:** User | **Other Names:** Player | |
| Actor Description | | Actor Assumptions |
| The actor for this system will be a user, or player. This user will be the main driver of the program, as most the systems functions will remain unused unless they specifically use it. The user will be able to interact with the system through a combination of the mouse and keyboard. With the mouse primarily being used for the menu and the keyboard being used to navigate the maze. | | The user will have the expectation that the system will work faultlessly with no visible errors during gameplay. |

### Documented Templates

|  |  |
| --- | --- |
| **Use Case:** Playing the Game | |
| **Functional Requirement Satisfied:** FR-1 | **Actor:** User |
| Description | |
| The user will interact with the game through the keyboard, using either the directional arrow keys or w, a, s, d to move “Pac-Man”. The user will navigate through a maze with the aim of picking up all the dots scattered throughout the level, whilst avoid the enemy ghost that pursue the player. The player will need to observe and learn the patterns of each “ghost” in order to complete the level in the smallest amount of time. | |
| Risk | |
| This requirement is the core of the game and without a user has no reason to use the system. Not implementing this requirement will leave a non-functional piece of software with little reason for existing. A user not being able to figure out the end goal is a possibility but implementing a tutorial of some description should suffice to negate this. | |
| Pre-Conditions | Post Conditions |
| The system started correctly or reset without fault and the user has selected their desired settings. | The game level successfully concludes with the player either completes the challenges or is caught be the enemies. |
| Basic Flow | |
| * The maze is generated, fully enclosed   + All dots are on the map   + The “ghost house” is located in the centre of the map   + The “ghosts” are placed in the ghost house   + The player character is placed somewhere in the maze * After a brief countdown the game begins   + Ghosts level the ghost house and pursue the player   + Player begins to pick up dots   + Timer begins   + Score increases with dot collection * Game ends if:   + Player collects all dots   + Or Player is touched by the ghosts whilst not under the effect of a power pellets | |
| Corresponding Diagram | Alternate Flow |
|  | The user terminates the program before the game could finish |

|  |  |
| --- | --- |
| **Use Case:** Tutorial Pop Up | |
| **Functional Requirement Satisfied:** FR-2 | **Actor:** User |
| Description | |
| A small dismissible pop up screen which will contain concise information regarding how the game is played. Includes: movement controls, win condition, lose condition, power-pellet function and score mechanics. Will be presented with corresponding graphics that allow the user to more quickly grasp the core concepts that are communicated. | |
| Risk | |
| The player does not interact with this portion of the software and is unaware how the system functions. | |
| Pre-Conditions | Post Conditions |
| The game system started correctly, and the user has clicked on the tutorial tab in a menu. | The user has read and understood the information conveyed and exits the screen. |
| Basic Flow | |
| * User selected the “Tutorial” tab * A screen pops up * User can cycle through information   + Controls   + How to win   + Enemy introduction   + Power ups   + How the scoring works * User exits the screen when satisfied | |
| Corresponding Diagram | Alternate Flow |
|  | User never interacts with the menu |
| **Use Case:** Score Scaling | |
| **Functional Requirement Satisfied:** FR-3 | **Actor:** User |
| Description | |
| A scoring system that will rate how well a user is doing, or has done, inside a level. The system will increase the score a user has accumulated through eating pellets, eating power-pellets and eating ghosts under the effect of a power-pellet. The system will scale the amount of score a player gets regarding how long the game has been running for. With the longer the level has been running the smaller the amount of score a player will receive. | |
| Risk | |
| The player does not understand how the scoring system works and potentially ends up with a score that is significantly lower than the amount they were expecting. | |
| Pre-Conditions | Post Conditions |
| The system and the game started and run correctly. | The level is terminated, through either completion or when the player is caught by a ghost. |
| Basic Flow | |
| * Player eats pellet/power-pellet/frightened ghost   + Score increases * As the timer increases   + Score amount decrease   + If the level has continued after a certain length of time no score is obtained | |
| Corresponding Diagram | Alternate Flow |
|  | The game is not started, and no score is accumulated. |

|  |  |
| --- | --- |
| **Use Case:** Winning the Game | |
| **Functional Requirement Satisfied:** FR-4 | **Actor:** User |
| Description | |
| After the player has collected all the pellets in a level the game will finish and the final score the player has achieved will be displayed. The game will then reset the level and allow the user to play again. | |
| Risk | |
| The player never completes the level due to the difficulty of the game. | |
| Pre-Conditions | Post Conditions |
| The system and level have started correctly. The game functions properly. | The game is reset |
| Basic Flow | |
| * Player collects all the pellets * A separate screen appears   + - Displays congratulatory text     - Displays score * Player exits screen * Game is reset | |
| Corresponding Diagram | Alternate Flow |
|  | Player never reaches the congratulations screen. |
| **Use Case:** Losing the Game | |
| **Functional Requirement Satisfied:** FR-5 | **Actor:** User |
| Description | |
| If the player has been caught by the ghosts and is not under the effect of a power-pellet the game level will terminated and the score the player achieved will be displayed. Afterwards the games will restart, and the player can play the game again. | |
| Risk | |
| The player never begins the game and never sees the screen. Or the player never loses. | |
| Pre-Conditions | Post Conditions |
| The game has started properly and the correctly detects when the player has lost | The game is reset |
| Basic Flow | |
| * Game begins and the player is caught * Some notification plays to alert the player that they have lost   + - Character animation     - Death sound   + Lose screen is displayed     - Some consoling message     - Score achieved   + Player exits screen   + Game resets | |
| Corresponding Diagram | Alternate Flow |
|  | Player completes the level and never sees the lose screen |

|  |  |
| --- | --- |
| **Use Case:** Power-Up Mode | |
| **Functional Requirement Satisfied:** FR-6 | **Actor:** User |
| Description | |
| Player collects a power-pellet and all of the ghosts enter a freighted state and scatter. During a brief amount of time the ghost can be eaten by the player for extra score. After the ghosts have been eaten, they will return to the ghost house and begin their pursuit again. When the power up wears off, the ghost will return to their normal state and renew their pursuit. | |
| Risk | |
| This feature is critical to the gameplay experience, as it gives the player a small window of relief which allows them to either get themselves into a more advantageous position or to gain some extra score. | |
| Pre-Conditions | Post Conditions |
| The level has started correctly, and the game is functioning properly. | The power-up wears off and the game continues like normal. |
| Basic Flow | |
| * Player eats a power-pellet   + Ghosts enter frightened state   + Some musical que is played   + Count down begins * Ghosts are eaten and points are awarded to the player.   + Ghost are changed to their eaten state   + Eaten ghosts return to the ghost house   + Revived ghost returns to pursuing the player * Power-up wears off   + Frightened ghost return to their normal state | |
| Corresponding Diagram | Alternate Flow |
|  | Player does not start the game |

### Activity Diagrams

Figure 1 (FR-1)  
Playing the Game

Might need to revise Diagram 1 and 5 to incorporate the lives system

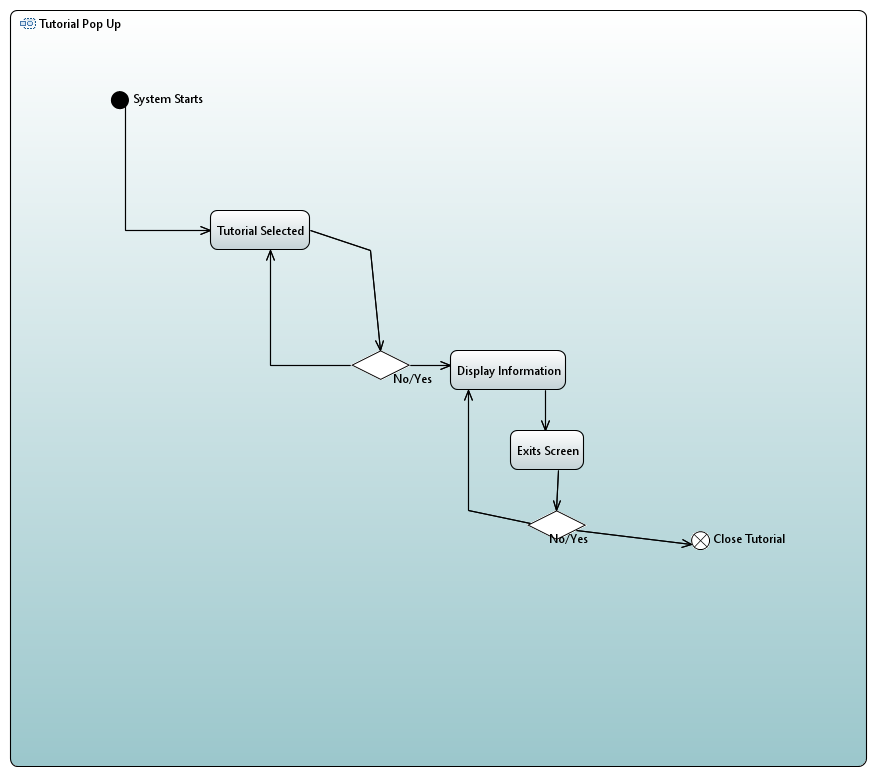


Figure 2 (FR-2)  
Tutorial Pop-up Screen

A screenshot of a social media post

Description automatically generated

Figure 3 (FR-3)  
Timer Based Score Scaling

A close up of text on a white background

Description automatically generated

Figure 4 (FR-4)  
Win Condition

A screenshot of a cell phone

Description automatically generated

Figure 5 (FR-5)  
Lose Condition

A picture containing screenshot

Description automatically generated

Figure 6 (FR-6)  
Player Power-Up Mode

## Use Case Diagram

A close up of a map

Description automatically generated

Figure 7 (Use Case)

## Non-Functional Requirements

Non-functional requirements describe abstract concepts that need to apply to the software, but do not entail components of the system itself. These requirements refer to the software system as a functional whole and how it is received by its user.

|  |  |  |
| --- | --- | --- |
| **ID:** NFR-1 | **Name:** Usable Interface | |
| Non-Functional Requirement | | Rationale |
| The system will need to feature an interface that a user will find easy to navigate, understand and use intuitively. This entails readable fonts pronounced buttons and complying with the standard interaction conventions for menu screens. | | This will allow a user to have a pleasant experience when navigating the system, where they do not need to learn overly cumbersome navigation controls. |
| **ID:** NFR-2 | **Name:** Reusable Systems | |
| Non-Functional Requirement | | Rationale |
| The system’s class and algorithm setup should be designed with reuse in mind. | | This would encourage a more efficient system and allow for further expansion of the software without any extensive changes needed. |
| **ID:** NFR-3 | **Name:** System Reliability | |
| Non-Functional Requirement | | Rationale |
| The system should always behave in the same manner for every possible instance. Under all possible stress loads, the game should always behave as expected. | | This will allow for a consistent experience across all possible users of the system. |
| **ID:** NFR-4 | **Name:** System Performance | |
| Non-Functional Requirement | | Rationale |
| The game needs to be quick and responsive to all user interactions. Maze generation should not take more than a second and there should not be any hang time during gameplay. | | The user will have expectations of the game, as “Pac-Man” is known as a lightweight game, to be fast. |
| **ID:** NFR-5 | **Name:** User Enjoyment | |
| Non-Functional Requirement | | Rationale |
| As the software’s primary purpose is that of a game, it should be enjoyable for any user to play. Challenging, but not too difficult. | | This will help to keep users engaged with the system and will create a positive reputation for the system. |
| **ID:** NFR-6 | **Name:** Software Maintainability | |
| Non-Functional Requirement | | Rationale |
| The software should be easy to apply updates and changes to. The systems need to be well organised and commented. | | This will allow for the possibility of rapid changes to the system if an error/bug is ever found. |

## Constraints

Constraints are the limitations placed upon the project. They entail function and non-function requirements to both the system and the developments process of said system.

|  |  |
| --- | --- |
| **ID:** CST-1 | **Name:** Time Limitation |
| The project only has a limited amount of time that is available for it to be worked on. After this time is up all progress must be uploaded and markable work is no longer possible. All work that is specified in the Milestone Briefs must be completed before the due date and to a high standard of quality. | |
| **ID:** CST-2 | **Name:** Restricted Features |
| As the project brief has specified that the game “Pac-Man” needs to be recreated, certain features need to be implemented in order to achieve that end. The game needs to take place in a maze, ghost needs to chase the player, all the orbs in a level need to be collected in order to complete said level, etc. | |
| **ID:** CST-3 | **Name:** Development Team Size |
| Only a single developer is permitted to work on this particular instance of the project. | |

# Project Risk

When undertaking a project, such as this one, there are always accompanying risks involved. These risks encompass a broad range of items of different levels of severity from acceptable, as low as reasonably practical (ALARP) and finally to intolerable. There are two distinct and relevant categories of risk associated with this type of project, with them being Software Process Risks and Software Product Risks. The former is related to the risks associated with the development of the system and the latter relates to mechanical faults in the product.

Risks can pose a serious issue to a project, but that is only if said risk is not identified and planned for. Through risk identification and analysis, these risks can be either mitigated or completely nullified with the formulation of appropriate plans to combat them. Contained below are some of the considered risks and the strategies that will minimise the likelihood of them becoming an issue.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID:** R-1 | **Name:** Deadlines | | **Risk Type:** ALARP |
| Risk Description | | Reduction Strategies | |
| Not meeting required deadlines | | Following standard software development practises, such as: prototyping key components, dividing work into phases and making use of reusable components. | |
| **ID:** R-2 | **Name:** Maze Generation | | **Risk Type:** ALARP |
| Risk Description | | Reduction Strategies | |
| Difficulty in generating the different required mazes for the game to be played. | | Prototyping the generation with static templates of the mazes that can work as the base. Proper research into the GUI software. | |
| **ID:** R-3 | **Name:** User Enjoyment | | **Risk Type:** Intolerable |
| Risk Description | | Reduction Strategies | |
| The end user/player does not enjoy the experience of playing the game, due to either being too easy, too difficult or not to their standards of quality. | | Adequate amounts of playtesting of gameplay. Tuning the ghosts AI as needed from the findings from the playtests. | |
| **ID:** R-4 | **Name:** System Security | | **Risk Type:** Intolerable |
| Risk Description | | Reduction Strategies | |
| Software unintentionally edits items on the user’s personal computer | | Ensuring the all the software will ever affect is contained within its root files. | |
| **ID:** R-5 | **Name:** Game Requirements | | **Risk Type:** ALARP |
| Risk Description | | Reduction Strategies | |
| Game not meeting the requirements for it to be considered as a “Pac-Man” game. | | Thorough research into the topic and implementing the core features to capture the essence of the game. | |

# Project Feasibility

Pac-Man is a video game created in 1980 for arcade machines, which eventually became a world-wide sensation. Created by nine people in just under 2 years with the limited technology of the era, it should be possible to recreate the game using modern technology in considerably less time. The only true limitation on the project being the experience of the developers themselves, when working with a system of this nature. However, with the appropriate amount of research and work, recreating the game within the given time limit is possible.

# Prototype Demonstration

# Conceptual Design

## Automatic Document generation

**Conceptual Design**