**Figure 0.1**: Image Credit: FreePik[1].

Title of the (Midterm) Report

"HUNGRY SPACE CAT"

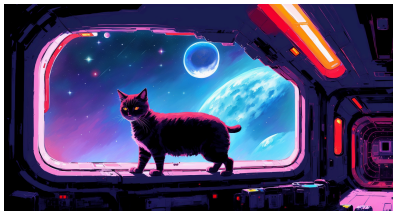
By Anita Pal (200208268)

[1] https://www.freepik.com/free-ai-image/cute-cat-space_82550977.htm

# 1. INTRODUCTION

## 1.1: Project Template

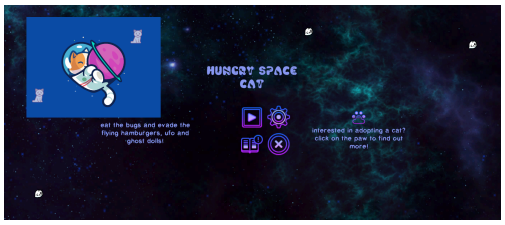
*Hungry Space Cat* builds on the 'Project Idea Title 1: Arcade Game' template.

| | |
|---|---|
|  | **Figure 1.1**: The game is cat-themed, as cats are fun and wonderfully naughty, even in space!<br><br>Source: Alla Lensky[2]. |

## *1.2: Hungry Space Cat: A Description*

*Hungry Space Cat* is a 2D feline-inspired (Figure 1.1) arcade game for the PC. In terms of gameplay, it takes light inspiration from arcade games such as Namco's *Pac-Man*, a 1980 maze action game [1], and -- regarding its appearance -- *Asteroids*, a 1976 space shooter by Atari [2] while including a more modernised look and feel. *Hungry Space Cat* uses arrow keys to control and simulate the experience of playing an arcade game: the only other control used throughout the game is ESC, which allows the player to pause it.

The game features an adorable astronaut cat trying to consume some purple space bugs while evading cute-looking enemies such as UFOs, flying hamburgers, ghost dolls, spaceships or a combination thereof. The player advances through increasingly more challenging levels by eating all available bugs without losing the cat's nine lives (Figure 1.2).

| | |
|---|---|
|  | **Figure 1.2**: The menu scene has a few buttons, an astronaut cat image[3] and a space-themed background[4]. A white ghost cat[5] further enhances the menu's design. |

Like in *Pac-Man*, the cat earns points for eating the space creatures while trying to avoid the annoyed enemies that could spell the end of its feast (and life!) [3]. The game's goal is for the cat to consume as many bugs as possible without getting hurt by its opponents. The more difficult levels include spaceships that try to shoot the space cat character that remains defenceless, with only its speed and wit allowing it to evade any shots.

The background of *Hungry Space Cat* engages a space theme to pay tribute to games such as *Space Invaders* and the previously mentioned *Asteroids*. However, realism is not at the heart of *Hungry Space Cat,* and its design and gameplay use silly/cute elements to appeal to a broader audience.

---

[2] https://www.deviantart.com/a2a5/art/Cat-in-space-station-00184-950887557.
[3] https://bevouliin.com/green-cat-astronaut-game-asset-sprites/
[4] https://screamingbrainstudios.itch.io/seamless-space-backgrounds
[5] https://kububbis.itch.io/kawaii-ghost-sprite-pack-neko-edition

The game aims to be immediately playable and fun without requiring too much investment or time for the player to understand its plot. It has two modes: one that aims to be easy and relaxing, while the other poses more of a challenge, adding momentum to the player's movements and more levels.

### *1.3: Why Hungry Space Cat?*

#### Motivation

Arcade games offer plenty of learning opportunities for aspiring programmers due to their familiarity and simplicity with graphics, giving learners a glimpse of the magic behind creating them[4]. Games such as *Pac-Man* also deal with pathfinding algorithms [4] that have inspired researchers to enhance the game with more modern and complex methods.

Simultaneously, titles such as *Asteroids* require collision detection and distance calculation, allowing for fun and visual exploration of these topics [4]. Consequently, creating an arcade game not only serves as a rewarding project for a final module but also provides a starting point for exploring other more challenging subjects, such as A* pathfinding.

On a personal note, *Hungry Space Cat* addresses the creator's affection and love for cats. Additionally, the game gives the creator a platform to hone her skills in writing games while having fun, with the long-term aim of creating ones accessible to all kinds of players.

#### Target Audience

The target audience of *Hungry Space Cat* mirrors those that Toru Iwatani, the creator of *Pac-Man*, intended for his game – in particular, he wanted it to be accessible to a diverse range of players [3]. The simplicity of *Hungry Space Cat* allows for instant immersion while offering a safe and quiet space, making it suitable for:

- ★ Busy people who want to enjoy a game without having to read instruction manuals.
- ★ Casual players who wish to play a game with a straightforward objective.
- ★ New gamers and seasoned ones alike who like challenges.
- ★ People with health issues that prevent them from playing a game for an extended amount of time.
- ★ People who enjoy light-hearted games.
- ★ Young players who enjoy cute creatures and bright colours.

#### Project Objectives

*Hungry Space Cat* aims to:

- ★ Be a game that is easy to play/pick up.
- ★ Have simple and enjoyable visuals.
- ★ Be respectful towards arcade games in their philosophy, design and approach.
- ★ Serve as a motivation for players to explore other arcade games.
- ★ Encourage people to adopt cats like the developers of An Arcade Full of Cats do.

### *1.4: Report Overview*

The literature review explores the legendary games *Pac-Man* and *Asteroids* before describing their significance in arcade games. The chapter concludes with an overview of other feline-based games and a web application before critically analysing them.

The design chapter explores the player character, enemies, and level designs for *Hungry Space Cat*. It also examines the game's flow and mechanics before discussing the main components of the overall project.

The implementation chapter covers code snippets and background information to underlie the programming logic.

In contrast, the evaluation chapter discusses the overall success of *Hungry Space Cat*, taking into account feedback from playtesting sessions and the overall project objectives.

The conclusion provides some final insights into the project.

The appendix covers sections relevant to the report that are too lengthy to be included in the main text. It contains more information about arcade games, their types and history. Moreover, the appendix covers all bugs and subsequent fixes found during playtesting sessions. It also provides a home for the work/evaluation plan, a Gantt chart, and prototype design documents that detail the various iterations the project went through.

(926 words).

**End of Chapter References**

[1] Wikipedia. Pac-Man. Retrieved March 19, 2024 from https://en.wikipedia.org/wiki/Pac-Man.

[2] Wikipedia. Asteroids (video game). Retrieved February 16, 2024 from https://en.wikipedia.org/wiki/Asteroids_(video_game).

[3] Retro Gamer. 2015. *Retro Gamer Book of Arcade Classics*. Imagine Publishing, Bournemouth, Dorset.

[4] Katrin Becker and James R. Parker. 2005. All I Ever Needed to Know About Programming, I Learned From Re-writing Classic Arcade Games.
In *Future Play, The International Conference on the Future of Game Design and Technology*, October 13-15 2005, Michigan State University, East Lansing, Michigan, 1-7.
http://hdl.handle.net/1880/46707

## 2. LITERATURE REVIEW

Please refer to the appendix for a more comprehensive definition of arcade games.

### *2.1: Pac-Man: A Cult Classic*

#### The History of *Pac-Man*

A 2008 report shows that 94 % of US consumers recognise *Pac-Man*, even beating *Mario* in popularity [1]. According to its creator, Toru Iwatani – a game designer with no formal training [1], the gentle gameplay of *Pac-Man* was intentional as, in the late 1970s, arcade centres only contained violent games focused on killing aliens [2]. He felt these arcade games were playgrounds for boys, leading him to develop one that could be played by women and couples [2].

A Japanese fairytale where a creature protected children from monsters by eating them inspired the creation of *Pac-Man's* prototype [2]. In *Pac-Man*, the monsters from the fairytale became four ghosts, each with a unique personality [2]. Furthermore, Toru Iwatani used the Kanji for eating – *taberu* 食 [3] – as a premise for the game and the Kanji for the word 'mouth' – *kuchi* 口 [4]-- with its square shape as the basis for the character's design [1][2].

**Figure 2.1:**

According to a popular urban myth, the shape of a pizza missing a piece influenced the overall shape of *Pac-Man* [1].

Source: vanked[6].

Iwatani insisted on the simplicity of *Pac-Man*'s design—a yellow disc with a basic mouth (Figure 2.1)—to streamline the game and keep the gameplay simple without the player requiring a manual [1]. Further, Iwatani restricted the gameplay to a maze where players could move up, down, left, and right [1].

#### Core Mechanics

*Pac-Man features* a two-dimensional maze, and the player controls the game character via a four-way joystick to navigate the labyrinth [5]. The maze has 240 non-flashing pills, each worth 10 points, and four flashing ones, referred to as energisers, worth 50 [5].

The game begins with the four ghosts stationed at the maze's centre, also known as the *ghost house* [6]. They are released individually, and upon contact, they chase after Pac-Man, aiming to consume them [5]. Pac-Man starts with two lives, losing one each time they are caught by a ghost [5]. When a life is lost, the ghosts and Pac-Man return to their original positions, but any eaten dots remain gone[6].

When Pac-Man eats an energiser, the ghosts turn blue briefly, allowing Pac-Man to eat them and score points [5] during the early levels of the game [6]. Upon finishing a level, the game moves to the next one [17]. While the game is technically unlimited, a bug prevents the player from surpassing level 255 [5].

[6] https://depositphotos.com/portfolio-1005524.html?content=photo

The speed of the ghosts is constant, except when travelling through a tunnel [5, 6]. The layout of the game remains the same, although each level increases in difficulty due to changes in Pac-Man's speed and the behaviour of the ghosts (Figure 2.2). There are no modifications after reaching level 21 [6].

TABLE I
CHARACTERISTICS OF THE GHOSTS IN PAC-MAN [4].

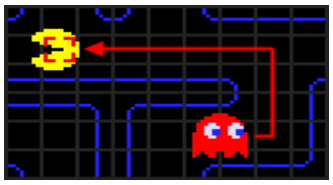
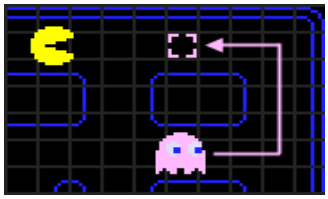
| Colour | Orange | Blue | Pink | Red |
|---|---|---|---|---|
| Name | Clyde | Inky | Pinky | Blinky |
| Aggressiveness | 20% | 50% | 70% | 90% |
| Territory | Southwest | Southeast | Northwest | Northeast |

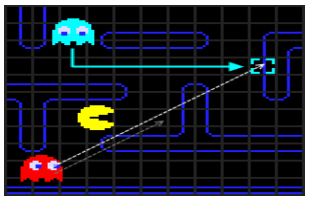
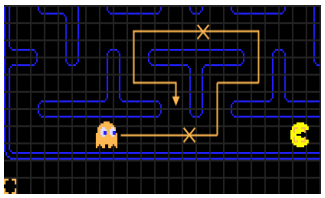
**Figure 2.2:**

The table shows the ghosts' four personalities and aggressiveness levels [5].

## Ghost Behaviour and Personalities

Each of the four ghosts has a target – a specific tile – assigned to them that they are trying to reach [6]. The colour of its eyes indicates which direction it travels, which cannot change unless Pac-Man has eaten an energiser [6]. The ghosts have three modes – scatter, chase, or frightened [5, 6], with the predominant one being chase, where the spectres use Pac-Man's position to find their target tile [6]. When in scatter or chase mode, the ghosts travel in waves with a set timer, only pausing when the ghosts are in frightened mode [6]. The table below further outlines the ghosts' behaviours:

| **Name of Ghost and Screenshot(s)** | **Core Behaviour** |
|---|---|
| Red Ghost / Blinky<br>  <br>**Figure 2.3**: The red ghost's target tile is where Pac-Man is located [6]. | Starting outside the ghost house, the red ghost, described as r 追いかけ, *oikake,* which means 'pursuer' in English, is seen as the first threat. It immediately starts following Pac-Man, as its target tile in chase mode is Pac-Man's current location (Figure 2.3) [6].<br><br>Blinky's speed increases when Pac-Man only has 20 pellets left to eat [7]. Its scatter mode also changes, ensuring it remains in chase mode all the time rather than targeting the tile at the upper right corner of the maze [6]. These behavioural shifts often make players perceive Blinky as the deadliest of the ghosts [7]. |
| Pink Ghost / Pinky<br>  <br>**Figure 2.4**: The pink ghost's target tile tries to move four spaces ahead of Pac-Man [6]. | The pink ghost starts inside the ghost house but exits immediately [6], moving around the maze's walls in an anti-clockwork pattern [7].<br><br>Although the ghost is nicknamed 'Pinky' [6, 7], the Japanese description of its personality 待ち伏せ, *machibuse* (ambusher), better suits its actual behaviour [6]. While Pinky does not move faster than the other ghosts, its target is where it anticipates Pac-Man will next be (Figure 2.4) [6].<br><br>More precisely, in chase mode, it looks at Pac-Man's current position [6] and the four spaces ahead of the player [6, 7] in an attempt to get in front of them [7]. However, this behaviour only works if Pac-Man moves left, down or right, with an overflow error preventing this behaviour when the player moves upwards [6].<br><br>In scatter mode, Pinky moves towards the top left |

| | |
|---|---|
| | corner in an anti-clockwise, circling motion around the corners of the wall [7]. |
| Blue Ghost / Inky<br>  <br>**Figure 2.5**: When calculating Inky's target, Pac-Man and Blinky's positions are considered [6]. | The blue ghost remains in the ghost house until Pac-Man has eaten 30 pellets [6]. Its Japanese description is 気紛れ, *kimagure*, which translates to 'whimsical' in English [6].<br><br>Inky's behaviour is difficult to predict as it uses Pac-Man's and Blinky's positions to calculate its target (Figure 2.5) [6]. It patrols the area close to the red ghost's location, becoming more random the further away it is from it [7]. In other words, if Blinky is close to the player, so will Inky [6]. In scatter mode, Inky guards the lower corners of the maze [7]. |
| Orange Ghost / Clyde<br>  <br>**Figure 2.6**: The image above shows how Clyde's behaviour changes when it is close to Pac-Man [6]. | The orange ghost is the last to leave the ghost house and does not move until a third of the pellets have been eaten in the first level [6]. The Japanese description for its behaviour translates to お惚け, *otoboke* or feigning ignorance' [6].<br><br>While Clyde gives the impression that it is doing its own thing, it has two modes for targeting Pac-Man that it switches depending on its proximity to the player (Figure 2.6) [6]. It first calculates how far it is from Pac-Man to determine its target. If it is eight spaces away, it behaves like Blinky, following the player directly; however, as soon as it is less than eight spaces from Pac-Man, its target chases to one of the ones it has in scatter mode, located at the bottom corner of the maze [6, 7].<br><br>The abovementioned behaviour means Clyde switches between following Pac-Man and retreating when it gets too close [6]. |

### Evaluating *Pac-Man*

Due to its immense popularity, *Pac-Man* has not only inspired various clones (e.g. *Cat Trax*) but has also been used in academic research [5]. *Pac-Man*'s easy-to-understand yet increasingly challenging gameplay shows what makes arcade games so popular and how these aspects should remain integral in any project aimed at creating one. Furthermore, its – intended – universal appeal [1] reveals how arcade games can bring a diverse community of players together and teach future generations how to code.

## *2.2: Asteroids - Atari's Response to Space Invaders*

### The Making of *Asteroids*

In December 1979 [8], Atari released the shoot-em-up *Asteroids* - the epitome of an easy-to-learn but difficult-to-master game, as people kept returning to it to stop dying quickly [9]. Not only did *Asteroids* make Atari a household name [8], but it sold 75,000 units, making it one of the best-selling coin-ops [9]. Beyond its immense financial success, *Asteroids* also helped arcade games become

popular among players from different walks of life, including professionals in their 30s and 40s who would play it during their lunch break [8]. As Atari's best-selling game, it dominated not only the scene in arcades but also areas such as waiting rooms, bars and shopping malls, amongst other locations [8].

Created by Ed Logg (Figure 2.7), the developer behind the 1988 *Super Breakout, Asteroids,* was a response to an unsuccessful game where players tried to shoot asteroids [8, 9]. Ed described the game as unfun and suggested an alternative: players should blow up the asteroids instead [9]. During a conversation with Lyle Rains, Ed stated in contrast to a successful title such as *Space Invaders* -- a game with one-directional controls that only let the player move left and right -- he envisioned a game that allowed for two-directional movements, creating more player satisfaction [9]. The proposed game had simple yet addictive gameplay involving a little flying spaceship and rocks that became smaller upon being hit until they disappeared altogether [8, 9].

**Figure 2.7:** Ed Logg poses next to *Gold Asteroids*, created to celebrate building 50,000 units [8].

While Ed and Lyle quickly agreed on the concept for *Asteroids*, they did not initially see eye to eye when it came to the game's format [8, 9] – Lyle wanted the game to use raster, while Ed wanted to utilise vector technology, which – according to him – had a higher resolution and thus allowed for more control [9]. To use vector technology on hardware, Ed tracked down the engineer Howard Delman [8, 9], who had worked on Atari titles such as *Super Bug* and *Lunar Lander* -- the company's first game to use vector technology [9].

### Ed Logg and His Approach to Game Design

While the hardware of *Asteroids* was being created, Ed Logg set about documenting the game's basic concepts [8], focusing on its design and tweaking various ship settings, such as its inertia, to find out what worked best in terms of gameplay [9].

As the Atari labs were open, engineers often walked between them and played games currently under development [9]. Consequently, Ed Logg received feedback from his colleagues, which was overwhelmingly positive [9]. However, Logg would also observe his colleagues playing and make amendments, such as moving the player away from the rocks to ensure that they hit more objects, as these increased the chance of collision [8, 9].

While there were no design processes in place in this era, Ed sketched many versions of the ship, and – as noted by Mark Cerny – a colleague of Ed's at Atari, the secret to his success was that he planned the game, developing features in the correct order rather than focusing on complex algorithms [8]. Ed also engaged in feedback from two field playing testing sessions to fine-tune his game [9].

### Evaluating *Asteroids*

While *Asteroids* has not enjoyed the same academic interest as *Pac-Man*, the game continues to be popular [8, 9]. Like *Pac-Man*, *Asteroids'* simple story and difficult-to-master gameplay show that these qualities make up a successful arcade game. Additionally, how Ed Logg designed and created his game shows that a project of such a nature is an incremental process that relies on player feedback to become a success.

### *2.3: Arcade-Based Cat Games*

#### Cat Trax

*Cat Trax* (Figure 2.8), released in 1982 or 1983[7], is a clone of the game *Pac-Man*, featuring a cat on the run from three canines [10, 12]. To gain points, the kitty has to eat the catnip in the maze, and – anytime a green potion appears – it will transform into a dog catcher truck that sends any captured dogs to a pound at the top of the screen [5, 6]. Once the potion wears off, the dogs start chasing the game character again [5, 6]. *Cat Trax* was developed for the Aradia 2001 [10] – an obscure home game system released in 1982 by UA.Ltd [11].

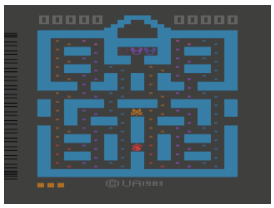


**Figure 2.8**: An image of the arcade game *Cat Trax* [11].

*Cat Trax* has a fun take on *Pac-Man*; however, it remains a clone with little to set it apart regarding creativity or originality. While games such as *Pac-Man* inspired *Hungry Space Cat*, it still aims to be something of its own, emphasising its space-centred/feline theme.

#### Mappy

Released by Namco in 1983 [13, 14], *Mappy (マッピー)* was a side-scrolling maze game featuring cartoon-like characters like cats and mice [13, 14]. It ran on the *Super Pac-Man* hardware modified to support horizontal side-scrolling [13, 14].

**Figure 2.9:** A screenshot of the arcade game *Mappy* [13].

In the game, the player guides the police mouse, Mappy (Figure 2.9) – a term derived from the Japanese nickname *mappo* [9] – through a mansion owned by cats called Mewkies to find stolen goods [13, 14]. To survive, Mappy must avoid the mansion's cats and traverse the building via trampolines [13, 14].

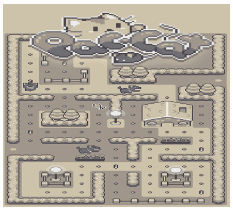
*Mappy* is more involved and complex than *Cat Trax*, but its story and motive are simple, not allowing the slightly more enhanced gameplay to get in the way. *Hungry Space Cat* builds on the philosophy of *Mappy* by extending a simple story with more elevated graphics.

#### Pac Cat

*Pac Cat* by Divok (Figure 2.10) is an indie arcade game with pixel graphics for mobile devices. In it, a cat has to eat all the points to pass the levels while running from bulldogs [15]. Reviews of the game describe it as cute but buggy, with the controls not working and the game being challenging from the get-go [15].

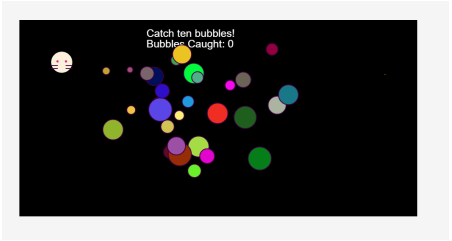
[7] The literature is not entirely clear here, see [11].

Consequently, *Pac-Cat* defeats the purpose that creators of games such as *Pac-Man* had in mind: for them to be simple but become more complex with increasing levels [3]. *Hungry Space Cat* starts simple but becomes more difficult at each level, thus keeping in line with the characteristics of an arcade game.

| | |
|---|---|
|  | **Figure 2.10**: An image of the indie game *Pac Cat* [15]. |

### *2.4 Web-Based Game Project(s)*

#### Space Cats

| | |
|---|---|
|  | **Figure 2.11**: The user can play one of the two games in *Space Cat* [16]. |

*Space Cats* [16] is a game web application where users can play games and view interactive art (Figure 2.12). The application aims to bring people together who enjoy cute/silly games. However, playtesters for the project described the gameplay of these two games as excessively simple and needing more substance. Using a web browser also affected the games' graphics in size/resolution, restricting gameplay further.

(2412 words).

**End of Chapter References**

[1] Retro Gamer. 2015. *Retro Gamer Book of Arcade Classics*. Imagine Publishing, Bournemouth, Dorset.

[2] Jamey Pittman. 2001. The Pac-Man Dossier. (April 2015). Retrieved February 12, 2024 from https://pacman.holenet.info.

[3] NIHONGO ICHIBAN. Kanji Card – 食. Retrieved February 12, 2024 from https://nihongoichiban.com/2011/04/10/jlpt-kanji-食/.

[4] NIHONGO ICHIBAN. Kanji Card – 口 – kuchi. Retrieved February 12, 2024 from https://nihongoichiban.com/2011/04/09/jlpt-kanji-口/.

[5] Philipp Rohlfshagen and al. 2017. Pac-Man Conquers Academia: Two Decades of Research Using a Classic Arcade Game. *IEEE Transactions on Games* 10, 3 (December. 2017), 233 - 256. https://ieeexplore.ieee.org/document/8207594.

[6] Chad Birch. 2010. Understanding Pac-Man Ghost Behavior. (December 2010). Retrieved April 1, 2024 from https://gameinternals.com/understanding-pac-man-ghost-behavior.

[7] Jason Brown. 2023. All Pac-Man Ghost Names and What They Do. (December 2023). Retrieved May 12, 2024 from https://retrododo.com/all-pac-man-ghosts/.

[8] Arcade Blogger. 2018. Atari Asteroids: Creating a Vector Arcade Classic. (March 2021). Retrieved May 11, 2024 from https://arcadeblogger.com/2018/10/24/atari-asteroids-creating-a-vector-arcade-classic/

[9] Retro Gamer. 2009. The Making of Asteroids. *Retro Gamer*, 68 (October 2009), 24-29.

[10] LaunchBox. Games Database. CAT TRAX. Retrieved February 17, 2024 from https://gamesdb.launchbox-app.com/games/details/76418-cat-trax

[11] atariprotos. Cat Trax. Retrieved March 30, 2024 from https://www.atariprotos.com/2600/software/cattrax/cattrax.htm

[12] TrekMD. 2016. Cat Trax. (June 2016). Retrieved March 30, 2024 from https://www.retrovideogamer.co.uk/cattrax2600/.

[13] Namco Wiki. Mappy. Retrieved February 18, 2024 from https://namco.fandom.com/wiki/Mappy.

[14] Namco Wiki. Mappy. Retrieved February 18, 2024 from https://www.retrogames.cz/play_008-NES.php.

[15] Google Play. Pac Cat: retro cat. Retrieved February 17, 2024 from https://play.google.com/store/apps/details?id=com.Divok.PacCat&hl=en&gl=US.

[16] HedonisticOpportunist. SPACE CATS - A GAME WEB APPLICATION. Retrieved May 12, 2024 from https://github.com/HedonisticOpportunist/Space-Cats.

## 3. GAME DESIGN

Please refer to the appendix for more theoretical discussion on designing a game.

### *3.1: The Game's Domain and Its Users*

*Hungry Space Cat's domain* is arcade games—games developed to have a simplistic design but challenging to master gameplay [1]. Throughout their history, not only did the unsophistication of early arcade games allow the player to grasp all of a game's intricacies, but they also made them feel accomplished and eager to replay them to encounter new obstacles to solve [1]. In contrast, modern games have excessive tutorials that affect a game's pacing, robbing players of the opportunity to feel challenged by cracking problems independently [1].

While *Hungry Space Cat's intended audience* is gamers of a more casual kind who want to play something without requiring a manual, the goal is to keep them engaged in the story and want to return for more. In essence, the game's design elements—such as character and level aspects—are intended to be straightforward, with the goals and game mechanics similarly easy to gauge. Nevertheless, the gameplay aims to remain challenging to keep the user's interest, even with the focus on clarity.

### *3.2: Hungry Space Cat*: Core Mechanics

The following table outlines the core mechanics of *Hungry Space Cat*. With each new level, the number of bugs to eat increases, and the type of enemy changes, making the game less predictable. The player can choose between an accessible/easy and challenging/hard mode, influencing how the space cat moves, with the more difficult mode employing momentum.

| **Action** | **Action Triggered** |
|---|---|
| In difficult mode, the space cat moves up, down, left, or right using momentum. In easy mode, the player can stop and start instantaneously. | The cat navigates within the screen's boundaries, restricting its movements like the player confined to a maze in *Pac-Man* [2]. |
| The cat eats space bugs.<br><br>The cat has no weapons or defences against the enemy. | The cat gains 10 points for eating bugs, paying tribute to *Pac-Man* by making an action similar to the player eating dots [2]. |
| Depending on the levels, various enemies chase the cat. The main ones are:<br><br>Four UFOs travel back and forth at the corner of the screen. The space cat has to avoid them while trying to eat bugs.<br><br>Four flying hamburgers and the UFOs from level one chase after the astronaut cat as it tries to hunt bugs.<br><br>Ghosts—who use waypoints to navigate the scene—serve as obstacles for the cat while it | Like *Pac-Man*, the cat loses nine lives, but the enemies do not run away when a pellet is eaten [2], as the bugs serve as a means to proceed to the next level.<br><br>Instead, *Hungry Space Cat* interprets the four ghosts as three different enemies, offering versatility like this. |

| | |
|---|---|
| tries to collect the bugs.<br><br>Randomly spawned planets enhance the background., while UFOs and ghosts chase after the cat.<br><br>A spacecraft follows the player and shoots at it.<br><br>A wave of spacecraft shoots at the player. | |

***Hungry Space Cat*: Gameplay**

Please refer to the appendix for a visual representation of the gameplay envisioned as a prototype. The gameplay of *Hungry Space Cats* is a sequence of the following:

- ★ The cat enters the scene from any random location on the screen within its boundaries.
- ★ The cat moves up, down, left or right to look for and then eats bugs.
- ★ The UFOs enter the scene from the edges of the screen and then move back and forth within the screen's boundaries.
- ★ The flying hamburgers enter the scene from various screen areas but follow the player.
- ★ The ghosts move based on wave points, while the UFOs – as in their scene – appear on the edges of the screen and move back and forth.
- ★ The spaceship that shoots follows the player.
- ★ Green skull UFOs move based on wave points.
- ★ The various spacecraft floating around the scene enter from the edges of the screen before disappearing at the corner. These spaceships also shoot at the player.

### *3.3: Game Components*

The table below shows the game components, including the UI interface. In terms of the tools, the game uses Unity's game engine [3] along with the C# programming language [4]. Unity [3] was the best choice because previous modules taught its basics. Moreover, the Unity community has many tutorials that offer support and guidance.

***Hungry Space Cat*: Game Components**

The following table outlines the game components of *Hungry Space Cat*, each dealing with a game object and their related actions. Please note that these components do not necessarily reflect the implementation on a one-by-one basis but are more of a guideline for the game's structure.

| **Component** | **Meaning / Significance** |
|---|---|
| SpaceCat | The *SpaceCat* component deals with the space cat's movements and reaction to the enemies, especially regarding collision. |
| Enemies | The *Enemies* component handles the individual movements of each enemy and how they appear in the scene. |
| Bugs | The *Bugs' components* deal with their movement and |

| | |
|---|---|
| | reaction to encountering the space cat. It also deals with how they first appear in the scene. |
| AudioManager | The *AudioManager* deals with the sound effects used during the game. |
| SceneLoadingManager | The *SceneLoadingManager* deals with how the scenes are loaded. |
| ScoreManager | The *ScoreManager* component controls how the user's score and life appear on the screen, including any changes. |

### *3.4: Character Design and Level Design*

To view prototypes of the menu, please refer to the appendix.

#### *Hungry Space Cat:* Player, Pickup and Enemy Designs

Like the original *Pac-Man*, the character and level design are light-hearted [2] but of a higher quality as the game is not restricted in terms of what it can display. The game also has a distinctly anime-inspired feel, paying tribute to *Pac-Man* and other Japanese arcade games. To fit in with the space theme, the player plays the role of a cute space cat that walks upright, emphasising the game's light-hearted atmosphere.

Apart from the UFOs [5], all of the sprites were sourced from *bevouliin*, a site that features game assets for game developers [6]. Using the same artist for most sprites keeps the game design consistent.

The table in the appendix provides an overview of all the sprites used for the character design

#### *Hungry Space Cat:* Level Design

The game scene has a scrolling background, creating the illusion of the player being in space. Early versions of the game design involved using a tilemap to draw borders around the screen. However, tiles were abandoned in favour of a more straightforward but effective space-related background [7] that required less work, especially as tile maps proved inflexible in the light of frequent changes.

The table in the appendix provides an overview of all the sprites used for the level design.

### *3.5: Project Management*

For a view of the work plan, please review the appendix.

#### Gantt Chart

Please refer to the appendix to view the Gantt charts related to *Hungry Space Cat.*

### *3.6: Evaluation*

For a view of the evaluation plan, please review the appendix.

(1155 words).

**End of Chapter References**

[1] bdatg2. 2013. Simplicity. (December 2013). Retrieved February 20, 2024 from https://vgprobs.wordpress.com/simplicity/.

[2] Jamey Pittman. 2001. The Pac-Man Dossier. (April 2015). Retrieved February 12, 2024 from https://pacman.holenet.info.

[3] Unity. Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine. Retrieved February 21, 2024 from https://unity.com.

[4] Microsoft. C# documentation. Retrieved February 21, 2024 from https://learn.microsoft.com/en-us/dotnet/csharp/.

[5] kububbis. 2024. Kububbis - itch.io. Retrieved March 23, 2024 from https://kububbis.itch.io.

[6] bevouliin. 2013. Bevouliin - Selling 2D Game Assets for Game Developers. (July 2013). Retrieved March 23, 2024 from https://bevouliin.com/.

[7] screamingbrainstudios. 2023. Screaming Brain Studios - itch.io. Retrieved March 23, 2024 from https://screamingbrainstudios.itch.io.

## 4. GAME IMPLEMENTATION

### *4.1: Overview of the Code Structure*

*Hungry Space Cat* spreads its scripts across several folders to organise the project and structure the code so that its purpose/function remains within an appropriate area. The scripts fall into the following categories[8]:

- ★ Animation,
- ★ Audio,
- ★ Controllers,
- ★ GamePlay,
- ★ Helpers,
- ★ Spawners,
- ★ UI.

### *4.2: Animation*

#### The *AnimatedSprite* Script

| | |
|---|---|
|   | **Figure 4.1:**The UFO and space bug sprites use the *AnimatedSprite* script to show visual changes. |

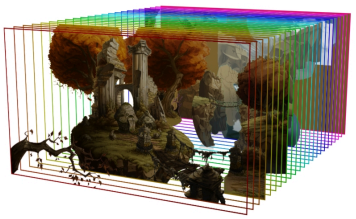
The following script is responsible for animating the sprites, including how an array of them is displayed during the game to reflect changes in movement over time [1]. The script is attached to the game objects, such as the space cat, enemy, and bugs (Figure 4.1).

| **Function/Code Snippet** | **Purpose** |
|---|---|
| void MoveAnimationForward()<br>{<br>…<br>{<br>animationFrame++;<br>CheckAnimationFrameConditions();<br>}<br>} | The following function increments the animation frame variable by one when the game object's *SpriteRenderer* component is enabled and the animation frame conditions have been met (see the *CheckAnimationFrameConditions* function in the next row below) [1].<br><br>A *SpriteRenderer* component is responsible for how a sprite is rendered and depicted on a screen [2]. |
| void CheckAnimationFrameConditions()<br>{<br>…<br>if (animationFrame >= 0 && animationFrame < spritesArray.Length && !PauseMenu.isPaused)<br>{<br>spriteRenderer.sprite = spritesArray[animationFrame];<br>} | The *CheckAnimationFrameConditions* function, called in the *MoveAnimationForward* method, checks that animation continues seamlessly if the sprite array is smaller than the animation frame by setting its current index to the one of the animation frame [1]. It also checks that the game is running/not paused [3].<br><br>If the animation frame is larger than the length of the |

[8] It is beyond the scope of this report to cover all of the directories or their relevant scripts.

| | |
|---|---|
| } | sprite array and smaller or equal to 0, then the animation frame is set to zero so that the sprite animation can continue looping again [1]. |

**The *BackgroundSpriteScroller* Script**

| | |
|---|---|
|  | **Figure 4.2**: The following image shows how parallax scrolling looks from the front using the layering method [4]. |

The *BackgroundSpriteScroller* helps with parallax scrolling, in which background images move more slowly than forefront ones [4]. Parallax scrolling became popular in the early 1980s with video arcade games such as *Jump Bug* [4]. Parallax scrolling uses layering (Figure 4.2), which involves multiple backgrounds and changing each layer's position by a different amount in the same direction [4].

The other method engages pseudo-layers using sprites drawn by hardware on top of layers [4]. To honour arcade games, *Hungry Space Cat* uses the layering method as shown in the table below:

| **Function/Code Snippet** | **Purpose** |
|---|---|
| void Update()<br>{<br>if (Effects.backgroundEffectEnabled)<br>{<br>_offset = moveSpeed * Time.deltaTime;<br>_backgroundMaterial.mainTextureOffset += _offset;<br>}<br>}<br>} | Within the script's *Update function*, a private offset variable is multiplied by the move speed and Time.deltaTime to create smooth movement [5].<br><br>Afterwards, the offset is added to the background material's texture offset to create the scrolling effect [5].<br><br>A static boolean checks whether this option has been triggered (the user has to choose to turn off the scrolling background effect in the *GameSettings* menu). |

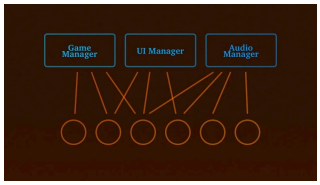
To ensure smooth movements with the scrolling backgrounds, Unity uses *Time.deltaTime* (Figure 4.3), which measures the interval between the current frame and the last one, ensuring that the animations and movement of game objects are consistent across platforms and different forms of hardware [6]

| | |
|---|---|
| $\text{Time.deltaTime} = \frac{1}{\text{frame rate}}$ | **Figure 4.3:** The formula helps normalise the calculation of the rate at which a machine renders a frame [5]. Calculating a game object's vector in Unity with the following equation returns a fixed speed, no matter the frame rate [6]. |

***4.3: Audio***

**The *AudioPlayer* Script**

The *AudioPlayer* script, attached to an empty game object of the same name, handles the game's background music. The script is a *singleton* (Figure 4.5), making it globally accessible but unique so that no other game object can create an instance [7].

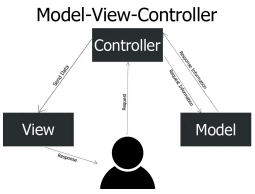
| | |
|---|---|
|  | **Figure 4.5:** While *singletons* are controversial, using them as single-game component managers is practical and avoids writing tightly connected scripts [7]. |

| **Function/Code Snippet** | **Purpose** |
|---|---|
| void PlayAudioClip(AudioClip audioClip, float volume)<br>{<br>…<br>AudioSource.PlayClipAtPoint(audioClip, cameraPosition, volume);<br>…<br>} | The *PlayAudioClip* function is reused within the script to play different background clips within game scenes [8].<br><br>More precisely, it uses a static method called PlayClipAtPoint that plays a clip from a given position in world space before disposing of it entirely [9]. |

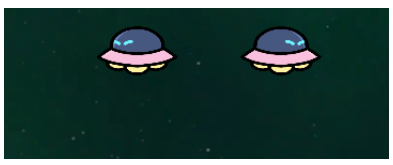
### *4.4: Controller Scripts*

Controller scripts come from the MVC pattern (Figure 4.6), which divides programming logic into three separate parts:

- ★ The model (the representation of information).
- ★ The view (interface).
- ★ The controller links the first two items together [10].

| | |
|---|---|
|  | **Figure 4.6:** The following images show how the MVC pattern breaks down an application into separate parts that reflect the logic flow [10].<br><br>Source: Medium[9]. |

While *Hungry Space Cat* does not fully implement the MVC pattern, the scripts in the *Controller* folder are responsible for updating the enemies' movements, pickups, and the player. It also deals with how game objects react to others, especially regarding collision and ensuring that the UFO does not leave the scene's boundaries (Figure 4.7).

| | |
|---|---|
|  | **Figure 4.7:** Code within the *Controller* script folder ensures that the UFOs do not leave the boundaries but change directions when 'colliding' with it. |

[9] https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5

### *Controller*-Script-Based Functions

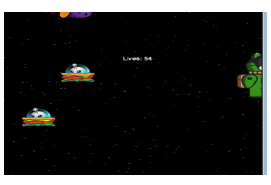
The table below lists some of the functions across all the *Controller* scripts.

| Function/Code Snippet | Purpose |
|---|---|
| if (other.CompareTag("Boundary") && !PauseMenu.isPaused)<br>{<br>speed = -speed;<br>_controllerHelper.FlipSprite(transform, _body);<br>…<br>}<br>…| In *UFOController*, using a reference to the *ControllerHelper* script, the UFO flips (Figure 4.7) and changes directions (as evidenced by the speed changing from a positive or negative value).<br><br>The code snippet is placed inside Unity's *OnTriggerExit2D* function, which handles how objects are triggered when another one leaves its trigger collider [11]. |
| if (other.CompareTag("SpaceCat") && !_wasEaten && !PauseMenu.isPaused)<br>{<br>_wasEaten = true;<br>_audioPlayer.PlayPickupClip();<br>_scoreKeeper.ModifyScore(pointsForBugsEaten);<br>gameObject.SetActive(false);<br>Destroy(gameObject);<br>} | The following code snippet in the *BugController* script shows how the space cat object coming into contact with the bug leads to a series of consequences, which include:<br><br>- The activation of the SFX audio pickup clip.<br>- The score's modification [13].<br>- The deletion of the bug itself [14].<br><br>The following code snippet is located in Unity's *OnTriggerEnter2D* function, which deals with how objects are triggered when they enter a trigger collider [12]. |
| if (other.CompareTag("UFO") && !PauseMenu.isPaused)<br>{<br>…<br>if (_healthKeeper.GetLives() == 0)<br>{<br>CatDeath();<br>}<br>} | The following code snippet, located with the *OnTriggerEnter2D* function of the *SpaceCatController* script, ensures that the space cat dies if its lives equal 0. |

## *4.5: Helper Scripts*

### The *ControllerHelper* Script

The *ControllerHelper* script contains reusable functions related to actions that make a game object behave in a certain way under specific condition, as shown in the table below:

| | |
|---|---|
|  | **Figure 4.8***:* The flying hamburgers pursue the space cat via its movements/position in the scene. |

| Function/Code Snippet | Purpose |
|---|---|
| public void ClampSpriteMovements(Transform transform)<br>{<br>Vector2 newPos = new()<br>{<br>x = Mathf.Clamp(transform.position.x, minimumBounds.x, maximumBounds.x),<br>y = Mathf.Clamp(transform.position.y, minimumBounds.y, maximumBounds.y)<br>};<br><br>transform.position = newPos;<br>} | One of the game prototypes used an empty boundary object to keep the player within the screen; however, that method was less effective than using Unity's *Clamp* function, which gives a minimum and maximum float value [15].<br><br>More specifically, the *Clamp* function defines a scene's x and y bounds and keeps a game object within it with these specified coordinates [16], as shown in the code snippet on the left. |
| public void FollowPlayer(Transform target, Transform transform, float speed, int lives)<br>{<br>…<br>Vector2 direction = target.position - transform.position;<br>direction.Normalize(); // Keeps the length of the direction to one, thus constant.<br>float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg; // This allows for smoother rotation.<br><br>transform.SetPositionAndRotation(Vector2.MoveTowards(transform.position, target.transform.position, speed * Time.deltaTime), Quaternion.Euler(Vector3.forward * angle));<br>… | Initial game prototypes did not include rotation towards the player. Normalising the direction between the player and the target position helps give a constant distance [17]; this helps avoid bugs when rotating the flying hamburgers towards the player (Figure 4.8) [18].<br><br>Further, the *Math.Atan* function [19] is applied to the x and y coordinates of the direction vectors to calculate their inverse tangent before being converted to degrees; this helps make the rotation of the enemy following the player smoother [19]. |

### The SpawnerHelper Script

The *SpawneHelper* script helps several game objects spawn across a scene. Several game objects, like planets and space bugs, use the *SpawnObjectsRandomly* function below.

| Function/Code Snippet | Purpose |
|---|---|
| for (int i = 0; i < objects.Count; i++)<br>{<br>…<br>Vector2 spawnPosition = new(Random.Range(minPosition.x, maxPosition.x), Random.Range(minPosition.y, maxPosition.y));<br><br>if (objects[i] != null && target != null)<br>{<br>float distance = GetDistanceFromPlayer(objects[i], target);<br><br>if (distance != 0)<br>{ | The *SpawnObjectsRandomly* function takes a list of game objects as an argument. Then, within a for loop, a Vector2 position is set that transforms random x and y coordinates into x and y positions within 3D space [20].<br><br>In order to avoid spawning on top of the player, the function also checks how far the potentially spawned object is from it. |

| | |
|---|---|
| Instantiate(objects[i], spawnPosition, Quaternion.identity);<br>}<br>}<br>…<br>} | |

(1494 words).

End of Chapter References


[1] zigurous. 2023. AnimatedSprite. (October 2023). Retrieved April 2, 2024 from https://github.com/zigurous/unity-pacman-tutorial/tree/main/Assets/Scripts.

[2] Unity Documentation. Sprite Renderer. Retrieved April 3, 2024 from https://docs.unity3d.com/Manual/class-SpriteRenderer.html.

[3] BMo. 2020. 6 Minute PAUSE MENU Unity Tutorial. (May 2020). (Retrieved April 28, 2024 from https://www.youtube.com/watch?v=9dYDBomQpBQ.

[4] Wikipedia. Parallax scrolling. Retrieved April 2, 2024 from https://en.wikipedia.org/wiki/Parallax_scrolling.

[5] Gary Pettie. 2021. 17. Scrolling Background. (July 2021). Retrieved April 4, 2024 from https://gitlab.com/GameDevTV/unity2d-v3/laser-defender/-/blob/master/Assets/Scripts/SpriteScroller.cs.

[6] educative. What is "Time.deltaTime" in Unity?. Retrieved April 13, 2024 from https://www.educative.io/answers/what-is-timedeltatime-in-unity.

[7] John French. 2023. Singletons in Unity (done right). (May 2023). Retrieved April 4, 2024 from https://gamedevbeginner.com/singletons-in-unity-the-right-way/.

[8] Gary Pettie. 2021. 25. Singleton Pattern. (August 2021). Retrieved April 4, 2024 from https://gitlab.com/GameDevTV/unity2d-v3/laser-defender/-/blob/master/Assets/Scripts/AudioPlayer.cs.

[9] Unity Documentation. AudioSource.PlayClipAtPoint. Retrieved April 4, 2024 from https://docs.unity3d.com/ScriptReference/AudioSource.PlayClipAtPoint.html.

[10] Wikipedia. Model–view–controller. Retrieved April 4, 2024 from https://en.wikipedia.org/wiki/Model–view–controller.

[11] Unity Documentation. MonoBehaviour.OnTriggerExit2D(Collider2D). Retrieved April 4, 2024 from https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerExit2D.html.

[12] Unity Documentation. MonoBehaviour.OnTriggerEnter2D(Collider2D). Retrieved April 13, 2024 from https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnTriggerEnter2D.html.

[13] Rick Davidson. 2021. additional bug fix for coin pickup. (November 2021). Retrieved April 4, 2024 from https://gitlab.com/GameDevTV/unity2d-v3/tilevania/-/blob/master/Assets/Scripts/CoinPickup.cs.

[14] Rick Davidson. 2021. 37 Scene Persist. (September 2021). Retrieved April 4, 2024 from https://gitlab.com/GameDevTV/unity2d-v3/tilevania/-/blob/master/Assets/Scripts/GameSession.cs.

[15] Unity Manual. Mathf.Clamp. Retrieved March 30, 2024 from https://docs.unity3d.com/ScriptReference/Mathf.Clamp.html.

[16] Gary Pettie. 2021. 13. Player Shooting. (July 2021). Retrieved March 23, 2024 from https://gitlab.com/GameDevTV/unity2d-v3/laser-defender/-/blob/master/Assets/Scripts/Player.cs.

[17] Stackoverflow. What is the correct way for an enemy to chase player?. Retrieved March 30, 2024 from https://stackoverflow.com/questions/65784085/what-is-the-correct-way-for-an-enemy-to-chase-player

[18] MoreBBlakeyyy. 2022. Unity simple 2D Enemy AI Follow Tutorial. (January 2022). Retrieved March 30, 2024 from https://www.youtube.com/watch?v=2SXa10ILJms.

[19] Unity Manual. Mathf.Atan. Retrieved March 30, 2024 from https://docs.unity3d.com/ScriptReference/Mathf.Atan.html.

[20] Unity Documentation. Camera.ViewportToWorldPoint. Retrieved April 6, 2024 from https://docs.unity3d.com/ScriptReference/Camera.ViewportToWorldPoint.html.

## APPENDIX

### *SECTION A: ARCADE GAMES*

#### Arcade Games: Definition

Due to the versatility of genres and technological advances within the gaming industry, a generalised definition of the *arcade game* remains complicated [1]. What generally sets them apart from modern video games is their simplicity in terms of graphics, with an emphasis placed on gameplay, storylines, and notable characters [2].

Another trait of arcade games is their presence in public places such as malls or restaurants and the fact they are coin-operated [3], designed to play one game at a time [4]. Further, while arcade games can be video-based, they also relate to pinball machines or electromagnetic games [3].

#### Characteristics of Arcade Games

Arcade games share the following qualities: intuitive and straightforward controls with equally simple physics, short levels that increase in difficulty with the game's progression, and a focus on gameplay [3, 4]. They are described as easy to learn but hard to master, and their simplicity is often deceptive[10] [4].

The purpose of intuitive controls is for the player to pick up the game without requiring any previous background knowledge, thus making games of its kind appropriate for any age group [4]. Meanwhile, short levels and high scores stored on a leatherboard table are designed to keep the player invested in the game and intrigued enough to continue [5]. Similarly, the game's increasing difficulty with each level and the remaining lives help the player feel challenged enough to keep scoring highly [5].

#### The Five Categories of Arcade Games

Arcade games belong into five categories or subgenres: maze/chase games, ping-pong games, shooters, puzzlers, and platformers [1].

#### Maze/Chase

Maze/chase games (see Figure 2.1) involve a player navigating an on-screen character through a maze of obstacles to achieve a goal while fleeing enemies [1]. These games can range from simple to complex, with the player's strategy proving more critical than their speed [1].

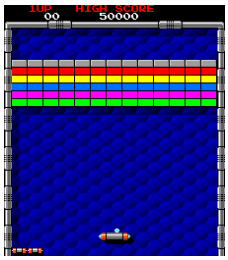
Moreover, maze games, contained to one game screen/playfield, involve little to no scrolling; multiple game levels keep the player's interest alive [1]. Famous representatives of maze/chase games are Namco's *Pac-Man*, released in 1980, and *Lady Bug*, published in 1981 by Universal [5].

[10] PC or console games with these characteristics can also be called arcade games [3].

| | |
|---|---|
|  | **Figure 1:**<br><br>A screenshot of a maze/chase arcade game called *Lady Bug*, released in 1981 by Universal.<br><br>Source: Wikipedia[11]. |

## Ping-Pong Games

Ping-pong games (see Figure 2.2), also known as bat-and-ball games, were among the first arcade games developed, with their first iterations dating back to the 1970s [1], with *Pong* released in 1976, being one of the earliest ones [4, 6].

| | |
|---|---|
|  | **Figure 2:**<br><br>A screenshot of *Arkanoid* released in 1986 by Taito.<br><br>Source: Wikipedia[12]. |

Ping-pong games involve one or two players manipulating a paddle on the screen to hit a ball; the other objective is to manoeuvre the ball past the opposing player to score points [1]. More complex versions of ping-pong also have the player use the ball to destroy walls or game creatures [1].

Pinball machines, seen as representatives of this genre, contain games with balls manipulated by paddles or flippers [1]. Usually, the action in ping-pong games takes place on static screens or playfields with no scrolling, and – like other arcade games – they contain multiple levels with varying difficulty levels [1]. *Pong* and *Arkanoid* [1] are ping-pong games released in 1972 (Atari) and 1986 (Taito), respectively [7, 8].

## Shooters

Arcade game shooters (see Figure 2.3), also called *shoot-em-ups*, were developed shortly after the first pong games [1]. They reached the height of their popularity from the mid-1980s till the early 1990s [1]. Shooter games feature one or more players controlling a character's movements on a screen [1]. The character acts as an aggressor or defender against a horde of enemies they must destroy [1].

Shooters can be *static* or *scrolling* [1]. *Static* shooters limit the player to a single area with a fixed location to shoot at their enemies [1]. In *scrolling* shooters, players are given more free range in movement by allowing them to scroll the playfield in one or more directions; most scrollers feature two—or four-way screen scrolling. Shooters have multiple levels, referred to as waves [1].

[11] https://en.wikipedia.org/wiki/Lady_Bug_(video_game)#/media/File:ARC_Lady_Bug.png
[12] https://en.wikipedia.org/wiki/Arkanoid#/media/File:Arkanoid.png

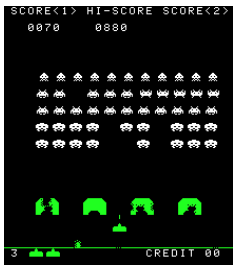



**Figure 3**: A screenshot of *Space Invaders* released in 1978 by Taito.

Source: Wikipedia[13].

Famous representatives of space shooters include *Space Invaders*, released in 1978 by Taito; it was also one of the first video games to use a joystick and sprites [9]. Another shooter is *Contra*, which Konami made available on arcade machines in 1982 [9].

## Puzzlers

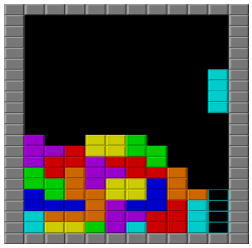


**Figure 4**: A screenshot of *Tetris* created in 1985 by Alexey Pajitnov.

Source: Wikipedia[14].

Puzzlers (see Figure 2.4) are arcade games that use puzzles to advance their plot and do not require fast reflexes on the part of the player to play or enjoy them [1].

Consequently, puzzlers work differently from other arcade games, as the player solves various puzzles to finish an objective [1]. Puzzlers use one playfield and multiple game levels – even more so than other arcade games, as they rely on these levels to keep the player interested [1]. *Blockbuster* – created by Alan Griesemer and Stephen Bradshaw – was a computerised version of the Rubik Cube puzzle, while *Tetris*, released in 1985 by Alexey Pajitnov, made the puzzle arcade game famous [10].

## Platformers

While platformers (see Figure 1.5) serve as symbols for arcade games, they did not become popular until the 1980s with the rise of 16-bit-home computers and video game systems [1].

All platformers share that they have the player control a character on-screen who has to overcome various challenges and score points to achieve an objective [1]. Platformers also contain game elements such as platforms, bonus objects, obstacles, and enemies [1]. Like shooters, platformers also come in two forms: *static* and *scrolling* [1].

In *static* platformers, the player faces the constraint of having one playfield, but using multiple game levels – called *worlds* – compensates for that lack of flexibility and challenges the player [1]. Scrolling platformers use scrolling, allowing players to scroll from left to right, though some allow for multi-dimensional scrolling [1].

[13] https://en.wikipedia.org/wiki/Space_Invaders#/media/File:SpaceInvaders-Gameplay.gif
[14] https://en.wikipedia.org/wiki/Tetris#/media/File:Typical_Tetris_Game.svg

| | |
|---|---|
|  | **Figure 5**: A screenshot of *Cadash* released in 1989 by Taito.<br><br>Source: Wikipedia[15]. |

*Cadash* (1989)—an early Taito RPG platformer [10]—and Capcom's *Black Tiger* (1987) are representative of this genre [12].

### The History of Arcade Games

In 1969, two office mates, Nolan Bushnell and Ted Dabney, started developing the first video game -- an attempt to fuse their love for computer technology with arcade gaming [13]. In August 1971, they tested the prototype of *Computer Space* -- the first video arcade game -- at a local college bar called the Dutch Goose [13].

After enjoying decent success with *Computer Space*, Bushnell and Dabney founded the engineering firm Syzygy Company in 1972 [13]. This firm would later become Atari Inc. due to the success of *Pong* by the company's first engineer, Al Alcorn [13]. With its simple controls and gameplay, *Pong* caused other companies to pay attention to the potential of arcade machines as a medium for video games, leading to the creation of new companies with their version of the game [13].

Early arcade games did not use microprocessors or software [13]. Instead, they relied on logic chips [13]. It was not until Midway's *Gun Fight* in 1975 that video arcade games would use microprocessors [13]. However, despite technological advancements, the public did not take notice of video arcade games until the mid-1970s with the introduction of TV-tennis home game systems like Atari's *Home Pong* and Magnavox's *Odyssey 100/200* [13].

The release of *Star Wars* in 1977 helped make video arcade games more popular because children wanted to live out their fantasies of space battles [13]. Fuelled by the popularity of *Star Wars*, *Space Invaders*, created in 1978, became a phenomenon, leading other space-oriented games such as Namco/Midway's *Galaxian* and Atari's *Asteroid* to gain a fanbase [13].

The 1980s are considered the golden age of arcade games, with many now infamous titles such as *Battlezone*, *Pac-Man*, and *King Kong* appearing between 1980 and 1983 [13]. The 1980s also saw the development of the first LaserDisc game, *Dragon's Lair (*1983) [13]. Although arcade game development originated in the US, Japanese developers started dominating the market due to their focus on deterministic and scripted games such as *Space Invaders* and *Pac-Man* [13].

The American company Atari broke new ground by improving its vector graphic hardware to include colour. It was also one of the first companies to produce 3D arcade games, starting with I Robot in 1984. In 1989, polygon hardware, first used in the driving game Hard Drivin, allowed for an even better visual experience [13].

In the 1990s, arcade games remained popular, with Taito releasing more *Space Invaders* games and Konami having hits such as *The Simpsons* and *X-Men* [13]. The sequel to Capcom's Street Fighter, *Street Fighter II* (1991), was the first best-selling game of the decade, bringing public awareness to the one-on-one fighting game genre [13].

[15] https://en.wikipedia.org/wiki/Cadash#/media/File:Cadash.png

Other companies, such as SNK, noticed the popularity of *Street Fighter II* and published games like *Fatal Fury* in 1991 [13]. Indeed, a rivalry would emerge between Capcom and SNK as they became more prolific [13]. However, the success of beat-em-up games was not without controversy, as the emergence of *Mortal Kombat* in 1993 depicted gore and violence with humanised sprites [12]. Despite the outcry of moral guardians, *Mortal Kombat II* was released in 1994, along with other gory titles such as Bloodstorm and Killer Instinct [13].

However, the rise of games with a high polygon count, like *Sega's Virtual Racing* in 1992, meant that companies struggled to keep up with these trends [13]. With the release of titles like *Ridge Racer* – the first game to use 3D and texture mapping- developers had to spend more money creating games, causing them to turn to the home console as an increasingly more cost-efficient option [13]. Arcade operators also had to pay more to keep the machines relevant, which -- in turn -- caused arcade games to be more expensive [13].

In the 2000s, many arcade operations closed as developers focused on making games for home consoles [13]. Although memory cards emerged to help players keep track of their scores and customise their characters, they were primarily available in Asia [12]. Meanwhile, in countries like the UK, the status of arcade games diminished due to the marginalisation of genres [13].

**End of Section References**


[1] Ari Feldman. 2001. *Designing Arcade Computer Game Graphics*. Wordware Publishing Inc., Plano, Texas.

[2] Rare Historical Photos. What arcade games looked like before video games, 1968. Retrieved February 10, 2024 from https://rarehistoricalphotos.com/arcade-games-history-1968/.

[3] Margaret Rouse. 2018. Arcade Game. (June 2018). Retrieved February 10, 2024 from https://www.techopedia.com/definition/1903/arcade-game/.

[4] Carl Therrien. 2017. "To Get Help, Please Press X": The Rise of the Assistance Paradigm in Video Game Design. In *DiGRA Conference 2011*, September 14-17, 2011, Utrecht School of the Arts, Hilversum. http://www.digra.org/digital-library/publications/to-get-help-please-press-x-the-rise-of-the-assistance-paradigm-in-video-game-design/.

[5] Jacob Sobolev. 2019. What Are Arcade Games?. (May 2019). Retrieved February 10, 2024 from https://remarkablecoder.com/what-are-arcade-games/.

[6] Wikipedia. List of maze video games. Retrieved February 10, 2024 from https://en.wikipedia.org/wiki/List_of_maze_video_games#Maze_chase_games/.

[7] Wikipedia. Pong. Retrieved February 10, 2024 from https://en.wikipedia.org/wiki/Pong.

[8] Wikipedia. Arkanoid. Retrieved February 10, 2024 from https://en.wikipedia.org/wiki/Arkanoid.

[9] Gamestate. 2023. An overview of the best shooter arcade games. (March 2023). Retrieved February 11, 2024 from https://gamestate.com/blogs/news/an-overview-of-best-shooter-arcade-games.

[10] Wikipedia. Puzzle video game. Retrieved February 10, 2024 from https://en.wikipedia.org/wiki/Puzzle_video_game.

[11] Wikipedia. Cadash. Retrieved February 14, 2024 from

[12] /v/'s Recommended Games Wiki. Puzzle video game. Retrieved February 11, 2024 from https://vsrecommendedgames.fandom.com/wiki/Arcade_games/Platformer.

[13] Retro Gamer. 2015. *Retro Gamer Book of Arcade Classics*. Imagine Publishing, Bournemouth, Dorset.

## *SECTION B: EVALUATION AND MANUAL TESTING*

### How Can Video Games Be Evaluated?

Video games, seen as an interactive art form or gaming experience, should be judged on the following criteria: screenplay, technical, gameplay and sound [1].

The table below outlines how each of these categories should be critically assessed [1]:

| **Category** | **Explanation of Category** |
|---|---|
| Screenplay | The plot, characters and relationships these characters share can give a title meaning and depth.<br><br>The developer's intention is essential, as some games are created for escapist reasons [2]. |
| Technical [1] including graphics | The following category investigates how a game was made, including aspects such as the camera view, scripted events, and character models. It also measures how emotions and messages are translated onto the screen.<br><br>Graphics should not just enhance the game but complement it, especially in terms of gameplay [2]. |
| Gameplay [1], including the duration, replayability [2] and gameplay loop [3]. | The interest in gameplay lies in how the player interacts with the game, including the menu, controls, and interactions between the game characters and their environment.<br><br>The balance between the game time and its variety is equally important, as a long game with many quests is not necessarily good quality.<br><br>Gameplay is also concerned with measuring ease of use and the effectiveness of the game mechanics [2]. Not only is the effectiveness of game mechanics a concern, but critics also measure their learnability [2].<br><br>The gameplay loop is essential for the following reasons: a challenging and rewarding one keeps the player invested, while a boring one quickly becomes stale [3]. However, a gameplay loop should not just be challenging but also accessible to players at all levels [3]. |
| Music and sound | Music and sound can enhance a game's quality, making it more beautiful and enjoyable. In this case, the enjoyability of sound effects is evaluated [2].<br><br>Music and sound can add to the atmosphere of a game, which – in turn – can determine how immersive and enjoyable a game is [3]. |

Other means of evaluating video games deal with marketability, branding, and franchising [3], which are beyond the project's scope, as there are no plans to make them commercially available.

### Testing of Video Games

Testing is integral to the development process as it can discover bugs or user experience problems [4]. Testing is an ongoing process that should underpin all other stages of development [4].

The table below outlines the test types that are available to game programmers:

| **Type of Testing** | **Available Tools for Unity** |
|---|---|
| Unit testing is a technique that tests individual units or components of a game in isolation [4]. It ensures that bugs are caught early in development [4]. | Unity offers the option to test video game code using the Unity Test Framework package [5], which integrates the NUnit library – a unit testing framework for all .Net languages, ported initially from JUnit [6]. |
| Manual testing involves the player or the developer playing the game to test its features and functionality [4]. Automated testing only sometimes covers bugs like UI issues and poorly executed gameplay or design [4]. | A test plan can help identify which areas require testing and document any bugs found during testing [7]. |
| Integration testing involves testing different techniques to ensure they work correctly [4]. This kind of testing can catch bugs, such as issues in communication between components [4]. | The Unity Test Framework contains 'SetUp' and 'TearDown' methods [8] that allow the creation of an environment to test multiple systems simultaneously [4]. |

### Evaluation Plan

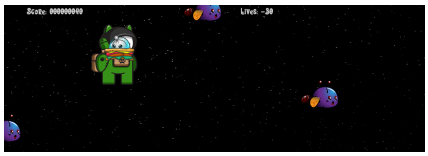
The table below displays a test plan that performs manual unit and regression testing to ensure that the most critical features of the game work. The actual result and Pass/Fail related rows are left blank as the test plan is for demonstrative purposes only.

| **Action** | **Expected Result** | **Actual Result** | **Pass / Fail** |
|---|---|---|---|
| Upon loading the game, the user sees the menu. | The menu contains buttons, the title of the game and an image of the cat. | | |
| The user is navigated to the first level when clicking the play game button. | The game scene is loaded correctly. | | |
| When clicking any individual play button, the user is directed to its specific game scene. | The specific scene is loaded correctly. | | |
| The cat cannot cross the boundaries of the screen. | The cat can only move within the boundaries of the spaceship. | | |
| The bugs disappear when the space cat collides with them. | The cat can devour the fish and milk sprites. | | |

| | | |
|---|---|---|
| The cat loses a life when colliding with one of the aliens. | The cat loses points. | |
| The SFX sound occurs when a cat picks up a coin or loses a life. | The sound clip plays with no issues. | |
| The score and health status text are updated whenever appropriate. | The UI displays of the scores and cat's health are updated promptly. | |
| When returning to the menu, the audio background of the game is muted. | The only audible background sound is the menu-specific one. | |

Along with the evaluation plan above, the game also relied on playtesting. Players were not given questionnaires but instead encouraged to observe any bugs as they see them / occur, allowing for an open forum.

### Bugs Found During the (Early) Manual Testing Phase

| | |
|---|---|
|  | **Figure 2**: The following image displays the lives' amount going into negative numbers when the game is loaded from a previous scene. |

Manual testing uncovered bugs and other errors, causing the game to undergo many changes. Testing was carried out during various phases of development, especially after the release of a new feature, to ensure that previous features were not affected by the change and verify that new ones worked as expected. Thus, regression testing was carried out, ensuring that software changes do not introduce bugs or break existing systems [3].

The table below displays any bugs found (as shown in Figure 2) and the fixes applied to them in order.

| **Bugs** | **Fixes** |
|---|---|
| The player character would not stay within the boundaries of the scene. | Addressed in the implementation section. |
| The introduction of helper methods caused the flying hamburgers to stop following the player. | The *ControllHelper* game object that has the FollowPlayer method was not attached to the *FlyingHamburgerController* script: the code snippet below shows how this has been improved:<br><br>void Update() => controllerHelper.FollowPlayer(spaceCat, this.transform, speed); |
| The flying hamburgers did not follow the player to the fullest extent, only following them to the left and right. | Addressed in the implementation section. |

| | |
|---|---|
| When transitioning from the menu scene to the game, the mute audio only applied to the background sound, not the SFX ones. | The addition of the *AudioListener* — a class that implements a microphone-like device [15]—and its volume function [16] applied to a menu-specific audio empty prefab that has a *MuteAudio* script attached to it ensures that all sounds remain muted when a mute button is pressed while preventing an overflow of sound into the next scene in the case said button has not been clicked. |
| After dying, the cat sprite would briefly show the sleeping cat before returning to the astronaut cat sprite, or the old sprite would keep flickering. | A separate *GameOver* scene addresses this issue (Figure 7.22). |
| The lives would not display correctly (Figure 2). | Within the *SpaceCatController* script, the *CatDeath* function has been improved in the following manner:<br><br>void CatDeath()<br>{<br>isAlive = false;<br>_scoreKeeper.ResetScore();<br>_healthKeeper.ResetLives();<br>StartCoroutine(DelayLoadEndScene());<br>} |
| The lives were not consistent across levels. | Within the *HealthKeeper* script, the lives variable has been changed from having a [SerializeField] attribute to not having one at all [17]. |

**End of Section References**


[1] Starico. How to evaluate Video Games. Retrieved February 22, 2024 from https://stari.co/scores/how-to-evaluate-video-games.

[2] slashint. 2023. How to evaluate a videogame effectively and objectively. (February 2023). Retrieved February 22, 2024 from https://stari.co/scores/how-to-evaluate-video-games.

[3] Frying Jelly. 2023. HOW TO EVALUATE GAME CONCEPTS. (March 2023). Retrieved February 22, 2024 from https://www.fryingjelly.com/blog/2022/how-to-evaluate-game-concepts/.

[4] Unity. Testing and quality assurance tips for Unity projects. Retrieved February 22, 2024 from https://unity.com/how-to/testing-and-quality-assurance-tips-unity-projects.

[5] Unity Manual. About Unity Test Framework. Retrieved February 22, 2024 from https://docs.unity3d.com/Packages/com.unity.test-framework@1.4/manual/index.html.

[6] NUnit. NUnit. Retrieved February 22, 2024 from https://nunit.org.

[7] Katalon. What is a Test Plan? Test Plan vs. Test Strategy (Ultimate Guide). Retrieved February 22, 2024 from https://katalon.com/resources-center/blog/test-plan.

[8] Unity Manual. 6. SetUp and TearDown. Retrieved February 22, 2024 from https://docs.unity3d.com/Packages/com.unity.test-framework@1.3/manual/course/setup-teardown.html.

## *SECTION C: PROTOTYPES*

### Design-Related Images and Prototypes

The images below were created during the project's design stage, and – consequently – they do not reflect the current stage of the game due to ongoing changes.

These changes occurred because the initial prototype relied too much on *Pac-Man*, which also came with an unwieldy maze design that needed to be more flexible when changes were required. As reflected in the conclusion, the lack of experience in game design similarly contributed to rehauling the game concept, especially in moving the location from being with a spaceship to just floating in space.

### Core Game Concept

The image below refers to the core game concept described in the introduction.

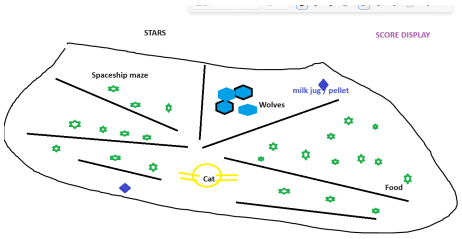



**Figure 1**: This is a rudimentary and awful sketch of the core game concept, showing the spaceship floating in space. Inside the spacecraft are wolves, cheese items, and jugs of milk. Of course, the cat is present as well. The image does not reflect the final state of the project.

### Gameplay

The image below refers to the gameplay as described in the design section.

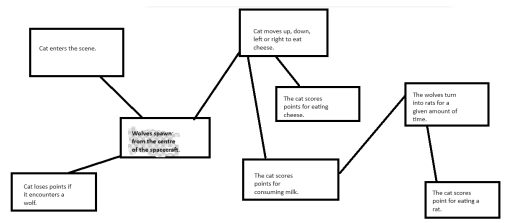



**Figure 2**: The image above shows the since-revised gameplay of *Hungry Space Cat.*

### Implementation Prototypes

### Menu Design

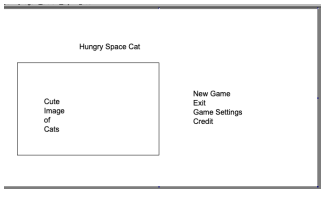


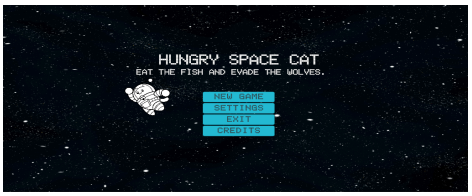

**Figure 3**: The screenshot shows the menu's initial design, with the title on top, followed by an image below and the menu settings on the right.

Even though the menu's design has changed since then, it still follows the principles laid out in this image.

The underlying principle behind the look and feel of the game menu is to have a simple background that calls back to the space theme. The design pays homage to the arcade style of the game, explicitly referencing game visuals from the 1980s. An image of a cat next to the menu items emphasises the feline-inspired touch of the game.

The menu, buttons and titles below were created using Unity's UI features [1], with the image of the astronaut cat created by *Lineart_Magic* and the space-themed background made by *kjpargeter* [3].

The table below displays the iterations of the menu (Figure 3), with the Hungry Space Cat: UI Interface serving as a starting point.

| **Prototype / Menu Versions** | **Notes** |
|---|---|
|  | **Figure 4**: The following table displays the first version of the game menu, designed using Unity's UI functionality [1].<br><br>The image of the cat [2] was sourced from pixabay[16]. |
|  | **Figure 5**: The more refined version of the prototype contains a space-themed background [3] sourced from Freepik[17]. |
|  | **Figure 6**: This version adds colour to the menu, making it less bland and fitting an 80s aesthetic. |
|  | **Figure 7**: The menu scene consists of a few buttons, an astronaut cat image[18] and a space-themed background image[19]; however, player feedback indicated that the font was unreadable and the text too small. |

## Creating the Tilemap (Prototype)

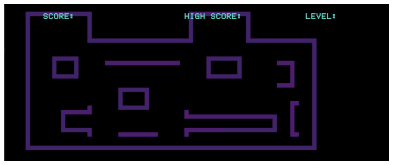
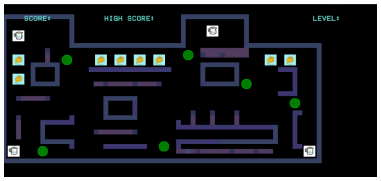
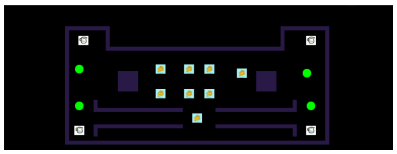
The game's prototype level design was created using Unity's tilemap feature [4] to create obstacles in the form of walls, the pellets – represented by fish [5] and milk sprites that were created in Paint – as well as the nodes that dictate which direction wolves move towards [6, 7].

Unity's tilemap feature allows for creating game levels, with the ability to prototype 2D worlds quickly [8]. Layers ensure no overlapping occurs between components that work together [8].

[16] https://pixabay.com/vectors/cat-astronaut-space-universe-7719975/
[17] https://www.freepik.com/free-photo/starry-night-sky_7061153.htm
[18] https://bevouliin.com/green-cat-astronaut-game-asset-sprites/
[19] https://screamingbrainstudios.itch.io/seamless-space-backgrounds

A *Tilemap Collider 2D*, which generates collider shapes, and a *Composite Collider 2D* – which is responsible for merging the shapes of the neighbour collider [9], have been added to the wall tilemap to ensure that the player does not move out of bounds.

The following table displays the prototypes of the game's level design. Please note that the prototypes do not represent the final implementation and that any assets listed here have since been abandoned.

| **Prototype / Level Versions** | **Notes** |
|---|---|
|  | **Figure 8**: The figure shows the initial feline-inspired cat maze. The pink tile asset was sourced from zigurous's *Pac-Man* tutorial [6, 7]. |
|  | **Figure 9:** The following figure shows the initial level design, featuring the cheese [10] and milk [11] pellets, along with the nodes that determine the direction the wolves will move towards [6, 7]. |
|  | **Figure 10**: A more finalised prototype that shows the cat-inspired maze/spaceship. |
|  | **Figure 11:** A prototype that abandoned some less useful sprites [10, 11]. |

**End of Section References**


[1] Vincent Taylor. 2021. Creating a Main Menu with Unity UI. (July 2023). Retrieved February 22, 2024 from https://vintay.medium.com/creating-a-main-menu-with-unity-ui-f724674f72ac/.

[2] Pixabay. Lineart_Magic. Retrieved February 22, 2024 from https://pixabay.com/users/lineart_magic-1156796/.

[3] Freepik. Kjpargeter | Freepik. Retrieved February 23, 2024 from https://www.freepik.com/author/kjpargeter.

[4] Unity Technologies. 2023. Introduction to Tilemaps. (January 2023). Retrieved February 24, 2024 https://learn.unity.com/tutorial/introduction-to-tilemaps.

[5] VoodooMoose. 2023. Free Fish Icons. (February 2023). Retrieved March 2, 2024 https://voodoomoose.itch.io/free-fish-icons.

[6] Zigurous. 2021. How to make Pacman in Unity (Complete Tutorial). (21 June 2021). Retrieved February 22, 2022 from https://www.youtube.com/watch?v=TKt_VlMn_aA.

[7] zigurous. 2023. unity-pacman-tutorial. (21 June 2021). Retrieved February 23, 2022 from https://github.com/zigurous/unity-pacman-tutorial/tree/main.

[8] Jordan Evans. 2021. Introduction to Tilemap in Unity. (September 2021). Retrieved February 25, 2024 from https://medium.com/nerd-for-tech/introduction-to-tilemap-in-unity-a27b1f0def78.

[9] Unity Documentation. Tilemap Collider 2D. Retrieved March 2, 2024 from https://docs.unity3d.com/Manual/class-TilemapCollider2D.html.

[10] Jaya Senantiasa. Cheese cartoon icon illustration Vector 8693552 Vector Art at Vecteezy. Retrieved February 25, 2022 from https://www.vecteezy.com/vector-art/8693552-cheese-cartoon-icon-illustration-vector.

[11] VectorStock. Cute milk jug cartoon hand vector image. Retrieved February 25, 2022 from https://www.vectorstock.com/royalty-free-vector/cute-milk-jug-cartoon-hand-vector-28099517.

## *SECTION C: DESIGN*

### Factors to Consider When Designing a Game

In order to create a successful title, it is essential to consider the following factors when designing a game [2], with an emphasis placed on not overburdening it with disproportionate design details or other unnecessary aspects [1]:

- ★ The core concept of the game.
- ★ The game mechanics and its overall objectives [3].
- ★ The user interactions [3].
- ★ The tools and platforms [3].
- ★ The character designs and level design.
- ★ The game's hook [2].
- ★ The narration and plot.
- ★ The sound design [2].

### Game Mechanics and Gameplay - A Brief Definition

*Game mechanics* (Figure 3.1) are the backbone of a game, determining how users and the overall interlocking pieces of a game, such as rules and states, work together [4]. Thus, it is crucial to understand the distinction between game mechanics and fluff [4]. Fluff refers to elements that may seem substantial but are distracting, not significantly benefiting the game or the player [4]. In other words, games are often affected by fluff in disguise, where an action appears meaningful but has little to no result [4].

Famous examples of game mechanics are 'jump' in *Super Mario* and 'rotate' in *Tetris* [4]. Game elements are not mechanics but provide links to an action that produces an outcome [4]. Meanwhile, *gameplay* refers to the flow between challenges, mechanics, and their outcomes. Gameplay loops, which represent repeatable actions and sequences of a player, support the analysis [4]. A smooth gameplay loop indicates that the game mechanics are solid, while an awkward one implies the opposite [4].

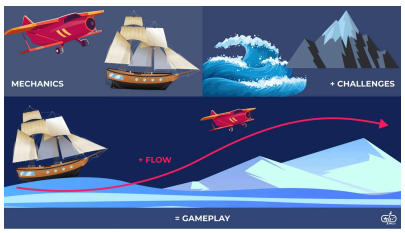
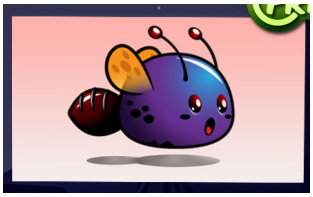



**Figure 1**: The following image shows how game mechanics influence the flow of the gameplay [4].

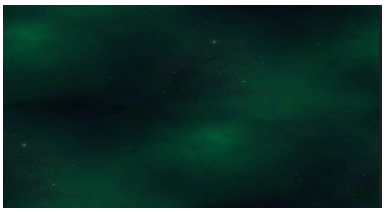
*Game mechanics* can be core, secondary or primary [4]. Core mechanics refer to the foundations of the game that make it what it is, while primary and secondary ones are there to support it [4].

### Character Design Sprites

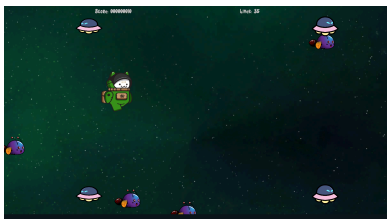
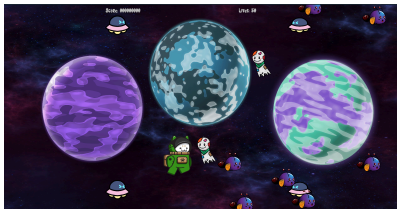
| Name of Sprite | Description |
|---|---|

| | |
|---|---|
|  | **Figure 2**: The cute astronaut cat sprite ensures the game's peaceful and soothing atmosphere.<br><br>Source: bevouliin[20]. |
|  | **Figure 3**: The bug sprite's design is also endearing, emphasising the game's cuteness.<br>.<br>Source: bevouliin[21]. |
|  | **Figure 4**: The enemy sprite for the first scene – like the other character sprites, is not meant to be threatening or dangerous-looking, fitting in with the game's theme of light-hearted cuteness.<br><br>Source: itch.io[22]. |
|  | **Figure 5**: The enemy sprite for the second scene, which follows the player, features a more adorable design than intimidating.<br><br>Source: bevouliin[23]. |
|  | **Figure 6**: The enemy sprite for the third scene involves ghost dolls that spawn based on wave points. Its design is also *moe* 萌え – a Japanese term that refers to intense feelings of affection [6].<br><br>Source: bevouliin[24]. |

## Level Design Design Sprites

| Name of Sprite | Description |
|---|---|
|  | **Figure 7:** The following green background is laid on top of another similar one to provide the impression of being in space.<br><br>Each of the three scenes features a differently coloured background [7].<br><br>Source: itch.io[25]. |

[20] https://bevouliin.com/green-cat-astronaut-game-asset-sprites/
[21] https://bevouliin.com/purple-bug-game-asset-sprites/
[22] https://kububbis.itch.io/kawaii-ufo-sprite-pack
[23] https://bevouliin.com/sandwich-alien-game-asset-sprites/
[24] https://bevouliin.com/doll-ghost-game-asset-sprites/
[25] https://screamingbrainstudios.itch.io/seamless-space-backgrounds

| | |
|---|---|
|  | **Figure 8:** The first level displays a green background [7] with the space cat chasing after bugs.<br><br>Four UFOs are located at each corner of a screen that the player might have to reach to catch a bug. The four UFOs were inspired by the four ghosts in *Pac-Man* but without their distinct personalities [8]. |
|  | **Figure 9:** The second level features a black background [7] with the space cat followed by three flying hamburgers.<br><br>This level is slightly more difficult than the previous one as the player does not have the option to avoid the hamburgers. |
|  | **Figure 10:** The third level in the game has a blue background [7]. It also features the four UFOs from level one and ghost dolls that appear/disappear based on a wave path.<br><br>It is the last and most challenging level for the easy mode. |
|  | **Figure 11:** The fourth level in the game has a purple background [7] and three planets [8]. It repeats the previous level but makes it harder for the player to track all bugs since the planets obscure them.<br><br>This level is available in the normal mode of the game. |

**End of Section References**


[1] bdatg2. 2013. Simplicity. (December 2013). Retrieved February 20, 2024 from https://vgprobs.wordpress.com/simplicity/.

[2] GameDesigning. Learn to Design Video Games with GameDesigning.org. Retrieved February 20, 2024 from https://www.gamedesigning.org.

[3] BE BLOGGERS. 2023. A Journey Through Game Development: Creating Arcade Games Like Space Invaders. (October 2023). Retrieved February 21, 2024 from https://medium.com/@social.hotcerts/a-journey-through-game-development-creating-arcade-games-like-space-invaders-8107a7d9b345.

[3] Unity. 10 game design tips for new developers. Retrieved February 20, 2024 from https://unity.com/how-to/beginner/10-game-design-tips-new-developers.

[4] Alexander Brazie. 2022. Video Game Mechanics: A Beginner's Guide (with Examples). (April 2022). Retrieved February 21, 2024 from https://gamedesignskills.com/game-design/video-game-mechanics.

[5] Jamey Pittman. 2001. The Pac-Man Dossier. (April 2015). Retrieved February 12, 2024 from https://pacman.holenet.info.

[6] Wikipedia. Moe (slang). Retrieved April 2, 2024 from https://en.wikipedia.org/wiki/Moe_(slang).

[7] screamingbrainstudios. 2023. Screaming Brain Studios - itch.io. Retrieved March 23, 2024 from https://screamingbrainstudios.itch.io.

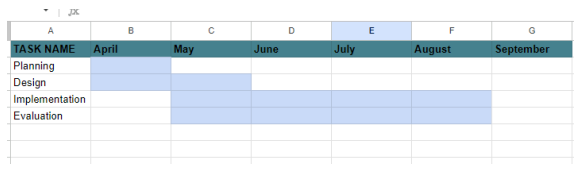
[8] Kenny. Planets. Retrieved April 2, 2024 from https://kenney.nl/assets/planets.

## *SECTION D: PROJECT MANAGEMENT*

### Gantt Chart/Progress Table

A Gantt chart – also referred to as a harmonograph – is a bar chart that allows for the visual tracking of a project's schedule [1]. Henry Gantt created it in the 1910s [1].

The image below refers to the initial Gantt chart (Figure 1), which shows the project's four stages. It should be noted that the initial stages of development, such as planning and design, took place within a month prior to the start of the module due to family constraints/health issues that would have impacted the successful completion of the project otherwise.



| TASK NAME | April | May | June | July | August | September |
|---|---|---|---|---|---|---|
| Planning | | | | | | |
| Design | | | | | | |
| Implementation | | | | | | |
| Evaluation | | | | | | |

**Figure 1:** The Gantt chart for the *Hungry Space Cat* project.

Consequently, the table below outlines the development progress:

| **Month** | **Tasks** | **Milestones** |
|---|---|---|
| February | Researching the template<br><br>Picking an appropriate topic<br><br>Research into arcade games.<br><br>Looking into cat-based arcade games.<br><br>Looking for sprites and other assets.<br><br>Setting up the GitHub account for the project.<br><br>Going through tutorials to learn more about Unity 2D programming. | Initial research and design document finished.<br><br>Developing an understanding of arcade games and their philosophy.<br><br>Start of implementation, with a rough prototype that, at first, was heavily based on *Pac-Man*, gaining an understanding of tilemaps in the process. |
| March | Refining the literature, design and implementation document.<br><br>Starting the implementation of the game, with a focus on controllers, spawn scripts and UI elements.<br><br>Going through tutorials to learn more about Unity 2D programming.<br><br>Preparing the game for | Refactoring the game to develop the concept it has been using since.<br><br>Putting the building blocks of the game into place.<br><br>Extensive manual testing to find any early bugs.<br><br>Uploading the game on itch.io with password protection. |

| | | |
|---|---|---|
| | playtesting. | |
| April | More playtesting sessions.<br><br>Refactoring bugs.<br><br>Refining the report.<br><br>Making the video for the pitch.<br><br>Going through tutorials to learn more about Unity 2D programming. | Game shows more polish, with a refactored menu and more levels.<br><br>Report is now in a stable state that only needs some editing here and there.<br><br>Increasing knowledge of how to make games in Unity. |
| May | Create a level or more for the game.<br><br>Look up game performance. | |

### Project Management Tools

While Gantt charts help schedule projects, an interactive tool geared for monitoring and design purposes, such as Trello or Jira, is even better for monitoring project progress [2]. However, Jira can be complex and better solutions exist for those looking for a cost-effective product [3]. Meanwhile, Trello has limited views and does not show task dependencies [4].

Considering that the final project uses GitHub, it is advantageous to utilise its feature — a tool integrated into the GUI to create tickets that can be moved across a board as work progresses [5]. Its key benefit is consolidating everything onto one platform, ensuring a streamlined project management experience (Figure 2).

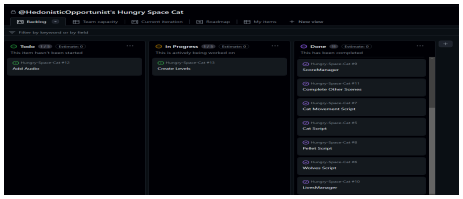



**Figure 2:** A screenshot of the project board shows how tracking the development process becomes manageable and less daunting.

### Work Plan

The following table defines a simple work plan that follows the report's structure; the reasoning behind this is that software development projects generally follow a similar lifecycle [1].

To view the actual progress of the project, please refer to the following section.

| **Project Stage** | **Time Frame** | **Key Goals** | **Milestone** |
|---|---|---|---|
| The discovery and literature review phase represents the planning stage in a software project [6]. | This stage should be completed within the first two to four weeks of April. | Based on the chosen template, a background search helps examine gaps in the market. | The proposal and literature review are comprehensive and complete and have helped with the project's |

| | | | |
|---|---|---|---|
| | | At the same time, the literature review enhances understanding of the subject and makes informed decisions about the project's objectives. | design. |
| The design stage. | The following stage should take place after the planning stage and take about two to three weeks.<br><br>It should finish around the middle of May to allow for the first iteration of the development process to take place simultaneously.<br><br>In the design stage, test plans that determine the success of a feature shall also be decided on.<br><br>In particular, unit testing of an arcade shall be researched to help make the testing more extensive. | In the design stage, various models are considered before a final one is selected.<br><br>Once the model has been chosen, a sketch of the game and some simple character designs are created.<br><br>Afterwards, the critical components of the game architecture are described to help understand how everything comes together.<br><br>The game's flow is also established, with diagrams present to help understand how the system and user interact. | The rough outline of the game components and the rudimentary character designs serve as a means to start developing the functioning prototype. |
| The implementation stage represents the building stage of a software project's lifecycle [6]. | While this stage will take place after the design stage, around the middle of May, it is an iterative process that should last until August.<br><br>In other words, development will consist of 'sprints' focusing on a specific project feature [7].<br><br>Each sprint will be about a week long [7]. | The goal of the implementation phase is to get the project working incrementally, with the ultimate goal of completing the project on time. | Each sprint ends with the completion of the feature. |
| The testing stage. | As with the implementation stage, this is an iterative process after each feature is developed.<br><br>This stage will be closely tied to the implementation stage, starting around the end of May and lasting till | The testing stage will help evaluate the project's success by combining manual testing and user play sessions.<br><br>These play sessions will include other users, | The testing stage ends up improving the project. |

| | | | |
|---|---|---|---|
| | August. | | |

**End of Section References**


[1] ProductPlan. Gantt Chart. Retrieved February 20, 2024 from https://www.productplan.com/glossary/gantt-chart/.

[2] Anne M. Carroll. 2023. 10 Best Project Management Software Buyers' Guide. (January 2024). Retrieved February 23, 2024 from https://project-management.com/top-10-project-management-software/.

[3] Lauren Good. 2023. The Pros and Cons of Using Jira Software for Project Management. (September 2023). Retrieved March 23, 2024 from https://project-management.com/the-pros-and-cons-of-using-jira-software/.

[4] Daniel Raymond. 2024. Top 8 Cons or Disadvantages of Using Trello Software. (January 2024). Retrieved March 23, 2024 from https://projectmanagers.net/top-8-disadvantages-of-using-trello/.

[5] GitHub Docs. About Projects. Retrieved February 21, 2024 from https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects.

[6] Alexandra. 2023. What Is SDLC? Understand the Software Development Life Cycle. (March 2023). Retrieved February 20, 2024 from https://stackify.com/what-is-sdlc/.

[7] Wrike. What Is a Sprint in Agile?. Retrieved February 20, 2024 from https://www.wrike.com/project-management-guide/faq/what-is-a-sprint-in-agile/.