



# Control Flow Bug

A Puzzle Game Based on Real-Time Flocking Behaviour

Bachelor's thesis in Computer science and engineering

Linda Bergevald  
Adrian Danielsson  
Jonathan Jansen Granö  
Erik Larsson  
Elsa Lenz  
Salar Loran Telo





BACHELOR'S THESIS 2026

# Control Flow Bug

A Puzzle Game Based on Real-Time Flocking Behaviour

Linda Bergevald  
Adrian Danielsson  
Jonathan Jansen Granö  
Erik Larsson  
Elsa Lenz  
Salar Loran Telo



UNIVERSITY OF  
GOTHENBURG

---



**CHALMERS**  
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering  
CHALMERS UNIVERSITY OF TECHNOLOGY  
UNIVERSITY OF GOTHENBURG  
Gothenburg, Sweden 2026

Control Flow Bug  
A Puzzle Game Based on Real-Time Flocking Behaviour  
Linda Bergevald Adrian Danielsson Jonathan Jansen Granö Erik Larsson Elsa Lenz  
Salar Loran Telo

© Linda Bergevald, Adrian Danielsson, Jonathan Jansen Granö, Erik Larsson,  
Elsa Lenz, Salar Loran Telo 2026.

Supervisor: Natasha Mangan, Department of Computer Science and Engineering,  
Chalmers University of Technology  
Examiner: Patik Jansson, Department of Computer Science and Engineering, Chalmers  
University of Technology  
Co-Examiner: Palle Dahlstedt, Department of Computer Science and Engineering,  
Chalmers University of Technology

Bachelor's Thesis 2026  
Department of Computer Science and Engineering  
Chalmers University of Technology and University of Gothenburg  
SE-412 96 Gothenburg  
Telephone +46 31 772 1000

Cover: Illustration of the game's logo and title.

Typeset in L<sup>A</sup>T<sub>E</sub>X  
Gothenburg, Sweden 2026

Control Flow Bug

A Puzzle Game Based on Real-Time Flocking Behaviour

Linda Bergevald, Adrian Danielsson, Jonathan Jansen Granö, Erik Larsson, Elsa Lenz,  
Salar Loran Telo

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

## Abstract

Flocking behaviour is a common sight in nature, seen in phenomena such as birds flying together in large groups. The purpose of this bachelor's thesis is to explore how flocking behaviour can be used as a primary game mechanic and what engagement implications can be derived from it. To accomplish this, a puzzle game was developed using the Unity game engine. In line with the project's purpose, a strong focus was put on game design and aesthetics. The game features a playable robot character and a swarm of bugs, which can be manipulated by the player. This swarm generates electricity used to solve puzzles by opening doors. The game was playtested and evaluated for its engagement implications, which successfully demonstrated that an engaging game can be made with flocking behaviour as a core mechanic.

## Sammandrag

Flockbeteende är en vanlig förekomst i naturen, vilket kan observeras hos fåglar som flyger tillsammans i stora grupper. Syftet med detta kandidatarbete är att utforska hur flockbeteende kan användas som en huvudsaklig spelmekanik och vilka implikationer detta har på spelarens engagemang. För att åstadkomma detta utvecklades ett pusselspel med spelmotorn Unity. I linje med syftet för projektet, låg ett stort fokus på speldesign och estetik. Spelet innehåller en spelbar robot och en insektsvärm, som spelaren kan manipulera. Denna svärm genererar elektricitet som används för att lösa pussel genom att öppna dörrar. Spelet testades och utvärderades med avseende på engagemang, vilket framgångsrikt demonstrerade att ett engagerande spel kan skapas med flockbeteende som en central spelmekanik.

Keywords: flocking behaviour, video game, puzzle, unity, boids



## Acknowledgements

We would like to thank our supervisor Natasha Mangan for her great support and guidance during this project. We would also like to thank all of our playtesters that provided us with helpful and insightful feedback on the project.

Group DATX11-38A, Gothenburg, June 2026



# Contents

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>List of Figures</b>                            | <b>xi</b> |
| <b>1 Introduction</b>                             | <b>1</b>  |
| 1.1 Purpose . . . . .                             | 1         |
| 1.2 Stakeholders . . . . .                        | 1         |
| 1.3 Scope . . . . .                               | 2         |
| <b>2 Theory</b>                                   | <b>5</b>  |
| 2.1 Emergent Behaviour . . . . .                  | 5         |
| 2.1.1 Flocks and Herds . . . . .                  | 5         |
| 2.1.2 The Boids Algorithm . . . . .               | 6         |
| 2.2 Data Oriented Design . . . . .                | 6         |
| 2.2.1 Spatial Partitioning and Indexing . . . . . | 7         |
| 2.2.2 Entity Component System . . . . .           | 8         |
| 2.3 Game Design . . . . .                         | 9         |
| 2.3.1 MDA Framework . . . . .                     | 9         |
| 2.3.2 Puzzle Design . . . . .                     | 9         |
| 2.3.3 User Experience . . . . .                   | 10        |
| 2.3.4 User Interface . . . . .                    | 11        |
| 2.3.5 Accessibility . . . . .                     | 13        |
| <b>3 Methodology</b>                              | <b>15</b> |
| 3.1 Tools Outline . . . . .                       | 15        |
| 3.2 Workflow . . . . .                            | 16        |
| 3.2.1 Agile . . . . .                             | 16        |
| 3.2.2 Distribution of work . . . . .              | 17        |
| 3.3 Flocking Behaviour . . . . .                  | 17        |
| 3.4 Game Design . . . . .                         | 18        |
| 3.4.1 Gameplay . . . . .                          | 19        |
| 3.4.2 Level Design . . . . .                      | 19        |
| 3.4.3 User Interface . . . . .                    | 20        |
| 3.4.4 User Experience . . . . .                   | 24        |
| 3.4.5 Accessibility . . . . .                     | 24        |
| 3.5 Aesthetics . . . . .                          | 26        |
| 3.5.1 Style and Theme . . . . .                   | 26        |
| 3.5.2 Art . . . . .                               | 27        |

|          |                                        |           |
|----------|----------------------------------------|-----------|
| 3.5.3    | Sound Effects . . . . .                | 33        |
| 3.5.4    | Music . . . . .                        | 34        |
| 3.6      | Testing & User Feedback . . . . .      | 36        |
| 3.6.1    | Testing Process . . . . .              | 36        |
| 3.6.2    | Tester and Conductor Bias . . . . .    | 37        |
| <b>4</b> | <b>Results</b>                         | <b>39</b> |
| 4.1      | First Test Phase . . . . .             | 39        |
| 4.1.1    | Quantitive Data . . . . .              | 40        |
| 4.1.2    | Received Feedback . . . . .            | 41        |
| 4.1.3    | Evaluation . . . . .                   | 42        |
| 4.2      | Second Test Phase . . . . .            | 43        |
| 4.2.1    | Quantitive Data . . . . .              | 43        |
| 4.2.2    | Received Feedback . . . . .            | 45        |
| 4.2.3    | Evaluation . . . . .                   | 46        |
| 4.3      | Finished Product . . . . .             | 46        |
| <b>5</b> | <b>Discussion</b>                      | <b>49</b> |
| 5.1      | Finished Product . . . . .             | 49        |
| 5.1.1    | Challenges . . . . .                   | 49        |
| 5.1.2    | Further Development . . . . .          | 50        |
| 5.2      | Testing . . . . .                      | 50        |
| 5.2.1    | Quality of Test Results . . . . .      | 51        |
| 5.2.2    | Potential Improvements . . . . .       | 52        |
| 5.3      | Research Question . . . . .            | 53        |
| 5.4      | Social & Ethical Aspects . . . . .     | 53        |
| 5.4.1    | Use of Generative AI . . . . .         | 54        |
| 5.5      | Conclusion . . . . .                   | 55        |
|          | <b>Bibliography</b>                    | <b>57</b> |
| <b>A</b> | <b>Appendix A</b>                      | <b>I</b>  |
| A.1      | First Testing Phase Template . . . . . | I         |
| A.1.1    | Introduction . . . . .                 | I         |
| A.1.2    | Tasks . . . . .                        | I         |
| A.1.3    | Interview Questions . . . . .          | I         |
| A.2      | Second Test Phase Template . . . . .   | II        |
| A.2.1    | Introduction . . . . .                 | II        |
| A.2.2    | Tasks . . . . .                        | II        |
| A.2.3    | Interview Questions . . . . .          | II        |



# List of Figures

|      |                                                                                                                                                                                                                                                             |    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1  | An example of a 2D-space with point objects and their associated buckets in the hash table after spatial hashing. . . . .                                                                                                                                   | 7  |
| 2.2  | <i>Portal 2</i> [24] incorporates both diegetic and non-diegetic elements. This screenshot was taken from a video walkthrough made by IGN [25].                                                                                                             | 12 |
| 3.1  | Screenshot of <i>Trello</i> [36] during the development process of the MVP.                                                                                                                                                                                 | 17 |
| 3.2  | Levels from final version of the game. . . . .                                                                                                                                                                                                              | 20 |
| 3.3  | The first level contains non-diegetic UI elements to display clear information as a tutorial. Diegetic elements can also be seen in the progress bar of the chargepad and the electrical sparks between it and the bugs. . . . .                            | 22 |
| 3.4  | The archetype selector has a button prompt, which is a non-diegetic UI element. . . . .                                                                                                                                                                     | 22 |
| 3.5  | Renderings of the UI for the main menu. . . . .                                                                                                                                                                                                             | 22 |
| 3.6  | The How-To-Play menu. . . . .                                                                                                                                                                                                                               | 23 |
| 3.7  | Iterations of the game title with the final version at the bottom. . . .                                                                                                                                                                                    | 24 |
| 3.8  | The deactivated (top) and activated (bottom) versions of the chargepad, attracting battery pillar, and door. The left image shows how greyscale versions show how light and dark values signal different states of activation, even without colour. . . . . | 25 |
| 3.9  | The attracting and repelling battery pillars in colour and converted to greyscale. Instead of value, the difference in shape is used to distinguish the function. . . . .                                                                                   | 26 |
| 3.10 | The yellow and green archetype selectors in colour and converted to greyscale. Instead of value, the shape of the logo is used to communicate the selected archetype. . . . .                                                                               | 26 |
| 3.11 | The first level with placeholder assets. . . . .                                                                                                                                                                                                            | 27 |
| 3.12 | The initial placeholder spritesheet for the player character facing in eight directions. . . . .                                                                                                                                                            | 27 |
| 3.13 | Initial placeholder floor tilemap and a finished floor tilemap. . . . .                                                                                                                                                                                     | 28 |
| 3.14 | Initial placeholder wall tilemap and a finished wall tilemap. . . . .                                                                                                                                                                                       | 28 |
| 3.15 | Cubes in the top-down perspective used in the game. Each side is one unit of length, except the height, which is 3/4 units. . . . .                                                                                                                         | 29 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                            |    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.16 | Mood boards. The images are taken from <i>Transistor</i> [46], <i>NieR: Automata</i> [47], <i>Rain World</i> [48], 天ノ弱/164 feat.GUMI [49], 天ノ弱 / Leo/need [50], 僕らまだアンダーグラウンド - Eve MV [51], 遊生夢死 - Eve MV [52], <i>Wall-E</i> [53], <i>Broken Age</i> [54], <i>Overwatch: The Last Bastion</i> [55], <i>Borderlands</i> [56], <i>Portal 2</i> [24], <i>NieR: Automata</i> [47], and <i>Star Wars: The Rise of Skywalker</i> [57]. . . . . | 29 |
| 3.17 | The concept artworks of the world and player character. . . . .                                                                                                                                                                                                                                                                                                                                                                            | 30 |
| 3.18 | The evolution of the appearance of the battery pillar. . . . .                                                                                                                                                                                                                                                                                                                                                                             | 31 |
| 3.19 | The final spritesheets of the idle pose and walking animations. . . . .                                                                                                                                                                                                                                                                                                                                                                    | 31 |
| 3.20 | A colour palette used by the artists when designing assets. . . . .                                                                                                                                                                                                                                                                                                                                                                        | 32 |
| 3.21 | The first two levels, fully decorated. The atmospheric perspective increases every level to show an increase in altitude. . . . .                                                                                                                                                                                                                                                                                                          | 32 |
| 3.22 | Textures of cityscapes. . . . .                                                                                                                                                                                                                                                                                                                                                                                                            | 33 |
| 3.23 | The <i>Wwise</i> [42] project hierarchy used to organise audio assets by function, including separate categories for ambience, music, sound effects, and UI audio. . . . .                                                                                                                                                                                                                                                                 | 34 |
| 3.24 | The initial piano sketch of the background music written in <i>MuseScore</i> [40], showing the harmonic and melodic foundation that was later arranged and produced for in-game use. . . . .                                                                                                                                                                                                                                               | 35 |
| 4.1  | Levels from the test version of the game, phase one. . . . .                                                                                                                                                                                                                                                                                                                                                                               | 39 |
| 4.2  | The distributions of self assessment data collected from the sign-up form, phase one. . . . .                                                                                                                                                                                                                                                                                                                                              | 40 |
| 4.3  | The distributions of self assessment data collected from the post-playtesting questionnaire, phase one. . . . .                                                                                                                                                                                                                                                                                                                            | 41 |
| 4.4  | Levels from the test version of the game, phase two. . . . .                                                                                                                                                                                                                                                                                                                                                                               | 43 |
| 4.5  | The distribution of self assessment data collected from the sign-up form, phase two. . . . .                                                                                                                                                                                                                                                                                                                                               | 44 |
| 4.6  | The distributions of self assessment data collected from the post-playtesting questionnaire, phase two. . . . .                                                                                                                                                                                                                                                                                                                            | 45 |

# 1

## Introduction

In nature, there are numerous examples of flocking behaviour. Birds fly in large groups consisting of potentially millions of birds, fish swim in shoals for protection, and ants form huge colonies [1]. Considering how commonplace this behaviour is, it is only natural that video game developers would want to include it in their games. For example, if the developers want to include birds flying in the sky, they might want accurate and efficient flocking behaviour to facilitate this. However, it is not a simple task, as it can be challenging design-wise. This project aims to develop an understanding of this aspect by developing a video game that allows flocking behaviour to be used as a tool for the player in a way that creates a satisfying experience. The scale of the flocking behaviour is bounded by the efficiency of the implementation, but also potentially by game design. For example, if the player interacts with a flock consisting of millions of entities, that might make it harder for the player to visually intuit what is happening on the screen.

### 1.1 Purpose

The purpose of this project is to gain a deeper understanding of flocking behaviour as a video game mechanic. To this end, the project aims to develop a video game that uses flocking behaviour. The goal of this video game is to give players a sense of enjoyment from intuitively interacting with flocks, without being frustrated by the experience. The video game developed will be implementing technical solutions to maximise performance. There will also be a large focus on game design and aesthetics with the goal of having an engaging user experience. This video game will then be used to study the implications of flocking behaviour on the experience of players.

In summary, the purpose of this project can be formulated as the following research question: *What engagement implications can be derived when including flocking behaviour as a core mechanic to a video game?*

### 1.2 Stakeholders

The stakeholders intended to benefit from the results of this project are the following:

1. Video game developers who are interested in using flocking behaviour in their games.

2. The target audience of video games which use flocking behaviour, who are interacting with the systems in the game.
3. Video game publishers, who have an interest in publishing games using novel ideas for game design.

### 1.3 Scope

When developing a video game, various limiting factors are present, such as a lack of time and experience. This introduces various challenges to the development team. One such challenge is learning the required knowledge to successfully design the game in time for the final completion date, with another being solving the technical problems that arise throughout the development of the game.

The primary goal of this project is to implement flocking and integrate it into a game. As such, the famous boids algorithm [2] should be implemented, and additional rules should be added to allow the player to interact with the flock. Additionally, the game should have puzzle elements for which the flock is the main tool used for solving. Other features are desired, but not considered top priority for the completion of the project. One such feature is the game being performant enough to have a smooth-running web-build. Furthermore, effective game and level design that provides an intuitive and enjoyable experience to the player would ideally be part of the final build. Visual fidelity, though agreeable, should only be implemented if all other aspects of the game are considered finished. All these features aim to make the game more intuitive and focused on the core components of this research.

To maintain focus on the core ideas of the project and manage prioritisation, the MoSCoW model [3] was utilised. The requirements and features of the game can be evaluated against the different MoSCoW classes (Must have, Should have, Could have and Won't have) to set an appropriate priority. This sets the scope of the project and decides the core functionality.

Certain areas will be considered out of scope for the project. These delimitations include character dialogue, replayability, procedural content generation, combat, quick-time events, multiplayer, and profitability. With this scope in mind, the MoSCoW prioritisation was formulated into:

**The must haves:** (Core features that must be implemented)

- A flock exhibiting flocking behaviour.
- A playable character with movement controls.
- Objects capable of attracting/repelling the flock.
- A way for the playable character to interact with the objects to activate them.
- A way for the flock to complete a simple puzzle allowing completion of the game/moving to the next level. For example, pressing a button or having a certain number of entities in the same spot.
- A simple starting menu, featuring a start button.

- Simple/placeholder graphics consisting of different shapes and, where possible, rougher sketches.

**The should haves:** (Features that are intended to be implemented)

- Different types of flocks exhibiting different properties.
- More levels increasing in difficulty by expanding on already existing mechanics and introducing new ones.
- Different obstacles for different levels.
- Movable objects.
- A way to save progress.
- Some form of tutorial introducing the game's mechanics.
- Original music.
- Original graphics.

**The could haves:** (Features that will be implemented if time allows it)

- Items the playable character can pick up that act as power-ups or change the way they interact with the flock.
- The flock being able to take damage and reduce in size.
- A way to respawn the flock to prevent players from getting stuck.

**The won't haves:** (Features that will not be implemented)

- Jumping. The game should not have platforming aspects.
- Combat. The focus should be on puzzle-solving.
- Any way for the player to directly control the flock (e.g. guiding them with the cursor).
- Additional characters. The game should only feature the playable character and the flock.
- No explicitly told story should be given to the players, as the core game relies on environmental storytelling.



# 2

## Theory

Before continuing to the implementation and results of the project, proper theoretical foundations have to be laid. Without an appropriate understanding of the subject, recommendations and research into the field are harder to verify. As such, a technical introduction to the relevant concepts is necessary.

### 2.1 Emergent Behaviour

The concept of emergence and the behaviour that follows have been examined through many schools of thought, from philosophy to biology, dating back to Aristotle's time, with arguments centred on how complex systems can display properties that do not come from their components alone [4]. Examples of emergent systems include simulations such as Conway's Game of Life [5], as well as the formation of complex snowflake structures which form from freezing water droplets in the air.

#### 2.1.1 Flocks and Herds

The canonical example of emergent behaviour in natural systems is that of flocks and herds. Across multiple species, such as birds, fish, and land mammals, a common and recurring theme is their tendency to move in large clusters that appear highly organised and coordinated.

This happens even with species that lack any clear leader that could act as the controlling body of the group, and instead have each animal only react to what happens nearby, as displayed by certain bird species [6]. These complex group behaviours often emerge in a decentralised, or rather, distributed manner solely guided by local interactions between neighbours [7].

When one of the animals in the group reacts, nearby animals respond to that movement, and this reaction spreads throughout the rest of the cluster. This chain reaction is the driving force that allows herds to change direction quickly, even though no single animal has a full overview of what is happening.

An important key point to note is that the behaviour of the group cannot be understood purely by looking at one individual alone. Each animal follows simple rules such as keeping a certain distance from others, avoiding collision, and heading in the same general direction as its peers. When many individuals follow these rules

together, the result is a coordinated and dynamic group movement that appears far more complex than the rules themselves.

### 2.1.2 The Boids Algorithm

Inspired by this collective motion observed by real-world flocks and herds, the boids algorithm [2] provides a simple computational model for reproducing similar emergent behaviour in simulations. The algorithm was introduced to the world by Craig Reynolds in 1987 and models the behaviour of individual agents, called *boids*, that follow a small set of local rules.

The classic boids model is based on three simple rules:

The **first** rule prevents agents from crowding each other by steering away from nearby boids that are too close, also called the **separation** rule. For a boid  $i$ , this can be expressed as a steering force away from its neighbours  $j$ :

$$\mathbf{F}_{\text{sep}} = \sum_{j \in N_i} \frac{\mathbf{p}_i - \mathbf{p}_j}{\|\mathbf{p}_i - \mathbf{p}_j\|^2}$$

where  $N_i$  is the set of neighbouring boids within the perception radius.

The **second** rule tells agents how to adjust their velocities based on the average heading of nearby boids, also called the **alignment** rule. This can be written as

$$\mathbf{v}_{\text{avg}} = \frac{1}{|N_i|} \sum_{j \in N_i} \mathbf{v}_j, \quad \mathbf{F}_{\text{align}} = \mathbf{v}_{\text{avg}} - \mathbf{v}_i$$

which steers each boid toward the average velocity of its neighbours.

The **third**, and final, rule ensures that the boids do not drift apart by steering them toward the average position of nearby boids, also called the **cohesion** rule. This can be expressed as

$$\mathbf{p}_{\text{centre}} = \frac{1}{|N_i|} \sum_{j \in N_i} \mathbf{p}_j, \quad \mathbf{F}_{\text{cohesion}} = \mathbf{p}_{\text{centre}} - \mathbf{p}_i$$

which moves each boid toward the local centre of the group.

The final steering force applied to each boid is typically a weighted sum of these three components:

$$\mathbf{F}_i = w_s \mathbf{F}_{\text{sep}} + w_a \mathbf{F}_{\text{align}} + w_c \mathbf{F}_{\text{coh}}$$

where  $w_s$ ,  $w_a$ , and  $w_c$  are tunable weights that determine how strongly each rule influences the motion.

## 2.2 Data Oriented Design

As the number of agents that have to be simulated grows, the algorithm becomes computationally expensive. Since the application is a real-time application in the

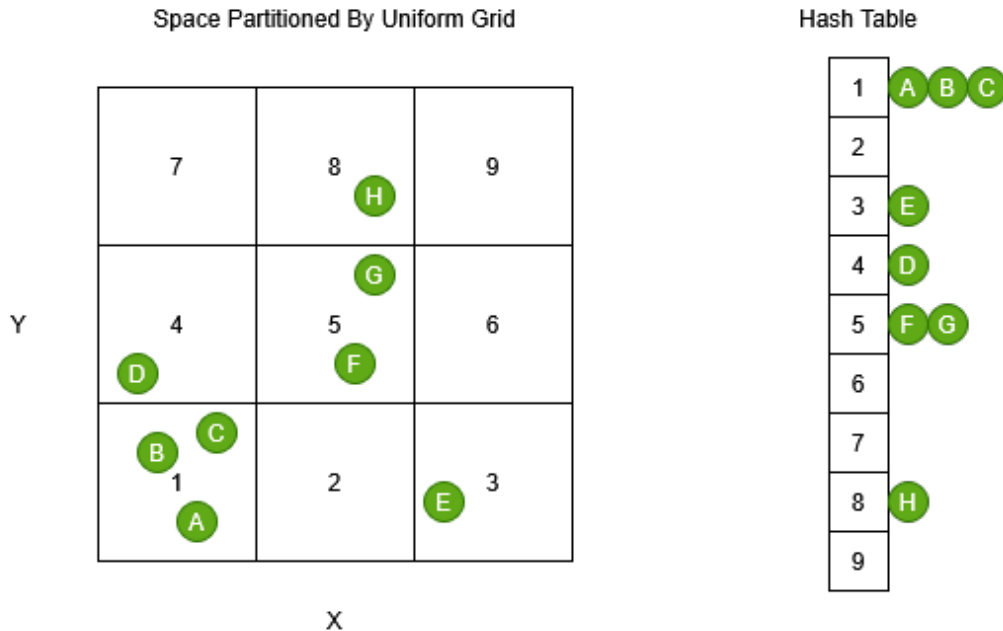


form of a game, performance is necessary in order to ensure a smooth experience. As such, different optimisation strategies should be employed in order to increase the performance. One such strategy is to reduce the neighbour search complexity with spatial data structures.

### 2.2.1 Spatial Partitioning and Indexing

In a naive implementation of the boids algorithm, every agent must calculate the distance to every other agent in order to perform the neighbour search. This results in a time complexity of  $O(N^2)$ . To combat this, spatial data structures which partition the space into separate regions can be utilised. This means that every agent only has to calculate its distance to other agents located in nearby regions. There are several ways to perform space partitioning, including spatial hashing, k-d trees, and octrees.

One variant of spatial partitioning, presented by authors Hastings and Mesit [8], is spatial hashing, which is a method that first subdivides the simulation space using a uniform grid, where each grid cell corresponds to a bucket in a hash table. A hash function then maps spatial coordinates to integer grid coordinates and combines these coordinates to index into the hash table, effectively interpreting the hash table as a one-dimensional representation of a multidimensional array of buckets. A visualisation of this can be seen in Figure 2.1.



**Figure 2.1:** An example of a 2D-space with point objects and their associated buckets in the hash table after spatial hashing.

Hastings and Mesit [8] also inform that objects are periodically rehashed during runtime to update their spatial locations in the hash table. Under the assumption of a well-chosen cell size and reasonably uniform spatial distribution of objects, this

enables  $O(1)$  average time complexity lookups for objects within a given cell. However, the article also emphasises that the effectiveness of spatial hashing depends critically on the chosen grid resolution. If many objects occupy a single cell, the method degenerates to a linear search in that bucket, negating the performance benefits of spatial hashing while still suffering the hashing overhead. Reducing the cell size may alleviate this issue, but doing so increases the number of cells and thus memory consumption. Moreover, when objects have spatial extent rather than being points, smaller cells increase the likelihood that objects span across cell boundaries, resulting in them being stored in multiple buckets and thereby increasing memory usage. Consequently, selecting an appropriate cell size represents a trade-off between memory efficiency, update overhead, and query performance.

Other variants of spatial partitioning often rely on hierarchical tree structures, which recursively subdivide the space into progressively smaller regions. An early example is the quadtree, a recursive spatial decomposition scheme for two-dimensional data presented by Finkel and Bentley [9] in 1974. Similarly, the k-d tree is a data structure presented by Bentley [10] in 1975, that partitions space using axis-aligned splitting planes. In these hierarchical approaches, each node in the tree represents a spatial region and may contain child nodes that further subdivide that region. Unlike spatial hashing, the subdivision is not always constrained to be uniform, nor is the number of regions fixed in advance. Instead, subdivision is typically performed adaptively during runtime.

Many tree-based spatial data structures subdivide a region only when the number of contained objects exceeds a defined threshold [11]. This adaptive behaviour allows for control of object density within each leaf node, ensuring that no leaf node contains more than a specified number of objects. During runtime, objects are inserted into a hierarchical structure by descending from the root to an appropriate leaf node based on spatial coordinates. Query operations similarly only traverse the parts of the tree whose regions intersect with the query. The hierarchical nature of these data structures leads to an average-case logarithmic time complexity  $O(\log(N))$ . However, as Bentley [10] notes, worst-case performance may degrade to  $O(N)$  if the tree becomes unbalanced. Consequently, performance depends on the spatial distribution of the objects.

### 2.2.2 Entity Component System

Another optimisation strategy is to adopt a data-oriented architecture, such as the Entity Component System (ECS) [12]. As opposed to the traditional object-oriented approach, where each game object encapsulates both its data and behaviour, ECS separates objects into three parts: entities, components, and systems. An entity represents an identity, components store only data, and systems contain the logic that operates on all entities with a given set of components. This separation makes it possible to organise data by component type rather than by object, so that the values of the same kind are stored contiguously in memory.

## 2.3 Game Design

There are multiple factors that game developers have to keep in mind in order to make a well-designed game. Some of these factors include the user interface, user experience, accessibility, and puzzle design, as well as game design frameworks. These factors form the main key points when designing systems for users, often the input and output components.

### 2.3.1 MDA Framework

The Mechanics, Dynamics, and Aesthetics (MDA) framework [13] breaks down the different parts that make up game design into three layers. The framework describes *Mechanics* as the algorithmic and data oriented side of a game's design. How these *Mechanics* behave within the game in response to inputs by the player is called *Dynamics*. Lastly, the *Aesthetics* describe the player's emotional response, and in turn, whether the game could be engaging. The game designer is to think about game design in the order of Mechanics, Dynamics, and then Aesthetics, while the player perceives them in the opposite order. By moving between these three layers, game designers can perform better analysis of their games and ensure a well-designed final product.

### 2.3.2 Puzzle Design

As Björk and Holopainen [14] claim, puzzles can be solved through two different approaches, namely inductive or deductive reasoning. Giving the player room to experiment and observe actions and consequences can aid them in coming up with theories about the solution. The authors argue that the process of testing and rejecting hypotheses is in itself puzzle solving, even though this may not necessarily lead to being closer to finishing the puzzle.

In an article they wrote, Davies [15] brings up the importance of making the player understand the puzzle, as opposed to simply solving it. Davies means that overcoming the puzzle through understanding provides a sense of accomplishment, yielding a "eureka moment", which is a goal for the puzzle designer. In line with this, Arvi Teikari [16], one of the creators of the puzzle game *Baba is You*, expresses that his motivation when designing puzzles is to amaze the player, or to pleasantly surprise them.

This is elaborated upon when Davies [15] interviews the game developer Jonathan Whiting, who suggests that an engaging puzzle game is one that applies the cycle of learning and mastering game mechanics effectively. The puzzle designer must thus be aware of how to guide the player toward the intended solution without giving away the solution outright.

Game designer Alan Hazelden, who is also interviewed by Davies [15], further elaborates on the importance of frequently leading the player to new revelations. Solving

several levels using the same method or thought process is not very interesting or engaging according to Hazelden. However, this does not mean that different game mechanics must always be introduced to keep the player engaged. Hazelden suggests exploring the full potential of existing game mechanics by applying them in new ways as this can lead to a deeper understanding and thus, more interesting and satisfying revelations.

### 2.3.3 User Experience

User experience (UX) [17] concerns the entire process of designing a product and encompasses the entire user journey, most importantly delivering a solution that addresses pain points and needs. User experience (UX) and user interface (UI) are often interchangeable terms, but UI can be considered as a subset of UX. Accessibility is also considered part of user experience as it relates to user needs. The next sections will focus more on UI as well as accessibility, while this section will focus solely on UX.

As Bernhaupt [18] discusses in their book, UX can also be seen as an umbrella term connected to Human-Computer Interaction (HCI) research with the goal of understanding the role of affect as an antecedent, a consequence, and a mediator of technology. Bernhaupt describes different dimensions of user experience which have been addressed by researchers in the domain of HCI and games, as seen below.

**Aesthetics:** This dimension describes how aesthetically pleasing or beautiful something is perceived. The pleasure gained from sensory perceptions is something the visual experience deals with.

**Emotion:** A key factor of UX is emotion. One of the three main factors contributing to product experience is emotional experience, including feelings and emotions that are elicited.

**Stimulation:** To what extent a product can support human need for innovative functions, interactions, and contents is described in this dimension. Stimulation can be described as something which can lead to new impressions, opportunities, and insights.

**Identification:** To what extent a certain product allows the user to identify with it is indicated in the identification dimension. The human need to express one's self through objects is addressed in this dimension. Self-expression through an object to communicate identity can be seen as identification.

Other dimensions, factors, and constructs exist that are related to user experience [18]. All of these concepts allow for the measurement of the characteristics of UX.

The experience of playing games is supposed to be positive. Bernhaupt [18] associates this experience with the terms *immersion*, *flow*, and *presence*. These terms

can be described in different ways. Llanos and Jørgensen [19] describe the term flow as a deep level of concentration or attachment. The passage of time can seem to disappear during flow due to deep focus on an activity. Regarding games, flow can be associated with different aspects depending on the player. For some, flow can be initiated by following a plot or exploring an environment. Bernhaupt [18] claims that flow can be characterised as ease of doing and enjoyment, as well as absence of boredom and anxiety. Furthermore, Llanos and Jørgensen [19] use the term immersion as another way of describing involvement. It is often associated with the experience of being absorbed in the game world and surrounded by its environment. Presence is the psychological sense of "being there" in a virtual environment.

### 2.3.4 User Interface

Compared to user experience (UX) a user interface (UI) focuses more on the surface and overall feel of a design [20]. A good UI is something users find easy to use and pleasant.

When developing UI, the concepts of diegetic and non-diegetic can be applied [21]. The terms originate from literary and film theory, but can be adopted to game studies. However, the meaning of the terms will change slightly when migrating from one medium to another. *Diegetic* describes features existing in the game world and its total world of narrative action, while *non-diegetic* describes features that are external to the world of narrative focus.

Diegetic elements are more narrative focused and can provide a greater sense of immersion [22]. The information to be displayed is embedded within the virtual environment, making them seen as "belonging" to the world. Displaying information in this way resembles the experience in the real world, which can enhance realism and immersion [23]. Non-diegetic elements do not necessary have to disconnected from the act of gameplay, rather they are often centrally connected to it [21]. Non-diegetic elements can be placed as an overlay, often as a Heads-Up Display (HUD). These overlays are independent of the world geometry of the virtual environment. Non-diegetic UI may appear intrusive or disconnected from the environment, resulting in disrupting the sense of "being there".

*Portal 2* [24] is an example of a puzzle game that uses both diegetic and non-diegetic elements. As can be seen in Figure 2.2, the portal gun is an example of a diegetic element, since it changes colour depending on the last portal placed. The game also uses non-diegetic elements to indicate what portals are active, which can be seen in the centre of the screen around the crosshair.



**Figure 2.2:** *Portal 2* [24] incorporates both diegetic and non-diegetic elements. This screenshot was taken from a video walkthrough made by IGN [25].

The concept of diegesis is discussed and studied by Pfister and Ghellal [22] in their conference proceeding. The authors concluded that diegetic and non-diegetic elements significantly influence the player experience, but it needs to be considered that the study was only made with a 2D prototype and is not general. The study findings suggested that when certain elements such as menu, pause function, score system, life indicator, or tutorial are implemented in a non-diegetic way, there is an increase of immersion.

Diegetic UI can improve immersion, it can also reduce usability, which Jiang et al. [23] claim in their research on diegesis in video games, but specifically VR environments. The article discusses how diegetic UI can improve immersion, but may also reduce usability. The authors state that non-diegetic UI can be favoured for its clear presentation of information, while diegetic UI can be valued for its immersive qualities.

Traditional overlay features are not typically perceived as particularly intrusive by players, even if they would otherwise break the immersion, as long as the UI features are clearly motivated and provide relevant information in the given game situation [19]. These non-diegetic elements (icons, menus, and windows) are accepted since it enables players to navigate and play the game effectively. But if these UI elements provide information in an unclear and inconsistent manner, giving more information than needed, there is a risk of players becoming annoyed.

The UI should show information in a sufficient and relevant way that allows players to interact meaningfully with the game mechanics and world [19]. For some players, emotional engagement in the fiction is important, while for others, the gameplay

structure is more important. While designing the UI, it is important to be mindful of what kind of game experience is desired to be created. The common factor is still that the UI elements should not contain more information than necessary, since this may risk ruining the involvement.

### 2.3.5 Accessibility

Inaccessible game design makes the experience of playing video games unavailable to certain groups of people, such as those who are unfamiliar with video games, or who have some kind of impairment that is not being accounted for. This alienates these people from the world of video games. Inaccessibility is often accidental, which is why it is important to consider accessibility when creating a game [26]. Making games accessible to everyone, however, is no easy task, as different players might have vastly different needs. There are available accessibility guidelines [27] that can function as checklists over what features a game could have to improve accessibility, but these can still not ensure a specific player is able to play the game [28].

Implementation of recommended accessibility features can be appreciated by a large share of the player base. Many players can, because of these features, enjoy games they might otherwise struggle with. Some accessibility features include allowing players to adjust the game's controls, using both audio and visual cues, and giving players the option to bypass certain content, such as quick time events and some puzzle mechanics. An example of a mechanic bypass is the game *It Takes Two* [29], which has a setting that enables players to press down buttons once every second instead of repeatedly pressing them in quick succession. One hurdle game designers might face when implementing these features is the balance of providing helpful hints and bypasses while keeping the game as challenging as intended.

Accessibility options can be divided into two categories: access and challenge [28]. The access options focus on the player-loop while the challenge options focus on the game world. Some categories touch on areas that would aid this project. The training options include tutorial levels which would be beneficial. The presentation option could also be of interest, but for this project, it is considered higher priority to have this type of accessibility in mind when designing the game and making assets rather than adding several options in the settings.

Using colours is a common way of conveying information, however people with hardships in differentiating certain colours, like those with colour blindness, risk not perceiving the information trying to be conveyed if colour is the only tool used. Avoiding using fixed colours as the only means of conveying important information is a basic requirement for accessibility relating to vision [27]. For example, 8-10% of men have colour blindness relating to the colours red and green, the most common type. It is instead recommended to use colours only secondary to text, symbols, patterns or shapes. It is furthermore advised to be aware of contrasts. For example, distinguishing similar motifs, like football teams from a distance, becomes easier if one team wears clothes of a light colour and the other a dark colour. The lightness

or darkness of a colour is also called its *value* [30] and the value contrast plays an especially important role in the readability of a subject. Furthermore, value contrast can be used to divide content into layers. An example of this is a book, which is readable due to its high value contrast, with black text against a white backdrop, despite the complete lack of colours.



# 3

## Methodology

This chapter presents the methods used to implement the game and the tools utilised in the development workflow. It includes the entire development process and the methods by which the game has been tested.

### 3.1 Tools Outline

The project was developed using *Unity* [31] game engine. The motivation behind choosing this game engine is that *Unity* provides built-in systems for rendering, input handling and physics. Additionally, it has its own data-oriented design paradigm in the form of *Unity DOTS* [32], which makes it overall suitable for implementing and testing real-time performance-heavy flocking behaviour within an interactive game environment. Another reason for choosing to develop the game in *Unity* using C# is because it has been used to implement flocking behaviour successfully in the past [33]. An additional reason for this choice was that this was the game engine that the group's members had most previous experience with.

Version control was handled by using *Git* [34] with a shared repository on *GitHub* [35], allowing for collaborative development and version management throughout the project.

For task planning and coordination, *Trello* [36] was utilised and used as a Kanban board, while day-to-day communication and quick feedback between group members took place through *Discord* [37] and in-person meetings.

2D-sprite assets, animations and some UI assets were created using digital art software and imported into *Unity* [31] during development. The software used were *Krita* [38] and *Affinity* [39] for 2D illustration, while animation was done only in *Krita*.

Music and audio assets were composed using *MuseScore* [40], *Cubase* [41] and *Wwise* [42] before being integrated into the game. These tools allow for the creation and iteration of sound and music that support the gameplay experience.

## 3.2 Workflow

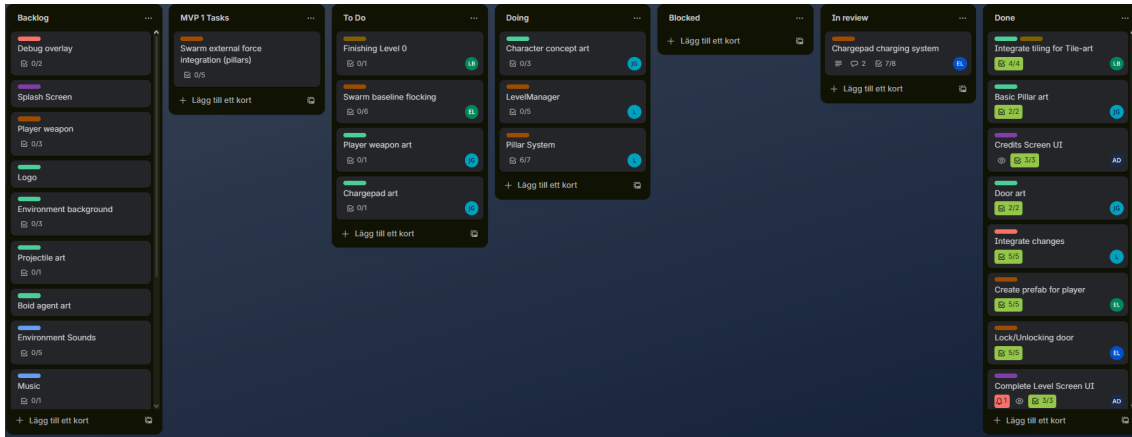
In order to maintain a steady progression for the project that ensures alignment across the team in terms of programming, design, and assets, an iterative approach with regular communication and continuous testing was used. This allowed features to be implemented, evaluated, and refined through continuous improvements in development.

The development process can be divided and represented by the following stages:

1. **Planning phase** with brainstorming and discussions. The initial phase included researching relevant topics, deciding the purpose and scope of the project, the core functionality of the game, and how the overall development process should work. The MoSCoW was created during this stage, which set the outline of the project. A preliminary timeplan was also created to map out both internal and external deadlines.
2. **Development of the minimal viable product (MVP)** that contained the must-haves of the MoSCoW model. After planning, the development of the actual game started, where the focus was on implementing core functionality with placeholder art assets.
3. **Test phase 1** was initiated after the MVP was finished, where the game was tested in multiple sessions. Continuous development and implementations occurred during this phase to add more features to the game and polish it.
4. **Additional development** that focused on the should-haves of the MoSCoW and polishing existing features from feedback gained from the playtesting.
5. **Test phase 2** was initiated after some additional features had been implemented.
6. **Finishing touches** were implemented after the last testing phase to conclude the development of the game.

### 3.2.1 Agile

The game was implemented using an agile workflow [43]. Using *Trello* [36] as the Kanban board, cards were created for different tasks. The first tasks created were focused on creating the MVP, meaning it focused on the must-have features of the MoSCoW. The tasks later focused more on the additional development of the should-haves and on polishing the game. Each card was assigned to one group member and placed in a specific status stage. For example, a card can be marked as "in progress" or "blocked". Each card also included a definition of done, along with checklists that provided more details about the implementation of a specific task. The cards had different labels attached to them to more easily distinguish the various work areas. For example, a card could have the labels "UI/UX", "Art" or "Programming". There were two scrum meetings per week and the sprints lasted from Tuesday to Friday, and Friday to Tuesday.



**Figure 3.1:** Screenshot of *Trello* [36] during the development process of the MVP.

### 3.2.2 Distribution of work

The work in the project was distributed across the group based on their primary areas of responsibility, while still allowing overlap when needed. Each member was assigned a domain, each with their own responsibility. The members were assigned according to personal interest and experience. Any major design decision still had to be discussed with the group, and a collective decision was made to ensure consistency throughout the project.

The distribution of the main domains were as follows:

- *Linda Bergevald* - Level design.
- *Adrian Danielsson* - UI/UX design.
- *Jonathan Jansen Granö* - Art, concept art, character design, and animation.
- *Erik Larsson* - Flocking behaviour and engine programming.
- *Elsa Lenz* - Gameplay programming and music.
- *Salar Loran Telo* - Project leader and audio.

## 3.3 Flocking Behaviour

A central technical objective of this project is the implementation and integration of flocking behaviour in a game. This consists of the actual implementation of the algorithm in the *Unity* [31] game engine, as well as integrating it into the game which enables meaningful player interaction with the flocking behaviour. Though, as previously shown, the boids model is a conceptually simple flocking algorithm, its deployment within a real-time interactive environment presents some practical implementation challenges.

Firstly, the base boids model consisting of separation, cohesion, and avoidance was implemented. The calculation of the separation from a neighbour scales with the inverse square of the distance to said neighbour, which during testing resulted in more believable motion of the flock when compared to using the inverse distance. Alignment force for a given boid is calculated as the average velocity of its neighbours

minus the boid's current velocity. Cohesion was calculated in the same manner as alignment, with the only difference being cohesion uses the position instead of the velocity of its neighbours. Lastly, each factor is multiplied with its assigned weight before combining, allowing for individual tuning of how strong the steering factors are compared to each other.

However, as previously discussed, the time complexity of a naive implementation of the boids algorithm is  $O(N^2)$ . Such a time complexity quickly became problematic as the number of agents in the flock was increased, especially considering the application is a game that runs in real time in which lacking performance is immediately noticeable to the player. To counteract this problem, spatial partitioning and indexing techniques were implemented. Of the most common techniques, notably k-d trees and spatial hashing (as discussed in subsection 2.2.1), this project elected to implement spatial hashing for the positions of the boids to accelerate neighbour queries.

Next, additional behaviours other than the base flocking were necessary to realise meaningful player interaction with the flock. These behaviours include boundary conditions, obstacle avoidance, as well as target attraction and repulsion. For the boundary conditions, this project used reflective boundaries, meaning a soft steering force back toward the interior is applied on agents outside of the level boundaries. Obstacle avoidance is implemented through a combination of soft boundary steering and hard normal force calculations, modulated by a hardness parameter. Two steering forces are computed for each obstacle boundary: a soft avoidance steering and a hard normal force. The soft steering force is applied when a boid is within a configurable margin of an obstacle boundary, scaling with its proximity to said boundary. The hard normal force activates only when a boid is inside an obstacle boundary, pushing it directly outward with a magnitude proportional to the penetration depth. Finally, these forces are combined using a hardness value, where a hardness value of zero relies entirely on soft steering and a hardness value of one uses only the normal steering force. This allows per-obstacle tuning of how boids avoid them. Target attraction or repulsion is the primary way in which the player interacts with the flock in the game, and was implemented in the form of configurable battery pillars that are placed in the game world, affecting the boids in its influence radius. Each pillar either attracts or repels boids within its influence radius using a soft steering force which scales quadratically with the boid's proximity to the pillar.

## 3.4 Game Design

The purpose of the project includes designing a video game, thus it is important to consider different aspects regarding game design and to recognise what challenges may arise.

### 3.4.1 Gameplay

In the context of the MDA framework [13], the gameplay encompasses both *Mechanics* and *Dynamics*. The swarm of bugs that exhibit flocking behaviour is part of the Mechanics and how the player can manipulate them via objects in the level corresponds to Dynamics.

The gameplay consists of a playable character and a swarm of bugs that exhibit flocking behaviour that can be manipulated by the player. The playable character is able to move around and can interact with certain objects by shooting a projectile on it. The gameplay contains different interactable objects that the player or flock can manipulate. These objects are as follows.

- **Chargepad** that the swarm of electric bugs can power up, which in turn produce electricity to open doors.
- **Battery pillar** that the player can activate by shooting a projectile on it. There are two types of pillars: *attracting* and *repelling*. When an attracting pillar is activated, the flock is attracted to it and gathers around it. When a repelling pillar is activated, the opposite happens, the flock is repelled and moves away from it.
- **Archetype selector** that the player can activate by pressing "E", which in turn will change the fundamental behaviour of the flock. For example, it can make the swarm flock more closely together or have them spread out more.

The game consists of levels where the player needs to complete puzzles in order to progress. After a puzzle is completed, a door will open where the player can exit the level and progress to the next one. The player managing to deduce the solution to a puzzle is supposed to lead to the revelatory "eureka moment", previously discussed in subsection 2.3.2.

### 3.4.2 Level Design

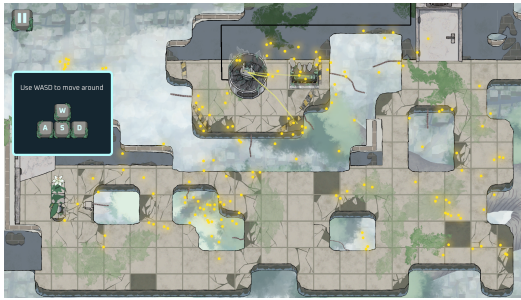
The game's levels were constructed with the intention of introducing the mechanics of the game one at a time in order to give players the opportunity to explore the gameplay while not getting overwhelmed.

**The first level** introduces the chargepad and the attracting battery pillar (see Figure 3.2a). This level can be completed by activating the attracting battery pillar, causing the swarm to get closer to it and in the process also entering the range of the chargepad, powering it up to open the door. The goal was to make this level simple to complete and provide players with an understanding of the core gameplay of influencing the movement and behaviour of the flock to complete the puzzles. By placing the battery pillar on a floating platform inaccessible through traversal, the player is also taught to use the hand-projectile and the bugs to cover for the player character's limited movement options. This level also includes a tutorial to further help players familiarise with the game.

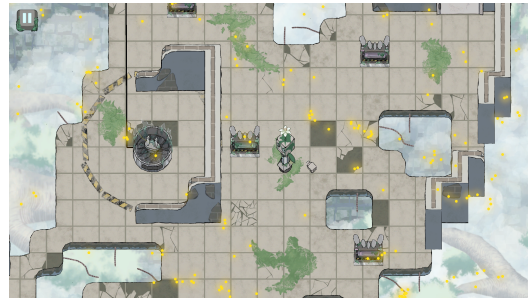
**The second level** introduces the repelling battery pillar and intended to feature

a more challenging puzzle (see Figure 3.2b). Similar to the previous level, it can be completed by guiding the swarm to the chargepad and thus unlocking the door. The additional challenge comes from players having to manipulate the movement of the swarm in a more intricate way, using the repelling battery pillars to concentrate the bugs on one side of the level and then guiding them into the walled off area with the attracting pillar. This level does not have a tutorial nor does it have inaccessible platforms, opting to instead give players more freedom to explore the game's mechanics at their own pace. The *dynamics* aspect of the MDA framework [13] are prominent here as several forces can be combined to affect the bugs in a multitude of ways. Ideally, players will be equipped to handle this challenge with the knowledge they get from the first level together with some experimentation of the new mechanic.

**The third level** is the last level in the game and introduces the archetype selector (see Figure 3.2c). This level's chargepad is surrounded by walls with a gap too small for the player to enter. They must instead get the bugs into the area. This can be achieved by guiding the swarm with the attracting pillars and using the archetype selector to make it easier for the bugs to first enter the area and power up the chargepad. The goal with this level was to expand on the behaviour of the flock itself by allowing players to change between archetypes.



(a) Level 1.



(b) Level 2.



(c) Level 3.

**Figure 3.2:** Levels from final version of the game.

#### 3.4.3 User Interface

The game contains different menus and screens as well as various other visual elements, all of which collectively constitute the game's user interface (UI). It was

discussed which UI elements would be necessary to implement and what would make the game feel cohesive and easy to navigate through. The following menus and screens were developed:

- **Main Menu** - The menu that is displayed when starting the game where the player can navigate to other menus.
- **Level Selection** - The menu where the player can select which level to play.
- **Settings Menu** - Where the player can change audio options. The goal was also to have the player be able to reset level progression, though functionality for this was never implemented.
- **How-To-Play Menu** - Contains information about controls input and basic explanation about the game mechanics together with visual illustrations.
- **Credits Scene** - Contains attributions and can be accessed via the main menu or after completing the last level.
- **Pause Menu** - The player can pause while in-game to quit, change options or look through the How-To-Play.
- **Level Completion Screen** - Screen that appears after the player finishes a level where they can either go to the next level or quit.
- **Loading Screen** - Screen that appears when the game is loading, for example between levels. The screen contains general tips.
- **Heads-Up Display (HUD)** - The game has a very simple HUD that only contains a pause button. As extensions to the HUD, pop-ups at times appear in the game, such as tutorial boxes and button prompts.

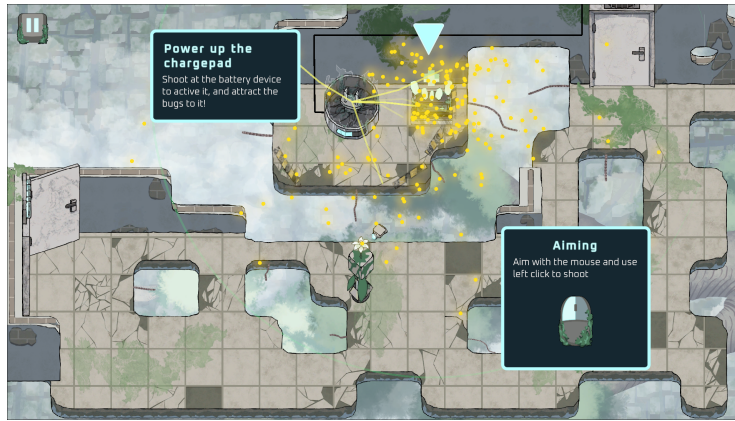
As mentioned in the theory section, it is important to have a UI with clear and relevant information. The UI should not contain more information than necessary, as this increases the risk of annoying the player. This was considered during the development of the UI. The information needed to be displayed in an efficient way without overdoing it and breaking the immersion.

Another consideration was to find a good balance between aesthetically pleasing UI and clear, informative UI. It is important to know which game situations warrant a clearer UI over an aesthetic one.

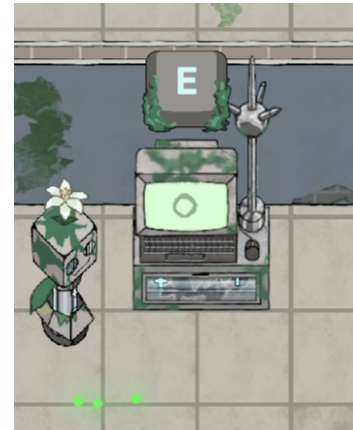
The visual design of the UI needed to be coherent with the style of the game. The UI should feel immersive while still retaining its usability. This was taken into consideration when developing the graphical profile of the UI. The menu buttons were designed to be clear, informative and readable. To make it more coherent with the style of the game, the same colour palette (see Figure 3.20) was used in all of the UI elements.

While in-game, the majority of the UI was diegetic and part of the game environment in order to not break immersion. One of these diegetic elements is the progress bar on the chargepad, which provides information while still being part of the game world. Non-diegetic UI was implemented in the HUD, like the pause button in the corner, as these elements are not part of the game world. In the first level, there is a tutorial where non-diegetic UI was implemented, as can be seen in

Figure 3.3. The reason for this was to have the information clearly displayed so that the player could easily learn the mechanics of the game. In the third level, where the archetype selector is present, the player will get a control input indication when moving close to the game object (see Figure 3.4). This indication is also non-diegetic and serve the purpose of conveying information.

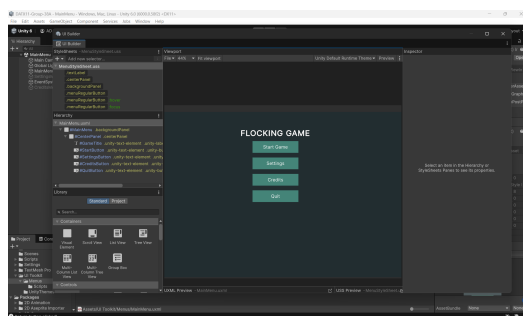


**Figure 3.3:** The first level contains non-diegetic UI elements to display clear information as a tutorial. Diegetic elements can also be seen in the progress bar of the chargepad and the electrical sparks between it and the bugs.

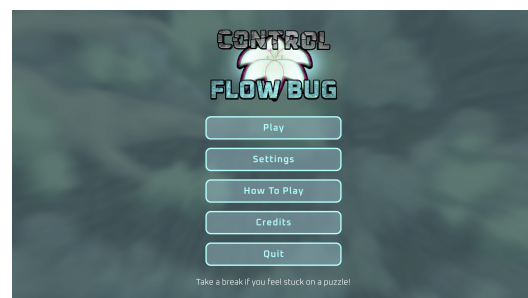


**Figure 3.4:** The archetype selector has a button prompt, which is a non-diegetic UI element.

The different menus started out as simple placeholders with focus on functionality, but were later polished to better fit the aesthetics of the game. The menus and different UI elements were developed in *Unity* [31] using the UI Builder. Figure 3.5a shows the early process of making the main menu in UI Builder. After continuous development and polishing, the main menu was changed to what Figure 3.5b shows. Each menu in the game was developed iteratively and polished throughout the whole development process.



(a) Early rendering of the main menu using the UI builder.



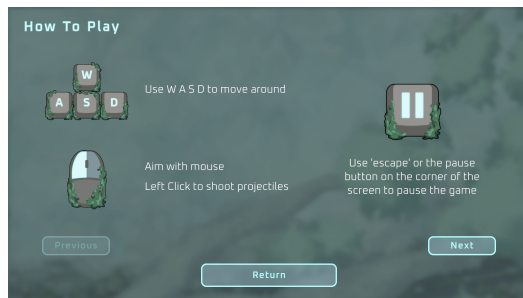
(b) Final version of the main menu containing original game title art and tips that interchange.

**Figure 3.5:** Renderings of the UI for the main menu.

Most of the UI elements were illustrated using the UI builder, which in turn is



inspired by HTML and CSS, meaning the elements were designed using style sheets. Some UI elements were illustrated using the creative software *Affinity* [39] to make them more detailed and stylised. These elements were designed with immersion in mind. Some examples are the art for the controls in the How-To-Play menu (see Figure 3.6a). These UI art assets followed the style of the game more closely.



(a) The first page.



(b) The second page.

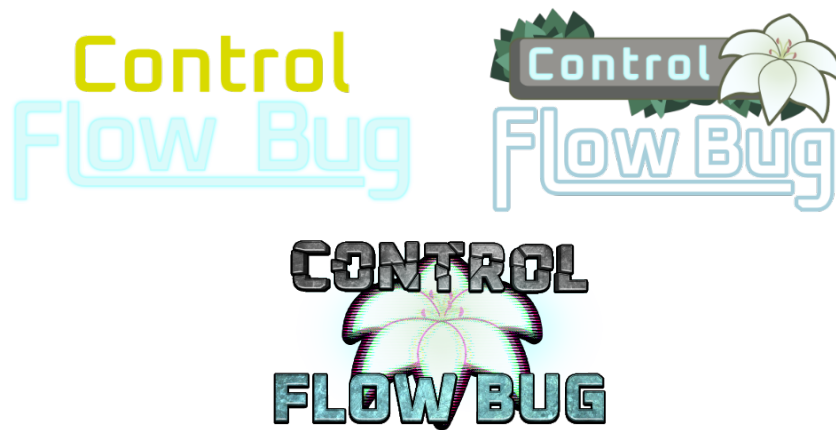


(c) The third page.

**Figure 3.6:** The How-To-Play menu.

The different UI elements gave visual feedback in the sense that the appearance slightly altered when the mouse is hovered over or the element is clicked on. In addition to this, audio feedback was also included by adding different sound effects when hovered or clicked. This was implemented to make the navigation easier and to better convey information about the UI functionality.

It was decided to make a logo and special game title art for the game that would be present in the main menu and credits scene. These were also made in *Affinity* [39] and iterated on throughout the development of the project. Figure 3.7 shows the different versions of the game title.



**Figure 3.7:** Iterations of the game title with the final version at the bottom.

#### 3.4.4 User Experience

While developing the game, the different user experience dimensions (aesthetics, emotion, stimulation, identification) were considered [18]. To be able to follow the different dimensions in addition to having the game give a good experience of immersion, flow and presence, was a challenge, especially for a shorter game.

By creating a puzzle experience, mental stimulation was possible to achieve. The puzzles varied in difficulty, requiring the player to think and thereby increasing the stimulation achieved.

In order to evoke emotion from the game, focus was given to layering the visuals. By focusing on having a good aesthetic with auditory experience and managing some environmental storytelling, the goal of evoking emotions could potentially be achieved. By evoking emotion, identification could possibly be achieved as well, by having the player feel a connection to the game and be able to identify with it.

To make the game more engaging, there was also a goal to achieve a feeling of immersion. By having an intriguing gameplay and puzzle mechanic together with a pleasing aesthetic, feelings of immersion, flow, and presence could be increased. Making the gameplay interesting and satisfying can increase the feeling of flow, creating a focus on the activity while evoking enjoyment and reducing boredom. Making the player feel involved in the game and absorbed by its environment increases the feeling of immersion as well as presence.

In order to determine if these dimensions of user experience were achieved, user testing was conducted to gather feedback.

#### 3.4.5 Accessibility

Accessibility was considered and discussed during early development. It was, however, not of high priority to feature any separate accessibility settings in the final

product. Accessibility was still kept in mind throughout development of the visuals, as the aesthetics from the MDA framework [13] are what make the first impression to the player and are thus important.

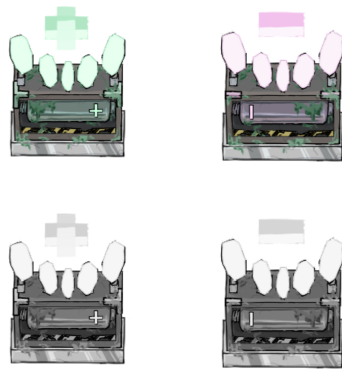
It should be noted that the colour palette was not chosen with colour blindness in mind. For example, as the most common colour blindness affects how a person sees the colours red and green, the battery pillars, seen in Figure 3.9, could have made use of different colours to convey their differences better. However, as green was already prominently used to depict plant life within the game, it was decided that other techniques could be used to cover for the suboptimal colour choice.

Being aware of value contrasts was one technique used to account for colour blindness. For example, deactivation of a game object was communicated by dark values, while the activated versions instead made use of light values, as illustrated by Figure 3.10.

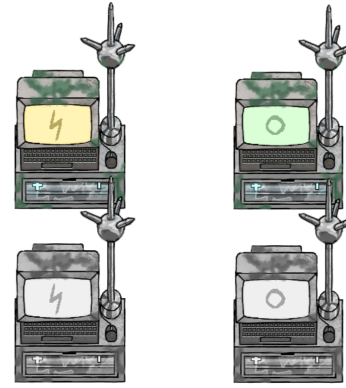


**Figure 3.8:** The deactivated (top) and activated (bottom) versions of the chargepad, attracting battery pillar, and door. The left image shows how greyscale versions show how light and dark values signal different states of activation, even without colour.

Similar activated objects conveying different states relied more on shape to distinguish between them. Examples of which being both the battery pillars, showing either plus or minus to convey their different functions, and the archetype selector, displaying graphical representation of the bugs' behaviours. These variations are shown in Figures 3.9 and 3.10.



**Figure 3.9:** The attracting and repelling battery pillars in colour and converted to greyscale. Instead of value, the difference in shape is used to distinguish the function.



**Figure 3.10:** The yellow and green archetype selectors in colour and converted to greyscale. Instead of value, the shape of the logo is used to communicate the selected archetype.

During the development of the game, it was considered that gaming experience can vary between players and that not every player may know basic video game controls or mechanics. The How-To-Play menu and tutorial textboxes were included as an attempt to make the game more accessible, regardless of previous experience. During early stages of development, only a basic How-To-Play menu existed with control inputs (see Figure 3.6a), but it was deemed insufficient for understanding the game. The tutorial was then implemented, in addition to having more slides on the How-To-Play menu to further increase the accessibility. The additional pages (Figure 3.6b and Figure 3.6c) explains the core game mechanic with text and illustrations, which serves as a complement to the tutorial level.

## 3.5 Aesthetics

Aesthetics and theme were considered when developing the game. As discussed in the theory chapter, *aesthetics* is part of the MDA framework [13] and an important layer for designing an engaging game. Additionally, aesthetics are related to user experience, being one of its dimensions [18]. With a pleasant aesthetic, the overall user experience could improve, potentially leading to a more engaging experience for the player.

### 3.5.1 Style and Theme

After discussions among the group members, it was decided that the game would take place in a post-apocalyptic world. The player would take control of a robot and explore the ruins of the old world, while solving puzzles with the help of a swarm of bugs. The goal was to have the game invoke emotions such as melancholy and loneliness.

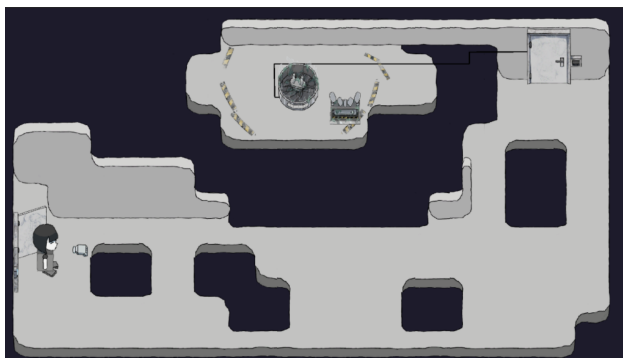
Inspiration was gathered from other video games and also other media such as

movies, TV-series, music videos, and illustrations, such as the ones in Figures 3.16a and 3.16b. This inspiration was reflected in the art, music and the different stylistic choices.

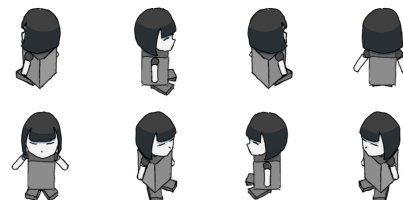
During the development process, the topic of storytelling was discussed. Among the group, multiple members had a preference for story-rich games, so there was an interest in incorporating storytelling into the game. However, it was not a highly prioritised aspect. It was also decided to include no explicit storytelling and instead focus on environmental storytelling that could convey certain emotions.

### 3.5.2 Art

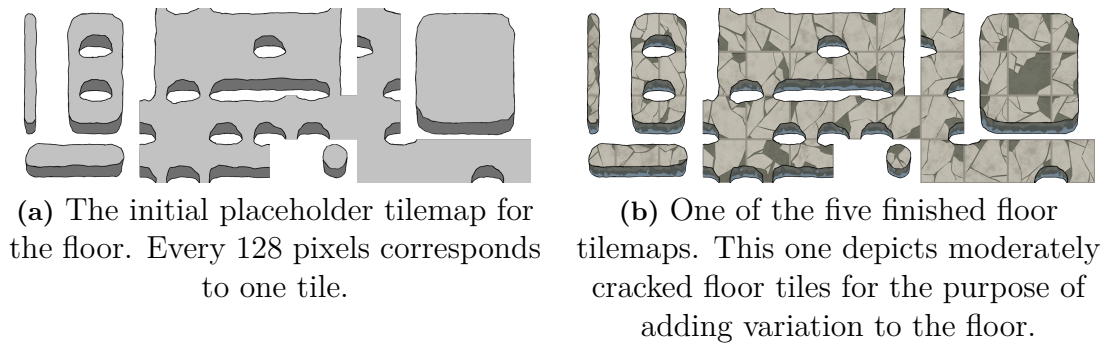
The creation of the assets was done in parallel with the level design. This was accomplished by first defining which elements were needed for the designed levels, followed by creating the bare minimum of placeholder assets to be used when designing and testing the levels. A leading designer for AAA games, Blake Rebouche [44], describes this methodology; the level designer "blocks out" a level with placeholders and later implements the finished art assets during *Art Passes*. This approach also allowed the placeholder assets to be thoroughly tested in different sizes and contexts before creating the final version of an asset. This was done to minimise the risk of completed assets having to be remade because a different level design was decided upon when testing. The "blockout" of the first level with placeholder assets can be seen in Figure 3.11. The earliest placeholder for the player character consisted of eight simple sprites to be able to tell the direction of the player, the spritesheet can be seen in Figure 3.12. Likewise, for level design to be performed in parallel with the asset creation, simple tilemaps were created and later refined. Five additional variations of the tilemaps for the floor and walls were later created for purely decorative purposes. The earliest, as well as the latest, versions of the floor and wall tilemaps are in Figures 3.13 and 3.14.



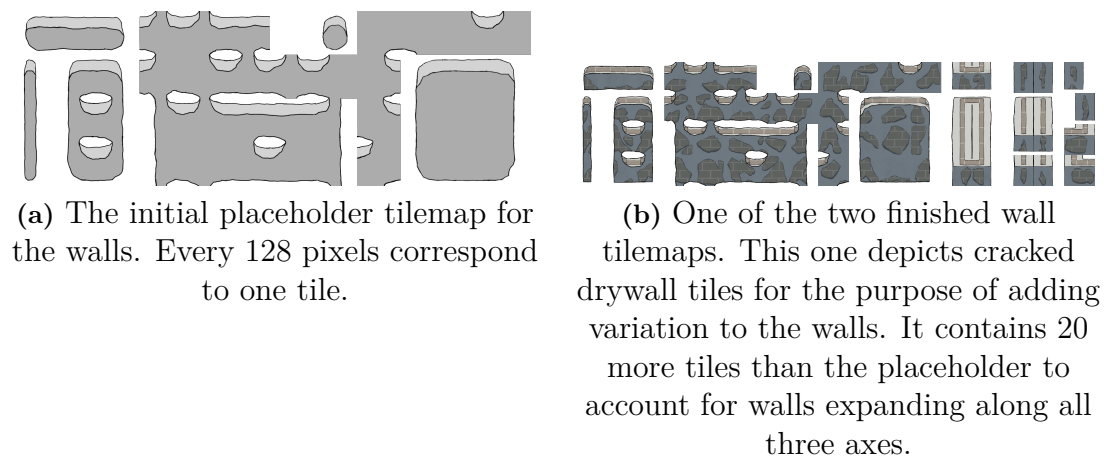
**Figure 3.11:** The first level with placeholder assets.



**Figure 3.12:** The initial placeholder spritesheet for the player character facing in eight directions.



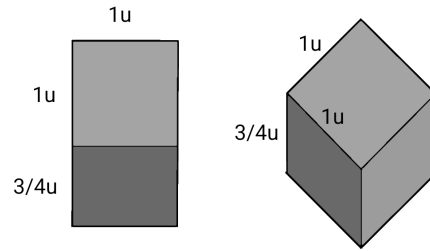
**Figure 3.13:** Initial placeholder floor tilemap and a finished floor tilemap.



**Figure 3.14:** Initial placeholder wall tilemap and a finished wall tilemap.

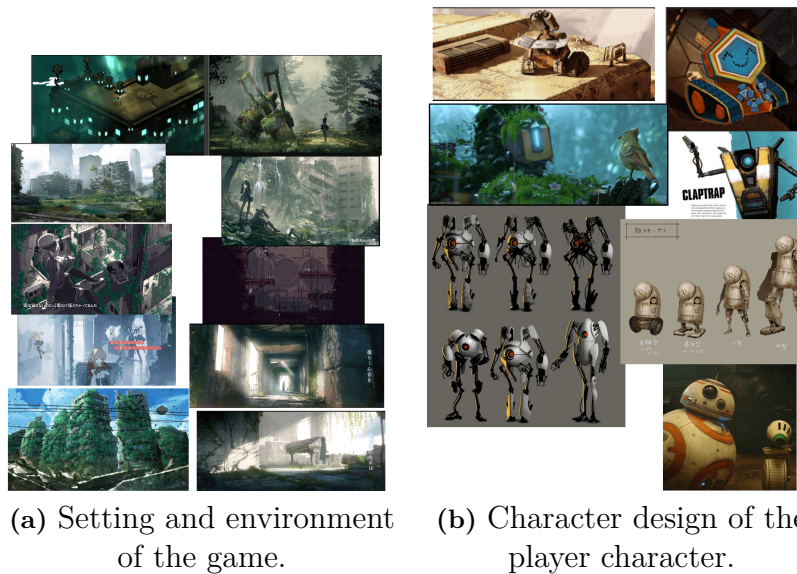
The game assets themselves were drawn using the 2D drawing software *Krita* [38], always using a canvas size close to or larger than the asset's intended size in the game. All animated assets were also created using *Krita*, as it contained tools for 2D animation. All assets were drawn using an established drawing praxis to ensure a unified art style, no matter the asset being drawn. For example, as the game follows the top-down perspective, all assets were drawn as though they were seen from the same angle. This was done by establishing how a cube would look in the correct perspective and using it as a reference for creating more advanced structures in the same perspective. Such a cube had a square base, but a height of  $3/4$  of the base's height, as seen in Figure 3.15.





**Figure 3.15:** Cubes in the top-down perspective used in the game. Each side is one unit of length, except the height, which is  $3/4$  units.

In order to solidify the game’s aesthetics and themes as early as possible, *mood boards* were assembled. The mood boards consisted of images featuring motifs that reflected the desired post-apocalyptic aesthetics of the game. Mood boards can be used as a helpful tool when communicating ideas between designers and non-designers [45]. As such, the use of mood boards helped ensure the entire development team could assist in inspiring and establishing the visual identity of the game. Examples of such mood boards can be seen in Figures 3.16a and 3.16b for the setting and the character design, respectively.



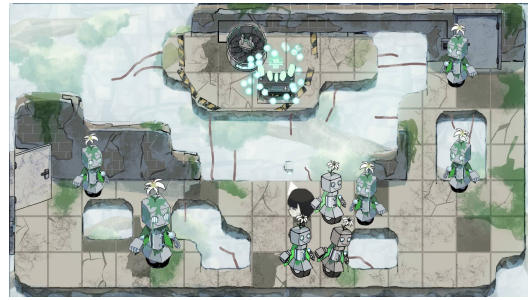
**Figure 3.16:** Mood boards. The images are taken from *Transistor* [46], *NieR: Automata* [47], *Rain World* [48], 天ノ弱 / 164 feat. GUMI [49], 天ノ弱 / Leo/need [50], 僕らまだアンダーグラウンド - Eve MV [51], 遊生夢死 - Eve MV [52], *Wall-E* [53], *Broken Age* [54], *Overwatch: The Last Bastion* [55], *Borderlands* [56], *Portal 2* [24], *NieR: Automata* [47], and *Star Wars: The Rise of Skywalker* [57].

After an idea of the aesthetics had been established, concept art was drawn early on in development. Multiple iterations of concept artworks depicting ideas of the different aspects of the game were drawn based on the aforementioned mood boards.

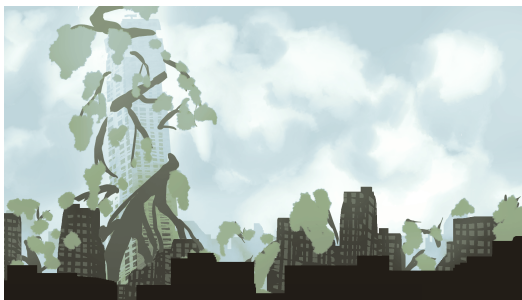
Examples of concept artworks depicting the evolution of the game's setting can be seen in Figure 3.17. The setting was decided to be a tall overgrown abandoned building. This was reinforced by including greenery, rebar, and general deterioration in all elements.



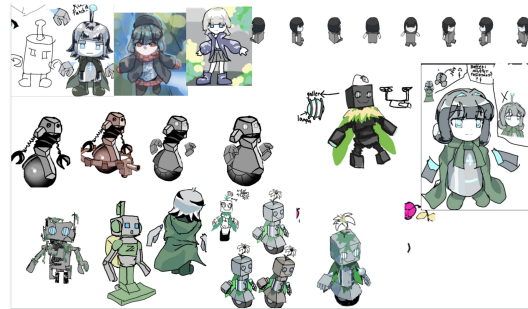
(a) The earliest concept art of a run-down building in a post-apocalyptic setting.



(b) The final concept art of a level in the game with variations of the player character placed in the scene.



(c) Concept art of the overgrown building the game takes place in from the outside.



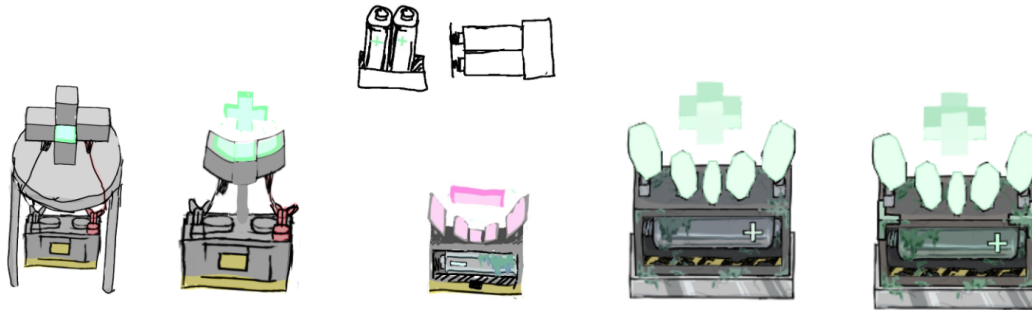
(d) The evolution of the concept artworks of the player character.

**Figure 3.17:** The concept artworks of the world and player character.

Similarly, multiple versions of the game's assets and the character design were explored before settling on final designs, examples of which are in Figure 3.18 and 3.17d. The final character design combines a man-made robotic appearance with organic greenery in order to reflect the post-apocalyptic setting and reinforce the idea of having been abandoned for a long time. The robotic parts made use of symmetrical shapes with clearly defined planes to look artificial. Furthermore, the player character's abdomen was made to be a battery, as it was decided that all machinery that did not require the electric bugs to function needed to contain a battery to power themselves. The greenery, on the other hand, made use of asymmetry and saturated green colours to appear more organic. The white lily on the robot's head further emphasised the idea of nature having reclaimed the world, using a clear plant-motif. The white lily was chosen due to it symbolising rebirth and purity in the language of flowers [58]. These terms aligned with the themes of the game as the

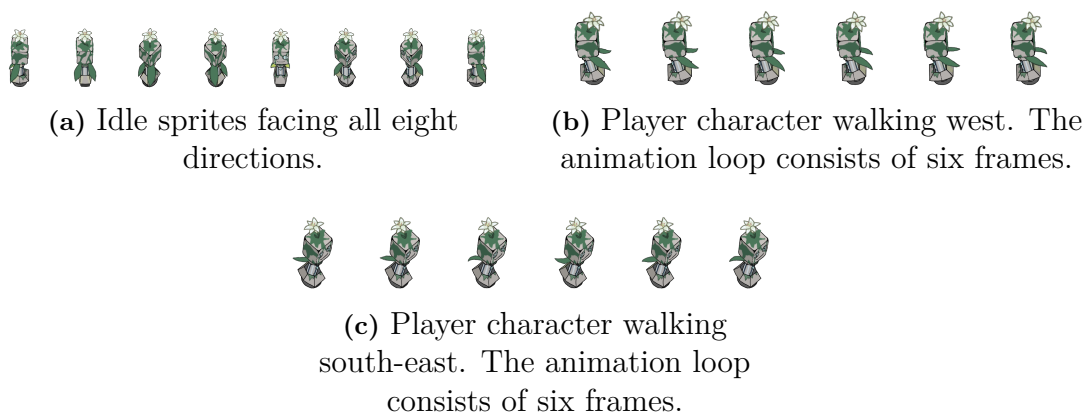


post-apocalyptic setting was to be more focused on the beauty of a world reclaimed by nature, rather than the darker implications of a destroyed world.



**Figure 3.18:** The evolution of the appearance of the battery pillar.

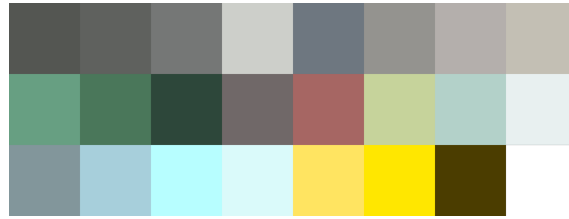
Minor details were, however, changed between the concept artworks and the finished assets due to the added context of being able to view the assets within the game. The most significant redesigns were done to the player character in order to look better in-game and become easier to animate. The redesigns amounted to more muted colours, the more symmetrical flower atop the head, and thinner wing-like greenery on the back, as seen in Figure 3.19a. Such redesigns helped save a lot of time as the player character in its entirety was made up of 56 unique hand-drawn images (walking for six frames in all eight directions, as well as standing idly facing all eight directions), meaning even small additions to the character design could lead to a lot of additional labour. As examples, a few of the finished animation frames of the player character can be seen in Figure 3.19b and 3.19c. Another example of a practical redesign was the colour of the bugs being changed from blue to yellow in order to be more visible against the pale-blue backdrop.



**Figure 3.19:** The final spritesheets of the idle pose and walking animations.

A colour palette was established by sampling the colours of the concept artworks, as seen in Figure 3.20. Said palette was then used by the artists when designing both

the game world's art assets, as well as the UI elements. This resulted in the artists being able to create unified aesthetics for all parts of the game.

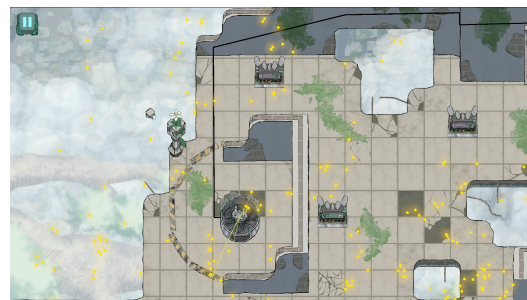


**Figure 3.20:** A colour palette used by the artists when designing assets.

Toward the end of development, purely decorative elements were drawn in order to further immerse the player in the game world. In order to show the decay of the building the game takes place in, broken variations of the base tilemaps, like the ones seen in Figure 3.13b and 3.14b, were overlaid over the existing levels to depict wear and tear over time. To further decorate the stages, assets for moss and rebar were created and placed above the tiles. In Figure 3.3, one can see an example of a level before the decorative elements were implemented, and the final version can be viewed in Figure 3.21.



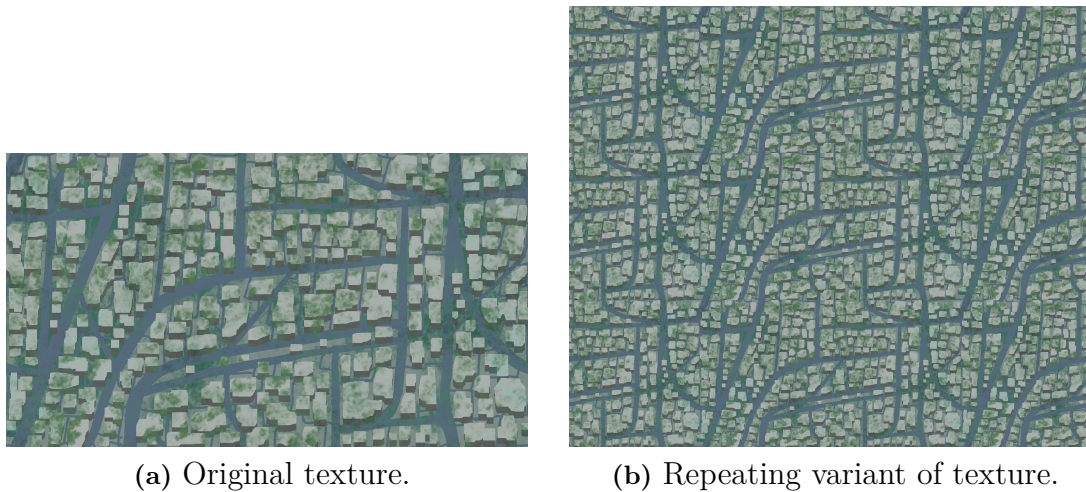
(a) The first level with all decorative elements implemented.



(b) The second level with more atmospheric perspective than the first level due to its higher altitude.

**Figure 3.21:** The first two levels, fully decorated. The atmospheric perspective increases every level to show an increase in altitude.

To help sell the idea of being inside a tall building, a backdrop was also constructed. It portrays a cityscape from above with clouds passing by and large tree branches stretching from the sides to illustrate the concept of a tree having grown around the building, as illustrated in Figure 3.17c. Atmospheric perspective increasing when being higher up was done by both having a bright image overlay the background and also by creating brighter variations of the background assets. This can be observed when comparing Figure 3.21a with Figure 3.21b, the latter being set at a higher altitude. In order to account for potentially having a high amount of levels, and thus a high building, the cityscape image was drawn as a texture that could be expanded by repeating it along its sides. Thus, the cityscape could potentially be zoomed out indefinitely. The textures can be seen in Figure 3.22.



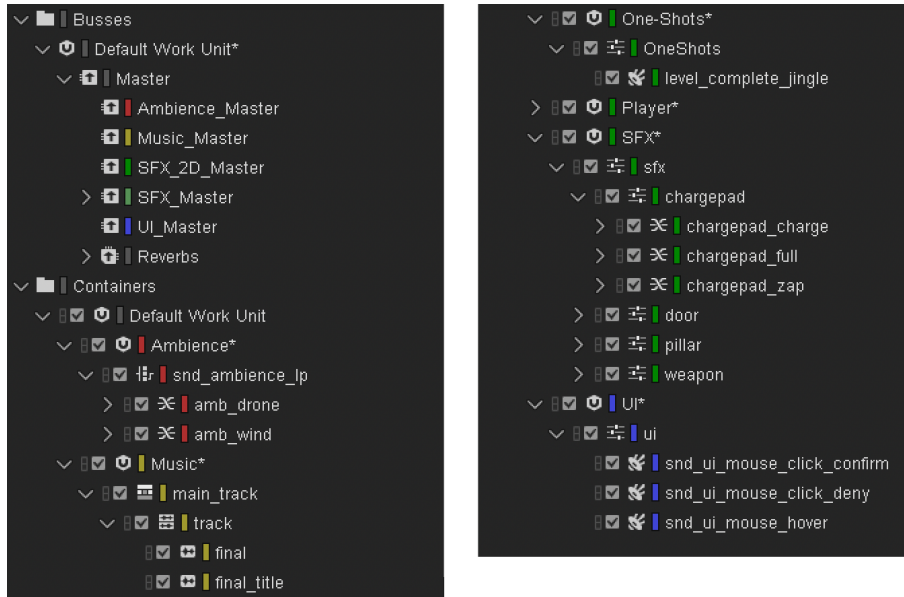
**Figure 3.22:** Textures of cityscapes.

Frequent inquiries were made to the entire development team to ensure the assets being drawn were of sufficient quality and followed the agreed upon style of the game. This could include whether the asset fit the game setting or not, whether the asset supported or distracted from a puzzle, or if the asset sufficiently kept accessibility in mind. Frequent feedback thus led to a game whose visuals were carefully constructed to aid the user experience as a whole.

### 3.5.3 Sound Effects

The sound effects were developed iteratively to support clarity, feedback, and overall atmosphere in the game. Given this project is a puzzle game built around players' indirect interaction with the flocking entities and the environmental objects, the sound design focused on making important actions easier to perceive and interpret. Priority was therefore given to sounds connected to movement, interaction, UI navigation, and charging related events.

The sound effects were produced using *Wwise* [42], which made it possible to organise, test, and adjust audio events directly in relation to gameplay. In *Wwise*, the sounds were organised into categories based on their function in the game, such as UI sounds, interaction sounds, and ambient or environmental sounds. This made it easier to iterate on groups of related sounds and maintaining a more structured implementation. This structure can be seen in Figure 3.23.



**Figure 3.23:** The *Wwise* [42] project hierarchy used to organise audio assets by function, including separate categories for ambience, music, sound effects, and UI audio.

The implemented sounds were connected to gameplay through audio events, allowing specific sounds to be triggered when the player performed certain actions. Some examples of audio events are sound effects for when the player launches the hand projectile, the door opening, and a jingle for when the player completes the level. The sound effects also had to remain noticeable without becoming overwhelming during repeated play. This was addressed by iteratively adjusting volume, timing, and overall character of the sounds in relation to the feedback and pacing of the game. In practice this meant that sounds with an informational purpose, such as a door opening, were kept short and distinct, while more atmospheric sounds, such as wind, were made less intrusive so that they support the world of the game, without interfering with the puzzle-solving aspect.

The final set of sound effects was essentially intended not only to enhance the immersion, but also act as a practical layer of feedback that complemented the game’s visual design and readability.

#### 3.5.4 Music

The music was developed with inspiration from the song *Dirtmouth* [59] from the game *Hollow Knight* [60], specifically the track as published in *Hollow Knight (Original Soundtrack)*. This source of inspiration was chosen because the members of the group agreed that it gave the correct kind of melancholic and lonely feeling that was desired for the game. Since the game takes place in an abandoned and to some extent overgrown environment, the music was intended to reinforce a sense of stillness, decay, and quiet reflection rather than urgency.

The composition was initially written in *MuseScore* [40], where the melody, bass, and chords were first sketched for solo piano. This allowed the musical ideas to be developed in a simpler form before any arrangement and production decisions were made. The initial piano sketch can be seen in Figure 3.24.



**Figure 3.24:** The initial piano sketch of the background music written in *MuseScore* [40], showing the harmonic and melodic foundation that was later arranged and produced for in-game use.

The piece was in rondo form, making it naturally loop smoothly by simply ending the piece with a dominant chord and beginning it with a tonic. This made it so that it could play continuously during gameplay without abrupt transitions. The rondo form allowed for reintroducing familiar material while still allowing for variation, which helped the loop feel coherent without becoming too repetitive. The harmonic language was centred around the key of D minor, along with the use of diminished chords with dominant function, to achieve the melancholic feeling sought. The chord of D minor was repeatedly used, grounding the music in that key. This made it so that the chords with dominant function were clear in their purpose. The music was written as four groups of eight measures, with each group structured as a perfect authentic cadence starting on the tonic and ending on a chord with a dominant function. This made it so that groups always resolve seamlessly into the next. This made each round of the rondo have a strong sense of tension and release.

After the music was written in *MuseScore* [40], it was arranged, orchestrated, and produced in *Cubase* [41] and adapted further to fit the intended atmosphere of the game. This stage involved selecting instrumentation, shaping dynamics, and refining the overall texture of the track so that it better matched the visual identity of the game. Attention was also paid to tempo, register, and rhythmic density, as a slightly slower and more restrained musical expression was considered more suitable for a puzzle game in which concentration and atmosphere were more important than tension. The produced track was then prepared for integration together with the rest of the audio assets.

## 3.6 Testing & User Feedback

To gather feedback on what could be further developed and whether the research question had been answered, playtesting of the game was conducted. These playtesting sessions were conducted simultaneously as part of the development process.

### 3.6.1 Testing Process

All playtesting sessions had the same structure, consisting of different steps that incorporated common game user research (GUR) methods [18].

1. Each tester filled out a sign-up form, which informed them about how the data will be handled and signed an agreement on the data usage. Additionally, they provided information about their experiences with playing puzzle games in particular, and video games in general.
2. Each playtesting session had two test moderators, one of whom took on the role of secretary, taking notes of what the player was saying. The other had the role of test leader and interviewer. At the start of the session, the moderators gave a short introduction and explained how the playtesting would work.
3. The tester was given different tasks, such as playing through a level. During the tasks, the tester was instructed to explain their behaviours out loud while playing the game, following the think-aloud protocol [18], a natural expansion of behavioural observation.
4. After the tester completed the tasks, an interview was conducted to ask follow-up questions and gather additional feedback. Interviews provide direct insight into the player's experience, making it a common method of qualitative subjective inquiry [18]. The skill of the interviewer influences the quality of the data collected during an interview session. Interviews enable a greater degree of depth when analysing player opinion, emotions, and reactions.
5. Lastly, the tester was instructed to anonymously fill in a questionnaire, enabling them to freely criticise the game and for data to be collected in the form of statistics. A questionnaire is another common GUR method for collecting large volumes of data from many different players [18]. A questionnaire can be presented directly after a gameplay event or a gameplay session. The experience remains present in memory, making it less biased compared to recalling an experience after the gameplay session.

The data being collected followed the GDPR [61] guidelines to ensure proper handling of personal data was in place.

The game was tested in two phases during the development process. The first phase was done after the MVP had been developed with some additional implementations. Meaning, the game included the must-haves of the MoSCoW, but also some should-haves. There were two levels in the game, with the second level having increased difficulty and introducing repelling battery pillars as a new mechanic.

The second phase was done after the feedback of the first phase had been reviewed and additional features had been implemented. The game version for the second test phase included all of the must-haves and most of the should-haves. Beyond the MoSCoW features, multiple visual feedback elements were implemented in addition to certain art assets being updated. Every session in each phase had the same testing structure with the same version of the game and interview questions, to ensure consistency.

The playtesting sessions focused mainly on game design, aesthetics, and, overall, how intuitive everything felt. The testing explored the different user experience dimensions and how well the game achieved them. The full list of different tasks and interview questions for each phase can be found in Appendix A.

Each testing phase aimed to have 12 playtesting sessions, with one participant in each session. Meaning, a total of 24 playtesting sessions. The participants were not repeated, since a tester of a puzzle game cannot test the same puzzle again, as the solution is already known. Therefore, it was decided to have different participants for every session. An attempt was also made to have a diverse group of testers, with different experiences in gaming and knowledge of puzzle games.

### **3.6.2 Tester and Conductor Bias**

During testing, it is possible that results are affected by bias, both from the tester and from the test conductors.

It is important to consider the accuracy of the test results and how much value can be given to the provided feedback. The people selected for the tests were personal contacts within the group. This should be considered when looking at the test results, since it can affect how the testers choose to criticise the game and provide feedback, since they may have an additional bias on top of those who are already experienced "gamers" or "puzzlers".

The tests were conducted by different people with the roles of interviewer and secretary, meaning an inconsistency may have occurred between the sessions. The interview questions may not have been asked in exactly the same way, meaning different kinds of answers may have been provided for different sessions.





# 4

## Results

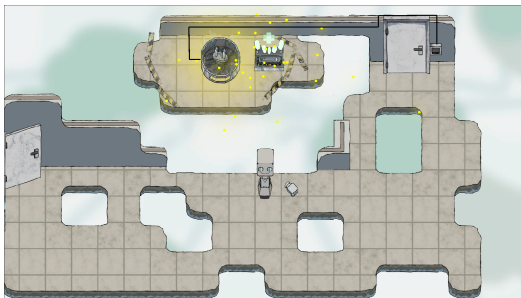
The results of the two test phases were evaluated and reflected upon to know what improvements to the game can be implemented, as well as determine some answers for the research. The results are presented by figures with data from the forms in addition to summaries of the observations and answers during the interviews of the playtesting sessions. Throughout the project, two testing phases were performed, allowing for iterative improvement on feedback. As part of this iterative approach, improvements were made to the game between testing phases. In the end, this resulted in three different game versions: one version for each testing phase, and one final refined version.

### 4.1 First Test Phase

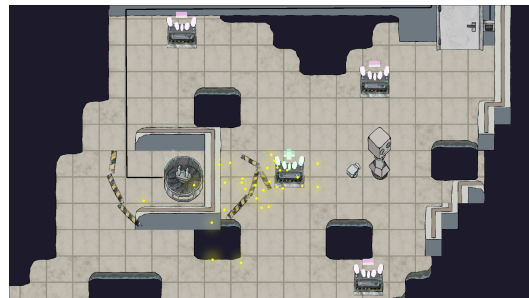
As mentioned in subsection 3.6.1, the game version used in the first testing phase included all the must-haves from the MoSCoW and some of the should-haves. This version had two playable levels.

The first level, as can be seen in Figure 4.1a, contained an introductory puzzle where the player was supposed to activate an attracting battery pillar. The first level did not yet have tutorial pop-ups implemented and the only tutorial that existed was the control input in the How-To-Play menu.

The second level, as can be seen in Figure 4.1b, contained both attracting and repelling battery pillars.



(a) First level.



(b) Second level.

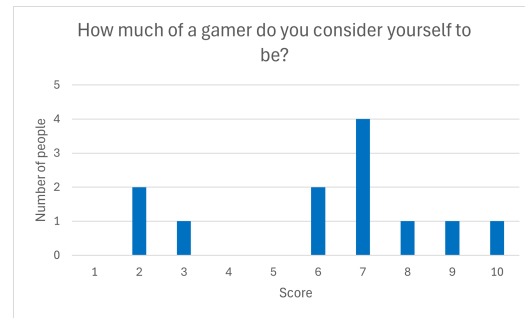
**Figure 4.1:** Levels from the test version of the game, phase one.

### 4.1.1 Quantitative Data

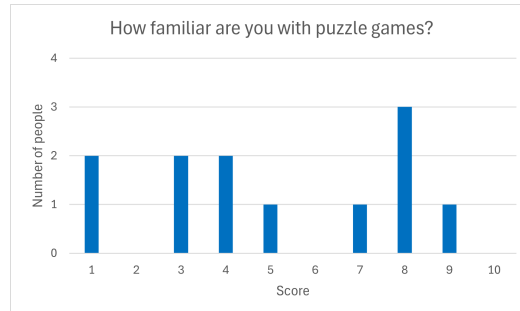
For the first phase of playtesting, the data shown in Figure 4.2 was collected from the sign-up form about the 12 playtesters involved. The ages ranged from 21 to 28, with most testers being in the lower end of that range. There were more people who played games frequently than not. The distribution of how much of a "gamer" the testers considered themselves was quite even, ranging from a score of two to a score of ten, and was biased slightly towards the higher end. The familiarity with puzzle games was also relatively evenly distributed, with scores ranging from one to nine.



(a) How frequently playtesters play video games.



(b) How much of a "gamer" playtesters identify as.



(c) Playtesters' familiarity with puzzle games.

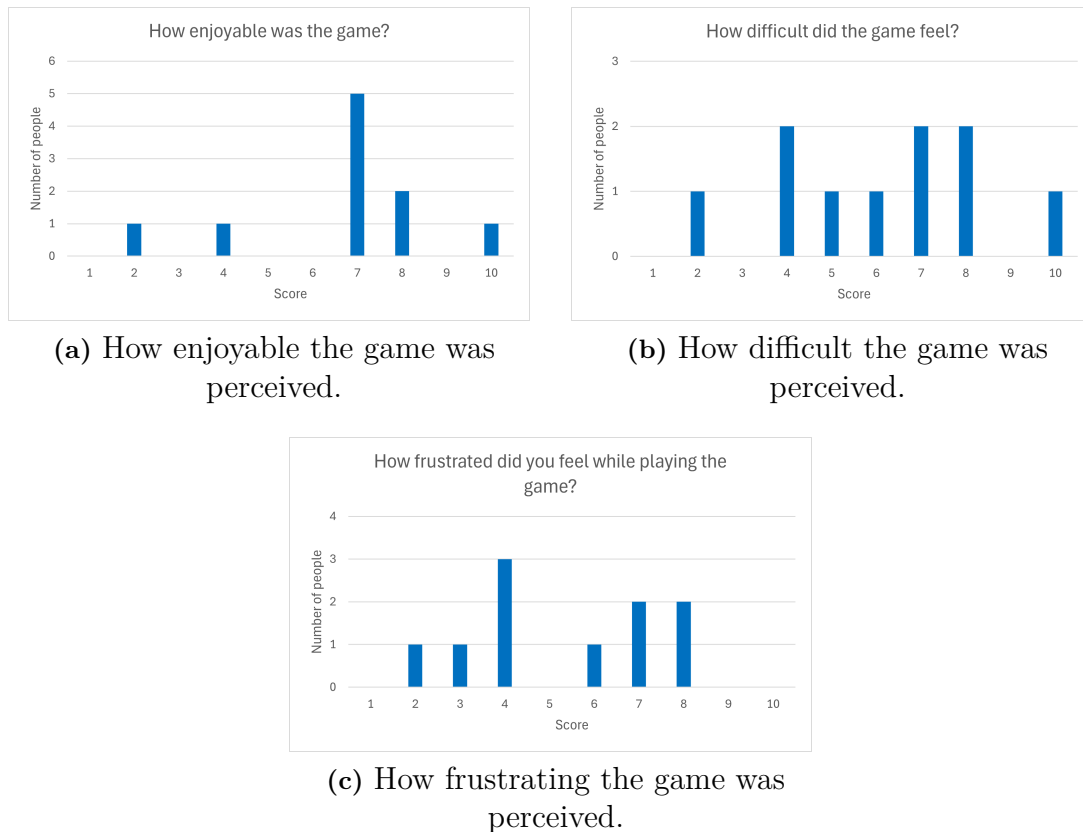
**Figure 4.2:** The distributions of self assessment data collected from the sign-up form, phase one.

The following Figure 4.3 shows the data collected from the post-playtesting questionnaire for the first phase of testing. In summary, testers mostly moderately enjoyed playing the game, while opinions about the difficulty of it were varied, ranging all the way from a score of two to a maximum score of ten.

The reason for this may be that testers with less experience playing video games may not have been properly introduced to the game, while more experienced testers were able to figure out how the game worked by themselves. However, it is not possible to verify this hypothesis. This is because the questions regarding testers' experienced difficulty were part of the anonymous post-playtesting questionnaire, while questions about their prior experiences were instead part of the sign-up form.

Finally, frustration levels were also to some extent varied, but not as much as the feeling of difficulty.

A hypothesis that can be drawn from the post-playtesting questionnaire is that testers who found the game to be more difficult also found it to be more frustrating, and vice versa. Testers that gave a high score in difficulty also put a high score in frustration. This pattern can be seen in majority of the responses in the questionnaire. However, the enjoyability varied where some testers that found the game difficult and frustrating could still have a quite high score in enjoyability.



**Figure 4.3:** The distributions of self assessment data collected from the post-playtesting questionnaire, phase one.

### 4.1.2 Received Feedback

In summary, the recurring problems were as follows.

1. It was hard to know what was interactable. This was generally attributed to the colour and style of the interactable objects being too similar to the non-interactable objects, making them hard to differentiate.
2. It was unclear how you should go about solving the puzzle, even after reading the How-To-Play page. This could have to do with the fact that players were told to look at the How-To-Play page before playing any levels, so the information might have seemed abstract, since players did not know what the

game was even about at that point.

3. Many testers tried to interact with or shoot at the door to open it at first before interacting with the battery pillars. This was most likely due to how the door was designed and that there was a square keypad beside the door that many testers thought they could interact with.
4. There is a lack of feedback from things like the chargepad charging up and what the battery pillars do, making their purpose and use unclear. This was caused by this feedback not having been implemented yet, instead being represented with either placeholder animations or no animation, making it hard to parse.
5. Testers did not understand that the bugs were what charged up the chargepad. This was due to the fact that the chargepads and bugs did not visibly interact, meaning there was no feedback to tell the player that the bugs were doing anything with the chargepad.

### 4.1.3 Evaluation

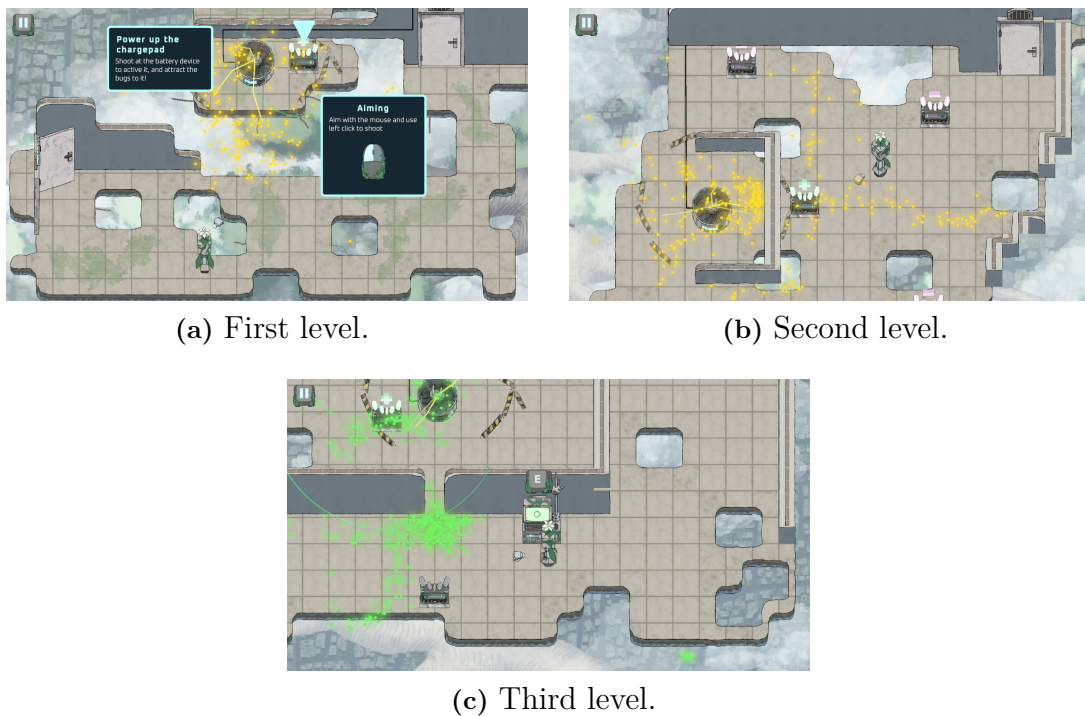
A majority of the testers experienced the game as enjoyable, but some frustrations existed. Testers often experienced confusion about how the game mechanics worked. Many testers completed the puzzle without understanding how or why it happened. The puzzles were completed by "randomly" interacting with the interactable objects until the door suddenly opened. To address these problems, the following features were added (along with other, minor changes). The quality of these solutions was then assessed in the second phase of testing.

1. A clearer tutorial in the first level, using modal dialogues to walk the player through the level. This avoids the problem of the player reading the How-To-Play menu without any context for what the game is like. In addition to this, the How-To-Play menu was updated with multiple pages to better explain the game mechanics.
2. A simple HUD was created with a pause button in the corner, to have an additional way to pause the game than exclusively using the "escape" key. Some testers commented that if you are not experienced with playing games, it might not be intuitive that "escape" is the key which pauses the game. Therefore, an additional pause button was implemented.
3. An electricity effect between the boids and the chargepads, to indicate that the boids charged up the chargepad, as well as what the radius of the chargepad's range was. This would serve as clear feedback to the player that the bugs and chargepads interact with each other.
4. Pulse effects emanating from the battery pillars, to make it clear that they either attract or repel. This would make the purpose of the pillars clearer to the player.
5. Art assets were updated to both make the game look more pleasing and to convey visual information better. One such example is the door getting a visual update to minimise confusion, which was done by removing the keypad.
6. The number of boids were significantly increased to give a better understanding of their behaviour.

## 4.2 Second Test Phase

For the second test phase, the game consisted of the same two levels as the previous test version, but with added improvements, and a new third level. The different levels can be seen in Figure 4.4.

The first and second level have the same level design as the previous versions, but with added visual feedback and general polishing. The first level additionally has the tutorial pop-ups implemented. The second level had wall collisions for the boids added, which was not present in the previous test version. Additionally, the walls were made bigger in order to reduce frustration when attempting to get the boids into the area. The third level introduced a new concept and included an archetype selector.

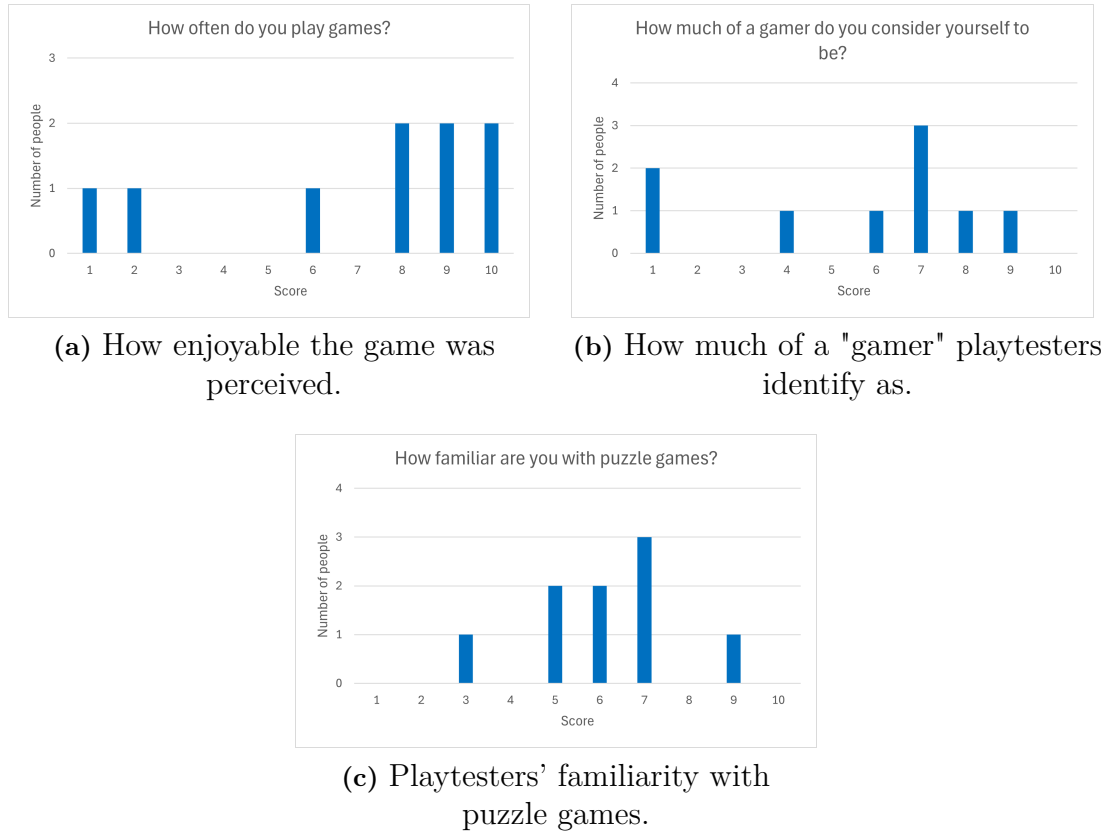


**Figure 4.4:** Levels from the test version of the game, phase two.

### 4.2.1 Quantitative Data

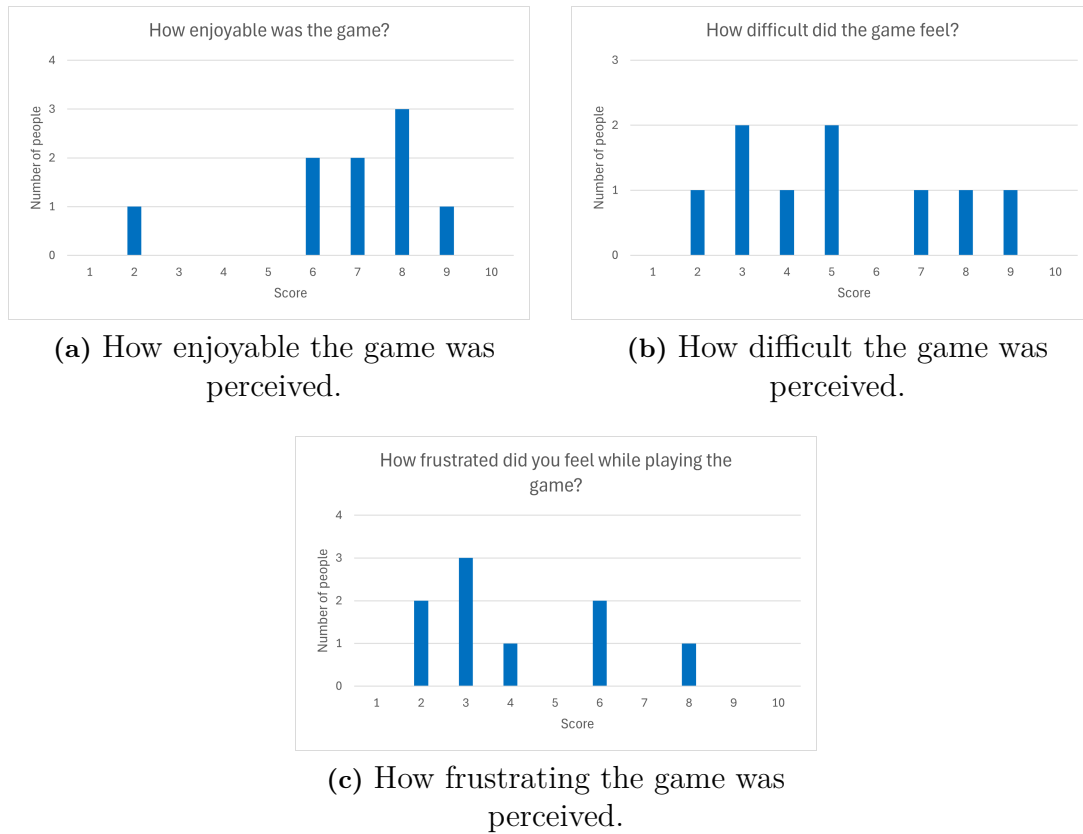
There were a total of 9 playtesters for this session, meaning the goal of 12 playtesters was not reached, and the second test phase had a somewhat smaller testing sample pool compared to the first phase. This was due to time constraints. Similar to the first test phase, the data that was collected from the sign-up form is represented in Figure 4.5. The ages this time ranged from 21 to 30, with most testers being in the lower end of that range. Once more there were more people who played games

frequently than not. The distribution of how much of a "gamer" the testers ranged from a score of one to a score of nine. The familiarity with puzzle games had scores ranging from three to nine, with a majority of testers between five and seven.



**Figure 4.5:** The distribution of self assessment data collected from the sign-up form, phase two.

The following Figure 4.6, show the data collected by the post-playtesting questionnaire for the second phase of testing. It shows a clear increase in enjoyment of the game, increasing from an average of 5.6 to 6.8, with almost all testers scoring it between six and nine. Similarly, the experience of difficulty seems to show a decrease, though it is not as clear of a change as it was for enjoyment, with the average score being almost unchanged. Another decrease can be observed in the frustration levels, decreasing from an average of 4.4 to 4.1. This tracks with the hypothesis presented in the section for the first phase, as it seems the frustration levels are connected with feelings of difficulty. In all, these results show an improvement as compared to the previous phase, indicating progress on the experience people have of playing the game.



**Figure 4.6:** The distributions of self assessment data collected from the post-playtesting questionnaire, phase two.

## 4.2.2 Received Feedback

In summary, the recurring problems were as follows.

1. There was still some confusion about the game mechanics and how everything actually worked. A possible reason is that there is no introduction or cinematic act to present the game's actual concept and environment.
2. Some testers felt that the How-To-Play did not feel that clear, but the tutorial still explained the basic game mechanics well enough. This could be due to the How-To-Play explanation might feel too abstract when the player does not know anything about the game or has not tried playing it yet. Some testers felt that the How-To-Play worked well as an extra addition to the existing tutorial.
3. The flock interacted or reacted with the environment too slowly, making some testers confused whether anything was actually happening. This was due to the flock not having a strong enough attracting force towards the battery pillars.
4. There was confusion about the archetype selector and how it worked, even if the testers managed to complete level three. Many completed it by accident, without a clear understanding of what actually occurred in the game. A factor in this confusion could be that the bugs change colours, which happen to

match the battery pillar, making some testers assume it changed how they would react to said pillar. Another factor that might have exacerbated the issue is that this puzzle in large part can solve itself, making it completable without an understanding of the mechanics.

### 4.2.3 Evaluation

Even if problems still occurred, many problems were eliminated or reduced when comparing to the feedback from test phase one. In general, the testers had a better understanding of the game and could more easily progress through the levels, even if some confused still existed. This change is likely due to the added visual feedback and added tutorial.

From the feedback gathered from the test phase, some minor polishing implementations were added to address the issues. However, the problems that occurred during the testing were mostly discussed and assessed for further development due to time limitations. The following features were added for the final version of the game.

1. Original music to give a larger sense of immersion and to evoke emotions.
2. Better aesthetics by decorating the levels with more varied tilemaps and purely cosmetic art assets like moss and rebar.
3. Adjusted level design to make the puzzles more intuitive. An example of which being adding a hole in the third level so that the player does not attempt to enter an area meant only for the bugs. Additionally, the wires leading from the chargepad to the door were adjusted to better hint at the connection between the two elements.
4. Bug fixing of undesired phenomena observed during the playtesting. A UI prompt in the third level never disappeared and could previously be accumulated to fill the entire screen. Furthermore, several stage elements overlapped one another incorrectly. Some areas where softlocking was possible were also fixed.

## 4.3 Finished Product

The finished version of the game includes all of the must-haves from the MoSCoW prioritisation model and almost all of the should-haves. None of the could-haves were implemented, which aligns with the intention, since there was no time left for them to be implemented. The should-haves that were not fully implemented were the "Movable objects" and "A way to save progress", but as stated below, these features were in the works and very close to being in the final product.

The finalised game was uploaded on itch.io and is available to play at the following link: <https://lorantelo.itch.io/control-flow-bug>

The finished game became similar to the version that was tested for the second



test phase, with some finishing touches added. Certain features were in development, but never completely implemented and therefore cut from the final version of the game because of time constraints. The features that were worked on but never fully implemented are the following.

- **Movable Pillars** - Battery pillar that could be moved by the playable character, in turn able to more freely manipulate the swarm of bugs by changing the location of the attraction/repelling force. The feature was functionally fully implemented, but missing some auxiliary features, for example visual representation of the tracks, as well as a dedicated level.
- **Box of Boids** - A box where bugs are trapped inside that could be broken by the playable character, releasing more bugs for the player to charge up the chargepads with.
- **Progress System** - Having a system which saves progress as the player goes through the levels. When the player starts the game for the first time, only the first level would be available, while the following levels would be locked and only unlocked if the previous level is completed. The UI exist for this system with place holder code, but the actual functionality was never implemented.
- **Lighting** - Utilising the built in lighting system in *Unity* [31] to create regions of darkness and shadows was experimented with, but never used in a level. This was because there was an unfinished idea to have some levels that were dark, where the player would use the boids to light up the environment. Since no such level was ever designed or implemented, however, this idea went unused.
- **More Archetypes** - Several variations of the bugs within the game were created by adjusting the *alignment*, *separation*, and *cohesion* parameters of the boids algorithm. The yellow and green variations are explored in the third level, however, pink and blue variants, exploring other behaviours were also created. Art sprites for the archetype selector, displaying graphical representations of the unimplemented behaviours were also created, yet unused.



# 5

## Discussion

Finally, with the project in its final stages, the group was able to look back and reflect on the finished game and the process of creating it. While the project went smoothly, there were still challenges and time constraints, leading to some features having to be cut.

### 5.1 Finished Product

The finished product has flocking behaviour implemented as a core game mechanic, which was an important aspect of the project, as it determines a large part of the quality, and thus the experience, of the game.

#### 5.1.1 Challenges

Various challenges were encountered during the course of this project. Some are related to design and others to technical abilities.

As most of the members in the group conducting this project had little to no experience with video game development prior, the design and development process took longer than it otherwise would. Furthermore, this inexperience was compounded by the added complexity and unpredictability of the flocking behaviour. As this project was conducted over just under four months, there was also a limited amount of time to learn how to handle these problems. During implementation, the game itself had around three months of development.

The aforementioned unpredictability of the flocking behaviour made designing levels around it rather challenging, especially as it came to the different archetypes. In order to give players an actual reason to properly utilise the archetype selector, the level must be designed in such a way that different obstacles only can be overcome by certain archetypes. Making an obstacle that could be passed by one archetype and not another proved difficult which was not helped by the short development time left. This resulted in the third level having certain gameplay issues that were not fixed during this project.

Another problem was performance. The boids algorithm is being run on a large number of agents, requiring a large amount of calculations. These calculations needed to be done fast enough to have the game run in real time, while also having

headroom to allow for the added complexity of the surrounding mechanics. This problem is especially true for web builds of the game. In the end, this did not end up being a significant issue, due to the fact that the number of boids in the game was relatively small. However, in web builds of later versions of the game, this performance ceiling was noticeably being reached, showing that it is not insignificant.

### 5.1.2 Further Development

While the game developed in this project is considered a finished product, there is still room for improvement. Various desirable features, such as a save and load system and more different boid archetypes, were not implemented, primarily due to time constraints. Additionally, some features were implemented but never used in any of the levels, and thus do not appear in the final product when the game is played, as listed under section 4.3. As such, further development includes refining content, integrating functionality into the game, and developing new mechanics and features.

Overall, implementing the remaining should-haves and the could-haves from the MoSCoW, as previously defined in the scope section, should take the most priority in further development. Moreover, the implementation of more story-telling elements, such as cinematic scenes at the start of the game or between levels, would significantly add to the gameplay experience by helping convey the game's story and further immersing the player in the game's setting. This would likely also mean the player has an easier time understanding the concept of the game, since a cinematic could help make it clearer what the player is supposed to do. During testing, it was apparent that the game suffered from a lack of levels. Thus, creating more levels would allow the player to further explore the existing mechanics of the game and could introduce more complex ways in which flocking behaviour can be utilised in game design.

Lastly, the game is lacking in accessibility features. These were considered throughout the development process, but it was often considered too low of a priority to warrant dedicated development time. Various desirable accessibility features include settings for people with colour blindness, support for additional input devices, and keybind remapping.

## 5.2 Testing

To answer the research question more accurately, it was decided to conduct testing to see how players perceived the flocking behaviour and how engaging the game was.

Many players expressed curiosity about the game and its concept and wanted more levels to play through. There were desires to be able to learn more about the game world.

Throughout testing, there was some confusion from the players in understanding the fundamental aspects of the game and how everything worked together, which was mostly present during the first testing phase. This confusion was likely connected to lack of visual and auditory feedback, but also to how the flock moved and acted in the game, which could be difficult to parse.

Occasionally, testers also failed to grasp the electric nature of the bugs. The actual boids consist of circular monocoloured sprites, which may not by themselves convey an electric motif. This could be addressed by complementing them with animations of sparks, reinforcing the electric theme of the boids.

As mentioned in the test result for the second test phase, the flock interacted with the environment too slowly, making the players not always understand what it did due to a delay in the reaction of the flock to the player's actions, which was only amplified by the seemingly random movements of the flock. By increasing how fast the flock interacts and reacts with its environment could decrease this sense of confusion, and in turn the frustration, while also potentially making the impact of the flock clearer.

### 5.2.1 Quality of Test Results

During the first testing phase, some errors occurred regarding the post-playtesting questionnaire that the testers were supposed to answer. The questionnaire was not always answered immediately after each session, and some answers were collected days after the session occurred. This means those answers were less reliable compared to the answers that were collected immediately after a session, impacting the accuracy of these results. The problem was largely caused by confusion about whether the questionnaire was optional. In trying to address this problem, it was unclear who had answered already and who had not, meaning some people may have answered multiple times. It is impossible to tell if this is the case, as the questionnaire was designed to be anonymous. These errors were reviewed and discussed, and it was concluded that during the second test phase, efforts would be made to prevent their recurrence. This was done by changing the settings of the form used for the questionnaire, enabling a setting that forces users to be logged in to an account in order to allow each user to respond to the form only once.

Due to delays in producing the test versions for both phases of testing, the test versions lacked basic refinements that could have improved the experiences of the playtesters. This was most apparent during the second test phase where complications occurred during the actual playtesting sessions, which had to be handled in real-time.

The questions about how familiar and how much experience the tester have was included in the sign-up form for the playtesting, while the questions about enjoyment, difficulty and frustrations were included in the post-playtesting questionnaire. As these were two separate questionnaires, and the latter was completely anonymous,

identifying relationships between the answers to questions in the sign-up form and the post-playtesting questionnaire was impossible. For example, identifying whether or not players with less prior puzzle-game experience had a more frustrating experience could not be done. By having these questions connected to a singular questionnaire, more conclusions could have been drawn.

### 5.2.2 Potential Improvements

In future iterations of projects akin to this one, there are several possible improvements that could be made to the testing process. For one, notes of the thoughts and opinions of the playtesters were taken quite informally, resulting in loose collections of bullet points rather than structured data. If a more clearly defined system had been used, the objectivity of the testing results would have been increased. One such improvement is to record the sessions and transcribe them afterwards, recording the exact words of the testers.

Secondly, more phases of testing would have been beneficial. Although, for this project, this would have been difficult to do given the short time frame. It is also possible that there would have been diminishing returns on the feedback from additional sessions. However, it is still worth considering in future projects if possible.

Thirdly, a small pilot testing session before the first phase could have identified glaring issues with the testing procedures and the game before the real testing begins. This would allow testers to provide higher quality feedback, especially for the first phase, due to better testing procedures and less distraction from simple and easily fixable issues with the game. The pilot testing session would interview only a few people and the data would not be included in the actual results, as the goal is to find obvious and common problems rather than more subtle ones. Alternatively, the session could be conducted with the group members of the project. This reduces the complexity of conducting the tests, but also reduces the quality of the results, as the group members developing the game may be biased by the fact that they know how the game works or is supposed to work in a lot of detail.

Fourthly, it could potentially have been advantageous to have two group members who conducted all the testing sessions. The advantage of this is that it makes the testing sessions more consistent, as it avoids different people handling the sessions slightly differently, for example by giving different amounts of help to the testers. However, an argument against this change is that it increases the risk of bias from the conductors being introduced, which is otherwise "averaged out" when multiple people conduct the tests.

Fifthly, a larger sample size, meaning a larger number of playtesters within each testing phase, would have been beneficial. The primary benefit of such a change would be that a larger sample size would provide more statistically significant results. Additionally, a larger group of playtesters would naturally include more variation in play styles, skill levels, and preferences, which would provide a more diverse

set of perspectives on the game.

Finally, to avoid problems with scheduling, such as the ones experienced during this project, a booking system for testers could have been beneficial.

### 5.3 Research Question

This project has aimed to answer the following research question: *What engagement implications can be derived when including flocking behaviour as a core mechanic to a video game.*

As previously mentioned, the testers experienced overall satisfaction with the game. However, there were also several instances of confusion regarding the puzzles that made use of flocking behaviour in more complex ways. This likely stemmed from a lack of familiarity with flocking behaviour. This could be alleviated by creating intermediate levels between new mechanics to help introduce and ease players into each concept, rather than rapidly introducing several new mechanics in a few levels, as currently is the case.

Flocking behaviour as a core game mechanic was found capable of providing both positive and negative engagement implications for gameplay. The experience of flocking behaviour was for many players a new concept, which in turn evoked feelings of curiosity and a desire to explore the game further. Some also found the act of interacting with the flock enjoyable in and of itself. However, there were also some negative experiences with confusion, though it is uncertain to what extent this confusion occurred due to the game's design rather than the flocking behaviour itself. Notably, some players viewed the inherent randomness of flocking behaviour as an obstacle to the underlying puzzle gameplay. This confusion and frustration could potentially stem from a lack of understanding of the actual gameplay, and not necessarily the flocking behaviour in itself.

### 5.4 Social & Ethical Aspects

The game developed in this project was intended to include settings for accessibility, as people with disabilities are often overlooked by game developers. Examples of accessibility in game design include filters for colour blindness, text descriptions of sounds, and screen reader support [27]. In the end, dedicated accessibility features were not possible to prioritise highly enough to include in the final version of the game.

As brought up in subsection 3.4.5, the design decisions made to account for colour blindness were far from all-encompassing. To improve the existing designs, values could be used to differentiate the similar objects, instead of using just shapes and colours. For example, both the active battery pillars in Figure 3.9 can be considered too similar in appearance, even with colours. Additionally, the colours could have

been better planned from the start to, for example, avoid using red and green, the most common colours that colour blindness affects.

Video games are often developed for people who are already familiar with how video games work. This lack of onboarding is a known problem [62], and can, albeit unintentionally, make access to video games feel restricted from people who are new to them. Due to this, an attempt was made in this project not to assume prior knowledge from the player. An example of a common assumption is to assume the player understands the objective of a level, e.g. not explaining that the player is supposed to reach a door to progress in the game. This was especially important for this project, as the goal of the project encompasses investigating how flocking behaviour can be included in video games in a way that is not confusing to the player. In order to make sure the game was developed to accomplish this goal, an attempt was made to invite people with varied experiences in playing video games to the testing sessions. Using their feedback, biases and assumptions about the player could be addressed.

During playtesting, it is important to get a varied perspective. This project was conducted at a school attended by primarily white men, which could have potentially biased the results of the playtesting in some way. The simple solution to this problem was to attempt to invite a diverse set of people (including people of different gender, ethnicities, and with different experiences and levels of comfort with video games) for the playtesting. In the end, it was decided that it was inappropriate to collect data about the playtesters ethnicities, gender identities, etc. as these are sensitive issues, which could cause discomfort to ask about. Instead, a more informal approach was used, where the group members made an attempt to invite a varied set of playtesters. In the end, this was only partially successful, as most testers were young, white, and male students at the same university as the group members. In future work on a project similar to this, it would be desirable to make a greater effort in dealing with this issue.

### 5.4.1 Use of Generative AI

Usage of generative AI during development was limited to assisting in writing code and minor suggestions for improvements on phrasing in this report. Any AI-generated code being used in the game was checked by the programmers and thoroughly tested to ensure that it fulfilled its intended function. Whenever AI was used, it was mainly for the purpose of supporting the group's programmers less experienced in *Unity* [31]. No art assets or parts of the music were created using any generative AI at any point, and all of the content of this report (aside from the aforementioned minor suggestions on phrasing) was written by the authors themselves. The generative AI models used for these purposes are *GPT-5.3 Instant* [63], *GPT-5.3 Mini* [64], *GPT-5.5 Instant* [65], *GPT-5.5 Mini* [66], and *Gemini 3.1 Pro* [67].



## 5.5 Conclusion

After the end of the project, it can be concluded that a game developed with flocking behaviour as a core mechanic has the possibility of providing engaging gameplay to players. Throughout this project, this group has learned much about the difficulties of exploring this idea, as it is often hard to differentiate between problems stemming from game design and problems stemming from the flocking behaviour. In all, improvements can be made to the methods presented in this report in order to gain deeper insights in future iterations of this kind of project. From the results presented previously, the use of flocking behaviour typically evokes contrasting opinions, showing that such a game can provide both positive and negative engagement implications. When implemented well, flocking behaviour seems worthwhile for game developers to explore.



# Bibliography

- [1] W. Hamilton, “Geometry for the selfish herd,” *Journal of Theoretical Biology*, vol. 31, no. 2, pp. 295–311, 1971, ISSN: 0022-5193. DOI: [https://doi.org/10.1016/0022-5193\(71\)90189-5](https://doi.org/10.1016/0022-5193(71)90189-5). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0022519371901895>.
- [2] C. W. Reynolds, “Flocks, herds, and schools: A distributed behavioral model,” *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987. [Online]. Available: <https://www.red3d.com/cwr/papers/1987/boids.html>.
- [3] D. Clegg and R. Barker, *Case Method Fast-Track: A Rad Approach*. USA: Addison-Wesley Longman Publishing Co., Inc., 1994, ISBN: 020162432X.
- [4] T. O’Connor, “Emergent properties,” *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., 2021, Winter 2021 Edition. [Online]. Available: <https://plato.stanford.edu/entries/properties-emergent/>.
- [5] A. Zucconi. “Conway’s game of life. ”[Online]. Available: <https://www.alanzucconi.com/2020/10/13/conways-game-of-life/>.
- [6] M. Papadopoulou, H. Hildenbrandt, D. W. Sankley, S. J. Portugal, and C. K. Himelrijk, “Emergence of splits and collective turns in pigeon flocks under predation,” *Royal Society open science*, 2022.
- [7] T. Vicsek and A. Zafeiris, “Collective motion,” *Physics Reports*, vol. 517, no. 3–4, pp. 71–140, 2012. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0370157312000968>.
- [8] E. Hastings and J. Mesit, “Optimization of large-scale, real-time simulations by spatial hashing,” Jan. 2005.
- [9] R. Finkel and J. Bentley, “Quad trees: A data structure for retrieval on composite keys,” *Acta Inf.*, vol. 4, pp. 1–9, Mar. 1974. DOI: 10.1007/BF00288933.
- [10] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, ISSN: 0001-0782. DOI: 10.1145/361002.361007. [Online]. Available: <https://doi.org/10.1145/361002.361007>.
- [11] H. Samet, “The quadtree and related hierarchical data structures,” *ACM Comput. Surv.*, vol. 16, no. 2, pp. 187–260, Jun. 1984, ISSN: 0360-0300. DOI: 10.1145/356924.356930. [Online]. Available: <https://doi.org/10.1145/356924.356930>.
- [12] R. Fabian, “Data-oriented design,” *framework*, vol. 21, pp. 1–7, 2018.
- [13] R. Hunicke, M. G. Leblanc, and R. Zubek, “Mda : A formal approach to game design and game research,” 2004. [Online]. Available: <https://api.semanticscholar.org/CorpusID:6056852>.

- [14] S. Björk and J. Holopainen, *Patterns in Game Design*. Hingham, MA: Charles River Media, 2005, ISBN: 978-1584503545. [Online]. Available: <https://www.bokus.com/bok/9781584503545/patterns-in-game-design/>.
- [15] M. Davies. “A good puzzle game is hard to build,” Accessed: Feb. 10, 2026. [Online]. Available: <https://www.rockpapershotgun.com/how-to-make-a-puzzle-game>.
- [16] “Baba is you developer arvi teikari talks indie innovation, influences and puzzle design,” 2019. [Online]. Available: <https://www.indiegamewebsite.com/2019/07/19/baba-is-you-developer-arvi-teikari-talks-indie-innovation-influences-and-puzzle-design/>.
- [17] IxDF, “What is user experience design?” *IxDF - Interaction Design Foundation*, Jun. 2016. [Online]. Available: <https://www.interaction-design.org/literature/topics/ux-design>.
- [18] R. Bernhaupt, Ed., *Game User Experience Evaluation* (Human-Computer Interaction Series). Cham: Springer, 2015, ISBN: 978-3-319-15984-3. DOI: 10.1007/978-3-319-15985-0. [Online]. Available: <https://doi.org/10.1007/978-3-319-15985-0>.
- [19] S. C. Llanos and K. Jørgensen, “Do players prefer integrated user interfaces? a qualitative study of game ui design issues,” *Proceedings of the DiGRA 2011 Conference: Think Design Play*, 2011. DOI: <https://doi.org/10.26503/dl.v2011i1.514>.
- [20] IxDF, “What is user interface (ui) design?” *IxDF - Interaction Design Foundation*, Jun. 2016. [Online]. Available: <https://www.interaction-design.org/literature/topics/ui-design>.
- [21] A. Galloway, *Gaming: Essays On Algorithmic Culture* (Electronic Mediations). University of Minnesota Press, 2006, ISBN: 9781452908687. [Online]. Available: <https://books.google.se/books?id=z8LibpVNIZ4C>.
- [22] L. Pfister and S. Ghellal, “Exploring the influence of non-diegetic and diegetic elements on the immersion of 2d games,” in *Proceedings of the 30th Australian Conference on Computer-Human Interaction*, ser. OzCHI '18, Melbourne, Australia: Association for Computing Machinery, 2018, pp. 490–494, ISBN: 9781450361880. DOI: 10.1145/3292147.3292190. [Online]. Available: <https://doi.org/10.1145/3292147.3292190>.
- [23] S. Jiang, H. Lee, and B. Rooney, “Designing usable information display in immersive virtual environments: A comparative analysis of diegetic and non-diegetic designs to identify design factors,” in *Proceedings of the 20th International Conference on Virtual Reality Continuum and Its Applications in Industry*, ser. VRCAI '25, Association for Computing Machinery, 2026, ISBN: 9798400723629. DOI: 10.1145/3779232.3779263. [Online]. Available: <https://doi.org/10.1145/3779232.3779263>.
- [24] Valve, *Portal 2*, 2011. [Online]. Available: <https://www.thinkwithportals.com>.
- [25] IGN, *Portal 2 walkthrough: Chapter 2 the cold boot*, Youtube, Accessed: 2026-04-13, May 2011. [Online]. Available: <https://www.youtube.com/watch?v=Qv4sRFzkjj0>.

- 
- [26] M. Heron, "Cultural integration and the accessibility of gaming," *The Computer Games Journal*, vol. 5, Nov. 2016. DOI: 10.1007/s40869-016-0028-x.
  - [27] Game Accessibility Guidelines, *Full list – game accessibility guidelines*, n.d. [Online]. Available: <https://gameaccessibilityguidelines.com/full-list/>.
  - [28] P. Cairns, C. Power, M. Barlet, and G. Haynes, "Future design of accessibility in games: A design vocabulary," *International Journal of Human-Computer Studies*, vol. 131, pp. 64–71, 2019, 50 years of the International Journal of Human-Computer Studies. Reflections on the past, present and future of human-centred technologies, ISSN: 1071-5819. DOI: <https://doi.org/10.1016/j.ijhcs.2019.06.010>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1071581919300801>.
  - [29] H. Studios, *It takes two*, 2021. [Online]. Available: <https://www.hazelight.se/games/it-takes-two/>.
  - [30] A. Aloumi. "The effect of colour contrast combinations on the simplicity and complexity of design." [Online]. Available: [https://eprints.bournemouth.ac.uk/20966/1/Aloumi%2CAhmad%20E.\\_Ph.D.\\_2013.pdf](https://eprints.bournemouth.ac.uk/20966/1/Aloumi%2CAhmad%20E._Ph.D._2013.pdf).
  - [31] Unity Technologies, *Unity*, version 6000.0.58f2, Computer software, 2025. [Online]. Available: <https://unity.com>.
  - [32] *Unity dots*, 2024. [Online]. Available: <https://unity.com/dots>.
  - [33] S. Lague, 2019. [Online]. Available: <https://github.com/SebLague/Boids>.
  - [34] Git, *Git*, version 2.53.0, Computer software, 2026. [Online]. Available: <https://git-scm.com/>.
  - [35] I. Github, *Github*, version 3.19, Computer software, 2025. [Online]. Available: <https://github.com/>.
  - [36] Trello, *Trello*, 2025. [Online]. Available: <https://trello.com/>.
  - [37] D. Inc., *Discord*, 2026. [Online]. Available: <https://discord.com>.
  - [38] Krita, *Krita*, version 5.2.15, Computer software, 2026. [Online]. Available: <https://krita.org/sv/>.
  - [39] Affinity, *Affinity*, version 3.0.3, Computer software, 2026. [Online]. Available: <https://www.affinity.studio/>.
  - [40] Muse Group, *Muscore*, version 4, 2025. [Online]. Available: <https://www.musehub.com/app/muscore-studio>.
  - [41] Steinberg, *Cubase*, version 15, 2025. [Online]. Available: <https://www.steinberg.net/cubase/>.
  - [42] Audiokinetic, *Wwise*, 2025. [Online]. Available: <https://www.audiokinetic.com/en/wwise/>.
  - [43] C. Fox, *Agile methods in software engineering*, 2023. [Online]. Available: <https://doi.org/10.1036/1097-8542.014730>.
  - [44] B. Rebouche, "Knocking on death's door: Designing a new bunker for 'horizon forbidden west'," Sweden Game Conference, 2022. [Online]. Available: <https://www.youtube.com/watch?v=KxcXUYx3eLQ>.
  - [45] D. McDonagh and I. Storer, "Mood boards as a design catalyst and resource: Researching an under-researched area," English (US), *The Design Journal*, vol. 7, no. 3, pp. 16–31, 2005, ISSN: 1460-6925. DOI: 10.2752/146069204789338424.

- [46] Supergiant Games, *Transistor*, 2014. [Online]. Available: <https://store.steampowered.com/app/237930/Transistor/>.
- [47] Square Enix and PlatinumGames Inc., *Nier: Automata*, 2017. [Online]. Available: <https://store.steampowered.com/app/524220/NieRAutomata/>.
- [48] Videocult, *Rain world*, 2017. [Online]. Available: [https://store.steampowered.com/app/312520/Rain\\_World/](https://store.steampowered.com/app/312520/Rain_World/).
- [49] 164, 天ノ弱 / 164 feat.gumi公式), Youtube, Nov. 2014. [Online]. Available: <https://www.youtube.com/watch?v=2-zPY0vrpjQ>.
- [50] 164 and Sega and Colorful Palette Inc and Crypton Future Media, INC, 天ノ弱 / leo/need.
- [51] Eve and Wit Studio, 僕らまだアンダーグラウンド, Youtube, Jan. 2019. [Online]. Available: <https://www.youtube.com/watch?v=nBte0-bU78Y>.
- [52] Eve and niL, 遊生夢死 - eve mv, Youtube, Sep. 2021. [Online]. Available: [https://www.youtube.com/watch?v=roh\\_p2l8DHo](https://www.youtube.com/watch?v=roh_p2l8DHo).
- [53] Walt Disney Pictures and Pixar Animation Studios and FortyFour Studios, *Wall-e*, Jun. 2008. [Online]. Available: <https://www.pixar.com/wall-e>.
- [54] Double Fine Productions, *Broken age*, 2014. [Online]. Available: [https://store.steampowered.com/app/232790/Broken\\_Age/](https://store.steampowered.com/app/232790/Broken_Age/).
- [55] Blizzard Entertainment, *Overwatch: The last bastion*, Youtube, Aug. 2016. [Online]. Available: <https://www.youtube.com/watch?v=to8yh83jlXg>.
- [56] Gearbox Software, *Borderlands*, 2009. [Online]. Available: [https://store.steampowered.com/app/729040/Borderlands\\_Game\\_of\\_the\\_Year\\_Enhanced/?curator\\_clanid=2428135](https://store.steampowered.com/app/729040/Borderlands_Game_of_the_Year_Enhanced/?curator_clanid=2428135).
- [57] Lucasfilm Ltd. and Bad Robot and British Film Commission, *Star wars: The rise of skywalker*, Dec. 2019. [Online]. Available: <https://www.starwars.com/films/star-wars-episode-ix-the-rise-of-skywalker>.
- [58] Bloom Wild. “What is the meaning of the lily flower?” Accessed 2026-05-13, Bloom Wild. [Online]. Available: <https://www.bloomandwild.com/the-blog/lily-flower-meaning>.
- [59] C. Larkin. “Dirtmouth.” Track from *Hollow Knight (Original Soundtrack)*, Accessed: May 13, 2026. [Online]. Available: <https://christopherlarkin.bandcamp.com/track/dirtmouth>.
- [60] Team Cherry. “Hollow knight,” Accessed: May 13, 2026. [Online]. Available: <https://www.hollowknight.com/>.
- [61] *Regulation (eu) 2016/679 of the european parliament and of the council, General data protection regulation (gdpr)*, <https://gdpr-info.eu/>, 2016.
- [62] J. Gambles, “Player onboarding: An exploration of tutorial systems in modern pc video games,” M.S. thesis, Victoria University of Wellington, 2025.
- [63] OpenAI, *Chatgpt*, version GPT-5.3 Instant, 2026. [Online]. Available: <https://chatgpt.com/>.
- [64] OpenAI, *Chatgpt*, version GPT-5.3 Mini, 2026. [Online]. Available: <https://chatgpt.com/>.
- [65] OpenAI, *Chatgpt*, version GPT-5.5 Instant, 2026. [Online]. Available: <https://chatgpt.com/>.
- [66] OpenAI, *Chatgpt*, version GPT-5.5 Mini, 2026. [Online]. Available: <https://chatgpt.com/>.

- [67] Google, *Gemini*, version 3.1 Pro, 2026. [Online]. Available: <https://gemini.google.com/app>.





# A

## Appendix A

The following appendix includes the templates for the two test phases. It should be noted that the templates original language is Swedish, but have been translated to English for the purpose of this report.

### A.1 First Testing Phase Template

#### A.1.1 Introduction

At the start of the interview, the test moderators introduced the game and the play-testing session process according to the following instructions:

- Introduce your names and who is the interviewer and secretary and what each of you will do during the session.
- Introduce what you do: “A game based on real-time flocking behaviour.”
- Present the game
- Walk-through the interview process
- Ask if the person has understood the information and if they have any questions about the process.

#### A.1.2 Tasks

The below tasks were given to each tester:

1. Go to the How To Play menu
2. Start level 1, pause the game, and return to the main menu
3. Start level 1 again and play through it
4. Start the next level, pause the game, and change any sound setting
5. Unpause the game and play through the level
6. Go back to the main menu and go to the credits page

#### A.1.3 Interview Questions

After completing the aforementioned tasks, the tester was asked the following questions:

1. Did any task feel harder to complete, or felt unintuitive?
2. Did the puzzle feel too difficult or too easy? Did you understand what needed to be done to complete the level? If not, what could have made it easier?

3. How did it feel to navigate the different menus? Were the purposes of the different buttons intuitive?
4. Was it easy to see which objects were interactable and which were not?
5. Did you feel immersed while playing the game? That is, did you feel absorbed by the game world and were curious to explore it more?
6. What did you think of the sound effects? Did they fit? Did it feel like any sound effects were missing?
7. What did you think of the aesthetics of the game? Did the style feel unified? Was it easy to differentiate between different visual elements?
8. Did you feel some kind of feeling from the game and its environment? e.g. anger, sadness, happiness.
9. Do you have any other thoughts about the game you want to share?

## A.2 Second Test Phase Template

### A.2.1 Introduction

At the start of the interview, the test moderators introduced the game and the play-testing session process according to the following instructions:

- Introduce your names and who is the interviewer and secretary and what each of you will do during the session.
- Introduce what you do: “A game based on real-time flocking behaviour.”
- Present the game
- Walk-through the interview process
- Ask if the person has understood the information and if they have any questions about the process.

### A.2.2 Tasks

The below tasks were given to each tester:

1. Browse through the menus
2. Play through the levels

### A.2.3 Interview Questions

After completing the aforementioned tasks, the tester was asked the following questions:

1. Were the puzzles easy or difficult? Did you understand what you had to do to complete the levels? If not, what do you think would have helped you understand better?
2. Was it easy to see which objects were interactive and which weren't? For example, that you could shoot the battery pillars.
3. Did you feel immersed while playing the game? That is, did you feel absorbed by the game world and were curious to explore it more?
4. Did you feel that flocking behaviour was central to the game? If not, what would have made it feel more central?

5. Did you find the game easy to understand? Were the tutorial and How-To-Play menu useful? Could anything have been better?
6. What did you think of the aesthetics of the game? Did the style feel unified? Was it easy to differentiate between different visual elements?
7. Did you feel some kind of feeling from the game and its environment? e.g. anger, sadness, happiness.
8. Do you have any other thoughts about the game you want to share?