KTH Royal Institute of Technology

Game Design Document

DH2650 Computer Game Design

May, 2022

Group 9

Adam Červeň
cerven@kth.se

David Ryan
dryan@kth.se

Jonas Lundin
jlundin9@kth.se

Jiatong Zhong
jiatongz@kth.se

Table of Contents

This page intentionally left blank.

1. Game Overview

1.1. Game Concept

Polarfish is a 2D cooperative puzzle platformer about two magnetic fish - Maggie, who is a female deep sea fish, and Polo, who is a male clownfish. They are rolling around in fish bowls trying to reach the ocean. The pair must use their ability to switch polarity at will to navigate complex and often hazardous obstacles.

The start of the game is set in a futuristic laboratory of an evil scientist who captured Maggie and Polo for one of his experiments - he wanted to transform them into two Polarfish with superpowers. His experiment was successful and Maggie and Polo gained the abilities to be magnetically charged and to freeze in mid-air. These abilities are at the core of the gameplay of Polarfish, as they need to be used cooperatively by both players to solve various puzzles and advance through the story.

The world of Polarfish is futuristic and cyberpunk styled, depicted using pixel art, which contrasts with the two cute and friendly looking fish. They feel out of place, which works well with the story of them not belonging and trying to return to the ocean. There is a lot of emphasis on the characters themselves, and the animations of the fish, the water and the bowl, as it was important for such a complex player model to look and feel right, and match with the player input, as the player models will be the visuals that the players will be seeing for the longest time (as the environments change).

A few of the key characteristics of the game concept are collaboration, communication and facing a non-frustrating and rewarding challenge together. The game has quite a forgiving set-up, with players dying often, but respawning closeby, which motivates trying out various creative solutions and not being worried about losing a lot of progress, which is perfect for a casual, family-friendly co-op game.

1.2. Genre

Polarfish is a local co-op multiplayer game. It is a challenging adventure platformer, which can only be played multiplayer (currently 2-player), and presents logical puzzles that require user collaboration, increasing skill (as the players get better at the game), and creativity. The game's puzzles are largely based on the unique game mechanics of Polarfish - magnetism and freezing. The game is highly collaborative, and can be characterized as family- and couple-friendly.

1.3. Target audience introduction

The nature of Polarfish opens it up for a broad range of players, with the gameplay satisfying both casual and skilled gamers. It is a family-friendly game, with the target age group being 12 and above. The aim of Polarfish is to appeal to players that seek a rewarding challenge and achievement, a social, bonding experience with another person and the feeling of gaining skills through playing (improving).

Action "Boom!"	Social "Let's Play Together"	Mastery "Let Me Think"	Achievement "I Want More"	Immersion "Once Upon a Time"	Creativity "What If?"
Destruction Guns. Explosives. Chaos. Mayhem.	Competition Duels. Matches. High on Ranking.	Challenge Practice. High Difficulty. Challenges.	Completion Get All Collectibles. Complete All Missions.	Fantasy Being someone else, somewhere else.	Design Expression. Customization.
Excitement Fast-Paced. Action. Surprises. Thrills.	Community Being on Team. Chatting. Interacting.	Strategy Thinking Ahead. Making Decisions.	Power Powerful Character. Powerful Equipment.	Story Elaborate plots. Interesting characters.	Discovery Explore. Tinker. Experiment.

Figure 1.1 – Main gaming motivations of our target audience[1]

1.4. Game Flow Summary

The game flow of the full game starts slowly - with a tutorial introducing the core game mechanics and how they can be used for collaboration. The first few levels help the players understand how they can help each other using magnetism and freezing, and the puzzles in these levels are quite simple, yet rewarding. As the players get better and better throughout the course of the game, the puzzles get more challenging, require more cooperation and more creative and complex ways of using the core game mechanics (polarity and freezing) are required. This keeps the game fresh and interesting over its entire course.

Further obstacles and enemies are added at later stages of the game, in new environments (which also keeps the game visually fresh and appealing). Once the players have mastered control over their fish and abilities, new, unexpected ways of interaction are introduced (e.g. new abilities, dark levels, new enemies and ways to

[1] Gaming motivations defined by Quantic Foundry (https://quanticfoundry.com)

die, new puzzles, styles of level - such as the "final boss" of running away from the scientist in the final scrolling level).

The flow of the demo is slightly different. Levels of the demo are not necessarily showing the first few levels, rather, they show a slice of the game's increasingly difficult puzzles. In this sense, the demo is "skipping ahead" in terms of difficulty and flow, but serves as a good indicator of what can be expected at later stages of the game (e.g. level 4 of the demo would probably be closer to level 20/30 in the full game).

The story develops along with the player's skill. At the start, the players have a very limited idea of what actually happened to the fish (as the fish also do not remember very much about what happened, just that they have new abilities and that something has changed and they need to escape). The story elements are uncovered more in depth as the players progress through the world of Polarfish. Cut scenes come into play at critical points, when transitioning between new areas and when significant plot points are reached - showing the players what happened to Maggie and Polo and why they are going where they are currently going. This way, players are on par with Maggie and Polo, as their knowledge of the story increases at the same rate as their skill.

1.5. Look and Feel

The world of Polarfish has a unique pixel-art artstyle with a heavy focus on player characters. The player models are quite complex, consisting of several independent layers that react to player input and the environment in slightly different ways. Each player model consists of three main parts - the fish, the water and the fishbowl. The sprites of the fish react directly to the inputs of the player - as if the player controls their fish directly. The water sprites are disconnected from this - they react to the forces applied to the fishbowl, meaning that the water is pushed in the opposite direction of the fish (and player input). Lastly, the fishbowl rotates based on how it rolls around the environment. Together, these three layers create a player model that "feels right" and simulates what a real fishbowl would look like in action. This feeling is extremely important as it helps the users understand what is happening and have better control over their fish.

Furthermore, the environments are contrasting to the player models in look, as contrary to the cute and friendly-looking fish, it is quite cold, futuristic, metallic and cyberpunk-esque. This contrast plays a role in selling the idea of Maggie and Polo not belonging and trying to escape back to the ocean. The world of Polarfish develops to keep a fresh and interesting visual look throughout the game, by

visiting different areas (laboratory, futuristic cyberpunk city, fishing village and beach). The aim is to keep the players engaged both through the gameplay and visuals, and to aid in developing the story through the visited environments (e.g. origins of powers are examined in the lab).

1.6. Project Scope

The project revolves around the demo of the full game, as it is limited in timespan and manpower, so we have decided to aim to create 5 short but challenging levels. All levels have a similar style to them, as if they occur back to back (which, in the full game, would be different). We only include a small number of mechanics, mostly because adding too many mechanics in these short levels would be overwhelming for the player. We also produced a simple website with descriptions and demonstrations of the game as well as a trailer video introducing Polarfish.

The scope of the full game would entail substantially more work. The game would take place in three different environments which would have separate visual identities, specific game mechanics, obstacles and game elements, as well as increasing demands on player skill. Levels would differ in length and complexity, and various story elements would require cutscenes, storyboards, voice acting and music creation. Furthermore, the game scope would leave it open for subsequent sequels, as there could be a lot of possibilities in developing new game mechanics and abilities, as well as creating new puzzles and levels (and possibly involving the gamer community through a level editor).

2. Gameplay and Mechanics

2.1. Gameplay

The gameplay of Polarfish is challenging, but forgiving. Player avatars are expected to "die" several times at each obstacle, but since they respawn closeby immediately after perishing, it shouldn't be too frustrating for the players.
The game is highly physics based so precise maneuvering is critical. The slightest mistake could send a player flying in the wrong direction.
The two players will need to coordinate their actions in order to pass most obstacles, and some obstacles may require some out-of-the-box thinking.

2.2. Mechanics

2.2.1. Physics

2.2.1.1. Physics Engine

The mechanics of Polarfish are heavily dependent on the 2D physics engine of Unity. The engine handles gravity and collisions, as well as forces applied to game objects by game mechanics. This makes the mechanics of Polarfish interesting, but also rather unpredictable. Unity calculates physics in what are called “physics cycles”, these cycles may run multiple times per frame.[2]

2.2.1.2. Magnetism

Magnetic objects (MO for short) fall into two categories, static (SMO) and dynamic (DMO). Both exert magnetic forces but only DMOs can be moved by these forces. Each MO has a charge, either positive or negative, and a maximum radius beyond which exerted magnetic forces are suppressed.

Some MOs are also “directed”, meaning the full magnetic force is mostly applied along a particular direction. The force affecting an object is multiplied by a factor between zero and 1, depending on the dot-product between the magnetic direction and the vector between the two magnetic objects, normalized.

In each physics cycle, magnetic forces from all MOs are applied to each DMOs according to this magnetic model:

$$F = magPower * chargeA * chargeB / distance$$

magPower is a global modifier of magnetic strength.
chargeA and chargeB are the charges of the two MOs.

[2] Unity Execution order: https://docs.unity3d.com/Manual/ExecutionOrder.html

The force is applied to the DMO in a direction away from the other MO, a direction which will be inverted if one of the charges is negative and the other positive.

Note that the force is modified by the distance, and not the distance squared, as per usual. We made this change because it is difficult for the player to account for how a force with a squared dependency changes as the two objects draw closer to each other. Linear dependency is difficult enough, so for a while we even considered a constant value, no relation to distance at all, but that was a bit annoying to play with.

The implementation of this mechanic was mostly drawn from Stuart Spence's video on the topic[3], with a few minor modifications.
You can view our source code in the following documents: MangeticManager.cs, MagneticObject.cs, DynamicMagneticObject.cs.

The only significant changes we made are:

1. A limitation that limits the maximum distance at which magnets exert forces (default 5 units of distance).
2. A few changes that allow magnets to be enabled and disabled.
3. A number of simple member functions that alter the state of the magnet, such as changing, locking, or disabling the magnet's charge.

Figure 2.1. A negatively charged DMO (the fish, Maggie) levitates above a negatively charged MO (the magnetic booster). The magenta line indicates the direction of the magnetic force acting upon Maggie.

2.2.2. Movement

2.2.2.1. Rolling and Jumping

Movement of the fish bowls works by acceleration, holding "right" on the input device adds a rightward force (acceleration) which results in a rolling motion when combined with the downward force of gravity. Holding "left" on the input device similarly adds a leftward force.
Only left and right are valid directions of this kind of movement, and these inputs are accepted even when airborne. These forces have no effect while the fish is frozen.

[3] Spence's Magnetism Simulation: https://www.youtube.com/watch?v=eBVim8kEhK8

Jumping is also an option, by pressing “jump” on the input device while under certain conditions, an instant force is applied to the fish bowl. The jumping force is applied upwards, regardless of the surface normal. The fish cannot jump while frozen.

Moving by means of forces, and weak ones at that, gives a certain feeling of sluggishness to the player characters. This feeling is intended, as it should give the players the feeling that they are fish stuck in a clumsy ball. This feeling is reinforced by a pitiful jump height (compared to many other games). After all, the fish need to lift the entire glass bowl when jumping! (Don’t think too hard about the real-world physics involved with any of these actions)
More importantly, by heavily restricting mobility, a greater sense of importance is placed on the magnetism mechanic, which becomes the most effective way of traversing great distances quickly.

The jumping mechanic is somewhat prone to glitches and bugs. The way it is implemented is to look for collisions between the fish bowls and other surfaces each physics cycle. If the fish bowl touches a surface with a normal that is within 45 degrees of an upward vector, then a boolean is flipped, and jumping inputs will be read again.

There are two exceptions to this rule. First, jump inputs within half a second of performing a successful jump will be ignored. This limit is enforced to prevent a glitch where jumps were registered twice in the same frame but it also makes sense realistically. Second, jump inputs within 0.2 seconds before landing will be saved as a preemptive jump, and will be acted upon once the fish bowl hits a valid surface. This preemptive jump window was added in order to capture the intent of the player (jumping as soon as the fish bowl lands) instead of the raw input. Performing a series of jumps feels better this way.

You can view our source code in the following documents: move.cs.

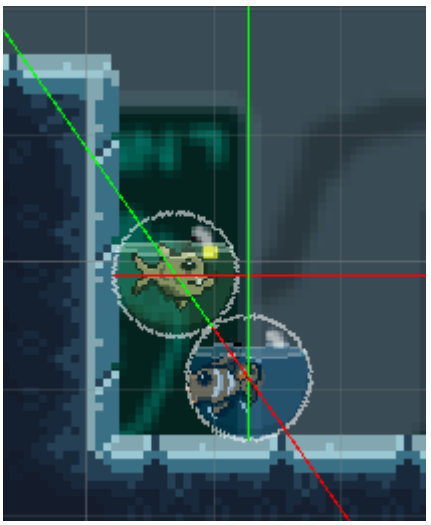


Figure 2.2. The lines indicate the normals of surfaces the fish bowls are in contact with. Green lines indicate valid surfaces, red lines indicate invalid surfaces. In this situation, Polo (bottom) could jump because he stands on solid ground, and Maggie can also jump because she is standing on Polo.

2.2.3. Action

2.2.3.1. Polarity Control

The player avatars are DMOs (Dynamic Magnetic Objects) with a default charge of zero. Both players can control their charge by holding down “positive” or “negative” on the input device, which increases or decreases the fish’s magnetic charge, respectively. This allows the players to determine if the fish are pulled toward each other, repelled away from each other, or undisturbed. This is a core mechanic of the game that is meant to be used frequently to navigate the levels.

Changing your polarity in order to leap or climb, for various reasons, are the primary means of effective traversal in Polarfish, when regular rolling or jumping are insufficient.

Throughout the game, various abilities and powers will be added, but this mechanic is the most important, everything revolves around this, so it is essential, vital, that this mechanic is intuitive and feels good to use. Many obstacles will alter how the mechanic functions, but these changes should be temporary or localized.

Polarity control is implemented simply by reading the analog input of both the positive and negative inputs and multiplying the difference with the maximum positive (and negative) charge. The fish DMO object is then given this new charge, and the magnetism system handles the rest automatically.

You can view the relevant source code in the following documents: move.cs , MagneticObject.cs.

Figure 2.3. Players can change their magnetic charge at will. This allows them to exploit magnetic forces to move around, using obstacles in the level, or each other.

2.2.3.2. Freezing

Both players can also stop their own movement by holding "freeze" on the input device. This causes the fish to freeze in place, potentially in mid-air, stopping and preventing all further motion. While frozen, all inputs other than ending the freeze (letting go of the input) will be ignored. The fish keeps whatever magnetic charge it

froze with, and so exerts magnetic forces, but cannot change its charge while frozen.

This ability serves as a way to ease magnetic interactions. By freezing one player with a charge, you can ensure that only the other player is moved, which is helpful in many situations. This was not the original intention of the mechanic, but merely an appreciated side-effect. Freezing was meant to hold significance in a puzzle-solving sense, but this has failed. Future level designs ought to find interesting ways to use this power as more than a handicap.

One potential alteration that has been considered is to remove freezing as an ability from Polo, and replace it with a double jump. This would make the two characters more distinct, and it would also prevent certain exploits that abuse the freezing mechanic. Right now, players can freeze right after thawing (unfreezing), with no limitation. This is not intended use, so if it is being used for exploitative purposes some limitation should be introduced.

Freezing is implemented in the simplest manner possible, by locking the X and Y coordinates of its position, via a Unity function. In order for this to work properly, there are also a few steps that check if the fish is currently on a moving platform, and if so, disconnects the fish from the platform. More on moving platforms later. In the future, it may be good to look at making freezing a sort of gradual process, where it takes about half a second to freeze fully, and until then motion is merely slowed. This would hinder "spamming" and "feathering" of the ability, but it is also more challenging to implement.

You can view the relevant source code in the following documents: Freezer.cs.

Figure 2.4. In this image, Polo has frozen himself in place while Maggie hangs on with strong magnetic forces. Gravity will cause Maggie to start swinging around Polo.

2.2.4. Level Mechanics and obstacles

2.2.4.1. Buttons

Some levels have buttons, usually at fixed positions, that the players can roll onto in order to activate them. Activating a button will have some effect on the level. It

might move a platform, it might open a door, etc. These buttons are used often to create platforming challenges or puzzle challenges in the levels.

Different buttons behave in different ways, some stay activated, others deactivate after a few seconds, others deactivate other buttons when pressed. There is currently no visual indicator of how the button will behave once pressed, they all look the same. This means that players will need to press each button to find out both what effects they have on the level, and how long these effects remain.

One visual indicator that would be useful, and should probably be implemented, is a timer of sorts that appears above activated buttons that deactivate themselves after a set time.

Buttons, and "Button responses", are implemented by giving each button a list of button response references that will be called upon once activated (and deactivated). Button responses is a superclass with functions that are overridden in each custom button response behavior.

You can view the relevant source code in this folder: Button.

Here are the button responses that have been made thus far:

- A few different responses will move around objects or change the way in which they already move.
- One response will enable or disable a specific magnetic object, usually a magnetic booster.
- There are two buttons in level 4 that only move a platform one pressed simultaneously.
- One response will diable polarity locks, another will switch polarity locks, but these are not in use.

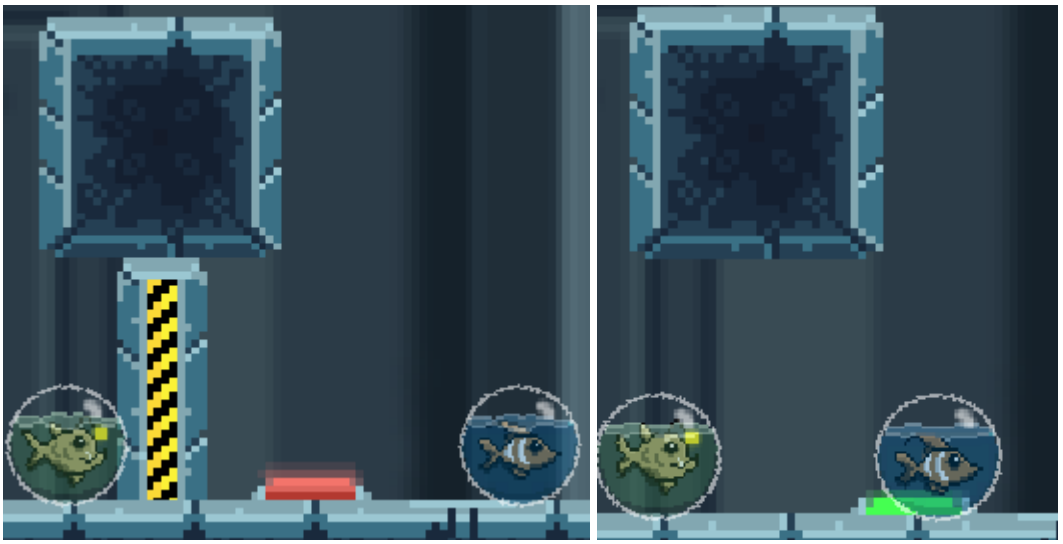


Figure 2.5. This button opens a barrier once pressed, allowing Maggie to pass.

2.2.4.2. Exit Portals

At the end of each level is a purple "exit portal". When both players are close to the portal they will be transported to the next level, or to the main menu if they are playing the last level.

This is a very simplistic and rudimentary way of transporting the players to the next level, but it also has the advantage of being recognizable to the players, since they'll see them so often.

You can view the relevant source code in this document: ExitPortal.cs.

2.2.4.3. Checkpoints

A feature of Polarfish is the frequent use of checkpoints. Once the players have cleared a significant obstacle, they will respawn after that obstacle if they perish. Each level includes 1-3 checkpoints, because the game might be somewhat frustrating without them. It may however be interesting to experiment with fewer checkpoints, or none at all.

Checkpoints consist of an area and a point, both are invisible to the players. The checkpoint is triggered once both players are within the area at the same time. This requirement is added in order to prevent cheating by sending in one player at a time. The areas should be set up in a generous manner, so that they cover more or less everything in the level that isn't a preceding obstacle. The other part of the checkpoint, the point, designates where players are respawned when killed.

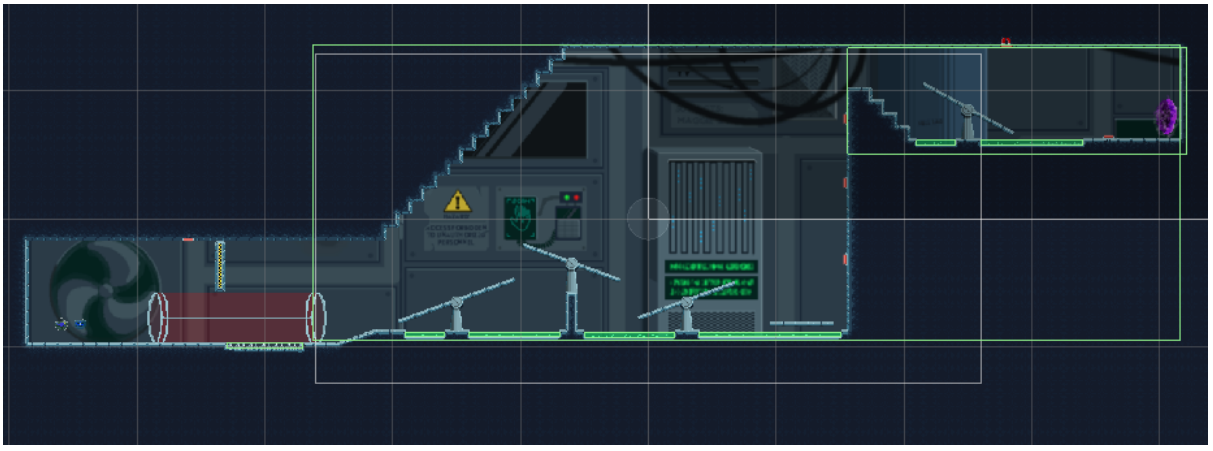
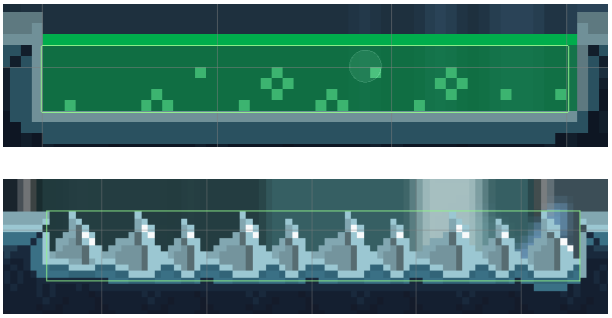


Figure 2.6. The green rectangles are two different checkpoint areas. This image indicates how areas should be set up, encompassing all that isn't a preceding area.

You can view the relevant source code in these documents: Checkpoints.cs, Respawn.cs.

2.2.4.4. Respawn Triggers

These are areas that, once by a player, force the triggering player to respawn at the latest checkpoint. Essentially, it "kills" the player fish. These obstacles can take a variety of shapes, such as acid or spikes, but they all act the same.



You can view the relevant source code here: Respawn.cs.

Figure 2.7. Acid pools, spikes, and many other surfaces, can act as respawn triggers.

2.2.4.5. Seesaws

A common platforming obstacle is the seesaw. It is a simple structure that rotates depending on how weight is distributed upon it. The rotations should be limited at 30 degrees away from a horizontal arrangement, and the seesaws should have a noticeable initial rotation, so as not to be confused with regular platforms.

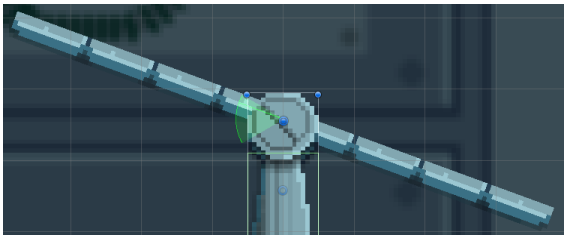


Figure 2.8. A typical seesaw in the lab, with an initial rotation of -20 degrees.

2.2.4.6. Boosters

Many levels contain "boosters", magnetic surfaces that emit magnetic forces directed away from the surface. These forces should be strong enough to be immediately noticeable, but may vary depending on its purpose in the level. By making boosters directed, it is easier for a charged player to make it onto them, and harder to use them for any unintended purposes.

You can view the relevant source code in the following documents: MagneticObject.cs, DynamicMagneticObject.cs.

Figure 2.9. Boosters may be placed with either a positive (blue) or negative (charge), with any orientation, as long as the magnetic forces are directed away from the surface.

2.2.4.7. Polarity Tubes

Polarity tubes are areas that enforce a certain magnetic charge upon the fish inside them. The tubes are often long and wide enough for other obstacles to be contained within them, but these obstacles are not affected by the polarity tube's effect.

Polarity tubes act purely as a hindrance to the players, for instance, it forces two players in the tube to keep their distance, preventing them from leaping from each other using magnetism.

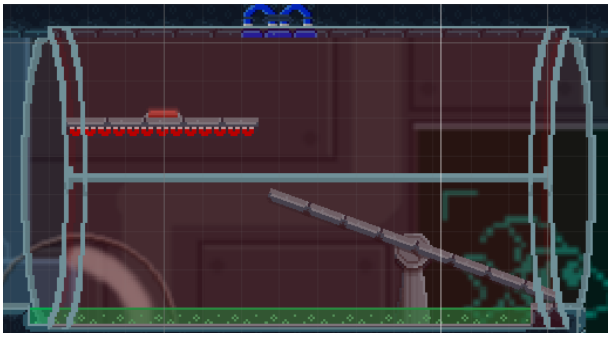


Figure 2.10. This polarity tube contains 4 other obstacles, including one of each polarity.

Despite appearances, tubes do not have solid walls, so player fish and other objects can pass into and out of the area from any side. This is fine as long as the tubes are placed between solid surfaces, but it may cause confusion in other cases. In the future, adding solid walls to the tubes should be considered.

You can view the relevant source code here: PolarityLock.cs.

2.2.4.8. Barriers

Possibly the simplest obstacle, barriers, are simple solid rectangular blocks that obstruct passages, as seen in figure 2.5. These barriers can often be moved by pressing some buttons, meaning that barriers usually function as doors or gates.

2.2.4.9. Moving Platforms

Certain platforms in the game will move automatically, or start moving once activated by buttons. These moving platforms typically move in a cycle between a set of points, marked by small white dots.

These platforms may test the players' timing, and coordination whenever two players must jump onto them within the same window of time. Often though, they simply serve as a means of moving players from one portion of the level to another.

You can view the relevant source code here: MovingPlatform.cs.

2.2.4.10. Drones

A category of future obstacles that should be implemented is drones. Unlike most other obstacles, drones will move and actively hunt the player fish. Despite this, they are not to be deployed as "enemies" in many other games that can be disposed of, they are instead puzzle elements like the other obstacles. Some drones could be harmed, perhaps by hitting them with an object, or a fish bowl, at great velocity, but even then the drones will recover soon after.

2.2.4.11. Darkness

In certain future levels, or specific areas of levels, will incorporate darkness as a mechanic. The players can normally not see any terrain or object placed within these areas, unless, except for the player fish themselves, which are always fully visible in order to ensure players don't lose track of themselves.

Darkness can be removed by lighting up an area, either with some level element, or more commonly, with the faint light attached to Maggie's head. Maggie's light has limited range though, so Polo will need to stay close by in order to see where he is going.

Maggie's light has a fatal drawback though, it is easily noticeable in the dark, meaning that Polo is better at sneaking past observers. In the levels that include darkness, this relation between the players will be an important puzzle mechanic.

Here is an example: A challenge where Maggie needs to first reveal a path for Polo who needs to memorize it, because Polo will need to traverse the dark path without Maggie's help later because at the end of the path is a drone who would easily spot Maggie in the dark, a drone that Polo could slip by.

3. Story, Setting and Character

3.1. Story and Narrative

3.1.1. Back story

Maggie and Polo are two magnetic fish aiming to get back to the ocean. They were captured by an evil scientist, who used them for his experiments, which is how they gained their magnetic abilities (and their freezing super ability). Maggie and Polo managed to escape from their prison fish tank into two small sealed off fish bowls, and they are trying to cooperate to navigate through the various obstacles found in the laboratory of the scientist. Ultimately, their goal is to reach the ocean and regain their freedom.

3.1.2. Plot Elements

The levels produced thus far take place in the first environment, the lab, but have no significant plot. The plot of the full game will be simple, two fish escaping to the ocean, that's the gist of it. Here follows a brief outline of the adventure we've planned for them:

The players see the two fish for the first time in the lab of the mad scientist, and they are soon set loose in the lab, without much of an introduction.
A number of levels, roughly 2 hours of real time in duration, take place in the lab, after which the pair escape the lab. The intensity throughout this environment should be mostly low.

Now outdoors, the two fish find themselves in a neon-futuristic metropolis at nighttime. Throughout the next set of levels, again roughly 2 hours of real time in length, the fish must navigate dark alleys, cross busy roads, and avoid the scientist's drones that are seeking them out. At the end of this environment, the two fish end up in the back of a transport van, which takes them out of the bustling city. Throughout the environment, the players should get the feeling that they're being hunted, with a lingering tension that occasionally spikes once the players are spotted.

Next, a few hours later in game world time, the van stops and the two fish find themselves in a remote little town by the sea, a fishing town specifically. The first half of this 2-hour environment should have a low intensity, with certain disturbing elements as the two fish explore the town centered around a perceived massacre of their kind. The game should still be child friendly though, so nothing too scary or gory, but unnerving.

The intensity then quickly ramps up in the second half, when the scientist shows up, having tracked the fish's location (by unspecified means). Suddenly, the previously slow pace becomes a mad scramble to escape the dreadful town and return to the ocean. This all culminates in an exciting final confrontation with the scientist and his drones at the edge of the harbor, a "boss fight" of sorts, but still puzzle oriented, as the two fish desperately attempt to make their way past the scientist, into the ocean, where the game ends.

3.1.3. Game Progression

The game starts out with easy obstacles utilizing the magnetic polarity functionality of the two fish to get the players familiarized with the game mechanic. With each successive level, the complexity of the puzzles increases, as players need to communicate, cooperate, and get creative while trying to navigate through the map to a portal that leads to the next level. Both fish need to stand on the portal to progress, which means that players cannot progress the game with only one fish.

With the potential future addition of new game areas, obstacles and powers, the idea is that the players will be learning these new features at a progressing rate. First, they will be introduced to the new addition with a simple level and then the following levels will use this addition in increasingly difficult manners. For some of the fully new mechanics, e.g. enemies such as drones, there might be an argument for having a more clear-cut tutorial with signs (see the tutorial level). One idea that could be explored is to have one unique power that is emphasized for each game area, similarly to the temples in the Legend of Zelda[4] series. The addition of new powers could also lead to new ways of using old powers. The most optimal solution would be having the old mechanics still have an impact at later stages of the game, so players would always have use of what they have learned before.

3.1.4. Cut Scenes

In the scope of the demo, cut scenes were not implemented.

As we follow the story of Maggie and Polo, the game would start "in medias res" - with the players being thrown straight into the gameplay, in a lab, trying to figure out what to do and where to go. This way, the experience of the players is similar to the fish - not knowing what happened, however being aware that you wish to escape. Consequently, no cut scenes at the start of the game would occur, but later on, as the players get the hang of their powers and improve in their puzzle-solving abilities, some short cutscenes would be added, which would explain how the

[4] https://zelda.fandom.com/wiki/Temple - See Characteristics section

players got to the lab and how they got their powers. The knowledge of the story would develop simultaneously with player skill.

The purpose of cutscenes would be twofold. Firstly, cutscenes would add meaning to the story, uncovering information about the story of Maggie and Polo - who they are, how they got their abilities and where they need to go. Secondly, cutscenes would serve as a bridge between environments - similarly to how portals bridge two levels. These cutscenes would uncover how players transition between areas - e.g. when transitioning between the City area and the Fishing village area, a cutscene would show Maggie and Polo getting closed in a van and being driven to the coast, or in the final level, a cutscene would show the evil scientist arriving at the location of Maggie and Polo, before starting to chase them.

There are a few defined cutscenes, however more would be added as the development of the game would progress.

3.1.4.1. Cut scene #1

The first cutscene occurs still in the laboratory. Maggie and Polo get to uncover more about how they got their powers and where they need to go, as they reach a CCTV room with a recording from their capture, transformation into Polarfish and the aftermath. They can see what they looked like before their transformation, and it helps them remember where they come from. Additionally, they see the evil scientist and finally get to know who their main enemy in the game is. Lastly, the cutscene would show them a path through the laboratory, which would set up the final levels in the lab area and lead as an introduction to the city area.

3.1.4.2. Cut scene #2

The next cutscene would be very short, showing how Maggie and Polo are exiting the lab and arriving at the new area - the cyberpunk, futuristic city. With this, a new visual identity for the levels is introduced, and the story advances.

3.1.4.3. Cut scene #3

The cutscene occurs at the end of the City area, showing Maggie and Polo entering a van that they believe will lead closer to the ocean - a van with a fishing village image on it. In the cutscene, Maggie and Polo enter the van moments before its doors close, which fades it into darkness and fades back as the doors open (after a timeskip). The players are once again introduced to a new area - this time a fishing harbor town with a beach, which is the final area of the game.

3.1.4.4. Cut scene #4

The final pre-defined cutscene occurs before the last level of the game, as it sets up a final showdown between Maggie, Polo, and the scientist. This is a leadup to a scrolling "escape the scientist" level, in which players will need to quickly clear the last few puzzles and obstacles to escape into the ocean, as the scientist runs after them. The cutscene would show how the scientist is looking for them using a tracking device, how he exits his car and it would end by him finally noticing them and starting to run after them.

3.2. Game World

3.2.1. Laboratory

As the backstory of the game situates Maggie and Polo into the laboratory of an evil scientist, the general look of the world matches this setting. Most of the maps consist of metal surfaces and platforms, with spikes and acids as the main obstacles. The background builds on this feeling by having various computers, vents, metal doors and other objects populating the space.

While the laboratory is cartoonish in presentation, it should be clear that there are usually humans in the building, researching a variety of things. This could be implied with notes lying about, whiteboards in the background with instructions, and yesterday's leftovers from lunch still on the table, as an example.

3.2.2. Metropolis

The metropolis consists mostly of dark structures, most light comes from an overabundance of neon signs and decorations. Main streets are well lit with street lights but the player fish move mostly along smaller walkways and alleys, so these street lights are usually not visible in the game. For the players, this means that only a few areas have proper lighting, most are moderately bright with a variety of coloured tints, depending on the color of nearby signs, and a number of locations, especially back alleys, are pitch black.

The open and public areas are kept neat and clean, with polished statues and fountains. But most other areas are dirty, to varying degrees, with trash abound, such as fast food packaging. Certain locations are also noisy while others are mostly quiet, but almost all locations are disturbed by the occasional vehicle driving by.

The city doesn't have a direct connection to the next environment, the fishing town, but the van that brings the players there should be parked in an appropriate location in the city. Also, nearby scenery should give some kind of hint as to where the van is headed. For example, the players might find the van in a warehouse with a few

advertisement posters on the walls saying something like: “Fresh fish from certified local fishermen!”

3.2.3. Town

This area is less technologically oriented than the city, but still has certain sci-fi elements, such as a hologram projector in a traditional bar, for instance. The entire town should really echo the fishing theme and it is a sort of middle ground between the oceanic world, that is the player character’s home, and the technological madness they witnessed in the metropolis.
The van that brings the fish to the location stops on a nearby hill, overlooking the town. This way, once the fish escape the van, they are greeted with a nice view of the town, the harbor, and the great ocean beyond. By the time the fish reach the village, it is almost morning, so the top of the sun could be visible on the horizon, casting a soft but warm light on the town.

The buildings are mostly wooden and old, with some portions looking almost derelict. It should seem like a once booming industry on the decline, of the few humans out and about this early in the morning, none of them should be young adults, as most have left for the city.

3.3. Characters

3.3.1. Maggie

Maggie is a female deep sea fish - with a light on her head (both for visual cosmetic effect and a gameplay mechanic in darkness, in later levels).

Figure 3.1. Maggie - neutrally charged

3.3.1.1. Maggie’s Personality

Maggie’s personality throughout the story would contrast the stigma about deep sea fish - their mean look and predatory behavior. In contrast, she would be the more friendly, relaxed and kind of the two fish throughout the story.

3.3.1.2. Maggie’s Look

Maggie was designed to look friendly, despite the type of fish she is. She has a slight smile and a light on her head, which, in the full game would also have a functional

reason - illuminating an area around her in a dark level. Furthermore, for better distinction between Polo and Maggie, Maggie's water was slightly tinted into a green shade, to simulate algae within the water, and make it easier to immediately know which player is controlling which fish bowl.

3.3.2. Polo

Polo is a male clownfish, with white stripes on the side of his body.

Figure 3.2 Polo - neutrally charged

3.3.2.1. Polo's Personality

Polo's personality in the story would be more hot headed than Maggie's. Polo would be a more dynamic and "aggressive" of the two throughout the story.

3.3.2.2. Polo's Look

Polo is a clownfish-like fish, with his color being altered to suit the polarity changing game mechanic. His neutral color is fuller than Maggie's (as he does not live so far away from the light in the ocean, and it helps distinguish the players from each other), and his water is a clear blue color.

3.3.3. Animations

Both fish have movement, jumping and ability animations. Each fish also has three different states (polarities), which require their specific sprite sheets, however the differences merely consist of different colors, and the core animations remain the same. Here is a list of all animations for Maggie:

Figure 3.3. Maggie - idle, blue (also comes in neutral and red)

Figure 3.4. Maggie - jumping left, red (also neutral and blue, for both left/right)

Figure 3.5. Maggie - moving right, neutral (also red and blue, for both left/right)

Figure 3.6. Maggie - frozen, red (also neutral and blue)

Here is a list of all animations for Polo:

Figure 3.7. Polo - idle, red (also comes in neutral and blue)

Figure 3.8. Polo - jumping left, neutral (also in blue and red, left/right)

Figure 3.9. Polo - moving left, blue (also comes in neutral and red, left/right)

Figure 3.10. Polo - frozen, blue (also comes in neutral and red)

3.3.4. Special Abilities

Maggie and Polo can both change polarities to attract/repel from each other and the environment. Furthermore, they have the ability to freeze themselves in place, which freezes their position and polarity (another meaning of Polarfish). Polo's abilities are identical to Maggie's currently, however, if the game is developed further, we plan to distinguish the players' special abilities (currently - freezing for both fish). Polo would then lose the freezing ability, and gain a double jump ability, which would introduce another layer to the necessity of collaboration between players, as well as unlock new possibilities for the puzzle design.

3.3.5. Physical characteristics

The movement of the fish inside the fishbowls must look and feel natural. The player models are quite complex, consisting of multiple layers of separate sprites that behave differently when physical forces are imposed on them. Each player model consists of a fish bowl - which rolls left/right, a water sprite - which has the characteristics of a liquid and must react as such, and the fish sprite - which is directly (visually) controlled by the player - when the player rolls the joystick to the left, the fish moves (and pushes the bowl) to the left, creating an illusion that the

user is actually in control of the fish, with the water and bowl merely reacting to the force of the fish.

3.3.6. Relationship of the characters

Polo and Maggie find an unexpected friendship as they attempt to escape the lab back into the ocean. Their relationship develops similarly to how the players' relationship in the game develops - they get closer and need to collaborate more to reach their goals. Their relationship to other actors in the story - the scientist and his drones is negative - they are trying to escape the evil scientist, who is trying to re-capture them to continue his experiments.

4. Current Levels

4.1. Tutorial Level

4.1.1. Synopsis

This level teaches the players the basic available operations including moving, jumping, analog polarity, and special ability of players. The interactable objects in surroundings like boosters, seesaws, polarity-lock areas, moving platforms as well as buttons, are involved for players to get familiar with.

4.1.2. Introductory Material

There are several screens situated throughout the level, which introduce the controls and game mechanics one-by-one, and let the user try them out directly in-game.

4.1.3. Objectives

To pass the tutorial and get familiar with the operations.

4.1.4. Physical Description

The encounters are placed one by one from the basic to the advanced to help players understand what they will meet with in the future levels.

4.1.5. Map

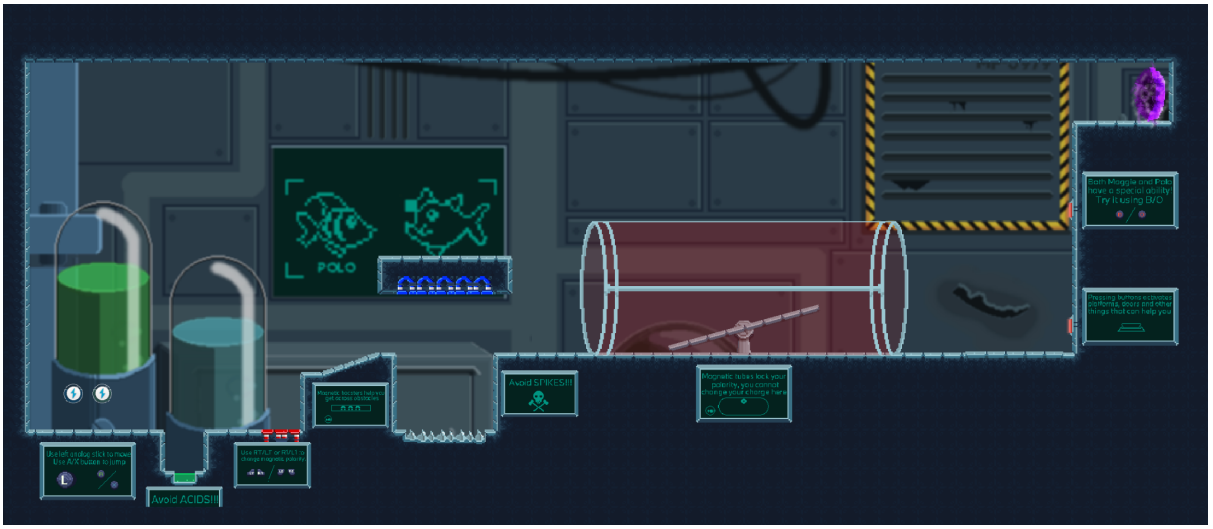


Figure 4.1. The map of the tutorial.

4.1.6. Critical Path

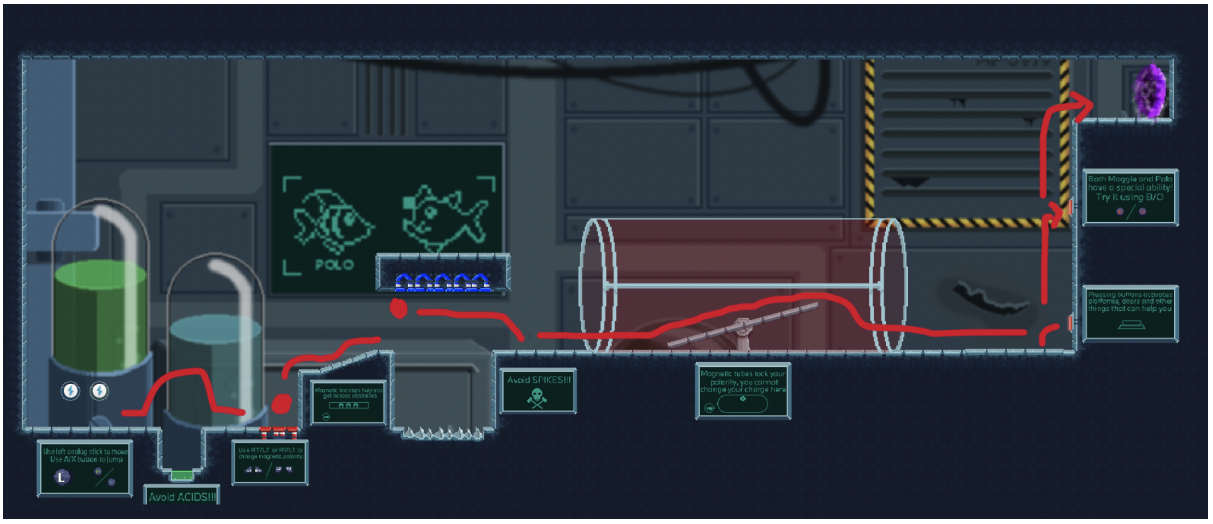


Figure 4.2. The critical path of the tutorial.

4.1.7. Encounters

- An acid Pool
- Magnet boosters
- Spikes
- A polarity lock tube
- A seesaw
- A moving platform
- Buttons

4.1.8. Level Walkthrough

To complete the level you have to do the following: jump over the acid pool, repel off the red magnet by turning red, attract to the blue magnet by turning red to avoid the spikes, cross the polarity lock with the seesaw one fish at the time, press the button to start the elevator and lastly press the second button to make the elevator reach the top.

4.2. Level #1

4.2.1. Synopsis

This level is a simple demo level after the tutorial, which will help players know how they will cooperate in this game.

4.2.2. Objectives

To pass the demo and get a general idea of how they move forward with their cooperation.

4.2.3. Physical Description

A boosters left with a wall blocking first make them feel like they can pass alone as before, while the next door blocking will remind them they are a team. The tilt platform also requires teamwork.

4.2.4. Map

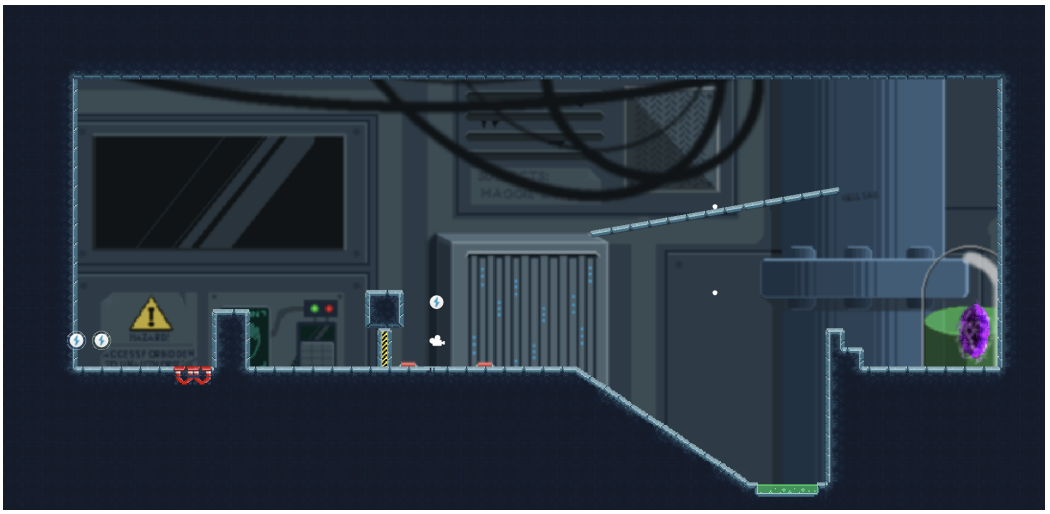


Figure 4.3. The map of Level #1.

4.2.5. Critical Path

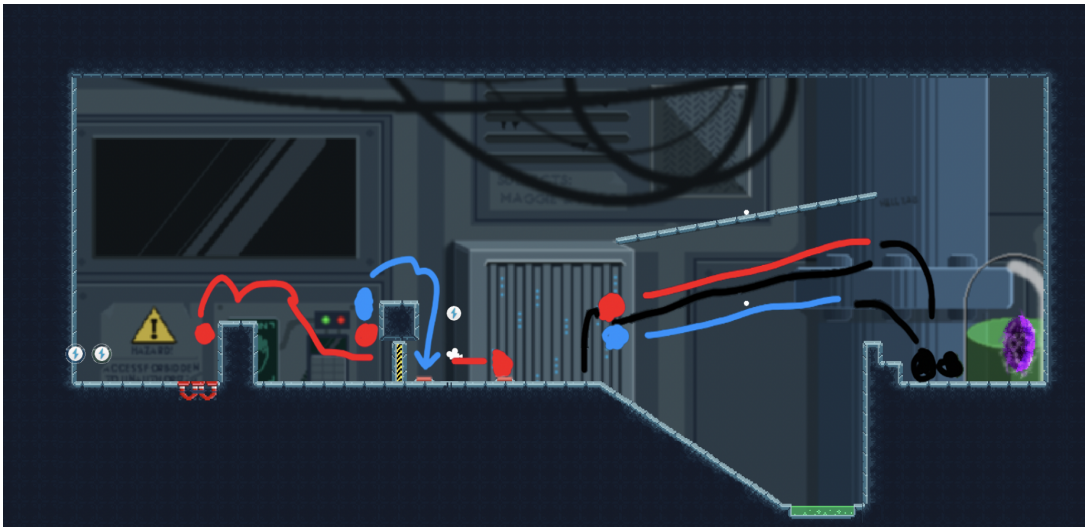


Figure 4.4. The critical path of Level #1.

4.2.6. Encounters

- Magnet boosters
- A blocking high wall
- A gate
- A tilt horizontal bar
- An acid pool

4.2.7. Level Walkthrough

First, the players will move to the booster, and analog the positive polarity to jump across the first barrier. Before the gate, one should jump up and freeze, and the other should jump upon it. When it arrives on the other side, the button should be pressed and the gate will open for the other. Then they press the button that controls the horizontal bar moving upwards and downwards. They should stop pressing when they find the height of the platform is proper to reach. One of them will jump upon the other, reach the bar and analog polarity. At the same time, the other should follow quickly and switch polarity to stick to the partner. Finally, they move forward together and reach the portal on the right.

4.3. Level #2

4.3.1. Synopsis

This level mainly includes magnet boosters on the ceiling, some seesaws and moving platforms.

4.3.2. Objectives

Players are expected to get more familiar with the operations, try to cooperate better and pass this more challenging level compared with the previous ones.

4.3.3. Physical Description

The different platforms are placed in an interval way. There is not much floor space but the platforms which increase their cooperation.

4.3.4. Map

Figure 4.5. The map of Level #2.

4.3.5. Critical Path

Figure 4.6. The critical path of Level #2.

4.3.6. Encounters

- Spikes
- A large acid pool.
- Two seesaws
- Two moving platforms
- Two boosters

4.3.7. Level Walkthrough

Sticking to the top magnet with analog polarity is the first step. Then both fishes should try to fall on the narrow ground and keep each other safe. One should jump upon the next tilt platform with the other's help. Then both fishes should change their polarity, stick to each other and move forward. They will fall on the seesaw to keep balance without falling. When the next moving platform comes close, they should jump onto it and try to balance on the next seesaw. When the second platform comes, they jump onto it and wait for the vertical moving platform to come. When they stand on the vertical moving platform, and are near the magnet boosters on the ceiling, one should jump and stick to the ceiling, move to the right,

and save the other one to the right. Going right and falling down will make them reach the portal.

4.4. Level #3

4.4.1. Synopsis

Level 3 combines the puzzle elements and platformer challenges, which requires a higher cooperation of players. The polarity lock was also involved besides those obstacles in previous levels.

4.4.2. Objectives

Find out how to help each other move forward, pass the challenging seesaw with good cooperation, and reach the portal.

4.4.3. Physical Description

The polarity lock at the beginning limits the players' ability and gives a simple puzzle. Then several seesaws in a row provide the challenge for cooperation. So do the moving platform, buttons, as well as crossing the long acid pool.

4.4.4. Map

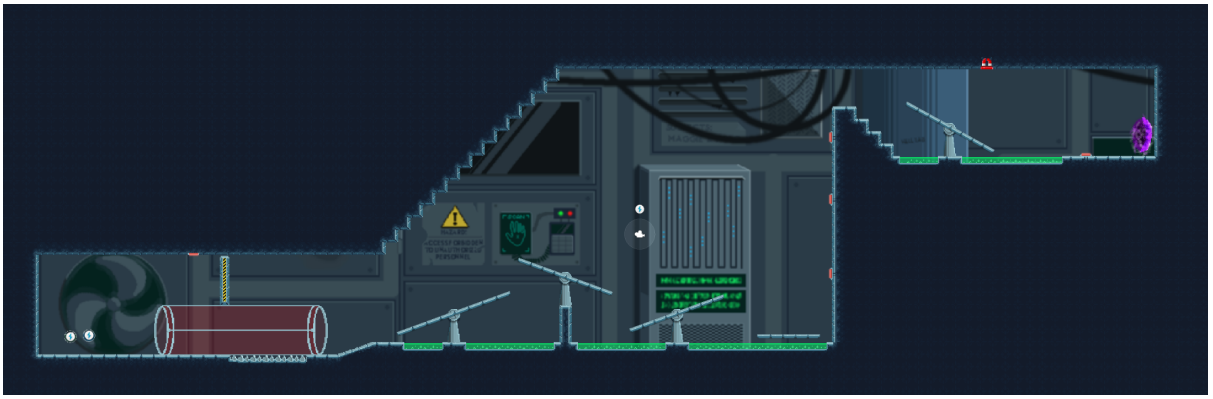


Figure 4.7. The map of Level #3.

4.4.5. Critical Path

Figure 4.8. The critical path of Level #3.

4.4.6. Encounters

- Polarity lock tube
- A gate
- Spikes
- Seesaws
- Several acid pools
- A moving platform
- Buttons
- A magnet booster

4.4.7. Level Walkthrough

At the beginning, one player should try to reach upon the tube through jumping on the other which is frozen when jumping. The other should enter the tube so that its partner can switch to the positive polarity and press the button on the ceiling to open the gate. After it falls down, it should switch to the negative polarity so that they can move forward together and pass the spikes in the tube safely. Then there will be a long way to challenge freezing skills. With good cooperation of freezing ability, they can cross these seesaws and push the buttons on the right side of the wall, lifting themselves up with the platform, and reach the right. The right part also should cross with freezing. They can use the booster on the ceiling to avoid falling into acid.

4.5. Level #4

4.5.1. Synopsis

This is the last level in the Lab, and the most challenging one. They have to think more when cooperating since more puzzles are involved. All elements can be seen in this level in a more complex way.

4.5.2. Objectives

The players should have a more positive attitude to face the challenges. They should keep patience, solve the puzzles together, and find a way to pass.

4.5.3. Physical Description

The beginning part with several gates and boosters was considered as a simple puzzle that players should think about. (Might be updated into a more difficult one in the future version). Then the middle part requires both players to explore the function of the buttons and try to help each other in a more challenging way on their individual path due to the high ceiling, long slope and the platform far away in the acid pool. The final part also wants the players to explore the buttons and involves the polarity lock which can be turned off. The magnets on the ceiling were intended to provide a falling force so that players on the other side could jump high.

4.5.4. Map

Figure 4.9. The map of Level #4.

4.5.5. Critical Path

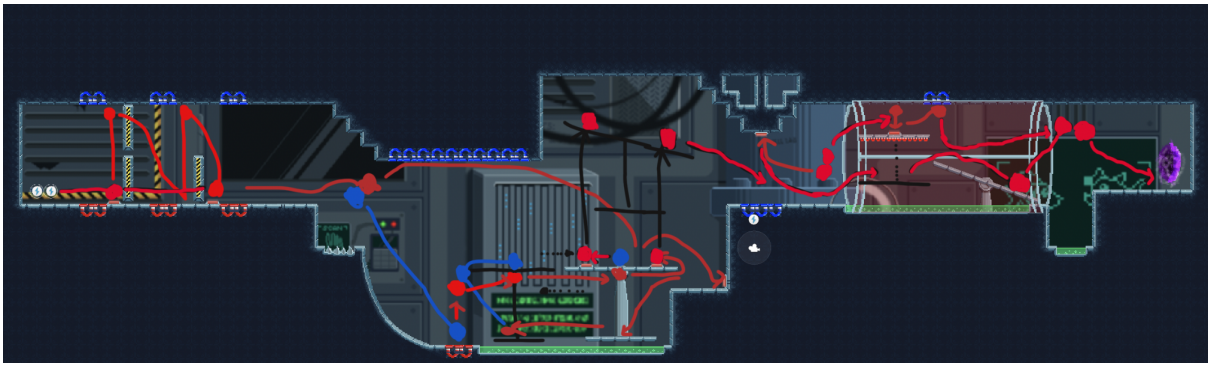


Figure 4.10. The critical path of Level #4.

4.5.6. Encounters

- Magnet boosters
- Gates
- Buttons
- Spikes
- Acid Pools
- Seesaws
- Moving Platforms

4.5.7. Level Walkthrough

One player should press the button on the floor to move the gate, then the other stick to the ceiling and move to the middle part. The button should be pressed again so that the gate moves back and it is possible to stick on the ceiling again and reach the right button. After that button is pressed, both players can meet each other before the spikes.

They are assumed to jump, analog polarity and freeze in the air. One could stick to the ceiling, and the other fall onto the slope and should stop through the magnet. After passing the ceiling magnets and going across the seesaw, the button on the right wall should be pressed so that the platform could go left. It should get to the platform and move with it. Through switching to positive, the player could jump high and freeze and the other player could reach upon the seesaw. With one player upon the seesaw and the other sticking to it, the button below the seesaw can be kept pressed and the platform moves back to right. After that, they both reach upon the seesaw and stand on the two buttons simultaneously, so that the seesaw will go up.

When they reach the last part, one of them should be sent to the high platform with positive lock through freezing, and the other should try to press the button on the top in order to let the button on the platform be pressed down. The platform will then go downwards and the player outside can reach the platform in the tube. When they both reach the seesaw, one of them can use the ceiling magnet to provide a falling force, helping the other jump high so that they can reach the portal side with freezing easier.

5. Interface

5.1. Visual System

5.1.1. Menus

To allow the players to navigate between the levels more easily and to pause the game, various menus were implemented.

5.1.1.1. Pause Menu

Whenever the pause button is pressed the game freezes and the pause menu is opened. In its current version it has three buttons: Resume, Menu and Quit. “Resume” starts the game (same as clicking the pause button again). “Menu” moves the scene to the Start Menu. “Quit” closes down the application, this will only work in a build of the game and not in the editor. The menu is very simple and has a few visual cues that signal to the player both that the game is paused and what button is about to be pressed. When pausing the game, a tint of black is added over the game. This allows you to still see the game, but get the feeling that it is paused. Each button in the menu also gets highlighted when hovered over or selected, which can be seen in the figure below.

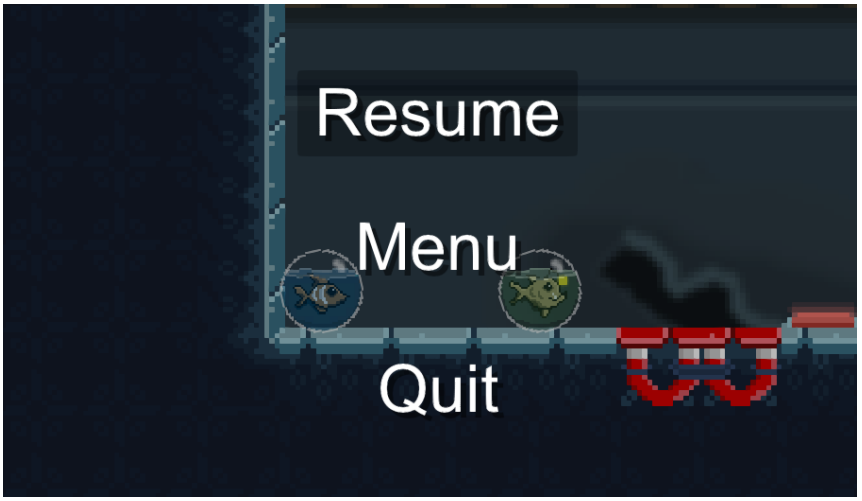


Figure 5.1. The Pause menu with the Resume button highlighted

5.1.1.2. Start Menu

At the start of the game, the scene with the start menu is presented. It has four available buttons: “New Game”, “Quit” and “Levels”. “New Game” starts the game from the tutorial by changing the current scene. “Quit” closes the application similar to the quit button in the Pause Menu. “Levels” moves the current scene to the Level Selection. In a future implementation there could also be a settings menu that could possibly include controls for the SFX and music volume, e.g. sliders, and options for controls and graphics.

Figure 5.2. The Start Menu

5.1.1.3 Level Selection

The Level Selection screen can be accessed by selecting "Levels" in the Start Menu. The screen consists of a list of Level Buttons and a "Back" button at the bottom. Each Level Button changes the scene to the specified level scene. The "Back" button moves the scene back to the Start Menu scene.

Figure 5.3. The Level Selection screen with level 1 highlighted

5.1.2. Camera

The default camera is designed to follow both players and zoom in and out to an appropriate level, in order to reveal as much artistic detail of the level while also making the players sufficiently aware of their surroundings. An extra margin of about 7 units of length is always added to the sides, so that players moving toward the edges can see where they are heading. This margin may need some adjustment.

In certain portions of the game, important level elements that the players should take note of would end up outside the camera's default view. For this reason, predetermined triggers that we call "vistas" have been set up to make sure that the camera pans and/or zooms out to ensure that all currently important points, players included, are shown on screen.

Vista objects have a list of viewpoints, references to transforms in the game world that will be kept in view, for as long as the players are within the vista area. Also, all of the vista object's children in the hierarchy will automatically be included in the list of viewpoints, unless disabled.

5.1.3. Lighting

Lighting has not yet been implemented in Polarfish, which consists mainly of 2D sprites, rendered with their normal colors. However, in order to implement the darkness mechanic in section [#Section number here, probably 2.2.3.11], some form of lighting mechanic must be present.

One option is to use Unity's normal point lights in levels with dark areas. While simple, it comes with the drawback of a gradient light. These lights are typically meant to look realistic, so they will gradually fade from white to black. Lights in Polarfish are meant to be a binary state, light or dark. It is also a very computationally expensive solution.

Another option is to slap a black sprite on the user interface with a circular cut-out centered on Maggie and any other light source, kind of like in old Pokémon games. The major drawback here is that we'll need a neat solution for handling multiple light sources, making sure that the cut-outs don't interfere with each other.

5.2. Control System

The controls are handled by the Unity Input System. The game automatically notices controllers or keyboards based on input and assigns players in the order they press their buttons. The input system can be described by two features: actions and control schemes. Actions are the available commands for the fish such as jumping, changing polarity, moving, etc. The control scheme is the mapping between physical buttons and actions. We currently have two control schemes implemented: keyboard and gamepad, with the gamepad being the main control scheme.

In the table below, the mappings between the two control schemes and the actions can be seen. Button south and button east would be the "A" and "B" buttons respectively on an Xbox controller.

Action	Controller input	Keyboard input
Jump	Button south	Space/right control
Move	Left joystick	WASD/arrow keys
Freeze	Button east	Left control
Positive Polarity	Right trigger	E
Negative Polarity	Left trigger	Q
Pause	Start	Escape

Table 5.1 Action and inputs

5.3. Audio

The audio in the game is handled in two ways: either with the AudioManager game object or through the audio sources placed on objects. Sounds and music can be added to the AudioManager, which can then be played from anywhere. Each sound in the AudioManager has a name (unique so that sounds can be called without issue), a pitch, and a specified volume.

5.4. Music

The music of the game had the goal of fulfilling two themes. One of them being an "aquatic" feel while the other being "cyberpunk" inspired. The game in its current state doesn't have any music implemented (only a short Ableton show file was created), but various concepts were explored during the project.

To fulfill the futuristic cyberpunk sound, an analogue synth bass was experimented with. When experimenting, the idea was to create a soundtrack for the levels in the demo. For the bass, both a steady chord progression and a bass line were created for different parts. The reasoning for this was to give the track some variation and to be less repetitive. Since this was a background music track, the concept was to keep it less emphasized in the background. Consequently, minimal drums were added - a steady "four on the floor" pattern occurs during the second section.

The aquatic feel was tested with various pads and reverb. However, the composer couldn't find a good sound and instead opted for recording an arpeggio pattern on a guitar and adding two reverbs on top of it. This made the sound feel very distant and almost submerged. Other effects that were tested for this were a chorus, vibrato, and auto panning. Chorus and vibrato are pretty similar, where the chorus is a combination of wet and dry signals; whereas vibrato is only the modulated wet signal. Chorus was preferred over vibrato since the vibrato effect was a bit too distracting and cacophonous. Auto panning didn't feel fitting either, since panning the signal steadily between the left and right speaker makes it a bit annoying and distracting from the game.

The music in a future variation of the game could have different soundtracks depending on which area the game is currently being played in. The concepts explored in this text could be used and extended in future musical arrangements related to the game.

5.5. Sound Effects

5.5.1. Fish sound effects

Maggie and Polo have different sounds that are played depending on their actions. The main sound is the **rolling sound**, which is played from an audio source on each fish. The rolling sound increases in pitch and volume depending on the speed of the fishbowl. A stationary fish makes no sound, while a full speed rolling fish bowl makes a sound with the highest pitch and volume. This increase in pitch and sound gives the impression of an accelerating rolling ball.

Whenever the two fishbowls collide beyond a certain speed threshold, a **clinking sound** is played from the AudioManager. The original sound comes from a recording of a cue ball colliding with a billiard ball. Similarly to the clinking sound, there is a thud sound that is played when a player and a non-player collide.

When we were using the old "polarity swap" mechanic, a **"zap" sound** was played. With the addition of the analogue polarity, this sound might find another purpose in the future.

Whenever the fish die by touching a respawn zone, **an acid sound** is played.

5.5.2. Platform sound effects

Whenever a door is opened, the **door opening sound** is played. This happens following a button press by the players. This sound was created by recording the opening of a kitchen drawer.

Moving platforms that are activated by the players holding a button have an **industrial hum sound**. This sound is played while the platform is moving and the button is pressed. If either the button is released or if the platform stops moving, the sound will stop.

5.6. Help System

Currently, the help system of the game consists of the first tutorial level. It explains the controls and different mechanics of the game with text next to the obstacles. After the tutorial, most of the guidance is handled by level design. Different design considerations have been made to help the players in the right direction, while still giving the players the chance to solve the puzzles on their own.

In the future, there would be an addition to the Pause Menu, where the players could see the controls. This could be useful if a quick refresh of the controls is needed. The option to remap controls could also be helpful.

6. Technical

6.1. Target Hardware

Couch co-op games are typically played on consoles, so the platforms Playstation 4 & 5, Xbox One & Series X, and Nintendo Switch will be our primary targets. We would also like to release the game on PC, using the same gamepad based controls, but this is secondary.

We will not release the game on mobile, because the controls would have to be reworked to a significant degree. Also, allowing two players would be difficult without online play, which we are not prepared to implement yet.

Releasing games on consoles means adhering to the rules and restrictions of the console developers. Here follows a few popular gaming consoles, and our options for publishing Polarfish on them. We should probably make an attempt with each of the platforms, unless it would extend development time too much. It may also be worthwhile to try to find out which platform tends to produce the most frequent successful indie games, and focus more on development on that platform.

6.1.1. Xbox

Xbox seems to have a variety of free publishing methods for independent developers to launch their games, such as ID@Xbox and Xbox Creators program. We'd need to submit an application to ID@Xbox, but it's probably preferable to the Xbox Creators program, because one quick look at the list of games published with that program reveals that interest in such products is very low.[5, 6]

6.1.2. PlayStation

PlayStation (PS) has a program called PlayStation partners that we could apply to. To do so, we must supply a game pitch in the form of this very GDD, which serves as our game pitch, along with some legal info and specific contact information.[7, 8]

[5] ID@Xbox: https://www.xbox.com/en-GB/developers/id
[6] Xbox Creators Program: https://www.xbox.com/en-GB/developers/creators-program
[7] Playstation Partners: https://register.playstation.net/prerequisite
[8] How to Pitch your Game to Playstation: https://www.sie.com/en/blog/how-to-pitch-your-game-to-playstation/

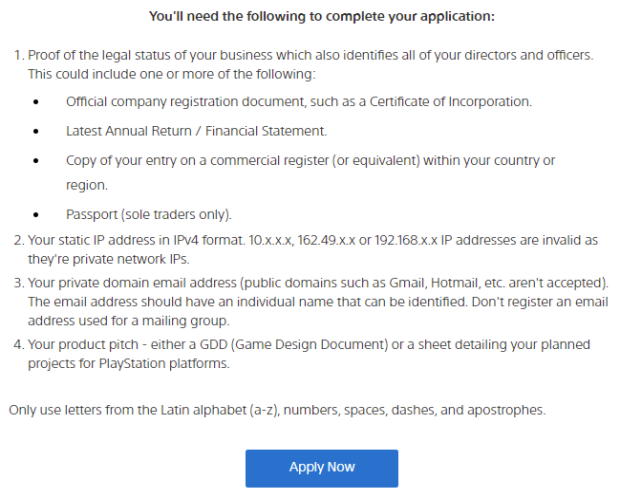


Figure 6.1. The list of requirements of a PS partner application.

6.1.3. Switch

Nintendo has a similar process for their Switch console, without any catchy name. They present a 6-step guide with brief instructions, which do not seem to include a pitch of any sort, you simply apply to the program, sign an (unspecified) NDA, and start developing. It is mentioned however, that the final product needs to be submitted for reviewing before it can be published, but this is to be expected.[9, 10]

6.2. Development hardware and software

The game is being developed on both PC and MAC, for no other reason than preferences and ease of access. On both platforms, Unity is being used to develop the game. The difference in operating systems has only caused minor issues thus far.[11]

PlasticSCM is used for version control, as recommended by Unity. When starting the project we intended to use Unity Teams, but it seems to have been deprecated and replaced with PlasticSCM.[12]

Aseprite is used to make pixel art, Level Designer Toolkit to create levels from said pixel art, and Tiled is used briefly to create colliders and import the level into Unity, using Seanba's "Super Tiled to Unity".[13, 14, 15, 16]

[9] Nintendo Developer Portal: https://developer.nintendo.com/
[10] The Process: https://developer.nintendo.com/the-process
[11] Unity: https://unity.com/
[12] PlasticSCM: https://www.plasticscm.com/
[13] Aseprite: https://www.aseprite.org/
[14] LDtk: https://ldtk.io/
[15] Tiled: https://www.mapeditor.org/
[16] SuperTiled2Unity: https://seanba.itch.io/supertiled2unity

We used Jira Software, by Atlassian, to plan our work in a Scrum-like manner, but we did not make any significant efforts to stay on schedule.[17]

Audacity was used to make the various sound effects in the game, and will likely be the program we continue to use.[18]
Microsoft's Visual studio was used for the coding of all scripts we made.[19]
Finally, Wix was used to create our website[20]. By using Unity to build Polarfish in webGL we managed to make the current state of our game accessible in the browser directly, but it had to be hosted by Simmer.[21, 22]

6.3. Development procedures and standards

6.3.1. Mechanics Development

Our mechanics are the result of wild discussions, and the programmers interpretations of those discussions. The mechanics were only vaguely outlined before work on scripting began. Once the mechanics were decently functional, we made sure to play around with them in test environments frequently, in order to figure out what felt right, and what didn't.

6.3.2. Level Design Process

Our levels were first sketched on paper, not all of these sketches ended up being implemented. Usually, only one team member produced these sketches, but the whole team would give feedback. Every level that seemed feasible, both from a developer's and a player's point of view, were created using Level Designer Toolkit and then imported into Unity. Then, functionality was added by placing obstacles and setting up scripts properly, and if everything still worked, we started testing the level ourselves. We frequently made minor adjustments, and sometimes scrapped entire sections that didn't work as intended.

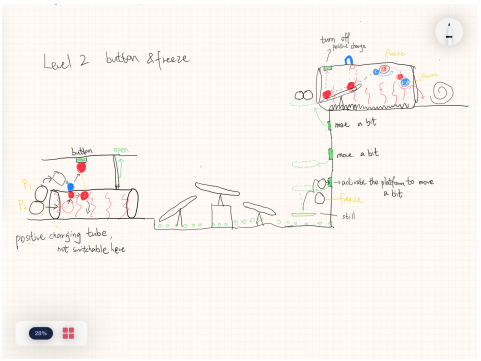


Figure 6.2. An early sketch of level 3.

[17] Jira Software: https://www.atlassian.com/software/jira
[18] Audacity: https://www.audacityteam.org/
[19] Visual Studio: https://visualstudio.microsoft.com/
[20] Our website: https://kthworks.wixsite.com/polarfish
[21] Wix: https://www.wix.com/
[22] Simmer: https://simmer.io/

Once we were satisfied, the levels were subjected to additional playtesting at the hands of other students in the course. Again, minor alterations were made to the levels in response to feedback they provided, but in most cases, the levels were enjoyable for the most part.

6.3.3. Playtesting

Playtesting was meant to test both the level design and the mechanics. We wanted to make sure that the mechanics were intuitive but fun, and the levels challenging but not frustrating. We collected feedback through observation and brief interviews, and concluded that for the most part, both our mechanics and levels succeeded in their tasks.

6.3.4. Collaboration Tools

Our usage of PlasticSCM, the version control system, is very simplistic, and usually just involves pulling and pushing the latest update. Occasionally an alleged merge conflict would emerge, but thanks to plasticSCM's interface these could be resolved without too much trouble.

In terms of planning, we would add new tasks to our Jira backlog, almost exclusively during meetings, after long discussions about what to work on next. The Scrum board on Jira was usually an afterthought, so many tasks would be moved directly to "Done" in batches. Despite this, the backlog in Jira was a good indicator of our progress.

6.4. Game Engine

The game engine we use is Unity 2D, the version is 2020.3.32f1. It is a common tool and has huge amounts of resources. The Unity 2D physics engine was an important part of the magnetism implementation.

6.5. Network

The game in its current state is only available for local players. This was intentional, as playing local coop has some irreplaceable qualities, and the overall experience is different from an online multiplayer game. Additionally, for the course, this made it possible for the developers to focus on more important tasks at hand. But the addition of playing together over a network connection has its pros as well. It allows people to play the game from wherever they are. This would bring the game to a larger audience but would need some extra implementations - some form

of peer-to-peer connection might work well for this game if networking were to be added.

6.6. Scripting Language

C# was used as the main development language for Unity scripts. Most of the important libraries that were used in the scripting come from Unity libraries.

7. Game Art

7.1. Player sprites

7.1.1. Water

Figure 7.1. Water - idle

Figure 7.2. Water - jump/vertical forces

Figure 7.3. Water - left/right / horizontal forces

Figure 7.4. Water - frozen

7.1.2. Full player model

7.1.2.1. Maggie

Figure 7.5. Maggie

7.1.2.2. Polo

Figure 7.6. Polo

7.2. World

7.2.1. Tilemap

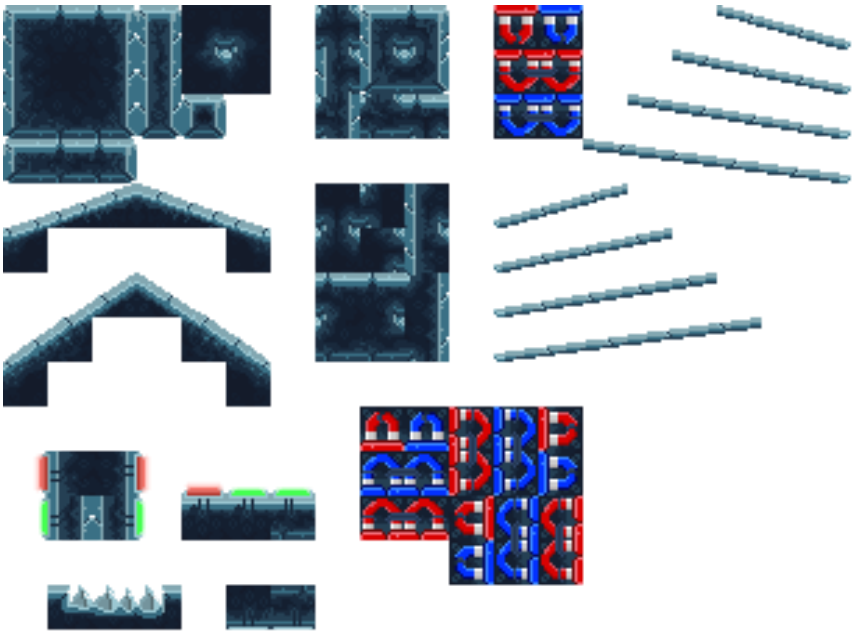


Figure 7.7. Tilemap

7.2.2. Background

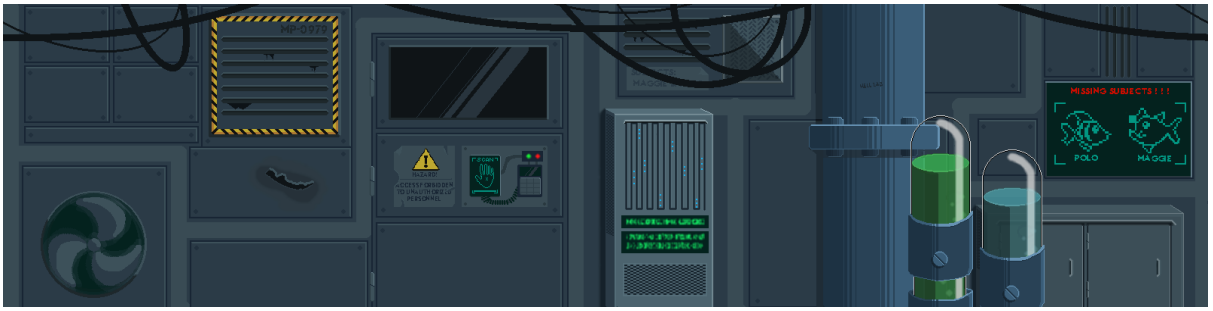


Figure 7.8. Background

7.3. Objects

7.3.1. Obstacles / Death triggers

Figure 7.9. Acid

Figure 7.10. Spikes

7.3.2. Other

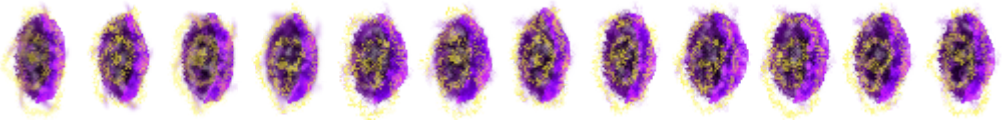


Figure 7.11. Portal

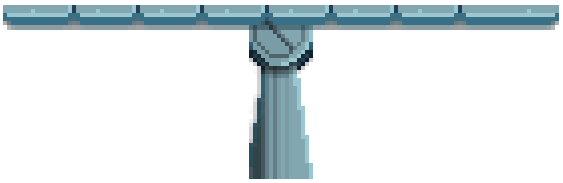


Figure 7.12. Seesaw

Figure 7.13. Door

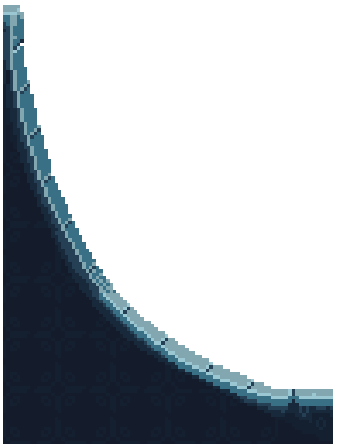


Figure 7.14. Ramp

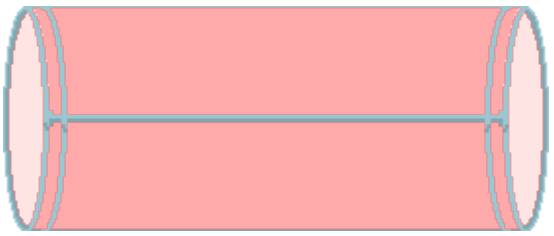


Figure 7.15. Polarity lock tube

7.4. Game Screenshots

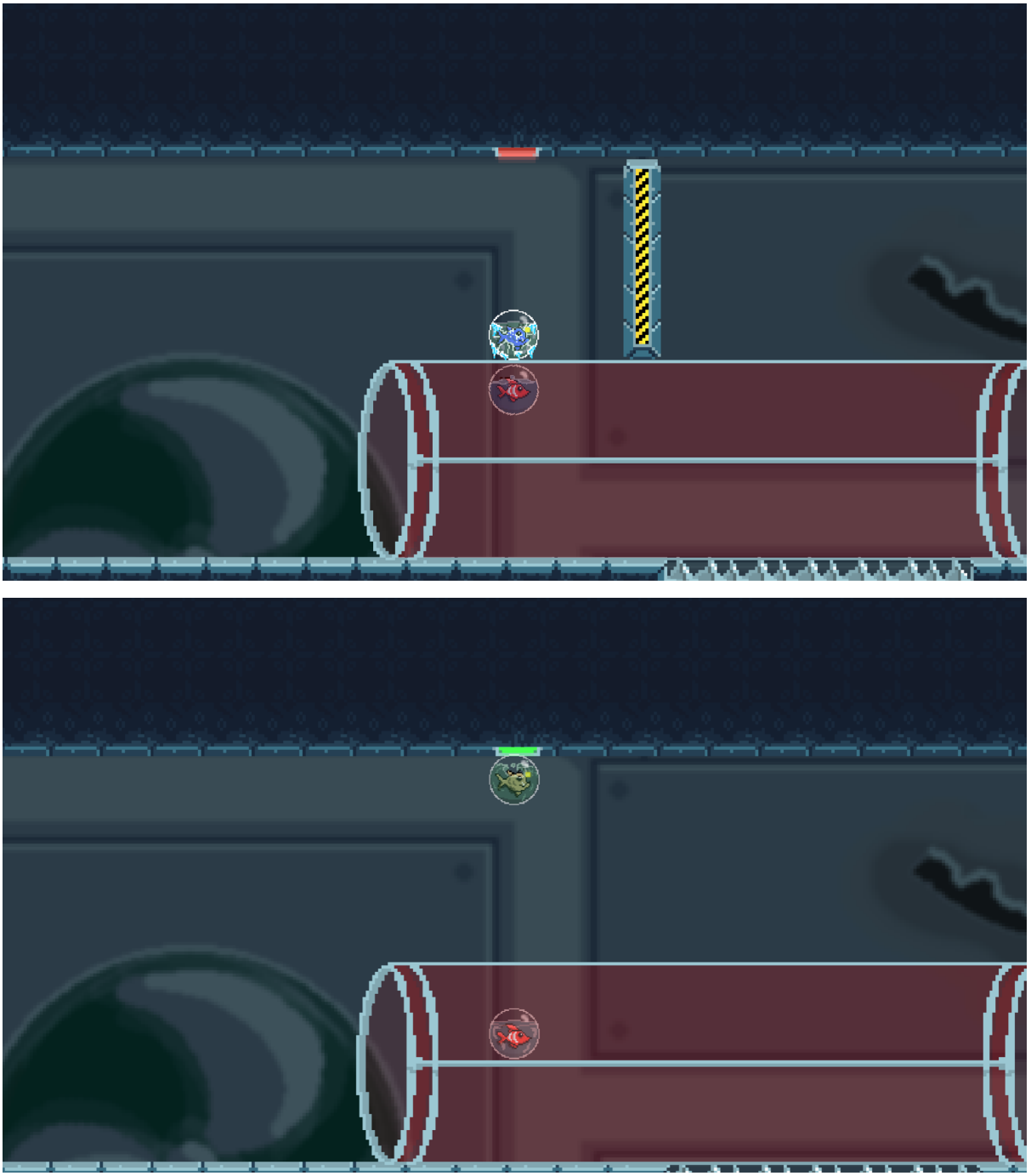


Figure 7.16. Opening a door using magnetic polarity

Figure 7.17. Player collaboration in accessing a seesaw

Figure 7.18. Death animation and instant respawn

Figure 7.19. The end of the tutorial with the portal

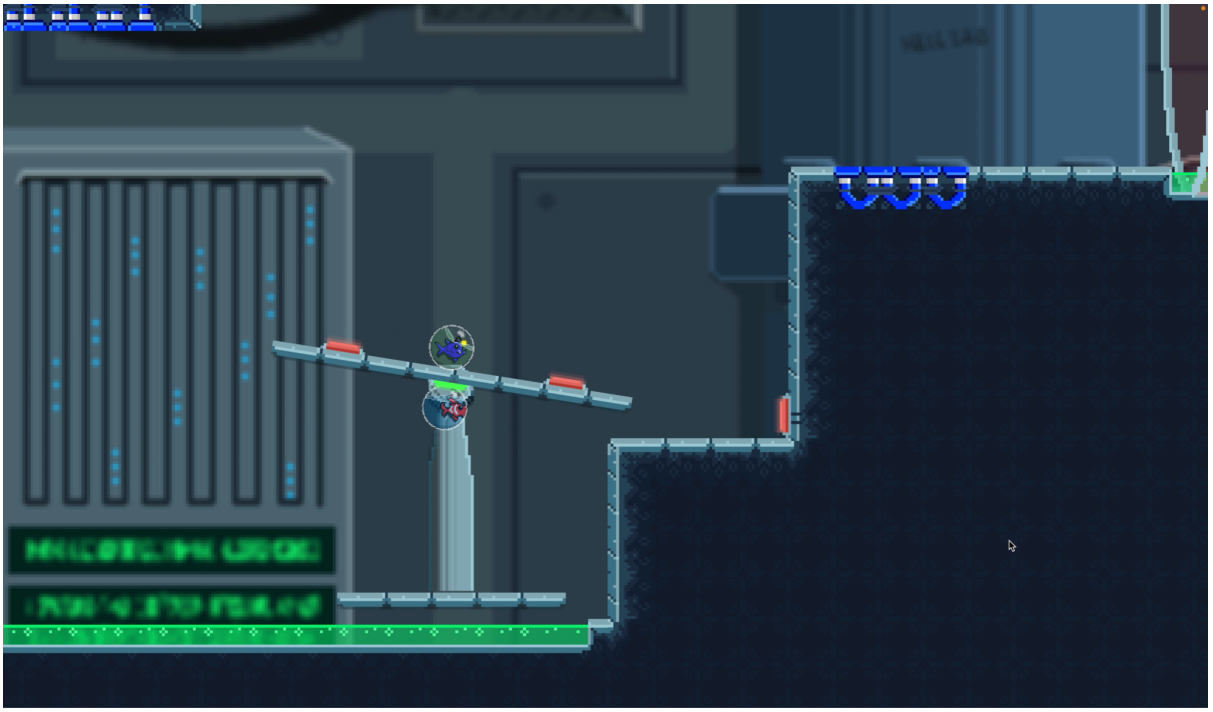


Figure 7.20. A complex puzzle which requires creative solutions

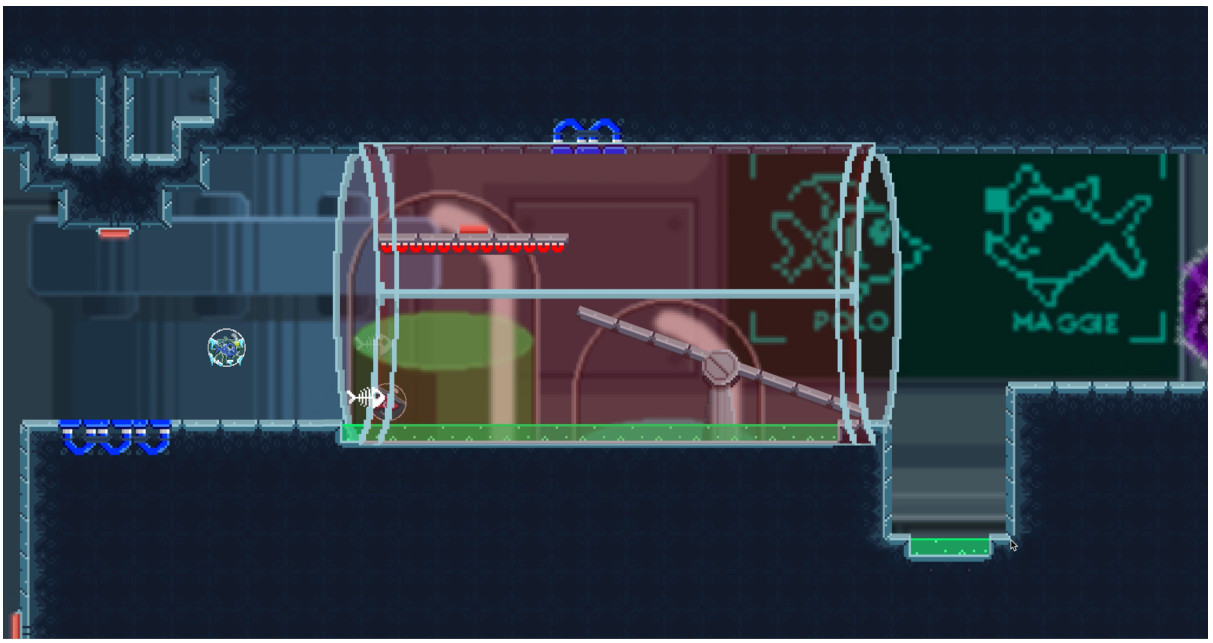


Figure 7.21. Final polarity lock puzzle before the end of the demo

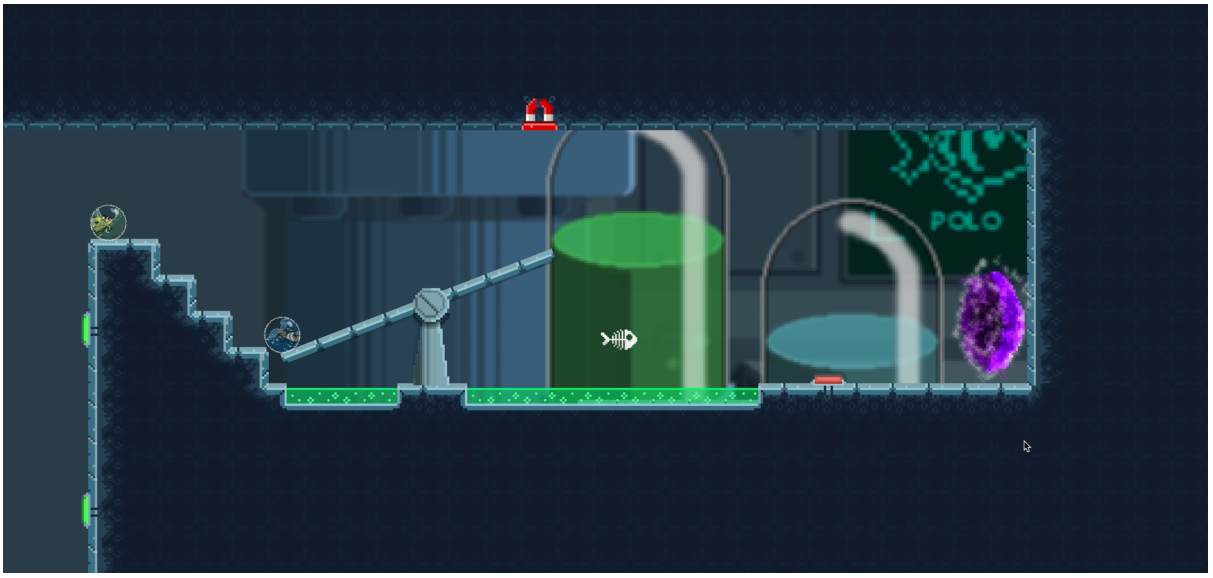


Figure 7.22. Player death before the end of a level

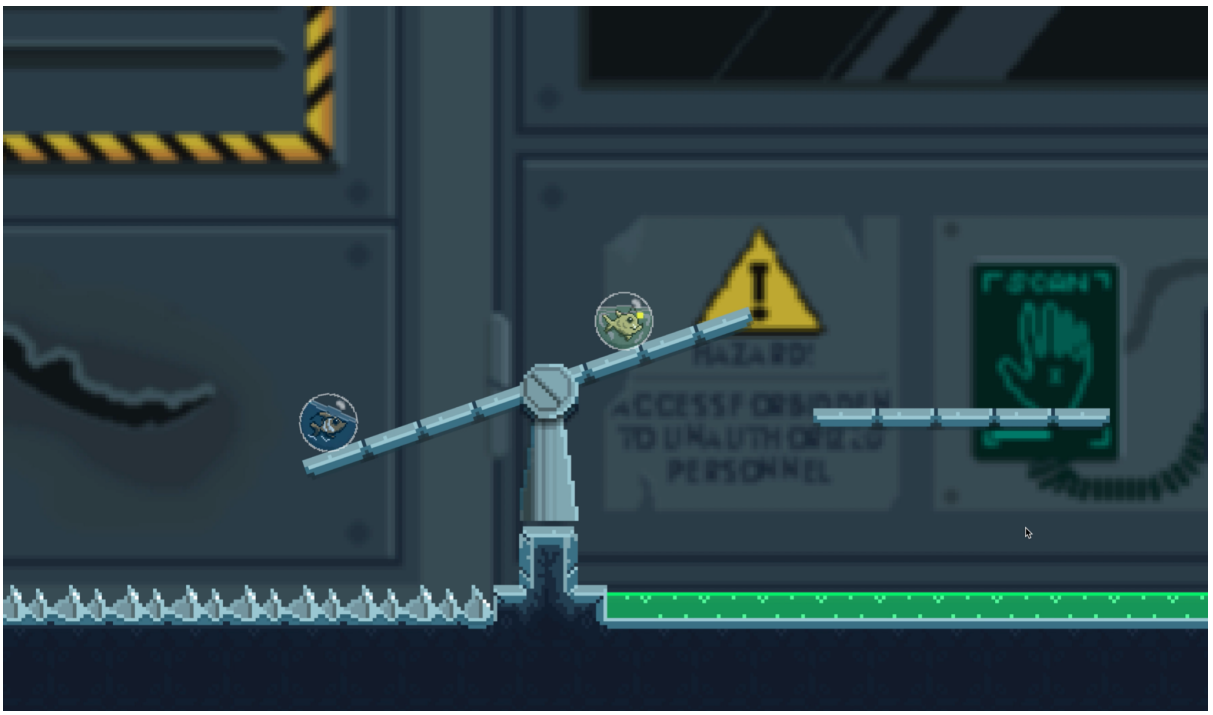


Figure 7.23. Balancing on a seesaw

Figure 7.24. Players getting familiar with how the magnetism can help them traverse the environment

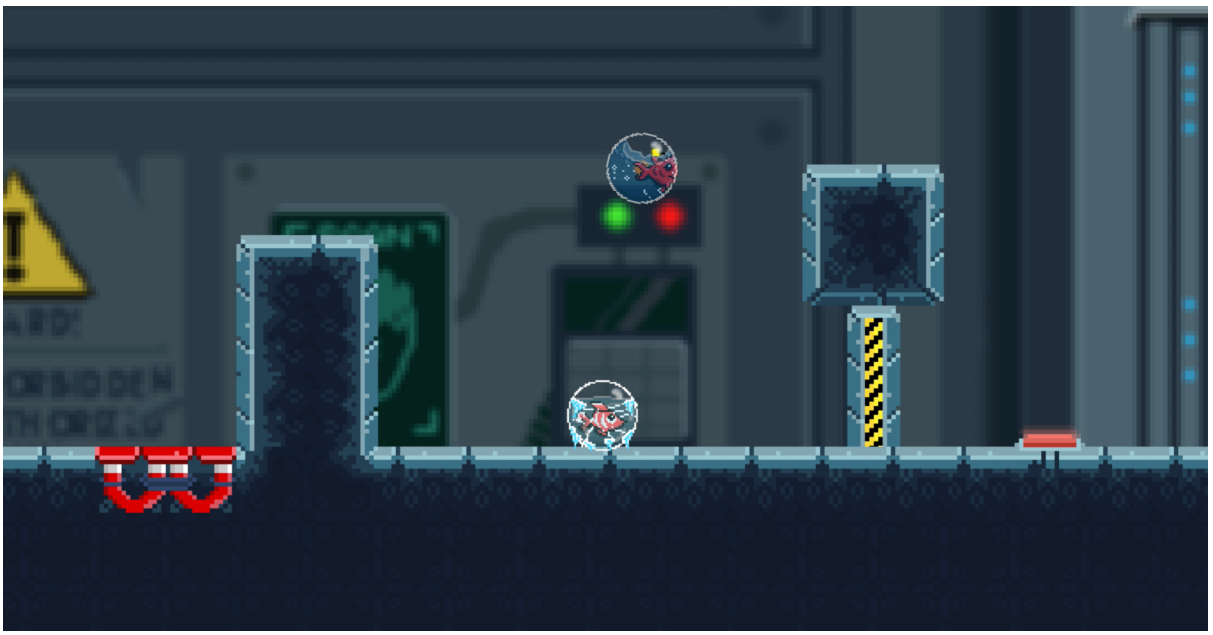


Figure 7.25. Player collaboration in accessing a button

8. Development Software

8.1 Unity Editor

Most of the work for combining the different parts of the game was done in the Unity editor. To collaborate and share resources between different computers, we used Plastic SCM. This version control program allowed us to work in a simple git-like manner by committing changes. It took some effort to set up, but afterwards, it was easily integrated into the Unity editor. The control system was also integrated into the game by using different windows available in the Unity editor. We used the personal pricing version of Unity, which is free. If the game was to be released, a paid version would probably have to be used.

8.1.1 Animation

The animations consist of various spritesheets which had to be correctly imported and then implemented using Unity's Animator. The spritesheets first needed to be sliced into individual frames, and then these frames could be used for implementing animations. Each animation for the player models is split into three sections - the start (transition to the animation), the middle (the looped animation itself) and the end (transition to idle). These are three separate animations that are linked through the animation controller to create a seamless, good-looking and smooth final animation. There needs to be a lot of emphasis on various conditions and situations where animations have to react quickly to player input and forces applied to the player models.

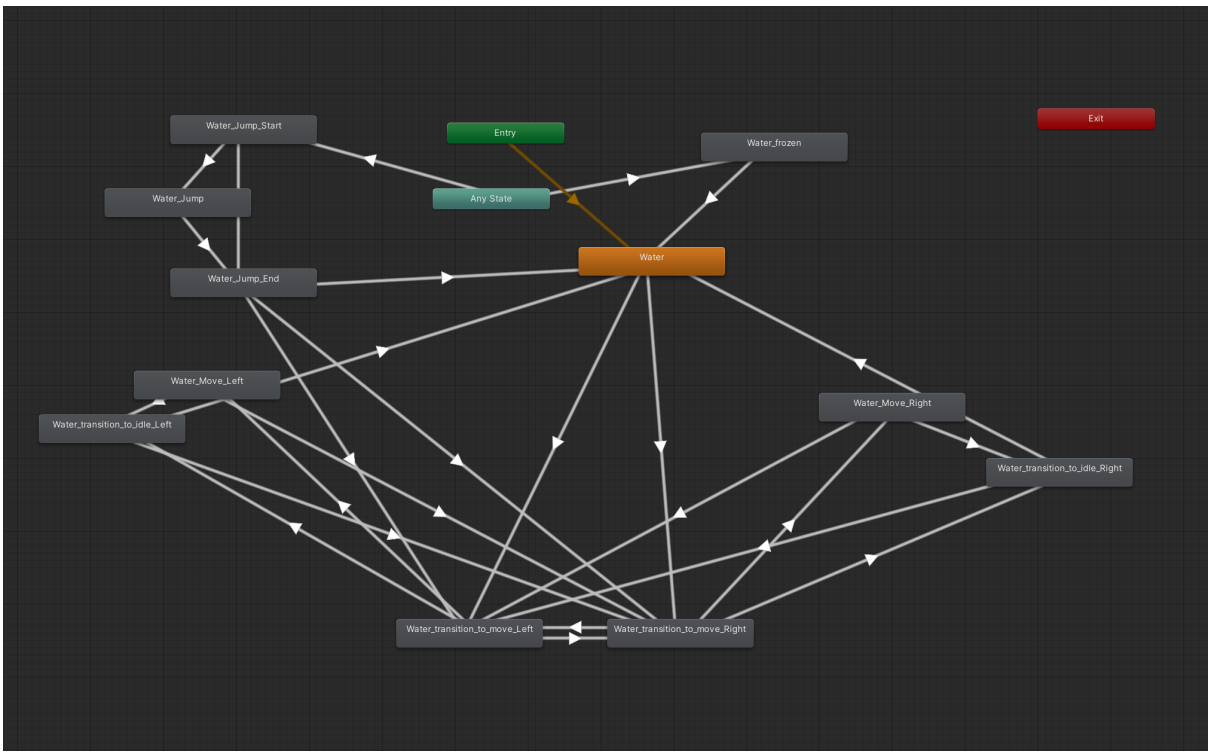


Figure 8.1. Water animation controller

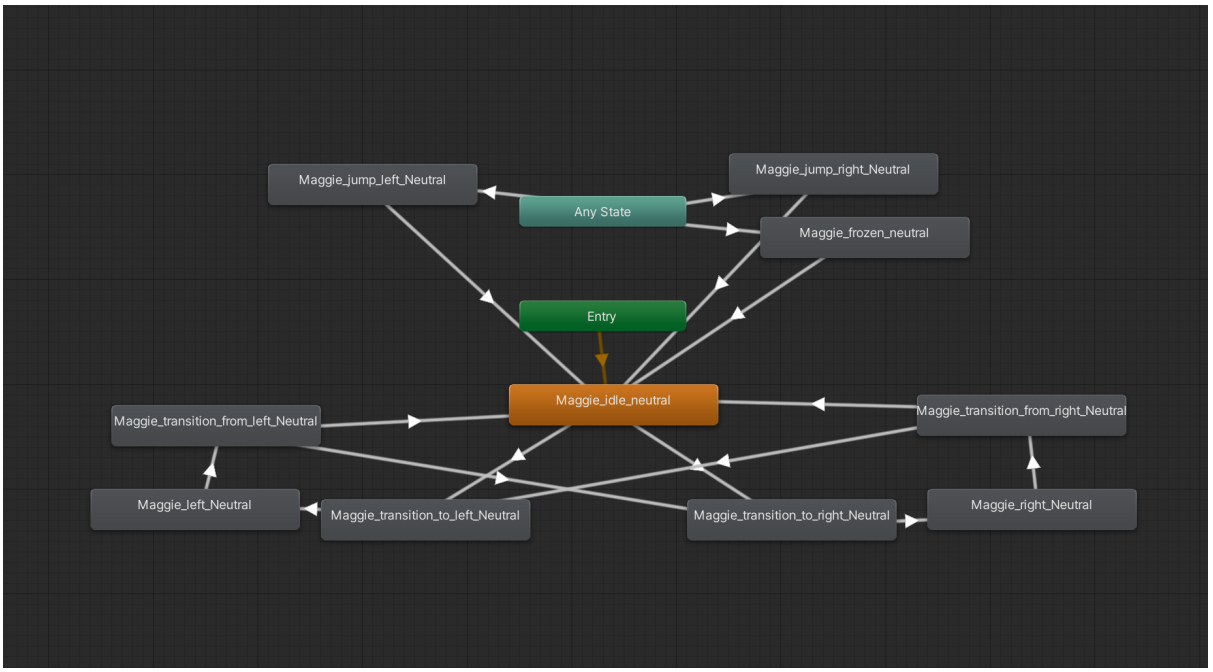


Figure 8.2. Maggie fish animation controller

8.2 Visual Studio

Visual Studio (VS) is an Integrated Development Environment by Microsoft that we used to code our scripts for Polarfish. If our team gets larger than 5 (we currently have 4 members), we may be required to upgrade from VS Community to VS Professional, which costs $45 per month. But for now the free version should be usable.[23, 24, 25]

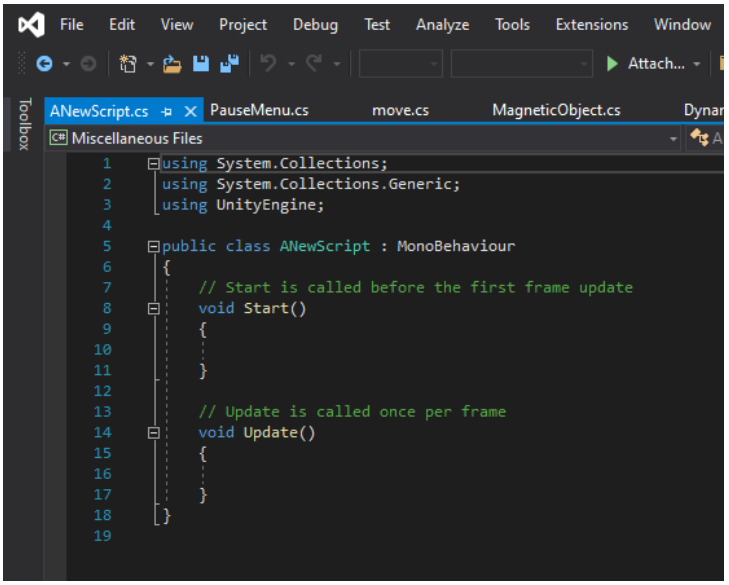


Figure 8.3. A new script in Visual Studio.

A new script created in Unity comes prepared with a few lines of code, allowing us to quickly get started. While working in Visual Studio, programmers would frequently consult the Unity API.

8.3 Aseprite

All visuals and sprites were created using Aseprite, due to its optimized workflow for pixel art and spritesheet/tilemap creation. For tilemap creation, Aseprite v.1.3 beta was used, as it contains new features that speed up the process of creating tilemaps, namely the different tile creation modes, which enable editing of new tiles, creating new tiles based on previous tiles and quick copying and checking of whether tiles match up.

[23] Visual Studio: https://visualstudio.microsoft.com/
[24] Visual Studio pricing: https://visualstudio.microsoft.com/vs/pricing/?tab=business
[25] Visual Studio license: https://visualstudio.microsoft.com/license-terms/mlt031819/

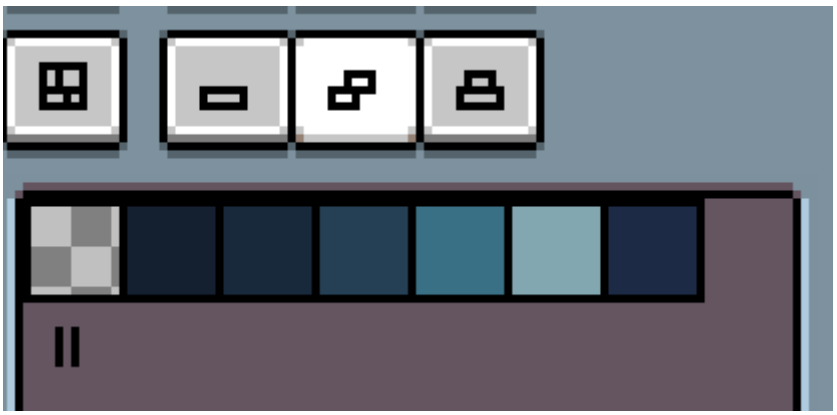


Figure 8.4. **Aseprite - tile map editing modes**

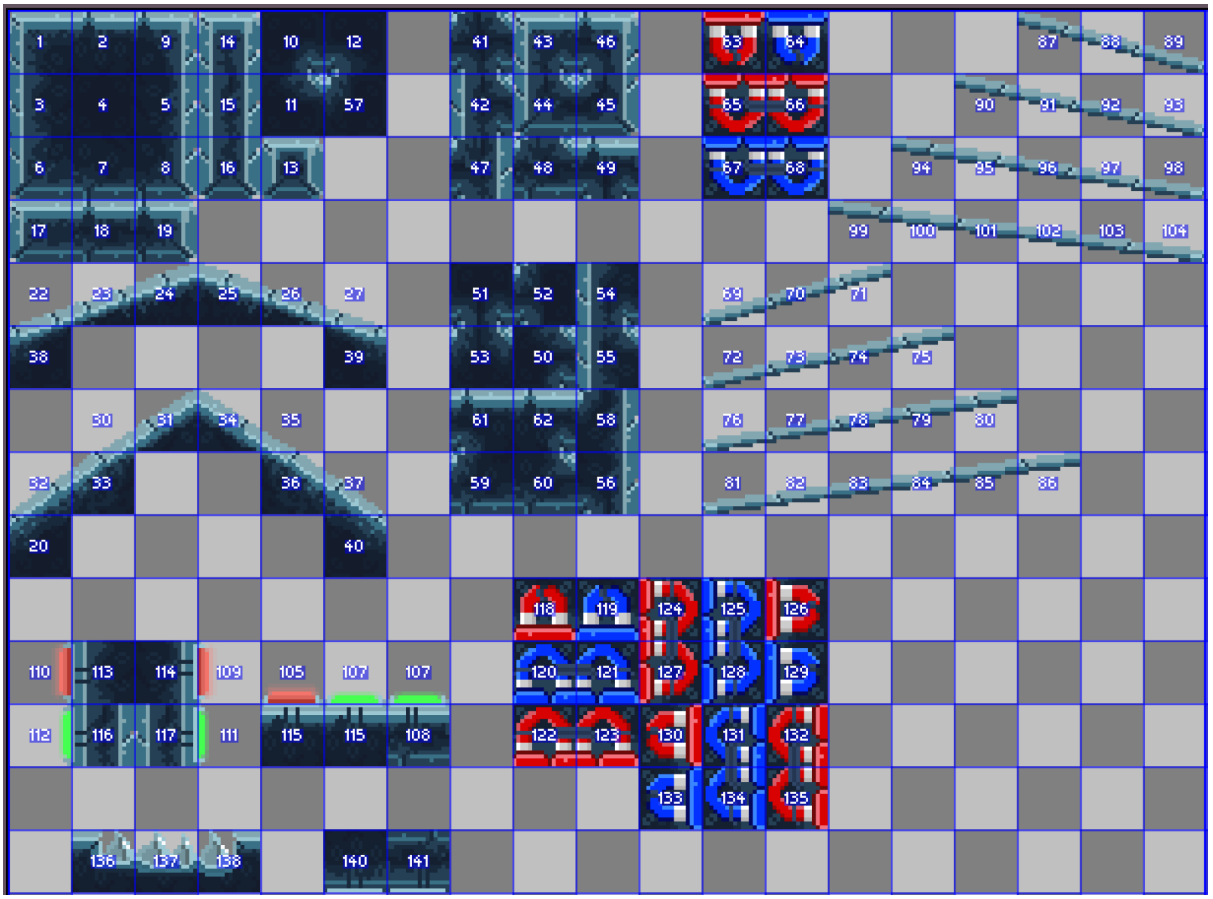


Figure 8.5. Aseprite - automatic detection of unique tiles

8.4 LDtk[26] and Tiled[27] for map creation

For level creation, the Level Designer Toolkit (LDtk) was used, as it offers an intuitive workflow once set up correctly. The main strength of the software are the rule tiles, which let the users determine which tiles should automatically be inserted based on pre-specified rules (See below). The entire workflow after correctly setting up the rules consists of using different "inks" to draw the map, and the correct tiles get automatically substituted into the map.

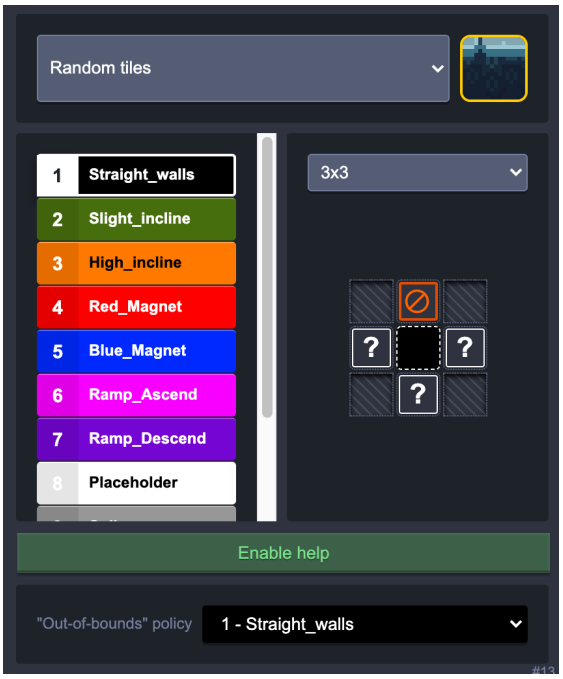


Figure 8.6. LDtk - defining a rule for the regular terrain with top-side walking (Translation of the visual rules - this tile will be substituted when the "Straight_Walls" ink is used, and there is nothing on top of the tile, and there are tiles on all other sides of the tile)

New tilesets can be added and their rules can be easily defined if they are created as "Auto-layers".

[26] https://ldtk.io
[27] https://www.mapeditor.org

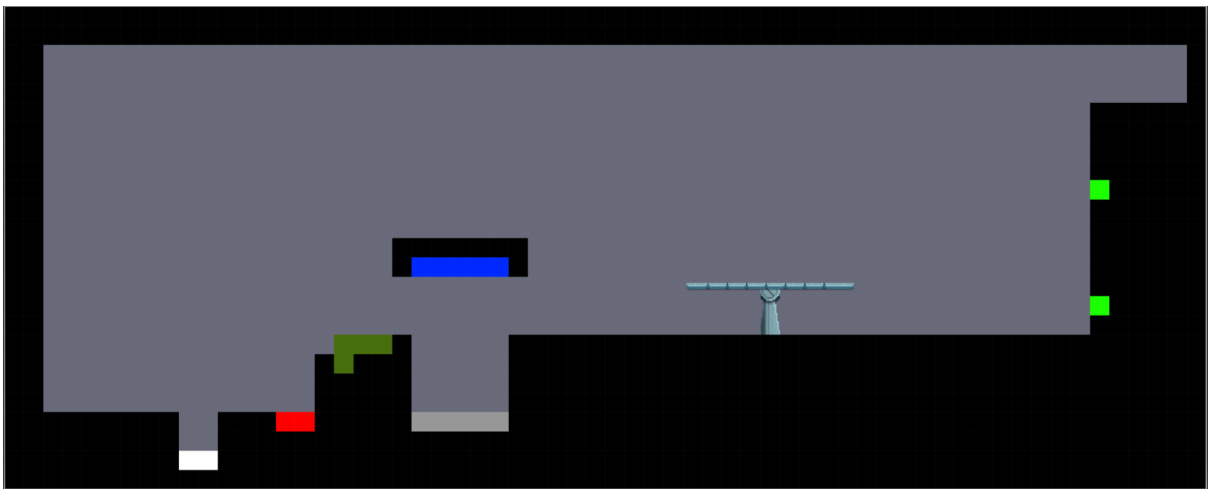


Figure 8.7. LDtk - using different inks (black for basic terrain, blue/red for magnetic boosters based on polarity, dark green for slight incline, light blue for buttons, gray for spikes and white for placeholders - used for level-specific components)

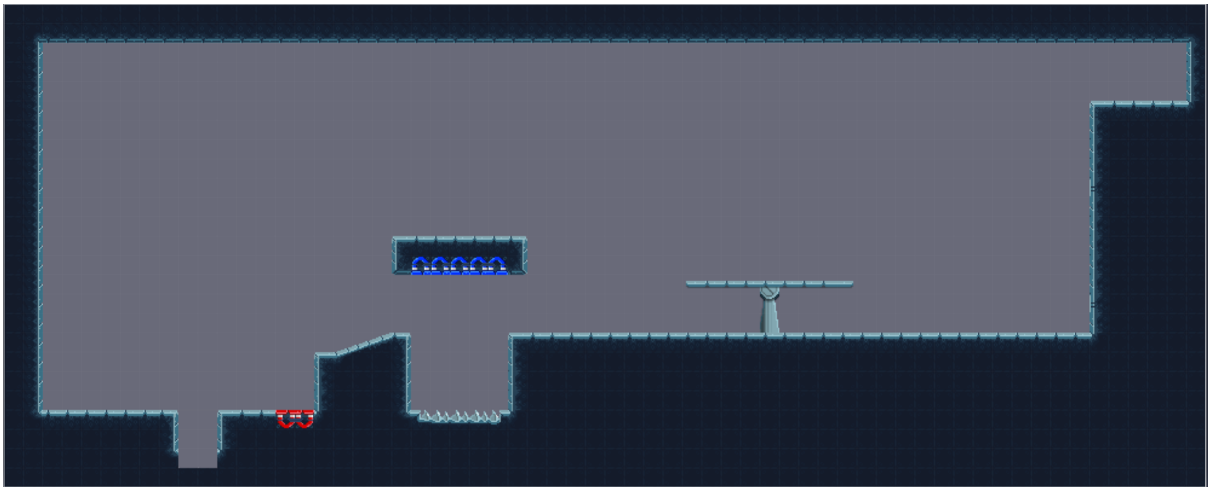


Figure 8.8. LDtk - the result of the previous image, after the tile rules are applied

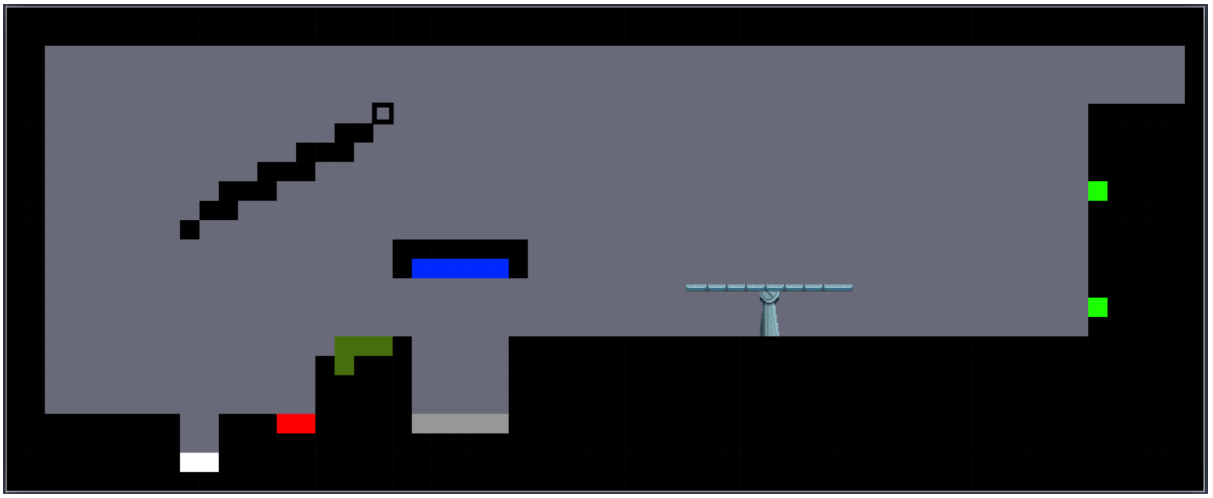


Figure 8.9. LDtk - workflow of drawing with the inks to create the map

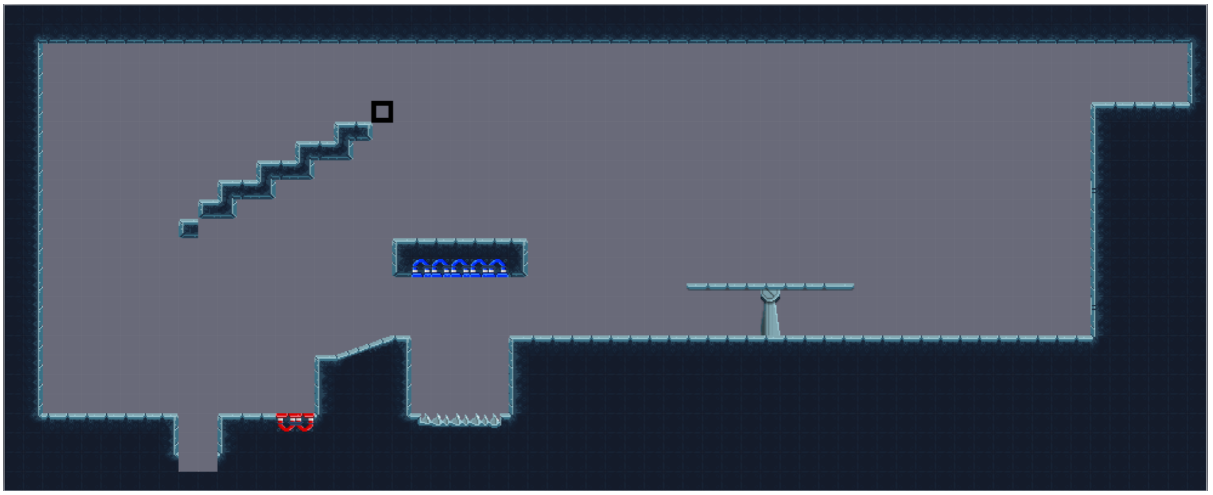


Figure 8.10. LDtk - real-time application of the pre-defined rules when using the inks

After creating the maps, LDtk offers the option to create both png files and TMX tiled files (in the settings) when saving. This is used so that Tiled can then be used for defining colliders before importing to Unity. One drawback is that it is important to be careful to create a local copy of the TXS tile map file used by Tiled for each specific tilemap, as it gets overwritten with each save in LDtk, and with it all defined colliders disappear. Consequently, after defining colliders and saving the Tiled TSX tile map, an additional copy should be saved in some other location of the project, and simply be copied over whenever a change in LDtk is made.

The workflow of Tiled collider creation is quite simple. After opening the TMX file generated by LDtk, a TSX of the embedded tile map should be exported, and any changes should be made on the new file. The colliders are easily defined with arbitrary shapes, but also with vector-based drawing.

Lastly, after all colliders are defined, SuperTiled To Unity[28] software is used to import Tiled files into Unity. This is a Unity package that is simply installed in the Unity project, and then any TMX Tiled file can be added directly in the Unity project as a prefab, with the correct tiles and colliders.

[28] https://seanba.itch.io/supertiled2unity?download

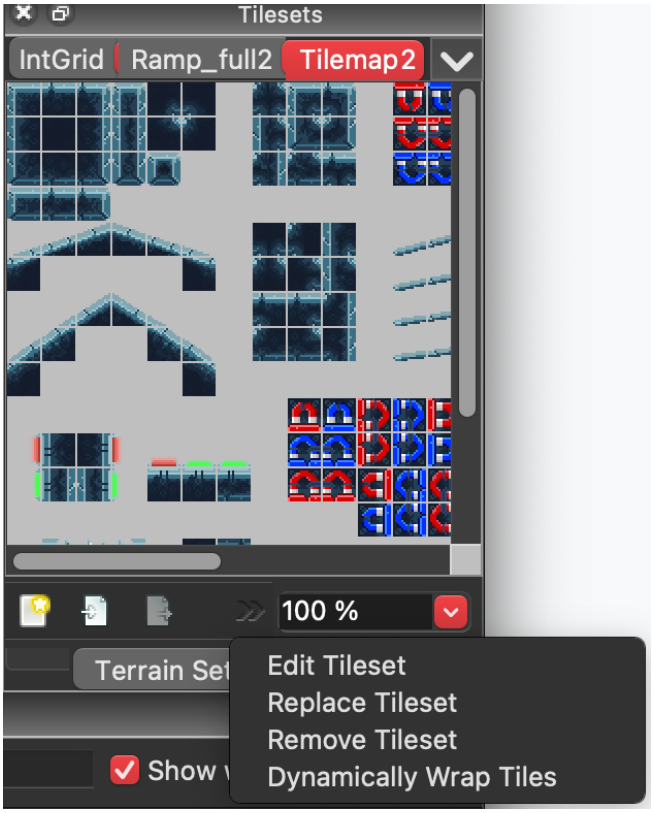


Figure 8.11. Tiled – for creating colliders, a specific tileset needs to be accessed through "Edit Tileset", after which each tile can be selected and it's collider can be defined

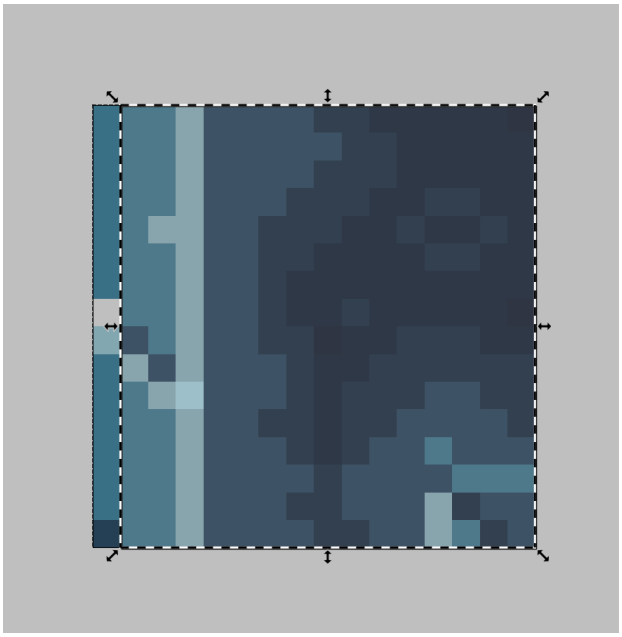


Figure 8.12. Tiled – defining colliders is done through drawing shapes

8.5 Audio software

For the sound effects and music, two different digital audio workstations (DAW) were used: Audacity and Ableton. Audacity is a free, simple audio editor that allows the user to work with audio with a limited amount of features. Audacity was primarily used to slice recorded sound effects and to give them better EQ and effects. Ableton is a more advanced DAW than Audacity and was used for the music.

8.6 Wix

The website was made through WiX, which is a tool for quick and easy website making in a visual way. One of the templates provided by the website was used, and the visual elements were redesigned into a new appearance. We used the free server plan of the platform. Therefore, an advertisement of WiX on the top is displayed and a domain name should be set including ‘wixsite.com’. The sections of the website were divided into Home, About, Gallery, Demo and Team. On the Home page, it shows the logo and the video. The About page tells the basic introduction. The screenshots of the game can be found on the Gallery page. The webGL was uploaded on Simmer.io (a platform for sharing WebGl games online), and then it was inserted into the Demo Page, so players can experience it online. The team page shows photos and roles of the team. People can always download the game with the button at the bottom.

9. Business Case

9.1 Executive summary

Polarfish is a local multiplayer cooperative platformer-puzzle game, revolving around magnetic fish.

The recent trends applicable to Polarfish are: co-op, local multiplayer, pixel art and challenging gameplay. Some recently successful games that have shown a selection of these trends are It Takes Two, Cuphead and most indie pixel art games, e.g. Vampire Survivors and Terraria. Polarfish could also benefit from its challenging aspect by appealing to the speedrunning community.

The target audience for Polarfish consists of a wide range of age groups (12+), due to its fun and easy-to-learn game mechanic and family friendly style. Typical users play on a console, spending 1-2 hours on the game, several times per week. The users of the target group typically spend 10-20 $ on a game similar to Polarfish.

There are several similar games in the market, four of them were picked as our competitors, including It Takes Two, Pico Park, I am Fish and Teslagrad. These games are somehow related to our theme so that we have examples to learn from, like how the gameplay and cooperation were designed. In the following section, they are introduced with more details including the sales information.

Polarfish will be marketed knowing that it is an obscure title by a new and small company. It will therefore be sold as a fairly cheap one time purchase, with some potential room for paid extensions later on. It is vital that the game is given some initial exposure, so the game should be showcased by streamers or other influencers. Additionally, a free demo ought to be provided, so that customers see that the game is worth its price tag.

9.2 Target sectors

9.2.1 Trends

In recent years co-op, and local multiplayer games have been getting more traction. Most notable is the success of Hazelight's 2021 game It Takes Two, which is strictly co-op. The game was a massive success for both critics and the public and has won multiple awards, such as the Game of the Year at The Game Awards 2021. One interesting feature the game offers is the Friend Pass. The Friend Pass allows a person that doesn't own the game, to play with a friend that owns the game for free.

Cuphead is another game that fulfills several trends that apply to Polarfish. The game is described as a classic run and gun game with a focus on boss battles, which at first glance does not sound similar to Polarfish at all. It is not co-op only, but it allows the players to have a fun time playing the full game in a local co-op manner. Cuphead also has a retro look in the form of old cartoon visuals, similar to very early cartoon animations from the 1930s. Furthermore, Cuphead is a challenging indie game as well. It is quite niche, but shows that there is a demand for challenging indie games with charming retro graphics and the ability to play with a friend.

Pixel art as an artform might remind you of the first games ever created, rather than the trendiest latest games. However, there seems to have been a resurgence within the indie scene, where the art style is popular among game creators and users alike. A very recent success story is Vampire Survivors which had a peak of upwards 77 000 players in February 2022. Other popular and acclaimed games that come to mind are Dead Cells, Stardew Valley, Terraria, and The Binding of Isaac, among others. Choosing pixel art makes sense for indie developers, because of its simplicity; however, judging from the success of these games, the aesthetics of this style are genuinely appealing to consumers.

Polarfish is quite a challenging game. It is relatively easy to learn, but hard to master. The mastery of such a game could be seen as impressive and encourage people to challenge themselves with the game or speedrun it. Speedrunning is a subsection of the gaming community that consists of a smaller group of dedicated fans. Most speedruns are usually done single-player, but there are some games where co-op speedrunning has been popularized. One example is Portal 2, where a third of the recorded speed runs for the game on “speedrun.com” are co-op runs. Speedrunning allows games that should have been “dead” years ago to increase their longevity. One of the most popular games to speedrun is Super Mario 64, a game released in 1996. Fostering and allowing the creation of a tight-knit group of speedrunners for Polarfish could improve the traction the game has and grow the game's community.

9.2.2 Target audience / Main actors

Figure. 9.1. User persona

The target audience for Polarfish is quite broad. The main genres our target audience is interested in are puzzles, logical games and cooperative games. It aims to appeal to both casual players and more experienced and skilled gamers. The aim is to bring friends, couples, and siblings closer together and make them communicate with each other as they try to solve various cooperative puzzles. The age group consists of 12-year-olds and older - as we aim to create a game that is not impossible or too frustrating to complete, but still challenging enough to be fun for kids, teens and adults alike. The typical playing sessions for such a game and audience consist of 1-2 hours at a time, a few times per week, but can also be stretched out into longer sessions, less frequently.

The main gaming motivations of our target group are being social - experiencing a bonding, fun moment with a close person; feeling an achievement - getting through a tricky task and succeeding in a satisfying way; and mastery - having the feeling of being competent as the player improves.

Lastly, users of our target group prefer to purchase games like Polarfish once, rather than spend more money through micro-transactions or subscriptions. This is also in line with the nature of the game, as it is a 2D couch-multiplayer platformer with a short plot lacking significant replayability.

9.3 Competitor Analysis[29]

It Takes Two - 5 million units sold (Feb 04, 2022)
It Takes Two is a very popular and successful 3D co-op game. It is a puzzle-solving adventure game as well. Players must learn to adapt to new game skills that are complementary, and only by thinking on both sides of the function is it possible to solve puzzles and defeat bosses. It has rich gameplay design, excellent plot design, cooperation-centered, and low-threshold game difficulty so it has won great popularity among the public, which is a game we aim at making.

Figure 9.2. Store Data of It Takes Two

Pico park - exceed 1 million (Oct 7, 2021)
Pico park is a 2D cooperative puzzle-solving game that can be played by 2-10 players simultaneously. Each player controls a cat and must work together to solve the challenges in each level. The main solution to these challenges is stacking one on top of the other. The game looks easy and simple, which is where the inspiration comes from. It tests the cooperation of teammates very much. A careless operation by one person will lead to the annihilation of the entire team. The more team members it has, the more challenging it will be.

Figure 9.3. Store Data of Pico Park

[29] https://www.vgchartz.com/; https://steamdb.info/

I am Fish - unknown sales number
I Am Fish is a single-player, physics-based, easy and fun 3D adventure game. The main characters are four cute and fearless little fishes. After being forced to separate from their homes in a pet store fish tank. They each swim, leap, roll and nibble their way along in an effort to reunite in the warmth of the ocean. This provided the background inspiration for us since being a fish rolling in the fish ball looks interesting, which we think has the potential to be a two-player game.

Figure 9.4. Store Data of I Am Fish

Teslagrad - 136,460 sold on PC (Mar 30, 2019)
Teslagrad is a niche single-player platformer game with simple puzzles. It uses magnetic polarity as their mechanics which is similar to ours, but they focus on operating the objects in the environment instead of the magnet of players. Though the puzzles are not that difficult, the skill demand is high.

Figure 9.5. Store Data of Teslagrad

9.4 Marketing Plan

The game will be sold as a one time purchase for no more than 200 Swedish Crowns. Our game has a low budget, and only 6 hours long, but we would make sure that those 6 hours are enjoyable all the way through, so we feel that this price is reasonable.

In order to promote the game, we intend to contact a few gaming influencers who might be willing to try our game. We cannot afford the attention of any larger individuals, but there are plenty of smaller personalities who might be willing to give Polarfish a go.

Perhaps the greater issue is that, since Polarfish is a coop game, we would effectively need to contact influencers in pairs. It would be easiest if we could find ones that work in groups on a daily basis.

Additionally, we should avoid those who specialize in puzzle games, and those that would tend to produce more than 2 hours of content. Such videos would ruin the puzzles for the viewers, so very few would be willing to buy the product afterwards.

Finally, to market the game, for those whose attention we manage to catch, we should provide a free demo spanning the tutorial and a selection of proper levels. There is a significant risk that potential customers refrain from making a purchase due to a lack of trust in smaller games companies. Providing a free demo should encourage such customers to at least give us a chance to impress them.

If Polarfish manages to sell well, there is room for extra content in the form of paid DLCs. Suggestions include: New levels or environments, A 3-player mode with its own set of levels, and a level editor for the players to use. However, none of these ideas are currently being explored, they are merely observations of areas where the game could be easily extended.

10. Team

10.1. Adam

Adam was responsible for the entire visual design of the game - ranging from the player characters through the game banner to the levels, tilemaps and obstacles. Furthermore, player and environment animations were implemented by Adam in Unity. He drew all pixel art using Aseprite, and made levels ready for Unity through LDtk and Tiled.

Additional responsibilities included video production and editing and project management.

10.2. David

David was mainly responsible for implementing the various game mechanics, everything in the game world that operates according to a script, really. This meant that most of his time was spent programming, often in a pair with Jonas, or rigging scripts in levels. Ultimately, there wasn't enough time for David to implement all mechanics on his own, so Jonas handled some of them. David also handled a significant portion of the public playtesting sessions, writing down feedback and talking to players. Sometimes he would even alter level design, usually when the original idea didn't work as well as intended, and needed some minor modification.

10.3. Jiatong

Jiatong was mainly in charge of the basic design of the levels. She gave several sketches of the levels and then updated them if problems were found, which provided the foundation for the map-making. She also took the responsibility of the website making of the game using the WiX platform. She helped a bit on the sound effect and the buttons on the start menu.

10.4. Jonas

Jonas had two main areas to work with: general programming and everything sound-related in the game. He worked together with David and on his own to implement many of the mechanics in the game. Most of the sound effects (except the ones by Jiatong) were created and mixed by Jonas and implemented into the game. He also did some menu work, such as creating the level selection and pause menus.

Appendices

Link to website (includes demo video, and playable demo)
Link to the final presentation
Paste all central scripts referenced here (As image and link to .txt document on drive)

Scripts

move.cs
https://drive.google.com/file/d/1C2Y6KB2OatGOQBomAKg-IAfe5bco8JGN/view?usp=sharing

MagneticManager.cs
https://drive.google.com/file/d/1RPPzDvD6YmJK_z1iG3J-Zhv3cz_hY1kD/view?usp=sharing

MagneticObject.cs
https://drive.google.com/file/d/1hJo7z9pD4MsnkeasrXrK5WnAWnxO2UmN/view?usp=sharing

DynamicMagneticObject.cs
https://drive.google.com/file/d/14NZJwIfmm5ERyljw7Yu-Z1RZNjAHi_iy/view?usp=sharing

Checkpoint.cs
https://drive.google.com/file/d/1ad_usYTO7yNyIZ_xkEt9secEXThxthU_/view?usp=sharing

ExitPortal.cs
https://drive.google.com/file/d/1LXPMp-D9TNO7yvkepz941JTA9Xqe8Rmi/view?usp=sharing

Freezer.cs
https://drive.google.com/file/d/1wiFVxOqR0AmMhCkCkVngTaGRc5fDm8Dk/view?usp=sharing

MovingPlatform.cs
https://drive.google.com/file/d/1sRIz706qowgiUk8ghrha3svTtCb-dpHs/view?usp=sharing

PolarityLock.cs
https://drive.google.com/file/d/1UrLtAB_UEPsLVxOKHsInSeOseIF3i_BS/view?usp=sharing

Respawn.cs
https://drive.google.com/file/d/1-g_67Uwhc4kHOSch58lWNFslnpRLAqAN/view?usp=sharing

Button scripts
https://drive.google.com/drive/folders/1xx56apFYRWgaKlGF9lj1_gr5HnHJ_wIq?usp=sharing