[Orbital 2024] 6016-FFS

SPIRIT DASHING

README/DOCUMENTATION

Peter Hadi Wijaya, A0281232N, Year 1 Computer Science
Ulrico Nolan Orlando, A0281284B, Year 1 Computer Science

List of Content

Getting Started

Windows

1. Download the Zip file through the link at the end of this document.

2. Right-click and choose ‘Extract All’ to extract the content to a folder of your choice

3. Open/Double-click on the folder and look for Spirit Dashing.exe

4. Open/Double-click Spirit Dashing.exe to enjoy the game :)

Other OS

1. Download Godot Version 4.1.1-stable from https://godotengine.org/download/archive/ for your respective Operating System.

2. Sign in to Github and open the repository link below: https://github.com/PeterHW963/ffs-game-clone

3. Fork/clone this repository into your device

4. Open your Godot app and click ‘Import’

5. Navigate to the repository folder and open the file named ‘project.godot’ within the folder.

6. Click the play button on the top right corner of your screen or press F5 to run the game. Enjoy :)

Overview

Motivation

Growing up with video games, we grew to love and cherish their presence in our lives. Through this Orbital project, we aim to flip the script in one of the world's most popular video games: Geometry Dash.

Growing up in Indonesia, we were given too much time and the cheapest past-time is to play video games readily available through Facebook, games.com, and eventually, the App Store. The games we played had a long-lasting and significant impact in our intellectual growth, along with our core personalities. It became our mission to create 'permainan yang bermutu' which translates to 'quality games' that can positively impact players all over the world.

Creating video games is also a window to the souls of the youths who play them. We strongly believe that the mindset we need to have when addressing video game addiction in Singaporeans (old and young) is not by removing, penalising, or restricting access, but by creating more soulful and meaningful games that induce critical and creative thinking. The objective should not be to ban pencils simply because some bad apples chose to use it to write profanities, but to educate the masses of the power held in their hands, and above all, encourage them to use it to create poetry, solve equations, and take us to the moon.

On top of that, inspired by significant historical developments of video game algorithm design from Pac-Man to Rain World, we wish to venture further into developing our own versions of deceivingly intelligent Non-Playable Characters. Many video game studios have achieved huge successes due to their unique perspectives on the potential of algorithms inside and outside of games. Through our project, we too wish to develop our own unique understanding of the topic and eventually, bring this experience with us as we start working in relevant fields.

Putting in countless hours to make the base game will also push us to fully master the whole process of making a video game from ideation to release. Additionally, coming up with algorithms to beat our game gives us first hand experience in diving deeper into the world of algorithms and thinking creatively as computer scientists.

Requirements

Target Level of Achievement: **Apollo 11**

Features: 6-8 features of sufficient complexity
Planning/Version Control: Github issues with (monthly) milestones / labels / tags / assignee + Intermediate version control (branching, pull request)
Design: Quality design diagrams and decisions
Implementation: Coding standard for main build
Testing: Multi-level (unit / integration / system) testing with automation + User testing
Proper test strategy (planning / test case design)
Documentation: Detailed description of features, quantification of complexity and justifications of design which surmounts to 30+ pages of documentation
Peer Evaluation Average: 4

User Stories

User Portfolio	Scenarios
Casual Gamer Looking to Unwind	As a casual gamer who is just looking for a fun and relaxing game, I want to be able to enjoy an immersive and interesting game that can help me pass the time.
	As a casual gamer who wants to do things at their own pace, I want to be able to freely make a level with intuitive controls and little to no constraint.
Experienced Gamer Looking for a Challenge	As an experienced gamer who is looking for a challenge, I want to experience challenging and difficult levels possibly made by other enthusiasts of challenging games.
	As an experienced gamer who wants to see my achievements, I want to see through all my previous attempts at a level to observe my own progress of improving.
Level-making Enthusiast	As a level-making enthusiast who wants to make custom levels to share, I want to experience an easy-to-use level editor where I can quickly create, edit, and test out my level with a robust array of enemies.
	As a level-making enthusiast who wants to share my custom levels, I want to freely share my levels with fellow enthusiasts who will test out my levels and potentially give me feedback.

SE Practices

Version Control in GitHub

Through the use of GitHub, we were able to manage the changes to the source code by taking turns in working on the code and maintaining clear and careful communication to determine who is currently working on the game. Alongside the support provided by GitHub, we were able to secure safe version control and continue to build the game together feature-by-feature.

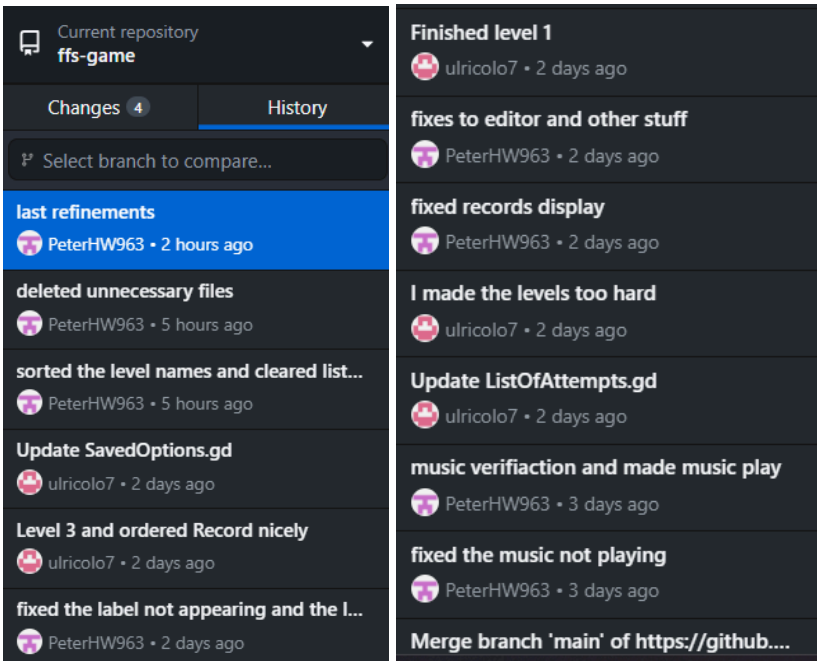


Figure 1: Project Version Control Using Github

Documentation

This document serves as a record that describes the technical capabilities of our game and all the positives to be gained by users/players of our game. These benefits can come from the myriad of features provided within the game or the overall experience of the game as a whole.

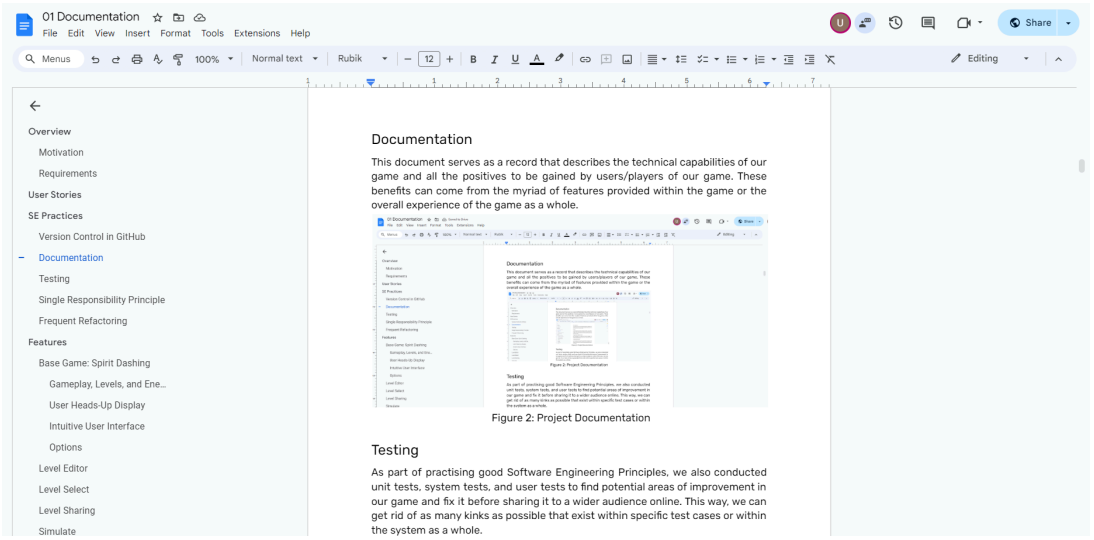


Figure 2: Project Documentation

Testing

As part of practising good Software Engineering Principles, we also conducted unit tests, system tests, and user tests to find potential areas of improvement in our game and fix it before sharing it to a wider audience online. This way, we can get rid of as many kinks as possible that exist within specific test cases or within the system as a whole.

Single Responsibility Principle

We also ensured that our code follows the Single Responsibility Principle where each script attached to its respective scene has one responsibility. The respective responsibilities generally include all the functions that will support the respective scenes/features.

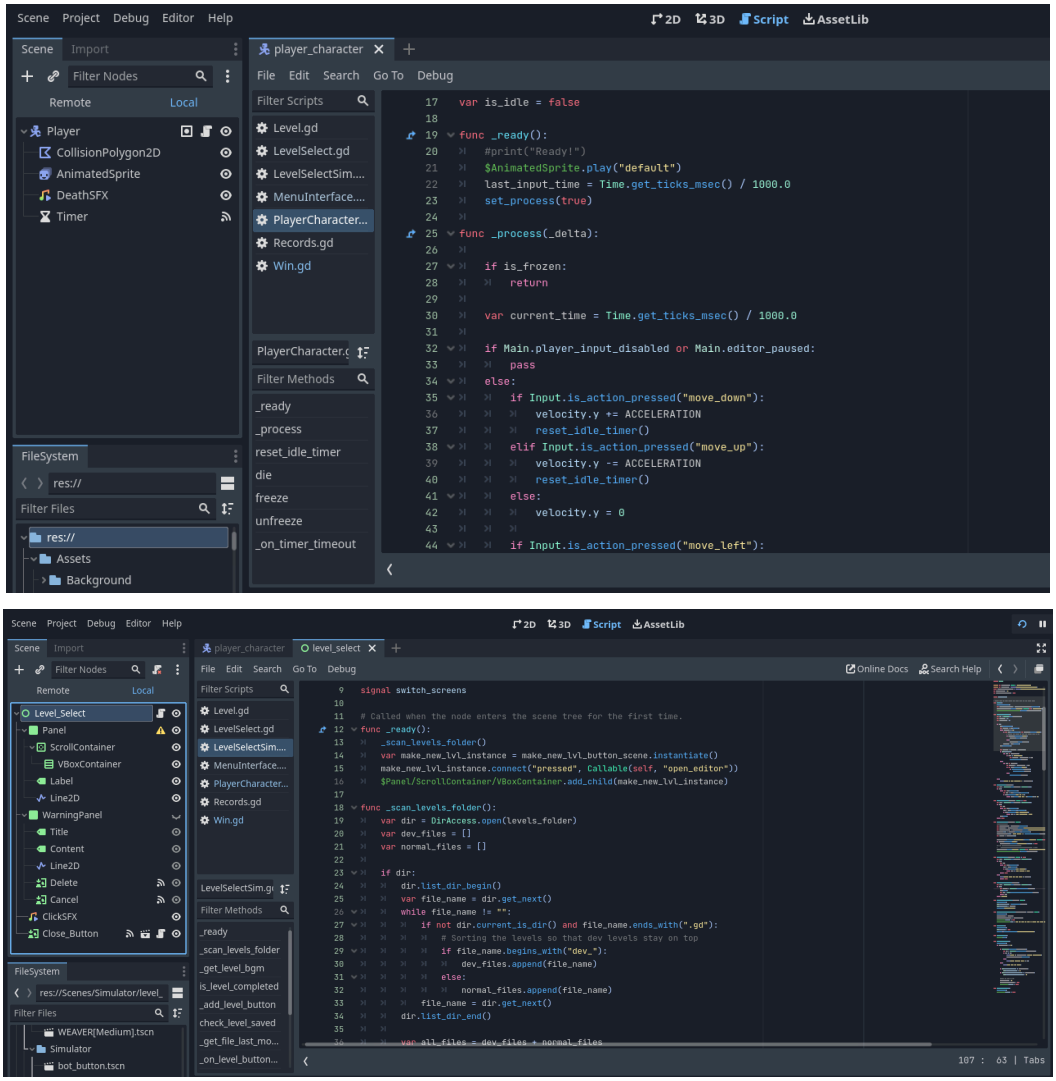


Figure 3: PlayerCharacter.gd and LevelSelectSim.gd

Frequent Refactoring

To ensure that our code works as efficiently and effectively as possible, we made sure to restructure the scripts to sustainably fit all the features we are implementing now and in the future. We also make it a good practice to remove excessive script that does not contribute to the functions to aid with efficiency and readability.

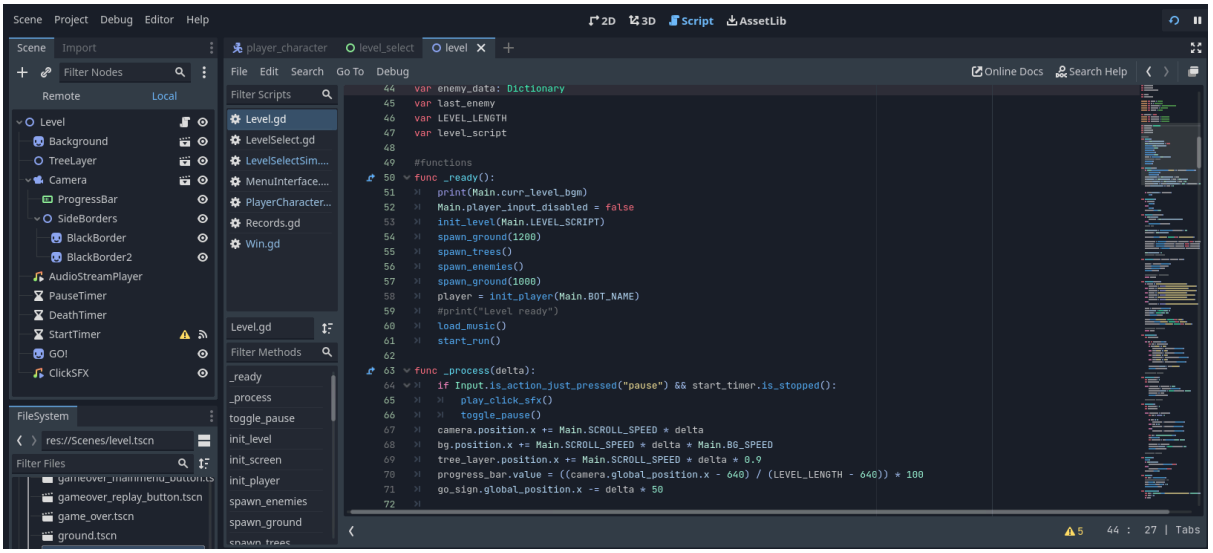


Figure 4: Level.gd

Features

Base Game: Spirit Dashing

The development of the base game itself requires us to utilise Godot, the video game-making app to its fullest extent. Generally, games are composed of three major types of components: Assets, Scripts, and Scenes.

Figure 5: Asset of Player Character, Main Menu Button, Ghaster

Assets are simply all the visual components that become the external representation of our game to the players. These components are situated and weaved nicely to form the User Interface, the immersive atmosphere of the game, and the animations of the game. The design of these assets should be well-planned as a balance is needed between visual clarity in conveying information and aesthetic immersion.

```
125
126 func spawn_enemies():
127     var enemy_instance
128
129     for idx in enemy_data.keys():
130         var data = enemy_data[idx]
131         if data["type"] == "gh":
132             enemy_instance = GhasterScene.instantiate()
133         elif data["type"] == "fl":
134             enemy_instance = FlapperScene.instantiate()
135         elif data["type"] == "cg":
136             enemy_instance = CrawlerGroundScene.instantiate()
137         elif data["type"] == "ca":
138             enemy_instance = CrawlerAirScene.instantiate()
139
140         enemy_instance.position = data["position"]
141         add_child(enemy_instance)
142
143     #print("Enemies spawned")
144
145 func spawn_ground(value):
146     var dist_covered = 0
147     var ground_instance
148
149     while dist_covered < LEVEL_LENGTH + 2000:
150         ground_instance = GroundScene.instantiate()
151         ground_instance.position = Vector2(dist_covered, 193.662)
```

Figure 6: Part of the Level.gd Script

Scripts are the backend components that allow the video game to run. Every single feature present in the game has a Script that determines what they will do. The Script is written in a language unique to Godot: GDScript. To note: the

base game automatically runs Main.gd upon launch as it holds many important in-game states that need to be accessed by the other Scripts.

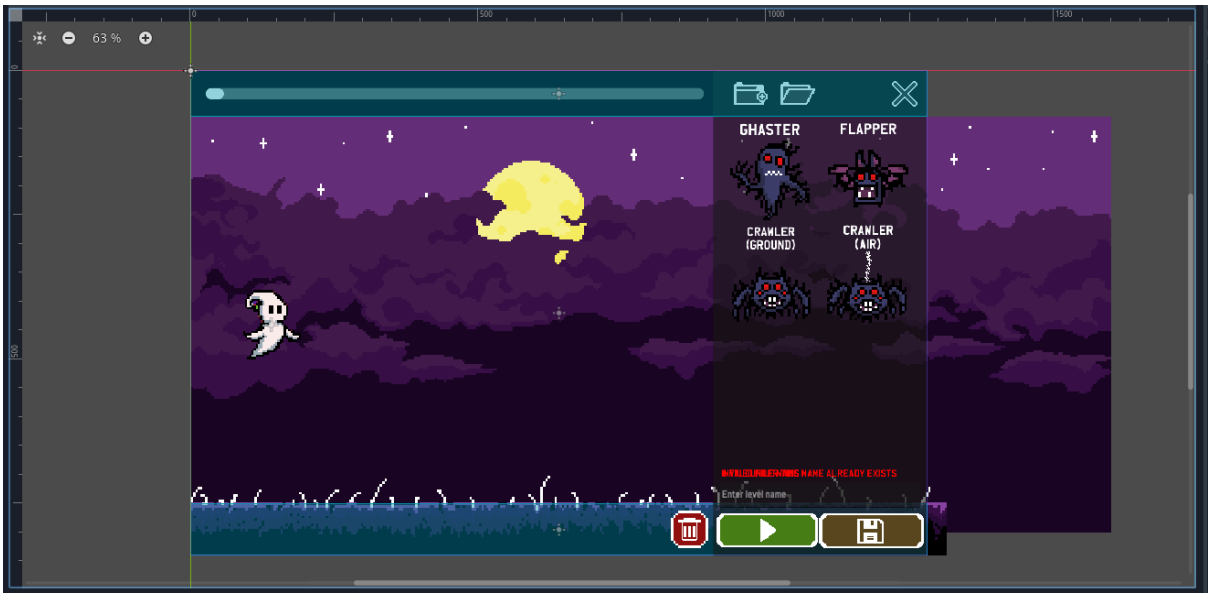


Figure 7: editor.tscn Scene

Finally, Scenes are replicable platforms that tie in both Scripts and Assets forming custom building blocks for video games. A Scene functions like a Node and each of its components have a parent-child relationship with it that allows whatever change done to the parent node to be reflected in all its children.

The foundation of the video game will contain several features we find essential for our game:

Gameplay, Levels, and Enemies

Originally designed by us, we create an immersive, challenging, and fun arcade game featuring a spirit who dashes through various intimidating enemies to reach the end! The gameplay itself revolves around the player dodging the incoming enemies by moving up, down, left, and right using the keys W, S, A, and D respectively. All the levels were intentionally designed with the enemies staying relatively stationary and the camera moving forward, creating an illusion that the enemies are coming fast towards the player. The logic is all handled within the Level.gd script.

Other than the levels that we developers created (prefixed with “dev_”), the users will also be able to create and share their own levels using our Level Editor.

Once again, the game scripts are coded within Godot using GDScript. Most of the assets used are designed, drawn, and animated using Aseprite and other graphic

editing tools such as Ms Paint. The animation is done by creating multiple frames of a single character or object's sprite and having these sets of frames looped indefinitely.

There are four kinds of enemies:

1. **Ghasters** - these ghastly phantoms have a bigger physique but a relatively straightforward move pattern.
2. **Flappers** - they are like bats but scarier. They move up and down.
3. **Crawlers (Ground)** - these arachnid monstrosities crawl on the ground and at a relatively faster speed than Ghasters.
4. **Crawlers (Air)** - these ones move up and down to and are not restricted to the ground. Still slightly faster than Ghasters.

Figure 8: Screenshot of dev_Level_4 Gameplay

User Heads-Up Display

Our Heads-Up Display (HUD) includes a progress bar that neatly and clearly displays how far has the player progressed through the level. The progress bar also automatically adapts to the length of the levels designed by users. The code is also handled within the Level.gd script where the end and step of the progress bar is calculated based on where the last enemy was placed.

Additionally, a HUD of the controls for our game will be shown at the start of every level to inform users who may not be used to playing games on a computer.

Intuitive User Interface

As per any modern video game and programme, the users will be met with an intuitive and friendly user interface that will let them navigate through the game seamlessly.

The User Interface of our game includes:

- A Main Menu - consisting of:
    - **“Play” Button** which will open the Level.tscn scene to run levels
    - **“Editor” Button** which will allow users to make custom levels
    - **“Simulate” Button** which will replace the Player with built-in bots and let them run through pre-made and custom-made levels
    - **“Options” Button** that will open a panel which includes some important in-game settings to better cater to the players
    - **“Quit” Button** that will close the game
- Level Selection Panels
- A Bot Selection Panel

This is fully implemented within Godot using the tools and settings provided in the form of Control, Texture Button, Line Edit, Label, and Panel Nodes.

Figure 9: Main Menu of the Game

Options

The main menu will also have an ‘Options’ button to help adjust audio levels and controls according to the preference of the User.

Currently, what is changeable is the audio levels of the background music and the auto-replay feature upon dying in a level. These features rely on manipulating a global variable under the Main.gd script to apply the preferences all throughout the game.

Level Editor

In addition to the base game, we implemented a level-editing feature to give players the freedom to create their own levels. This includes several unique and complex interactions between the UI and the game itself. It involves the EditorGUI.gd script that helps simulate an editor through the interface and to translate what has been designed into a playable level.

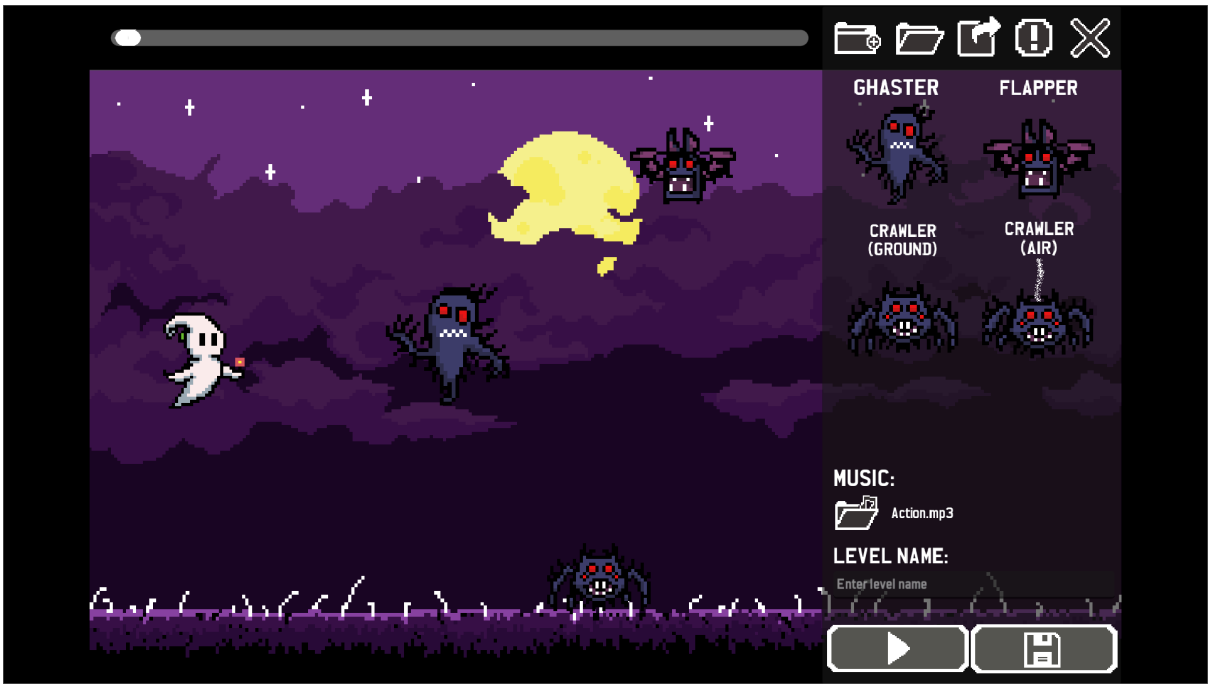


Figure 10: Level Editor

In our current implementation, we used provided tools such as FileAccess to create new files that will hold the levels and DirAccess to access these files from the perspective of a Directory.

The respective enemy icons on the right side of the editor will spawn an enemy right in the middle of the camera position. These enemies will then be draggable to set a new position and this increases the customizability of each level as the

level designers can intuitively move the enemies around according to their imagination. If a mistake was made, a delete button is also provided to remove the enemy clicked last by the level designer. In case more space is needed to place more enemies, the scroll bar at the top will allow players to fully utilise 30000 pixels worth of horizontal space to populate with our cleverly designed enemies.

After designing the level, the players can type a custom name fitting for their custom-made level. **The next step will be to press play and attempt to win the level before you can properly save and share the level with other players.**

There will be three more buttons at the top of the Editor; The first one is the Create New File button that opens a blank canvas for level designers to create a new level. Next is the Open File button that allows players to open a Level Select screen and choose a different level to edit. This level will have to be validated one more time to make sure that it is still winnable. Finally, the X is a button to close the editor and return to the Main Menu.

Level Select

Players will be given the choice to pick which level to play every time they select the Play or Simulate options in the Main Menu. These levels will initially only include those built internally through the Editor and developer built levels, but after Level Sharing is implemented, the levels displayed will also include those created by other users that have been validated.

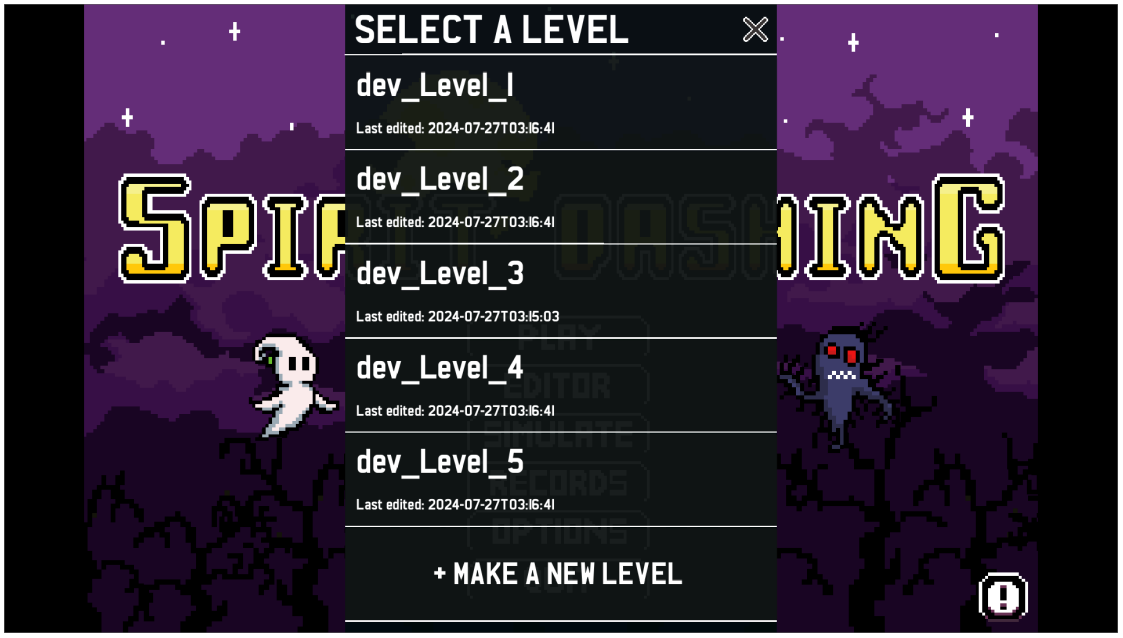


Figure 11: Level Select Screen

This involves a User Interface that will display all the available levels and a database to store the relevant information of the levels.

Level Sharing

Related to the previous feature, users will also be able to share levels designed by them with other users, vice versa. To implement this thoroughly, the game would have to have a system/script to encode and decode custom levels designed by users. Levels can only be shared (or simulated) if the level-designer can finish it themselves.

Simulate

With the intention of deepening our own understanding of algorithms and making our game more interesting, players who have designed their own levels will get a chance to deploy unique, built-in bots to act as a player in a simulation.

The bots vary in terms of skill due to the tweaking of the algorithm and/or independent decision-making system according to the intended effect of the individual bots. So far, there have been 3 bots created with their unique algorithms and flaws: MAIDEN[Easy], WEAVER[Medium], and SHAKER[Medium].

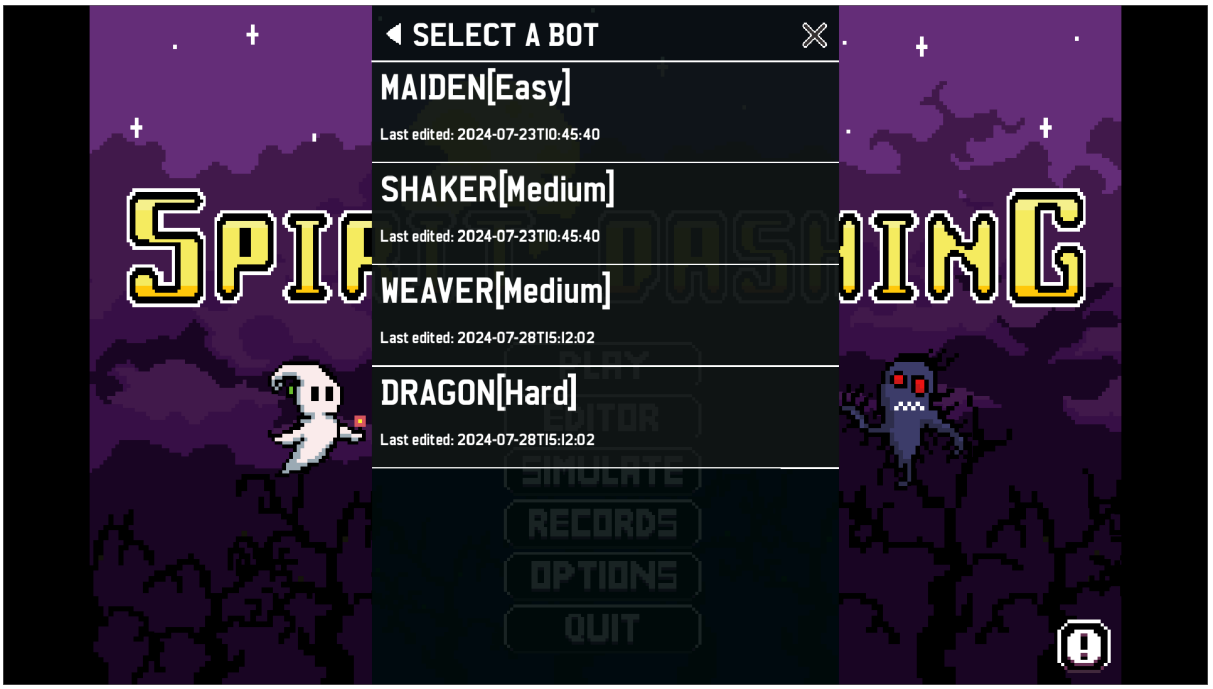


Figure 12: Bot Select Screen

MAIDEN[Easy]

The Maiden bot is as naive as it is beautiful. It simply utilises a dodging mechanic that allows it to dodge incoming enemies according to their relative positions. The bot stays at the centre and accelerates in the opposite direction of enemies that enter its scanner. This results in an elegant but naive bot that is unable to see potential obstacles beyond its immediate vicinity.

WEAVER[Medium]

The Weaver bot is created by introducing a further scanner that looks out if there are enemies approaching at a distance. This then preemptively prepares the bot to move to a favourable position first before dodging it gracefully similar to the Maiden bot. It then has a state when no immediate enemies are nearby to return to the centre region where there are more options to dodge enemies to keep it a good distance away from the top and bottom of the levels.

SHAKER[Medium]

The Shaker bot is created by adding 5 different areas that represent the different y-positions and counters in an array that will help the bot weigh which position is likely the safest position. This will help the bot reposition to the safest regions throughout the level as it still dodges the immediate dangers around it. Due to kinks in the implementation, it shakes.

DRAGON[Hard]

The Dragon bot is the latest bot to be added and it has proven to be able to solve all kinds of levels that we throw at it. It uses a combination of 7 different scanners and 3 different states (“dodging”, “adjusting”, and “returning”) to read the surrounding obstacles which helps it run smoothly across the levels. The front scanner is the primary scanner that guides the bot to know where to move. The far scanners and the surround scanner help it adjust to its surroundings. Finally, the back, head, and leg scanners help the bot stop itself before it ‘dodges’ right towards an obstacle.

Records

To improve on the quality of life features of the game, every attempt made by users or bots will be recorded locally to the server/device. This record will contain information of the user/bot that attempted, the level played, and the progress made by the player during the attempt.

Adding this feature will require us as developers to create a database to store the relevant data within the server/device the game is played in.

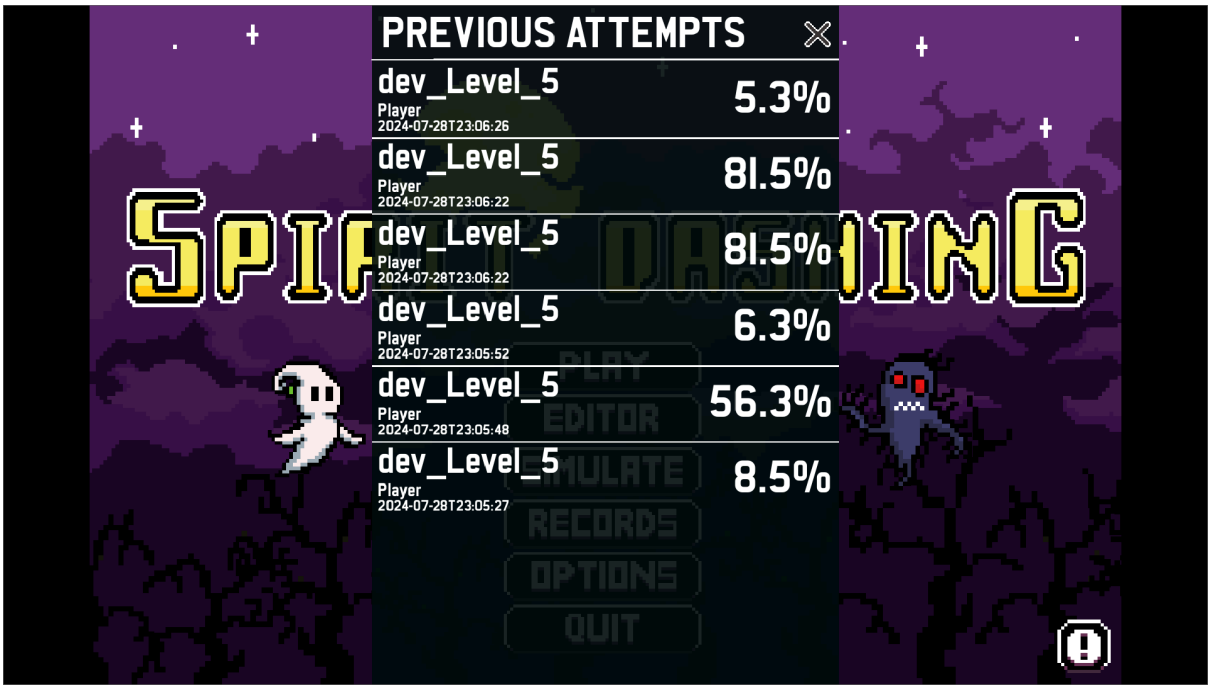
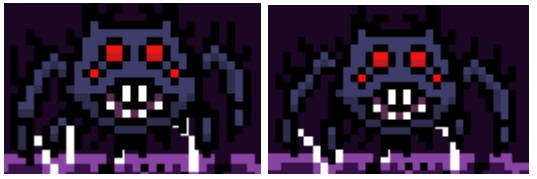


Figure 13: Records of Previous Attempts

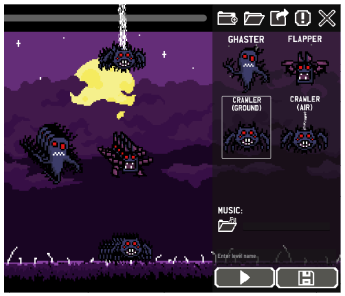
Testing

Unit Testing

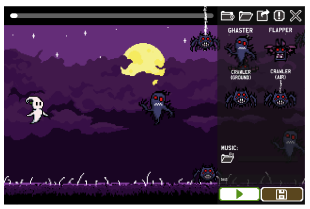
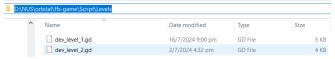
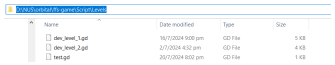
Feature	Test	Expected	Output	PASS/FAIL
Main Menu	Buttons sprite change	Different sprites between normal, hovered, toggled, and pressed states	Normal:  Hovered:  Pressed/Toggled: 	PASS
Main Menu	Clicking main menu buttons	Produces sound effect	Sound effect produced	PASS
Character sprites	Player character animation	Player sprite smoothly transitions between its (four) animated frames continuously		PASS
Character sprites	Enemy characters animation	Each enemy smoothly transitions between its animated frames continuously	Ghaster (2 frames):	PASS

			 Flapper (4 frames):  Crawler (ground) (2 frames):  Crawler (air) (2 frames): 	

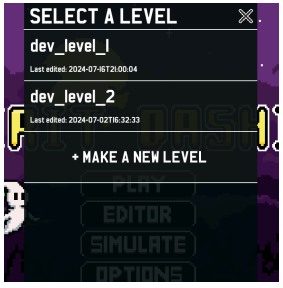
Level Select	Detecting levels in the folder	Levels in the levels select should be alphabetically ordered, except the dev levels (dev levels should be pinned on top		PASS
Editor	Delete enemy	Right clicking an enemy on the editor screen deletes the enemy	Enemy deleted upon right clicking	PASS
Editor	Drag and drop enemies	Clicking and holding the left-mouse button on an enemy instance in the editor screen allows for drag-and-drop of enemies	Enemies dragged and dropped	PASS
Editor	Drag and drop enemies	Editor panel hidden when dragging and dropping enemies	Editor panel is hidden when dragging and dropping enemies	PASS

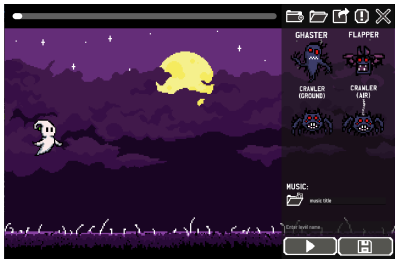
Editor	Enemy spawn location offset	Repeatedly spawning an enemy of the same type without moving any of the previous instances of that enemy type cause the newer enemies spawned to have an offset spawn location		PASS
Editor	File management button sprites	Different sprites between normal, hovered, and pressed states for all file-related editor buttons	Normal:  Hovered:  Pressed: 	PASS
Editor	Level management button sprites	Different sprites between normal, hovered, disabled, and pressed states for all level-related	Normal:  Hovered: 	PASS

		editor buttons	Pressed:  Disabled: 	
Editor	Level name line edit	Clicking on the “Level name” line edit focuses on it. and typing in the line edit changes the content of the line edit		PASS
Editor	Level name warnings	Typing a level with the prefix “dev_” causes an “Invalid File Name” warning to appear		PASS
Editor	Level name warnings	Typing a level with the same name as an existing level causes a “A file under this name already exists” warning to appear		PASS
Editor	Level management buttons	• PLAY: loads the level for user to play • SAVE: saves the level that user is	PLAY button loads the current editor level:	PASS

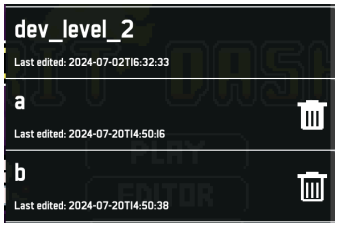
		currently editing	 ->  SAVE button saves the editor level:  ->  -> 	

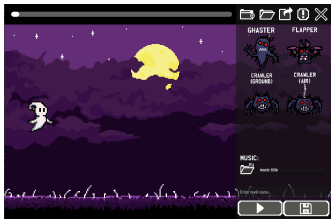
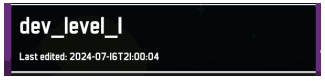
Integration Testing

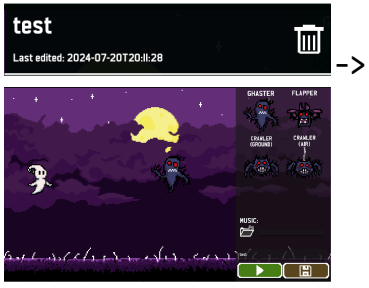
Feature	Test	Expected	Output	PASS/FAIL
Main Menu	Clicking main menu buttons	• PLAY: brings up the level select screen • EDITOR: opens the level editor • SIMULATE: opens the (simulate) level select screen • RECORDS: opens the records panel • OPTIONS: opens	 -> 	PASS

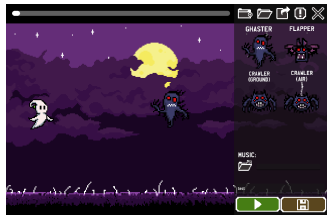
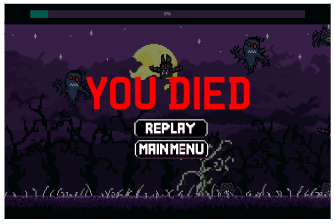
		the options panel • QUIT: closes the game • INSTRUCTIONS: opens the instructions panel	EDITOR ->  SIMULATE ->  RECORDS ->  	

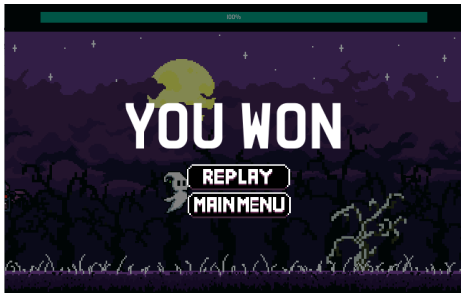
				
Game Instructions	Instructions button toggle	Clicking the instructions button when the instructions panel is visible closes the instruction panel	Instructions panel closed	PASS
Game Instructions	Clicking exit (X) button	Closes instructions panel	Instructions panel closed	PASS

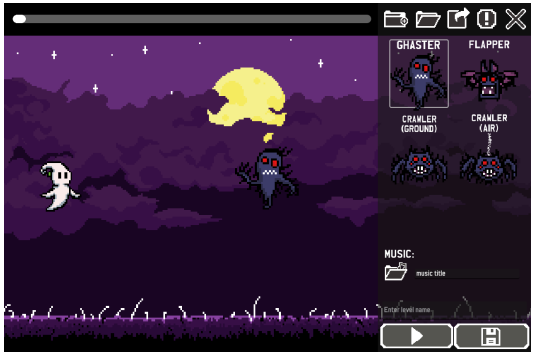
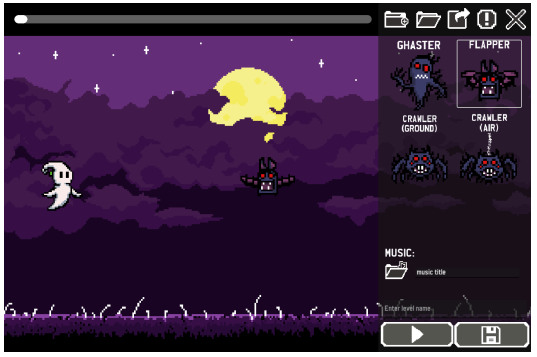
Level Select	Delete level	Clicking delete would cause a delete confirmation popup to appear	 -> 	PASS
Level Select	Delete confirmation popup	The delete confirmation popup shows the name of the file that user is about to delete		PASS
Level Select	Confirm delete button	Clicking the delete button on the delete confirmation popup removes the level from the level select screen	 -> 	PASS
Level Select	Back button	Clicking the back button on the delete confirmation popup does not change anything in the level select screen	 -> 	PASS
Level Select	Clicking “MAKE A NEW LEVEL” button	Loads the level editor	 ->	PASS

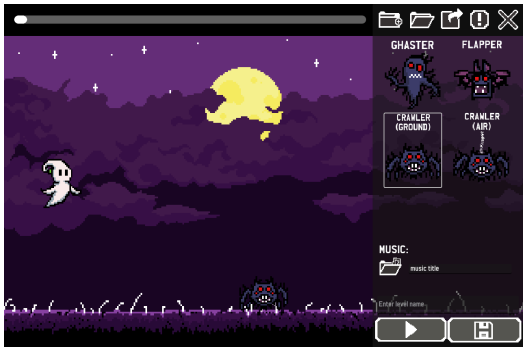
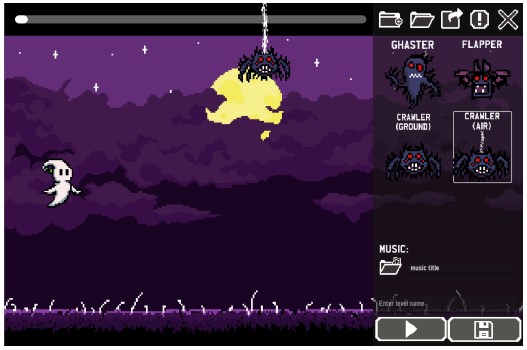
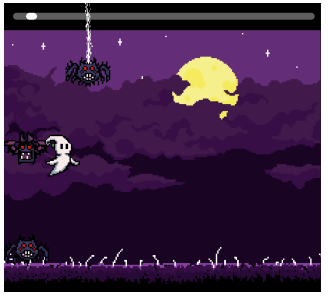
				
Level Select	Level select screen scroll	When there are too many levels, the level select screen becomes scrollable		PASS
Level Select (PLAY & SIMULATE)	Clicking exit (X) button	Returns to the main menu	 -> 	PASS
Level Select (PLAY)	Clicking level buttons	Loads the selected level and immediately starts the game	 ->	PASS

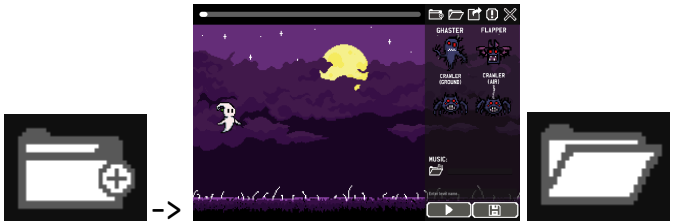
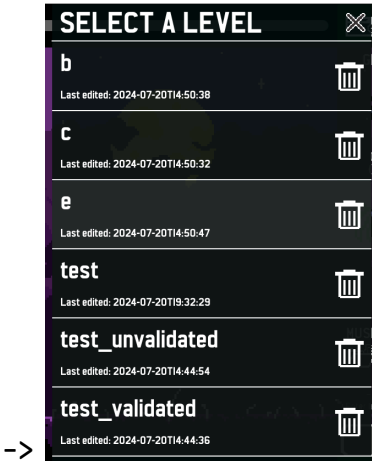
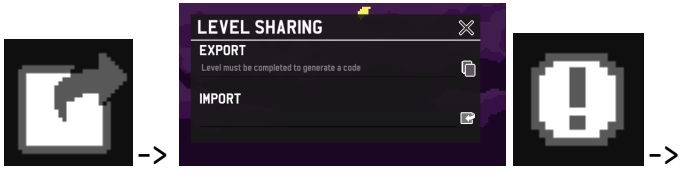
				
Level Select (SIMULATE)	Clicking level buttons	Switches to a bot select screen		PASS
Level Select (EDITOR)	Level buttons	Clicking a level button in editor loads the level into the editor		PASS

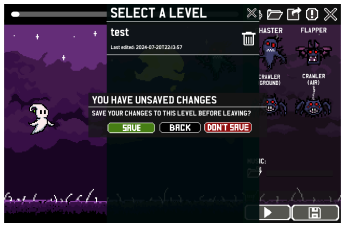
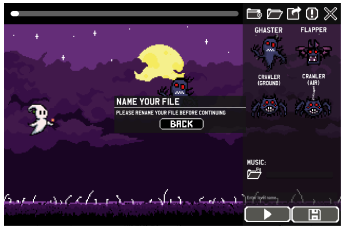
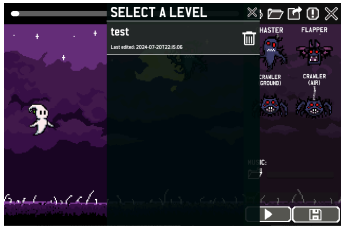
Level Select (EDITOR)	Clicking exit (X) button	Clicking the exit button closes the level select screen in the editor	 -> 	PASS
Level Gameplay (PLAY)	Movement inputs	W,A,S,D and UP. LEFT, DOWN, RIGHT causes the player character to move up, left, down and right respectively	W,A,S,D and UP. LEFT, DOWN, RIGHT causes the player character to move up, left, down and right respectively	PASS
Level Gameplay (PLAY)	Player death	Colliding with an enemy kills the player	 -> 	PASS
Level Gameplay (PLAY)	Death screen pops up	Player death causes a death screen to show		PASS

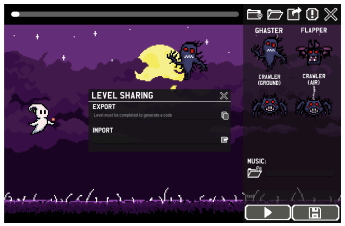
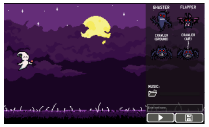
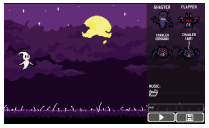
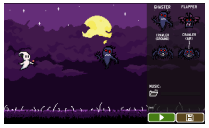
Level Gameplay (PLAY)	Victory screen pops up	Reaching the end of the level without dying causes victory screen to show		PASS
Death Screen (PLAY)	Clicking buttons on the death screen	• REPLAY: reloads the level • MAIN MENU: returns to the main menu	 ->   -> 	PASS
Victory Screen (PLAY)	Clicking buttons on the victory screen	• REPLAY: reloads the level • MAIN MENU: returns to the main menu	 -> 	PASS

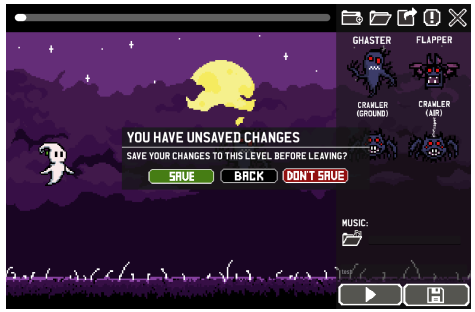
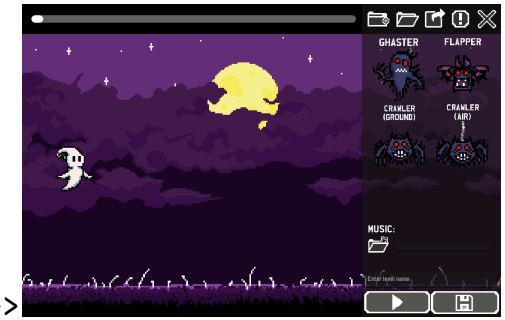
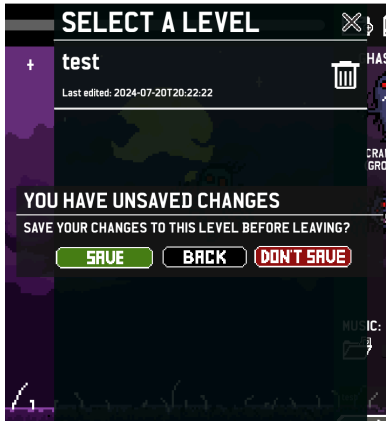
			 -> 	
Editor	Clicking enemy buttons	Corresponding enemy spawns on the editor screen	Ghaster button:  Flapper button:  Crawler (Ground) button:	PASS

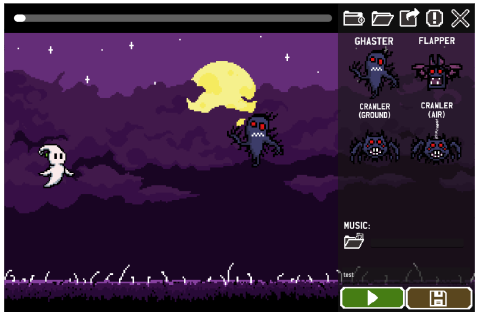
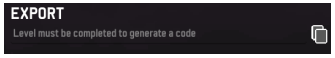
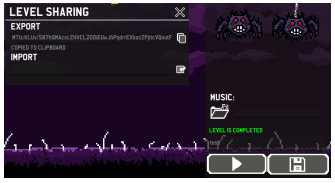
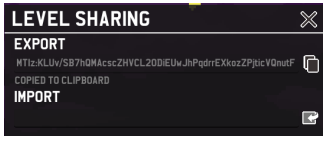
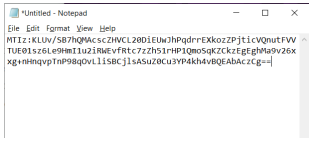
			 Crawler (Air) button: 	
Editor	Map scroll bar	Moving the scroll bar on top shifts the camera on the editor screen	 -> 	PASS

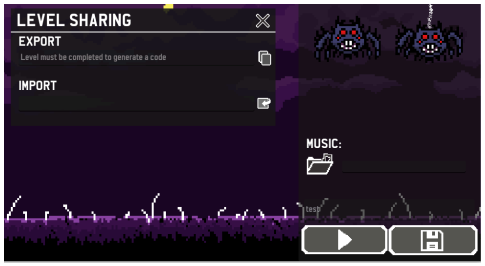
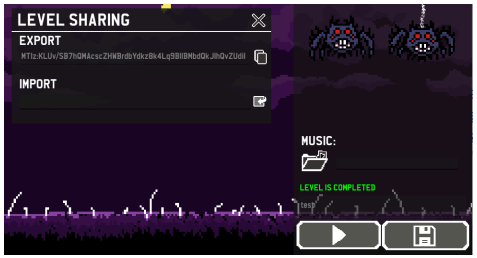
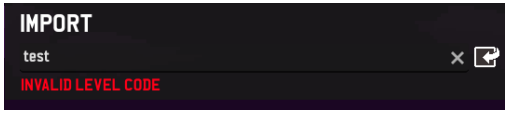
Editor	Editor buttons	• CREATE NEW: causes the editor to reload with a blank editor screen • OPEN: opens the level select screen for users to choose which level do edit • SHARE: opens the level sharing panel • INSTRUCTIONS: opens the instructions panel • QUIT: closes the editor and opens the main menu	 ->  ->  -> 	PASS

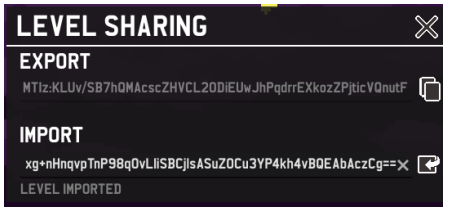
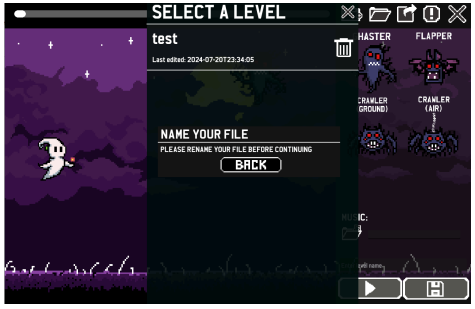
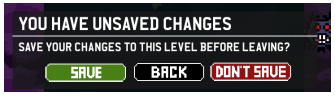
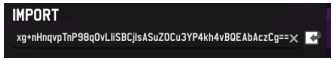
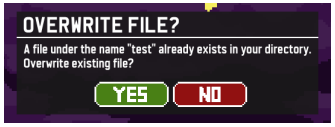
			 -> 	
Editor	Pop Ups	All pop ups cause most editor buttons (Line edit, file management buttons, level management buttons) to be disabled	Save pop up (focus on the play and save buttons on the bottom right being disabled):  Level name change pop up:  Level select screen:  Sharing panel:	PASS

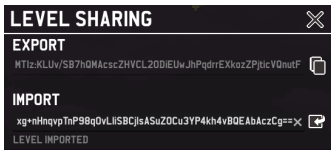
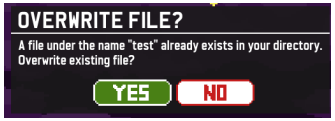
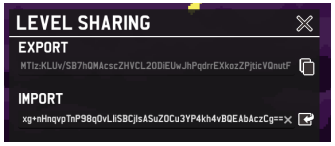
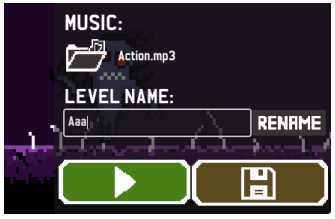
				
Editor	Level management buttons	Both play and save buttons are only enabled when the level is named (i.e. line edit is not empty) and there is at least 1 enemy in the level	Case 1: No enemy AND no level name  Case 2: Has an enemy AND no level name  Case 3: No enemy AND has level name  Case 4: Has enemy AND has level name 	PASS
Editor	Create new button	Clicking create new button causes a file save popup to appear if the current level is not saved,	Unsaved level:	PASS

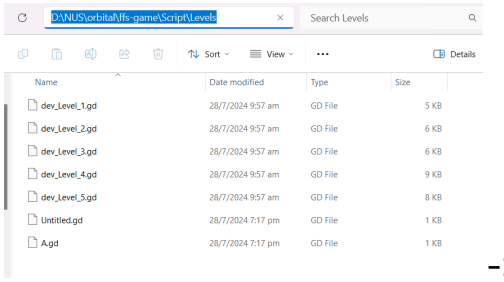
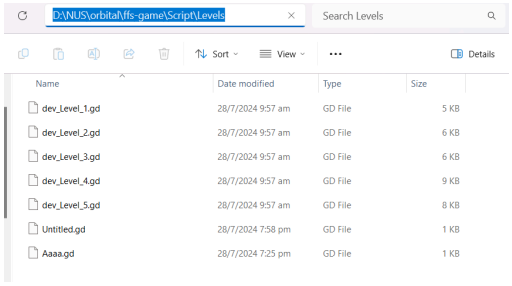
		immediately reloads the level otherwise	 ->  Saved level:  -> 	
Editor	Open level button	Clicking a level in the level editor's level select causes a file save popup to appear if the current level is not saved, immediately loads the selected level otherwise	Unsaved level: 	PASS

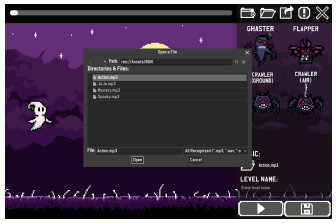
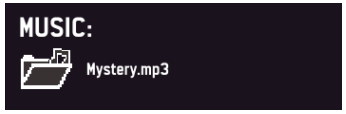
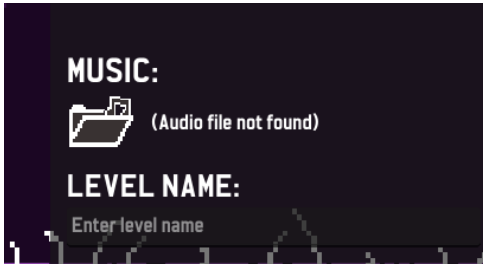
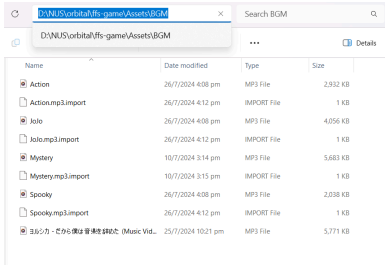
			Saved level: 	
Editor	Level completed label	When a level has been validated, a “level is completed” label is displayed above the line edit		PASS
Editor	Share level panel (copy button)	When a level is not yet completed, the export button is disabled		PASS
Editor	Share level panel (copy button)	Clicking the copy button on a valid code causes a “Copied to clipboard” notification to appear		PASS
Editor	Share level panel (copy button)	Clicking the copy button sets the computer’s clipboard value to the export code	 	PASS

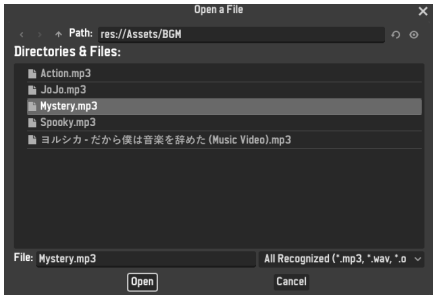
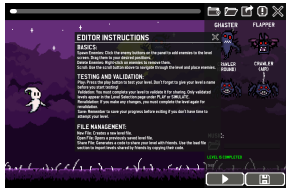
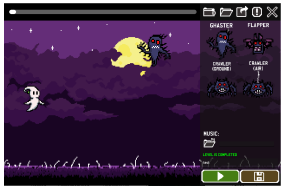
Editor	Share level panel (export code)	When a level is not yet completed, the level cannot be shared (i.e. level code is not produced). If the level is completed, then the level code is shown	Not completed level:  Completed level:  	PASS
Editor	Share level panel (import button)	Import button disabled when the export code line edit is empty, not disabled otherwise	  	PASS
Editor	Share level panel (import code)	Invalid import codes cause an “Invalid import code” warning to appear	 	PASS

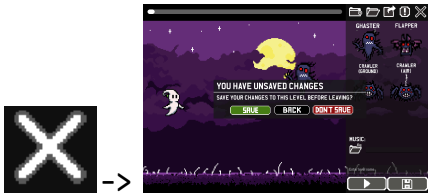
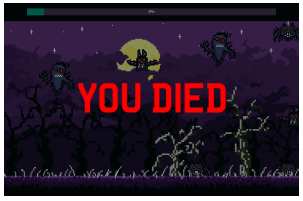
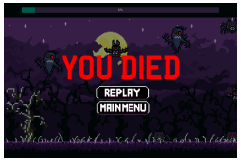
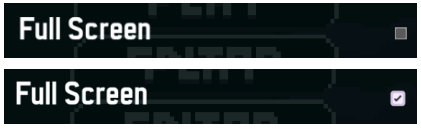
Editor	Share level panel (import notification)	Importing a level with a valid level code produces a “level imported” notification		PASS
Editor	Name your file pop up	If the level’s name in the line edit is empty when the level “save” button under file popup is clicked, a “name your file” warning pop up appears		PASS
Editor	Name your file pop up	Clicking the back button in the “name your file” pop up closes the pop up and reopens the file save pop up	 -> 	PASS
Editor	Level overwrite pop up	Importing a level with the same name as a level that already exists in the levels folder produces a “file overwrite” pop up	 -> 	PASS
Editor	Level overwrite pop	• YES: level	Yes:	PASS

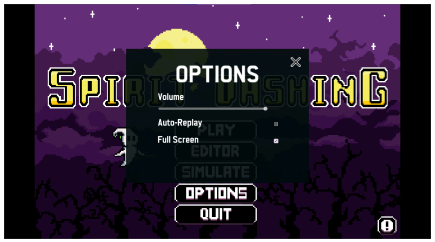
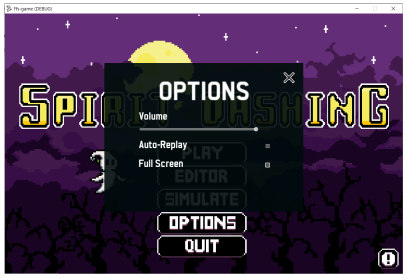
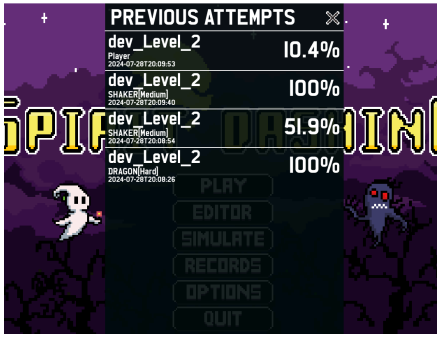
	up buttons	overwritten • NO: “overwrite file” popup disappears and sharing panel is shown again	 ->  No:  -> 	
Editor	Level rename	Rename button pops up only when a level is initially saved and has its name now changed	 -> 	PASS

Editor	Level rename	Clicking rename after changing the name of the level changes the name of the current level file into the new name, and not create a new level file	  ->  	PASS
Editor	Level rename	Renaming a level that is already completed does not change its completion status		PASS

Editor	Select music button	Clicking the select music button opens the res://Assets/BGM/ folder	 -> 	PASS
Editor	Select music label	The label under MUSIC section shows the selected music file's name	 -> 	PASS
Editor	Select music label	The label under MUSIC section will give an "Audio file not found" warning if the music is deleted from the BGM folder or if a level is imported with a music that does not exist in the importing device		PASS
Editor	Select music file dialog	Select music file dialog shows any new mp3 or wav or ogg file added into the res://Assets/BGM folder	 ->	PASS

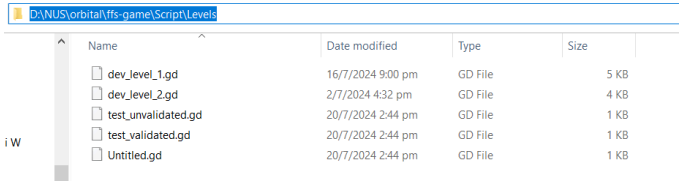
				
Editor	Select music	Selected music plays as the background music for the level	Selected music plays as the background music for the level	PASS
Editor	Instructions Button	Clicking the instructions button causes the instructions panel to appear	 -> 	PASS
Editor	Instructions panel exit (X) button	Clicking the exit button on the instructions panel closes the instructions panel	 -> 	PASS
Editor	Exit (X) button	If the current editor level is saved, pressing the exit button will bring the user to the main menu. Otherwise, the	Saved level:  -> 	PASS

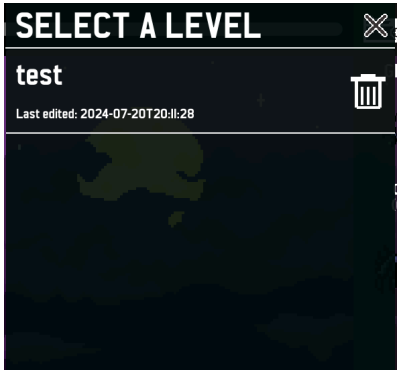
		file save pop up will appear	Unsaved level:  ->
Options	Auto-replay settings	Clicking the auto-replay tickbox toggles the auto-replay option.	
Options	Auto-replay settings	If auto-replay is ticked, a game level auto-replays after a short delay whenever the player dies. Otherwise, show the death screen with buttons	Auto-replay ticked:  ->  Auto-replay not ticked: 
Options	Full screen settings	Clicking the full screen tickbox toggles the full screen option.	

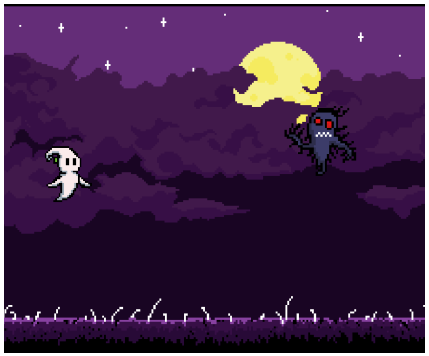
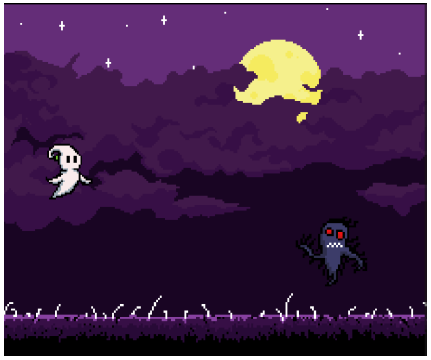
Options	Full screen settings	If full screen is ticked, the game goes into full screen mode. Otherwise, it remains windowed	Full screen ticked: (full screen screenshot)  Full-screen unticked:  	PASS
Options	Clicking exit (X) button	Closes options panel	Options panel closed	PASS
Records	Record entries	Victories or deaths in a level will be recorded under the RECORDS	 	PASS
Records	Record entries	Record entries will appear in the order of most recent to least recent		PASS

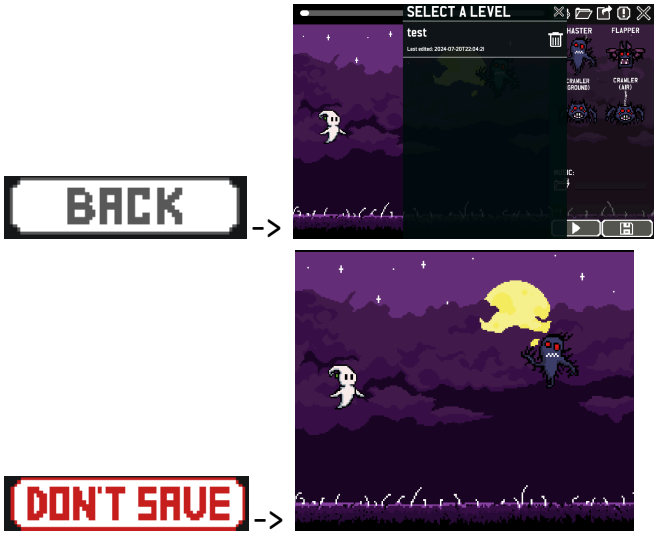
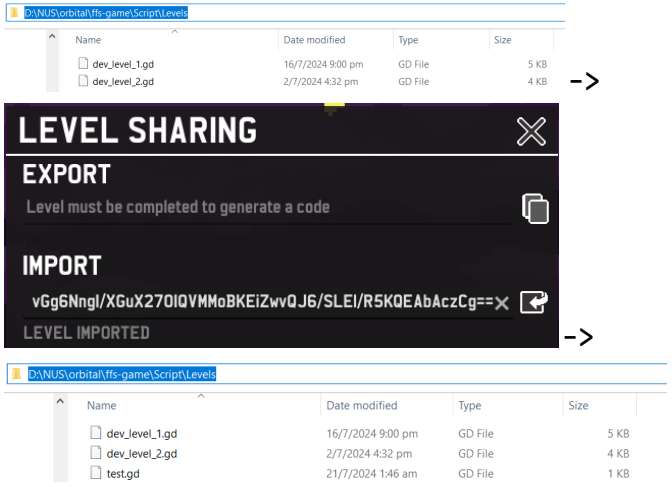
Records	Records exit (X) button	Clicking the exit button returns the user to the main menu		PASS

System Testing

Feature	Test	Expected	Output	PASS/FAIL
Main Menu	Player character idle animation	Player sprite plays an idle animation after receiving no input for a few seconds		PASS
Level Select (PLAY & SIMULATE)	Detecting levels in the folder	All (validated) levels are shown		PASS
Level Select (PLAY & SIMULATE)	Detecting levels in the folder	“Untitled.gd” file not shown		PASS
Level Select (PLAY & SIMULATE)	Detecting levels in the folder	“dev_” levels (indicating levels that we developers		PASS

		designed) are not deletable - i.e. the delete button should not be there for dev levels	->  	
Level Select (EDITOR)	Detecting levels in the folder	Level select in editor displays all levels (validated or unvalidated) except dev levels	 	PASS
Editor	File save popup buttons	• SAVE: Saves the current editor level before exiting the current level • BACK: closes the	Test level before modifications:	PASS

		save popup, stays in the level select screen and in the current editor level • DON'T SAVE: discards any new changes to the current editor level and exits the current level	 Test level after modifications:   -> 	

				
Editor	Level import	Importing a level causes a level file with the corresponding name to be produced in the levels folder		PASS

Simulate	Bot select back (<) button	Clicking the back button returns you to the level select (simulate) screen		PASS
Simulate	Bot select exit (X) button	Clicking the exit button returns you to the main menu		PASS

User Testing

We sent a zip file of our game to some of our friends, along with a google form for them to leave their feedback:
https://docs.google.com/forms/d/e/1FAIpQLScC1S12iLX5aZdktN09XNbRh3zQeGrT_dg78NDBrEYTJRJLzA/viewform

Link to the results:
https://docs.google.com/spreadsheets/d/1oXmtD4hMaGT-KIb__npw6CbaHOBW0WnRZfJrdWayVuc/edit?resourcekey=&gid=1005013788#gid=1005013788

Development Roadmap

20th May 2024: Ideation and consultation with adviser completed; Lift-off documents submitted.

3rd June 2024: Rough prototype of base game and other features completed; Milestone 1 submitted.

10th June 2024: Improve on Level Editor; Completion of HUD and Level Select.

17th June 2024: First bot for Simulation implemented; UI for Simulation completed.

24th June 2024: Second bot for Simulation completed; Level Editor fully implemented.

1st July 2024: Buffer time and time to improve game quality; Milestone 2 submitted.

29th July 2024: Instructions, Level Sharing, and Record features completed; Third bot developed; Game fully functional, presentable, and fun; Milestone 3 submitted.

28th August 2024: Polishing up minute details and outstanding issues that may be found from peer review. These details possibly include:

1. Implementation of an 'Endless Mode'
2. Refresh the instructions (Play & Level Editor) according to feedback provided:
    a. It could be more clear
    b. It is not intuitive to follow
3. General refresh of game elements:
    a. Level Editor enemy buttons
    b. Level Editor scrollbar and game progress bar
    c. Resizing certain panels and its contents (e.g. level select panel, records panel, etc.)
4. Upload the game into itch.io

Links

Project Log
Video
Poster
Game Download