Hoang Minh Duc PROJECT' S NAME: HYPER-CASUAL MOBILE GAME: MATCH 3 2024

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: INFORMATION SECURITY

PROJECT TITLE:

HYPER-CASUAL MOBILE GAME: MATCH 3

Instructor: **Ph.D. PHAM CONG THANG**
Student: **HOANG MINH DUC**
Student ID: **102190257**
Class**: 19TCLC_DT6**

Da Nang, 06/2024

THE UNIVERSITY OF DANANG
DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

GRADUATION PROJECT THESIS

MAJOR: INFORMATION TECHNOLOGY

SPECIALTY: INFORMATION SECURITY

PROJECT TITLE:

HYPER-CASUAL MOBILE GAME: MATCH 3

Instructor: **Ph.D. PHAM CONG THANG**
Student: **HOANG MINH DUC**
Student ID: **102190257**
Class**: 19TCLC_DT6**

Da Nang, 06/2024

DA NANG UNIVERSITY
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY……

SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT COMMENT

I. General information:

1. Student name: Hoàng Minh Đức
2. Class: 19TCLC_DT6 Student ID: 102190257
3. Topic title: Hyper-Casual Mobile Game: Match 3
4. Instructor: Phạm Công Thắng Academic title/ degree: Ph.D.

II. Reviews of graduation project

1. About the urgency, novelty, usability of the topic: (2 points)
   ……
   ……
2. About the results of solving the tasks required by the project: (4 points)
   ……
   ……
3. About the form, structure and layout of the graduation project: (2 points)
   ……
   ……
4. The topic includes scientific value/article/problem solving of the enterprise or school: (1 point)
   ……
   ……
5. Existing shortcoming need to be supplemented or modified:
   ……
   ……

III. Spirit and attitude of the student (1 point):

……
……

IV. Evaluation:

1. Evaluation point: …./10
2. Suggest: Defense permitted/ Edit to defend/ Defense not permitted

Da Nang, date……month……. 2024

Instructor

INSTRUCTOR'S COMMENT

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

…...

Da Nang, date……month……. 2024

STUDENT'S COMMENT

…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...
…...

Da Nang, date……month……. 2024

SUMMARY

Topic title: Hyper-Casual Mobile Game: Match 3

Student name: Hoàng Minh Đức

Student ID: 102190257 Class: 19TCLC_DT6

This project is a mobile game in the casual-puzzle genre aimed at providing high entertainment value for everyone at any age and gender. Connecting people through a shared experience.

The main appeal of this game will be its cute art style, simple-to-understand gameplay, and engaging game mechanics.

This project also aims to develop an easy-to-use level designing tool, a game structure that is fully customizable through the use of configuration files, and a sophisticated game prediction algorithm to manage the apparent randomness of player luck.

DA NANG UNIVERSITY
UNIVERSITY OF SICIENCE AND TECHNOLOGY
FALCUTY……

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: Hoàng Minh Đức Student ID: 102190257

Class: 19TCLC_DT6. Faculty: Information Technology.

Major: Information Technology.

1. *Topic title:*

……..……

……..

2. *Project topic :* ☐*has signed intellectual property agreement for final result*
3. *Initial figure and data:*

……..……..……...

...……

…..……..……..……

Content of the explanations and calculations:

…...……

…...……

…...……

…...……

…...……

4. *Drawings, charts (specify the types and sizes of drawings):*

…...……

…...……

…...……

…...……

5. *Name of instructor:*	*Content parts:*

6. *Date of assignment : ……/……/202…..*
7. *Date of completion : ……/……/202…..*

Đà Nẵng, date month year 2024

Head of Division……. **Instructor**

PREFACE

This project report documents the efforts, discoveries, and achievements of myself in creating a captivating mobile game, a project undertaken as part of my studies in Information Technology here at Da Nang University of Science and Technology.

Throughout this report, I will provide an in-depth exploration of the development of this game, from the initial concept and ideal phase to the implementation stages. I will discuss the technologies utilized, the methodologies employed, and the decisions made to overcome various technical and creative challenges.

I would like to convey my heartfelt gratitude to Mr. Pham Cong Thang, PhD from Da Nang University of Science and Technology for his support and assistance in the completion of this project. The completion of the project would not have been possible without his help and insights.

Thankyou.

Hoang Minh Duc

ASSURRANCE

I hereby declare:

1. I affirm that all the content in this thesis was carried out under the direct guidance of Professor Pham Cong Thang.
2. All referenced materials used in this thesis are clearly cited and comply with copyright regulations.
3. I declare that I have not engaged in any act of plagiarism in the process of completing this thesis.
4. If there are any acts of cheating or violations, I take full responsibility and am willing to accept the consequences of those actions.

Student Performed

Hoang Minh Duc

TABLE OF CONTENT

LIST OF TABLES, PICTURES

LIST OF SYMBOL, ACRONYM

Acronym	**Abbreviations**
API	Application Programming Interface
GUI	Graphical User Interface
UI	User Interface
UX	User Experience
SFX	Sound Effect
VFX	Visual Effect
Lvl	Level

INTRODUCTION

1. **Aims of the project**

- This project aims to develop a casual mobile game based on the classic game concept: match 3, which is easy to understand, easy to play and have a high entertainment value

2. Target audience of the project

- The target audience for this project are: casual gamers, children, and older players who enjoy simple and relaxing gameplay

3. Structure of this project

- Research about the chosen gerne/concept
- Decide on the core gameplay loop and mechanic of the game
- Design the game architecture, flow state and data structure
- Develop the game
- Writing project report and presentation

Chapter 1: OVERVIEW

1.1. Why game as a project?

In an era where smartphones have become ubiquitous companions, mobile games have emerged as a vibrant segment of digital entertainment. And with the current demand being so high, this project aim to tap into the high potential market of mobile game with a classic game concept: matching puzzle game or Match 3 in detail.

1.1.1. Why match 3?

Within the puzzle games genre, match-three is a fan favorite, mostly because they're easy to understand, simple to play, and entertaining.

The levels are quick and with each passing level, you feel a sense of accomplishment, which in turn, drives positive feelings towards the game.

Bright colors and creative art and graphics make the game more enjoyable and interesting.

1.2. Objectives of this project.

To create a mobile game with all the traditional functionality including:

- Implement gameplay.
- Sound effect system and a simple visual effect system.
- A gameplay loop consists of levels, gameplay with target and reward.
- Design a system allowing easy tweaking of game difficulty-based parameter.
- Implement user interface.
- Implement a level design tool.
- Design engaging level.

1.3. Technical Feasibility.

1.3.1. Game Engine

Game engine: Unity

Game engine version: 2021.3.35

Programing langue: C#

1.3.2. External tool

Code editor: Rider 2023

Image editor: Gimp 2.10

Audio editor: Audacity

1.4. Game Feature

For player:

- A full gameplay loop of classic match 3 mechanic consisting of:
  - Matching item
  - Game booster and a large variety of in game item
  - Hint
  - Level and its component (targets, moves …)
  - Game flow for win, lose and retry

For developer:

- A level design tool with user interface
- A debug menu for testing
- Fully customizable game config for every aspect of the game

Chapter 2: Game Design

2.1. The classic match 3 formula

The goal of match-three games is to form lines, chains, or groups of three or more of the same elements. Your task is to line up the tiles and achieve three-in-a-row. When this happens, it causes a chain reaction of more tiles to match, before the line or matches disappear.

The board has various tiles that you need to move around, shift in order to get them to drop down and match into lines or groups.

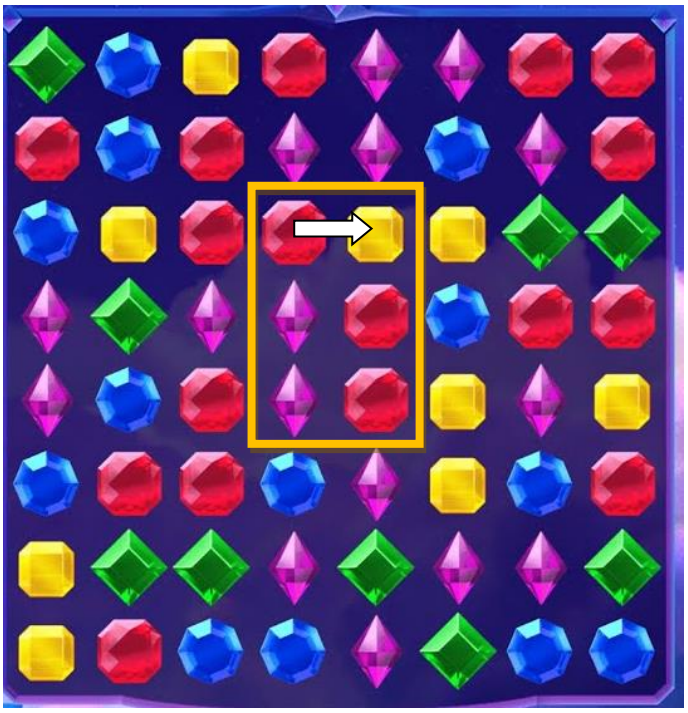

Figure 2.1: Microsoft Jewels

The role of the player in this game will be to control the game board by touch and swipe an item toward a direction (up, down, left, right) to swap position of that item to the one next to it in that direction.

2.1.1. Game rule

When player entered a level. The player must complete the level target (Destroy/collect an amount of items) before running out of available move.

The player win when target is reached within the move limit, otherwise the level is failed

2.1.2. Control

Player control the game by swiping an item, swapping its place with the item next to it in the direction of the swipe. Or taping items that support this interaction.

2.2. The modern twist

The classic match 3 mechanic have been around for very long time since the first ever match 3 game that is *Shariki* a puzzle game by Russian programmer Eugene Alemzhin and since the boom of the classic Microsoft Jewels this formula have already been experienced by many players worldwide.

With that in mind, to keep the experience fresh and more up to date with the current match 3 design, this project will introduce some new gameplay element:

2.2.1. Special items

This project will introduce some item that are more special than the regular color based item:

Figure 2.2: matching special pattern

The Tnt: created by matching 5 item in the pattern resembling the letter T or L

Figure 2.3: The Tnt

- When activated, explode and destroy all item in a square radius of 5x5 cell

The Rocket: created by matching 4 item in either horizontal or vertical direction

Figure 2.4: The Rocket

- When activated: split in to 2 item, each fly in either the direction the rocket is facing, destroy any item in its path.

The Propeller: created by matching four same-colored items in a 2x2 square.

Figure 2.5: The Propeller

- When activated: fly to an item and explode, destroying that item

The Light ball: created by matching five same-colored items in a row or column.

Figure 2.6: The Light ball

- When activated by tap: destroy all the item with the same color type that are most populated on the game board
- When activated by swipe: destroy all the item of the color type swapped with it.

2.2.2. Combination of special item.

Any 2 special item can combine with another when they are next to each other by swiping them together

- Tnt and Rocket: create 6 rocket, 3 in the vertical and 3 in the horizontal direction and activate them all at the same time
- Light ball: turn all the colored item with the most population into the combined special item type and activate them all at the same time
- Light ball and Light ball: destroy all item on the game board
- Propeller and Tnt/Rocket: fly to the target position and activate the special item it combined with at the target position.
- Propeller and Propeller: create 3 propeller.
- Tnt and Tnt: produce an explosion with a radius of 9x9 cell
- Rocket and Rocket: activate both in different direction

2.2.3. Target item.

Target item are item that are required to be destroyed in order to complete the level. Each have unique mechanic in how it is interacted with.

The Box:

Figure 2.7: The Box

An obstacle with 3 level. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one until destroyed completely.

Cannot fall and block everything above from falling down. Cannot be swiped

The Vase:

Figure 2.8: The Vase

An obstacle with 2 level. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one until destroyed completely.

The Shell:

Figure 2.9: The Shell

An obstacle with 3 level. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one until destroyed completely and give player the pearl inside.

The Mail box:

Figure 2.10: The Mail box

An obstacle that cannot be destroyed. Interact with it by matching item next to it or by special item, each interaction give the player a mail item for completing the level.

The Tree:

Figure 2.11: The Tree

An obstacle that can have any length. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one making it 1 cell shorter until destroyed completely.

The Grass:

Figure 2.12: The Grass

An lower-layered item that lie beneath normal item. Destroy by matching item above it or interacted by special item, each interaction reduced it level by one until destroyed completely.

The Cabinet:

Figure 2.13: The Cabinet

An obstacle with size of 2x2 cell, has 11 level. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one, destroying one plate inside until destroyed completely.

The Bush:

Figure 2.14: The Bush

An obstacle with size of 2x2 cell, has 3 level. Destroy by matching item next to it or interacted by special item, each interaction reduced it level by one until destroyed completely, producing a 4x4 field of grass beneath.

2.3. The unpredictability of luck and skill, and how to control them

The gameplay of match 3 game is almost entirely random since there are so many possibility in just one move the player made and the outcome of any move is almost based on pure luck alone because of the new item that are created each time any match are made and how the interact with the board can lead to another random outcome.

In a game design stand point, a system that rely almost purely on random to decide the outcome of a level can lead to a inconsistent or event frustrating user experience.

For example: A hard level can be beaten easily if the user are lucky enough but that same hard level can take many try to beat for an user with the same skill because of the random factor.

The ideal:

This game will try to manage and control that aspect of randomness for the benefit of the user experience by control how a level progress as best as it can. Limiting the random factor to a minimum by:

- Preset a set of probability
- No random item spawn allowed. Calculate the future state of the game board and try predict the best out come of the next board state with the probability above.
- Control the item spawn to achieve that desired game state by automatically create match for user with the spawned item when they fall down the their final position. In other word: intentionally create match combo to destroy more target or produce free special item for user to use

And with the randomness out of the way. We can control how difficult a level is where ever we want. In this project, this system will gradually try to make the game easier as the player approaching the losing point of a level. This artificial luck is better than sometime no luck at all.

Chapter 3: Theoretical Foundation

3.1. Game Engine

Unity is a versatile game engine developed by Unity Technologies, initially introduced at the Apple Worldwide Developers Conference in June 2005 as a Mac OS X game engine. Since its inception, Unity has expanded its capabilities to encompass various platforms including desktop, mobile, console, augmented reality, and virtual reality. It has gained significant popularity for mobile game development on iOS and Android, and is renowned for its accessibility to novice developers and its widespread use in indie game development. Beyond gaming, Unity is utilized in diverse industries such as film, automotive, architecture, engineering, construction, and by the United States Armed Forces. The engine supports the creation of both three-dimensional (3D) and two-dimensional (2D) games, as well as interactive simulations.

3.1.1. Advantages

User-Friendly Platform:
Unity stands out for its agility, speed, and ease of use, making it a preferred engine for developers and gamers alike. Its extensive Application Programming Interface (API) and powerful scripting capabilities contribute to its reputation as a user-friendly technology.

Efficient Debugging:
A robust open-source error tracking tool enables real-time monitoring and resolution of crashes, minimizing downtime. With a vast community of over 2.5 million developers, issues are tackled collaboratively and efficiently.

Ideal for All Skill Levels:
Unity's accessibility with free availability and minimal coding requirements caters to both beginners and seasoned developers. Online tutorials provide continuous learning opportunities, keeping developers abreast of the latest trends.

Multiplayer Gaming Fun:
Unity's multiplayer features facilitate community-building and enhance player retention. They enable global gaming interactions, fostering longer and more engaging gaming sessions.

Cross-Platform Compatibility:
Unity excels in portability, enabling games to launch seamlessly across multiple platforms with a single click. Its compatibility spans 25 different platforms, optimizing development time and broadening audience reach.

3.1.2. Disadvantages

Limited External Code Libraries Linkage:
Integration with external code libraries in Unity requires manual effort, lacking a centralized library folder. This can be time-consuming for developers handling multiple projects.

Costly License:
While initial Unity license costs may be steep for new companies, the investment yields consistent returns. Additional features and support options can increase development expenses, necessitating adequate initial funding.

Higher Memory Consumption:
Unity games often demand more memory, potentially leading to out-of-memory issues on mobile devices. Efficient memory management is crucial to mitigate debugging challenges.

Source Code Navigation:
Unity's search functionality for connected scripts is adequate but could benefit from enhancements. Manual searches for specific elements may disrupt the development workflow, posing occasional challenges.

3.1.3. Monobehaviour

MonoBehaviour serves as a foundational class from which many Unity scripts inherit. Most systems in a Unity project derive from MonoBehaviour, sharing its lifecycle and providing entry points and update points for logic execution (see Figure 3.1).

MonoBehaviour provides essential lifecycle functions that simplify development in Unity.

MonoBehaviours always exist as components of GameObjects, when the parent GameObject is destroyed, all components, including MonoBehaviours, are automatically deleted.

Even after the underlying component is destroyed, the C# object for the MonoBehaviour remains in memory until garbage collection occurs. In this state, a MonoBehaviour acts as if it is null.

When a MonoBehaviour is serialized, the values of C# fields are included according to Unity's serialization rules.

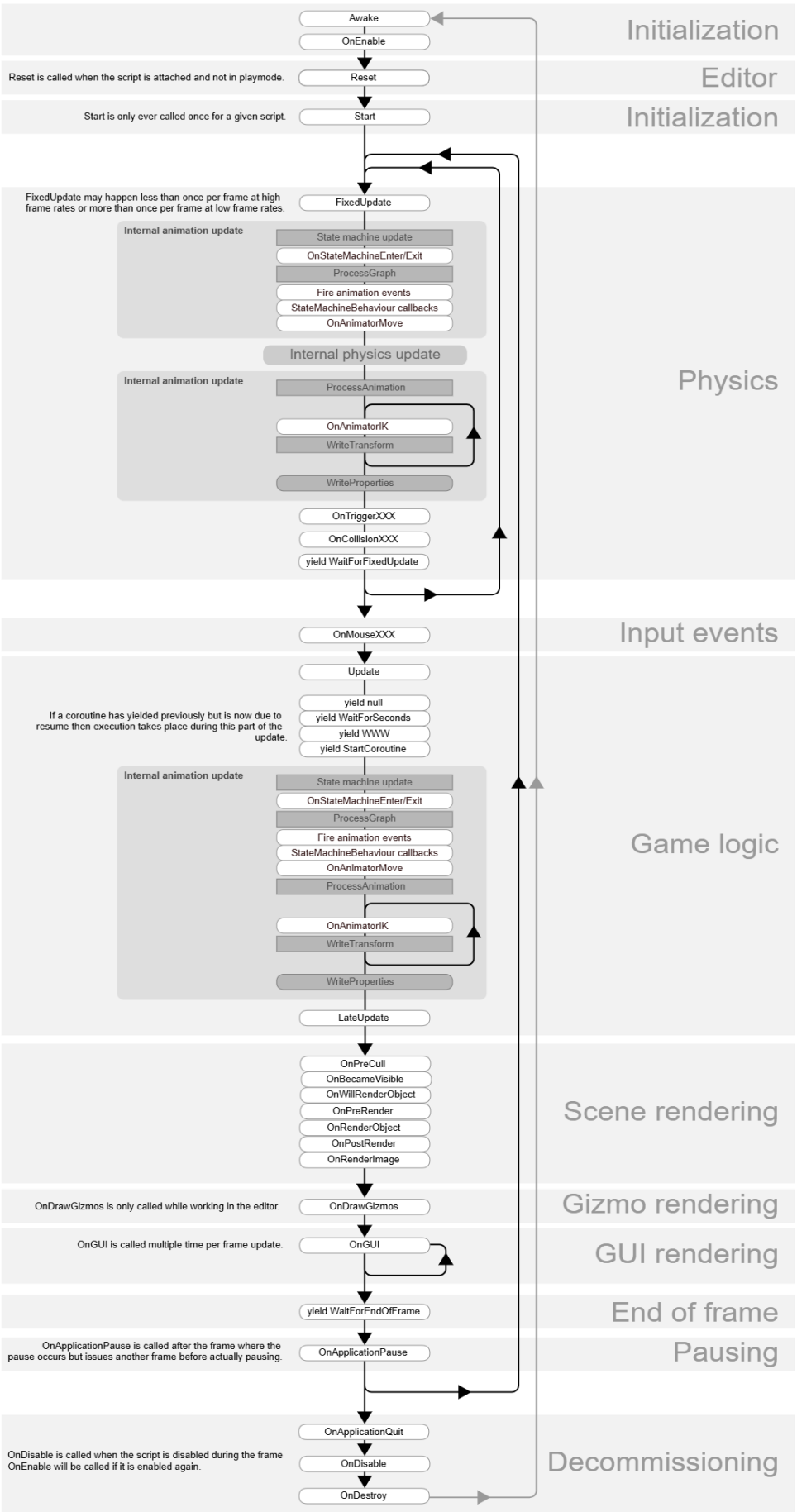

Figure 3.1: MonoBehaviour life cycle

3.2. The C# programing langue

C# is a versatile high-level programming language that supports multiple paradigms.

It includes static typing, strong typing, lexical scoping, and supports imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming styles.

Advantages of C# with Unity:

- ***Object-Oriented Programming:*** C# promotes structured and modular code organization, enhancing software design in Unity.
- ***High-Level Language with Memory Access:*** It offers efficient memory management and high-level abstractions suitable for game development.
- ***Integrated with .NET Platform:*** Leveraging cross-platform runtimes and extensive .NET framework resources enhances compatibility and development capabilities.
- ***Built-In Garbage Collector:*** Automatic memory management simplifies memory handling, reducing manual memory management errors.
- ***Type-Safe with Dynamic Capabilities:*** Provides type safety while supporting dynamic programming techniques for flexibility.
- ***Comprehensive Documentation and Community Support:*** Extensive resources and a large community facilitate learning, troubleshooting, and innovation in Unity development.

Disadvantages of C# Programming:

- ***Performance Concerns:*** C# may exhibit slower performance, particularly in complex or resource-intensive game projects, which could impact larger-scale game development.

Chapter 4: System Design

4.1. List of base class/component and their responsibility

There are many class and component involved in this project so I will only list out base classes that all other game system are derived from without going into the detail of each and every child subclass, giving an overview of all the core system in the game(Table 3.1)

Name	**Responsibility**
BoardManager.cs	Manage the creation, loading, saving data and the behavior of the board and the physic of the items on it like match, fall…,
Cell.cs	One cell unit of the board. Holding items in it, sharing data with the board and the item inside to determine the behavior of the item on the board
Item.cs	Base class for all item, holding data of that item, provide the base API for processing interaction with other item or system
GameManager.cs	Controlling game logic like win, lose…, provide API for loading level, data
UserPrefs.cs	Manage user data. Provide API for loading, saving user data. Determine how data are saved
FallController.cs	Manage the creation of item and their fall behavior. Working in tandem with Probability Controller to determine the item spawned, the difficult of the level
MatchController.cs	Manage the match behavior and logic of items, the activation, combination logic of special item and interaction between all of them.
ProbabilityController.cs	Calculate and predict the game state to decide what item will be spawned next. Keeping the game difficulty as configured in difficulty config for every level, basically controlling the “luck” the player receive
PatternDetector.cs	Detect match pattern of a collection of items and their coordinate.
HintController.cs	Search and manage how hint are found and behave
Manager.cs	Application-wide manager, handling initialization, communication between systems.
MoveController.cs	Base class for controlling game object movement. Providing API for start/stop movement and how an

	object move(speed, trajectory).
AnimationController	Base class for controlling animation of object. Providing API for playing, controlling animation
ItemManager.cs	Manage the creation, pooling, destruction and initialization of items
EffectManager	Manage the creation, pooling, destruction and initialization of visual effect created by Unity Particle system
SimpleCell/SimpleItem FallPrecalculation.cs	A simplified cell data system for game state prediction and decide the future game state based on configured data
BoosterController.cs	Base class for Controlling one booster, how it behave, how input system interact with it.
TouchController.cs	Receiving input and distributing input data to any system that required it
ITouchHandler.cs	Base class for receiving input from touch controller, deciding what to do with it and how subsequence input are handled
SingletonBehavior.cs	Base class for singleton entity
TargetController	Control one target in a level, listening to gameplay event and managing its corresponding UI element
UIManager/UIButton	Managing game UI/ UI element
AudioManager.cs	Manage in game audio system. Provide API for playing sound and manage how they are played, pooled and destroyed
SingletonScriptableObject.cs	Manage a single persistent data holder entity. Used for storing configurable data
Vibration.cs	Handle vibration on phone
ShuffleController.cs	Handle how board shuffle
CombineController.cs	Control how item are combined and how the combination result are behaved
LevelData.cs	Data class for storing data of a single level and how to serialize/deserialize them
EditorManager.cs	Manage the level editor tool. Similar functionality to GameManager but mostly handling saving level data
EditableBoardManager.cs	Derived from board behavior, Managing the game board inside level editor mostly working with constructing level data and how to display them for easy level creation

Table 4.1: List of classes and their responsibility

4.2. Use case diagram

4.2.1. Use case of player

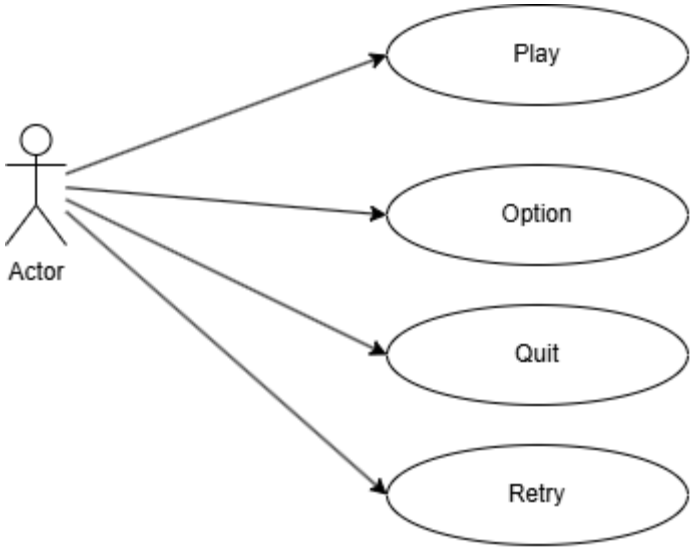

Figure 4.1: use case of player

Use case description for player(Figure 4.1):

1. Use case: Play

Primary actor: Anyone playing the game.

Goal in context: to play the game.

Precondition: Player select play from the main menu.

Scenario:

- Go to main menu of the game
- Select “Play”

2. Use case: Option

Primary actor: Anyone playing the game.

Goal in context: To change the game setting.

Precondition: Player already in a level

Scenario:

- Select setting button

3. Use case: Quit

Primary actor: Anyone playing the game.

Goal in context: To leave the current level and return to the main menu.

Precondition: Player already in a level

Scenario:

- Select setting button
- Select quit button in the setting menu

4. Use case: Retry

Primary actor: Anyone playing the game.

Goal in context: To restart a level.

Precondition: Player already in a level

Scenario:

- Select setting button
- Select retry button in the setting menu

4.2.2. Use case of Developer

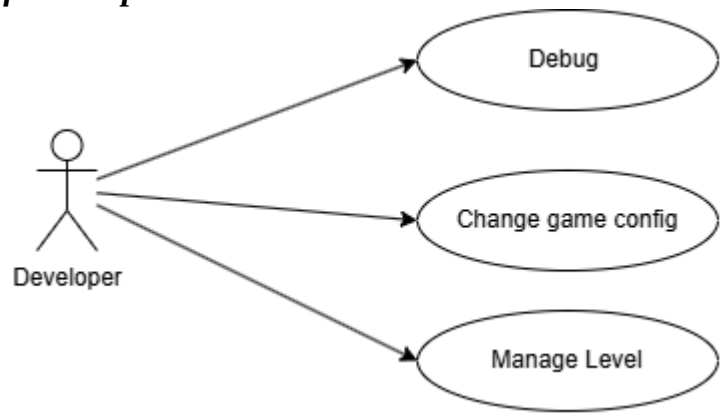

Figure 4.2: use case of Developer

Use case description for developer(Figure 4.2):

1. Use case: Debug

Primary actor: Developer playing the game.

Goal in context: to play the game with assisted tool for debugging.

Precondition: Played on a debug build version of the game.

Scenario:

- Go to main menu of the game
- Select “Play”
- Select option “show debug menu” or double taping the top left corner of the game

2. Use case: Change game config

Primary actor: Developer.

Goal in context: To change how the game behave

Precondition: Developer has access to the Unity project.

Scenario:

- Open Unity Editor
- Select the config file containing the entry the developer wanted to chane
- Change the configured value.
- Save the new config file

3. Use case: Manage level

Primary actor: Developer.

Goal in context: To manage levels for the game

Precondition: Developer has access to the Unity project.

Scenario:

- Open Unity Editor
- Open the LevelEditor scene
- Create/Edit/Delete level with the tool developed

4.3. Activity Diagram

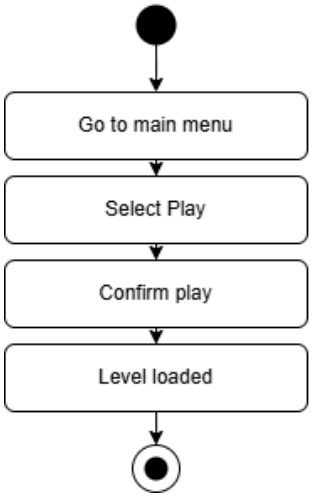

Figure 4.3: Activity diagram for "Play"

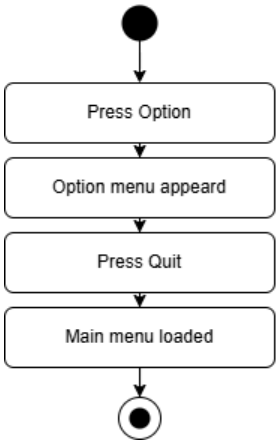

Figure 4.4: Activity diagram for "Quit"

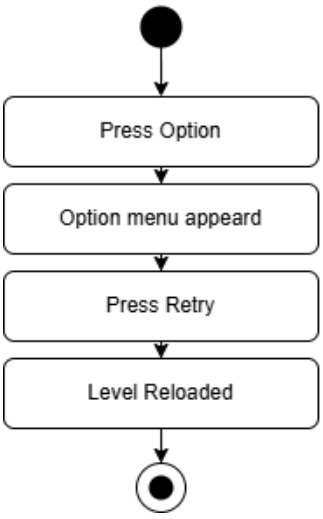

Figure 4.5: Activity diagram for "Retry"

4.4. Class Diagram

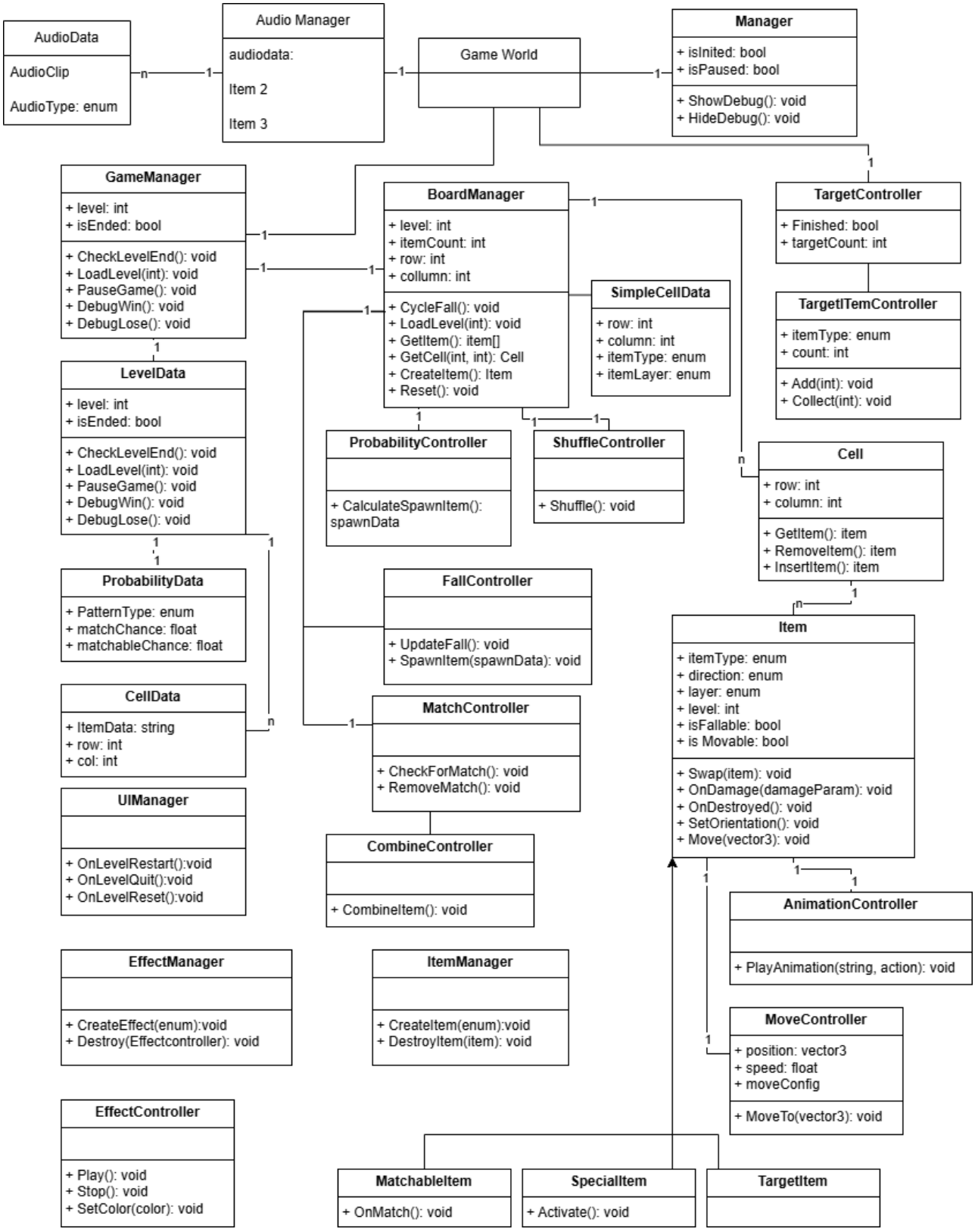

Figure 4.6: Class diagram

4.5. User data

Figure 4.7: UserData

All user data saved will be saved to Application's persistent data path (which depend on the platform)

All user data can be edited in Realtime via Debug Panel in Debug Mode

Description:

Field	Description
MaxLevel	User's max level achieved
CurrentLevel	User's currently playing level
Coin	Amount of coin owned
BoosterHammer	Amount of Hammer owned
BoosetBow	Amount of Bow owned
BoosterWand	Amount of Wand owned
BeginBoosterLightBall	Amount of Light ball BeginBooster owned
BeginBoosterRocket	Amount of Rocket BeginBooster owned
BeginBoosterTnt	Amount of Tnt BeginBooster owned
SoundEnabled	Is sound effect enabled
MusicEnabled	Is music enabled

Table 4.2: User data description

4.6. Configurable Game configs

Game config are file that hold and render data inside Unity Inspector window. Used for easy view and edit game parameter. Suitable for game designer to edit how the game behave without any code modification required.

The config file will be read when the game start to load all the parameter into the game.

Here are some of the configuration file this project provided

4.6.1. AudioConfig

Contain configuration of all the audio used in the game and its type

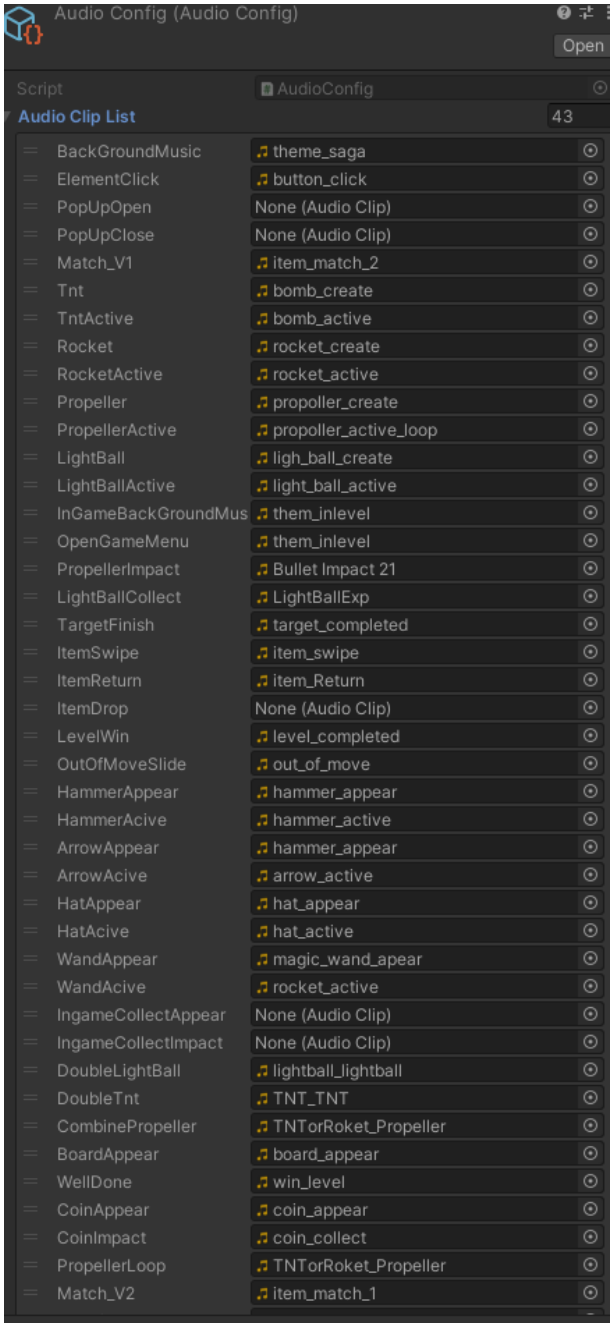

Figure 4.8: AudioConfig

4.6.2. Game Config

Contain comon data used all through out the game

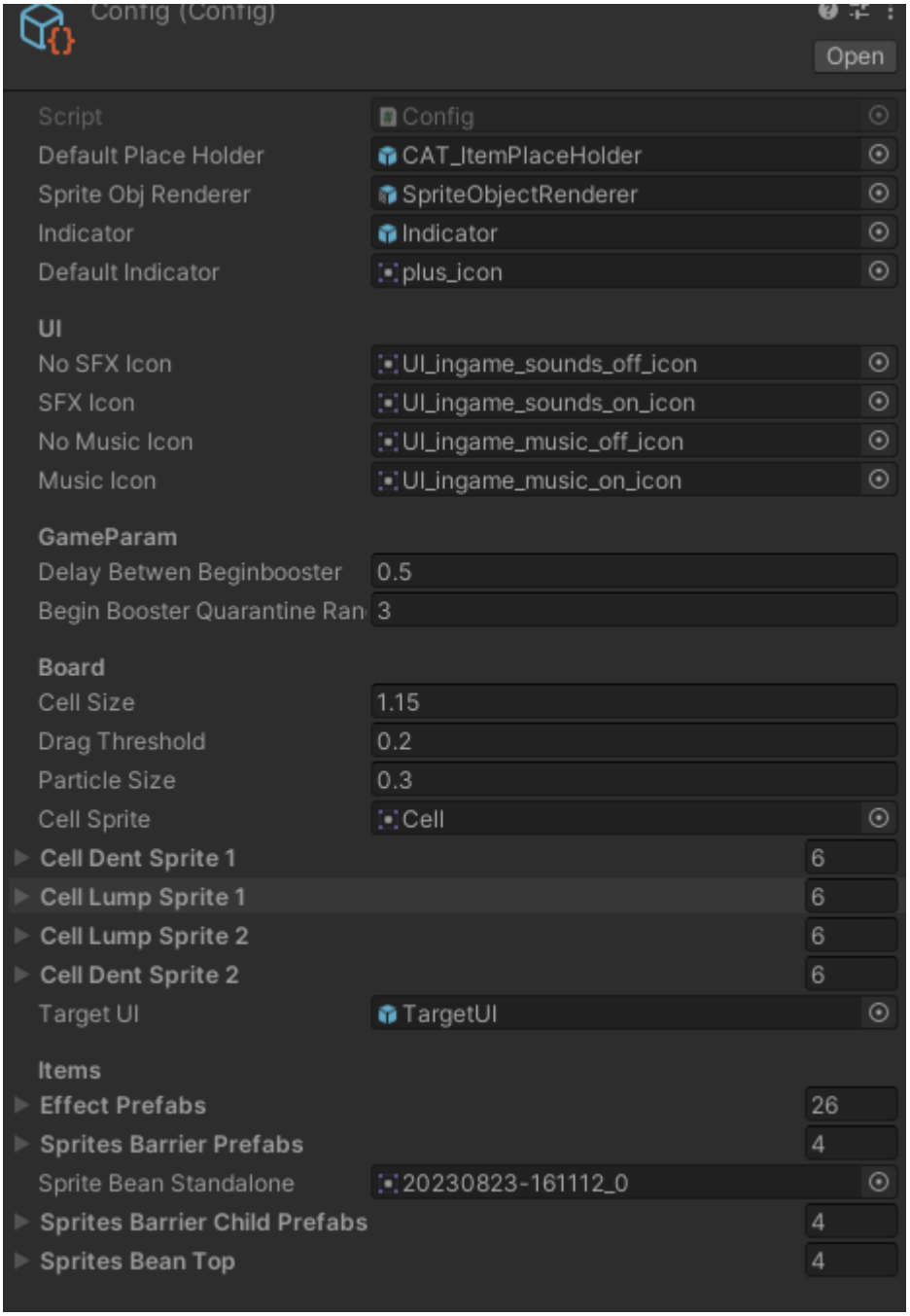

Figure 4.9: Game config

4.6.3. *Item resource config*

Contain configuration for all item in the game include:

- The item Prefab
- The image of that item
- The visual effect that item might produce

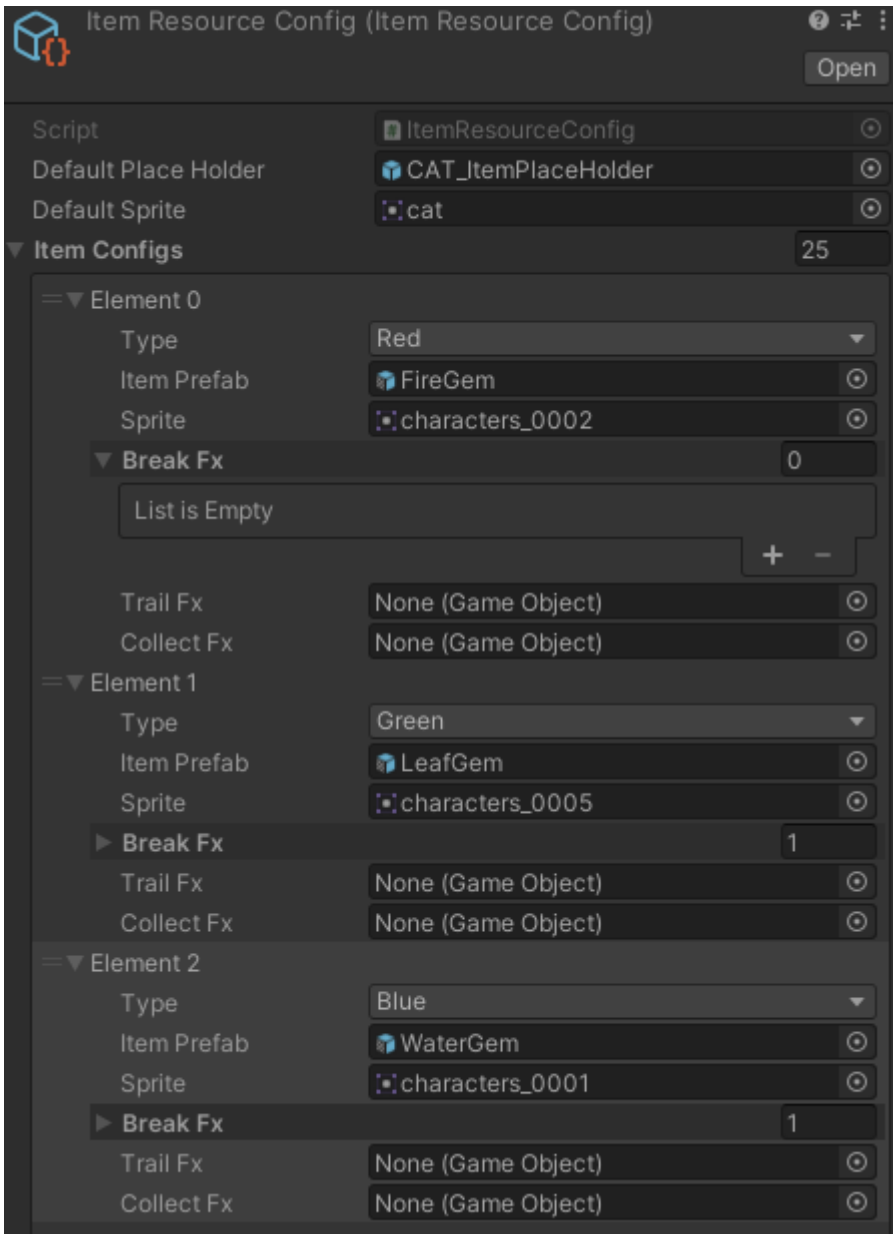

Figure 4.10: Item resource config

4.6.4. Logic config

Contain probability parameter for all the difficulty type:

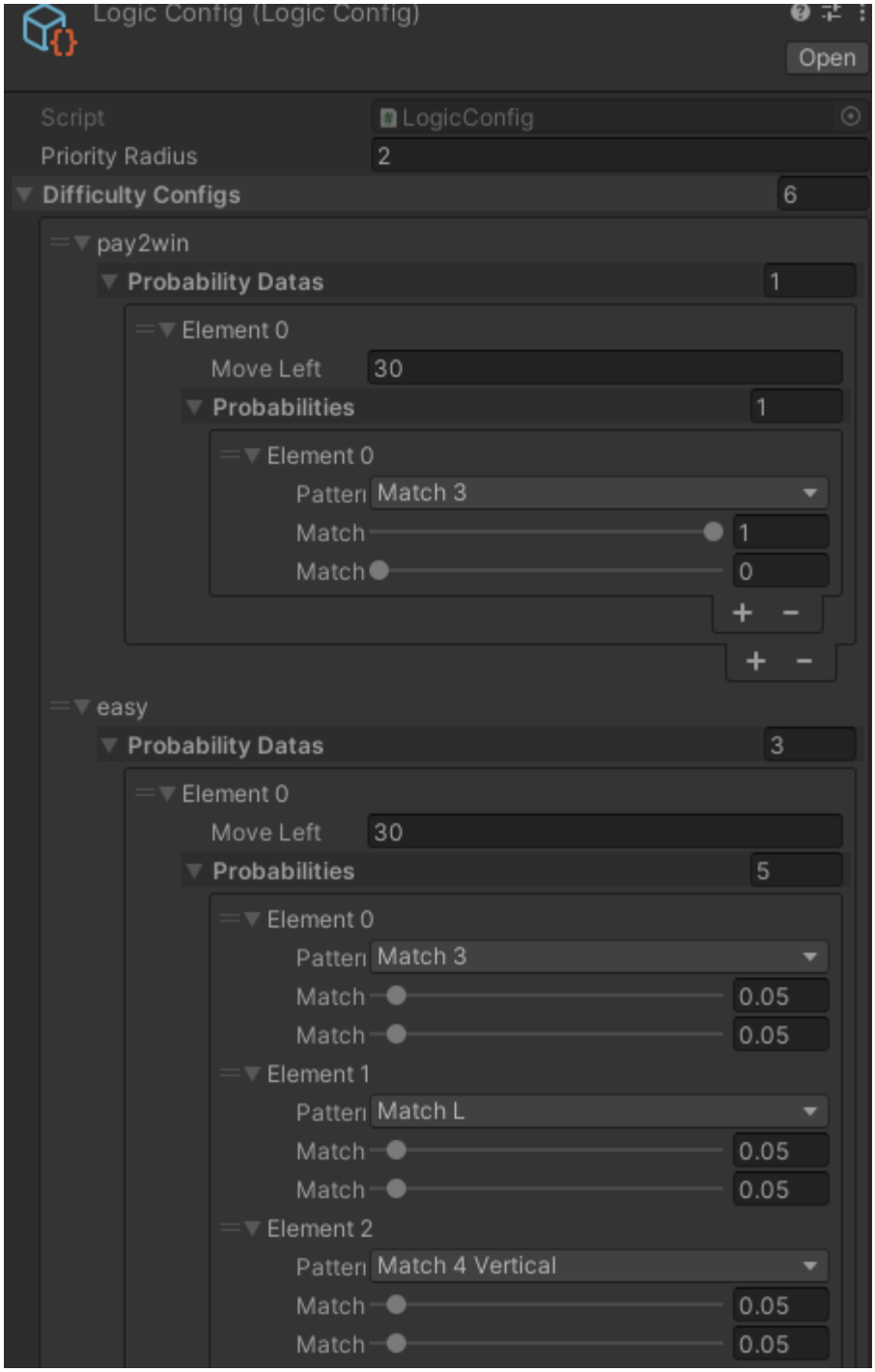

Figure 4.11: Logic config

4.6.5. *Movement config*

Contain all movement parameter for calculation of movement and trajectory.

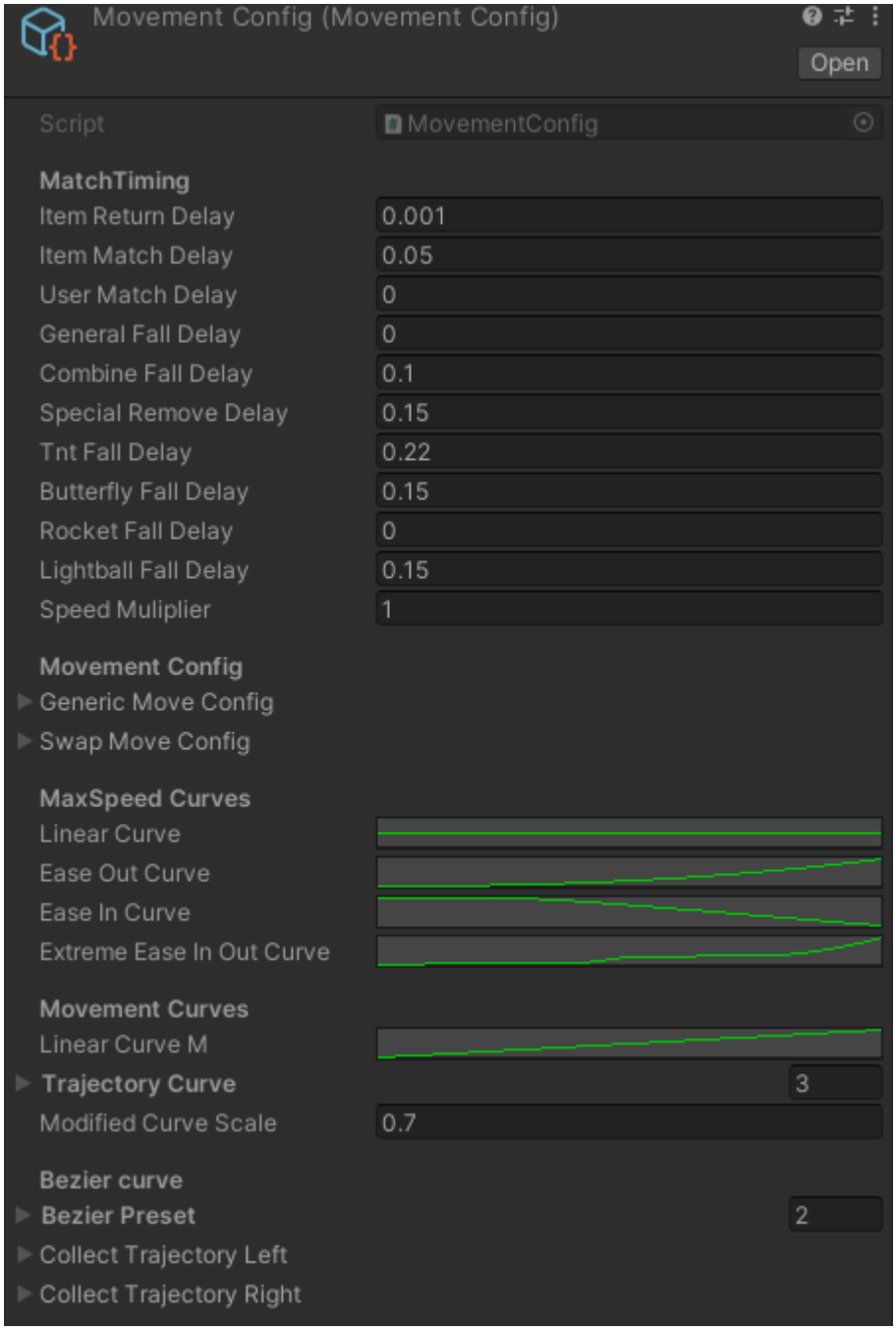

Figure 4.12: Movement config

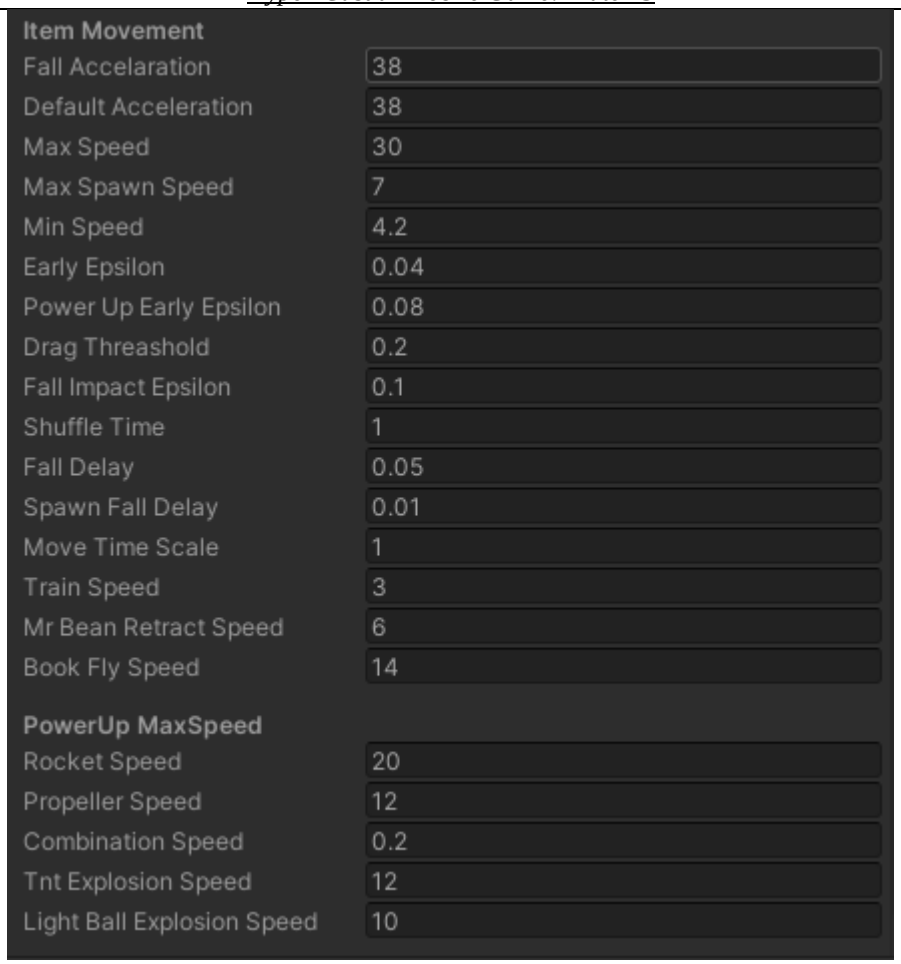

Figure 4.13: Movement config

4.7. The probability control loop

4.7.1. The core game logic cycle

To understand how the probability control system work, let first take a look at the main control cycle of the game(Figure 4.14)

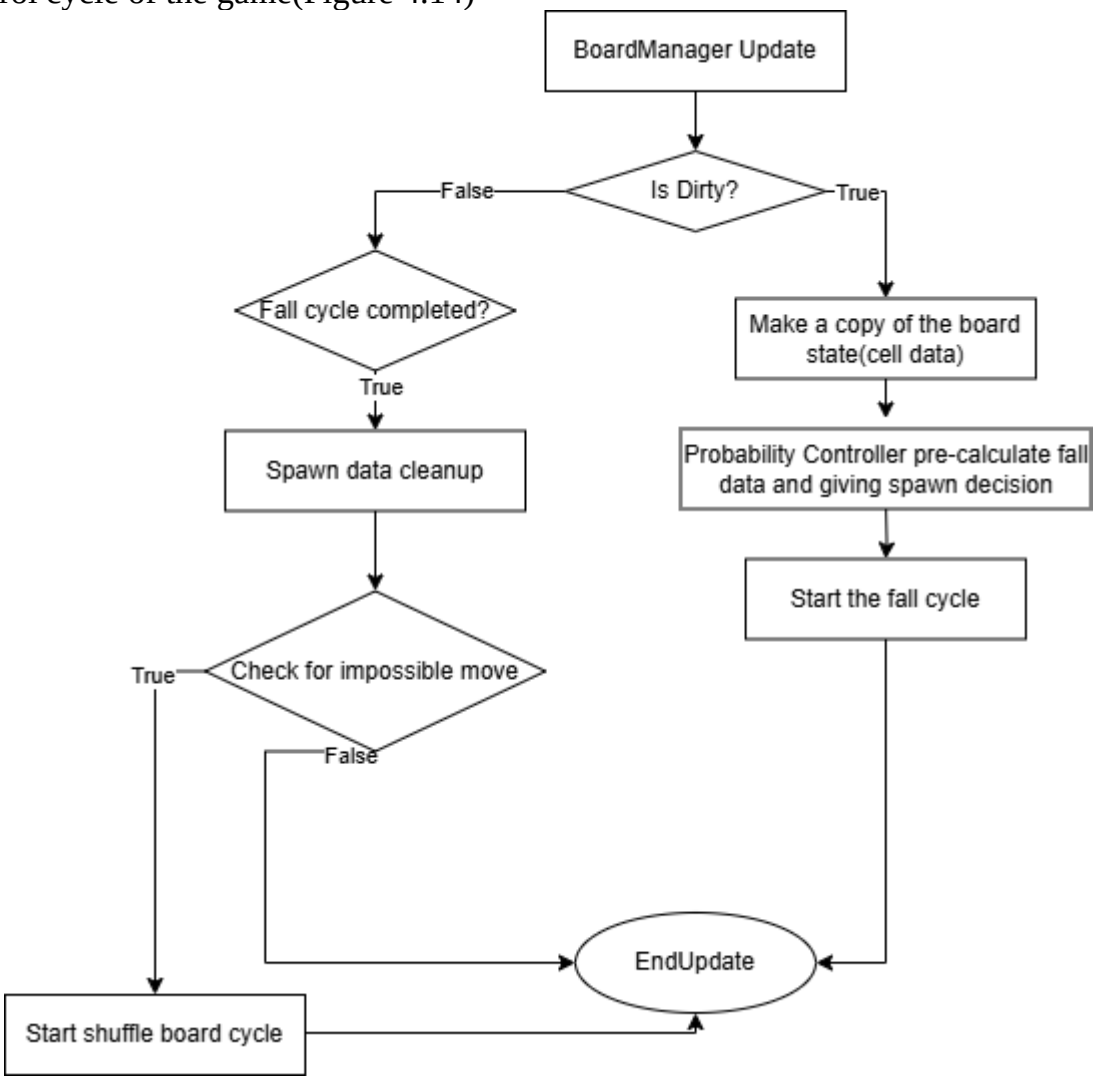

Figure 4.14: Board update cycle

The Probability controller always update it calculation cycle before the fall cycle ensuring the spawned item are always predicted first.
The calculation only performed when there are changed to the game board, this is a big overhead since every match made during the fall cycle will invalidate its previous prediction, forcing a recalculation and an recycle of the fall system. This is why a simpler representation of the board are created each time the calculation begin, keeping every calculation only occur in a simple data structure.

4.7.2. Probability calculation cycle

In order to keep calculation overhead at the minimum, only simple conditional check and basic math operation are used in the calculation cycle shown bellow(Figure 4.15)

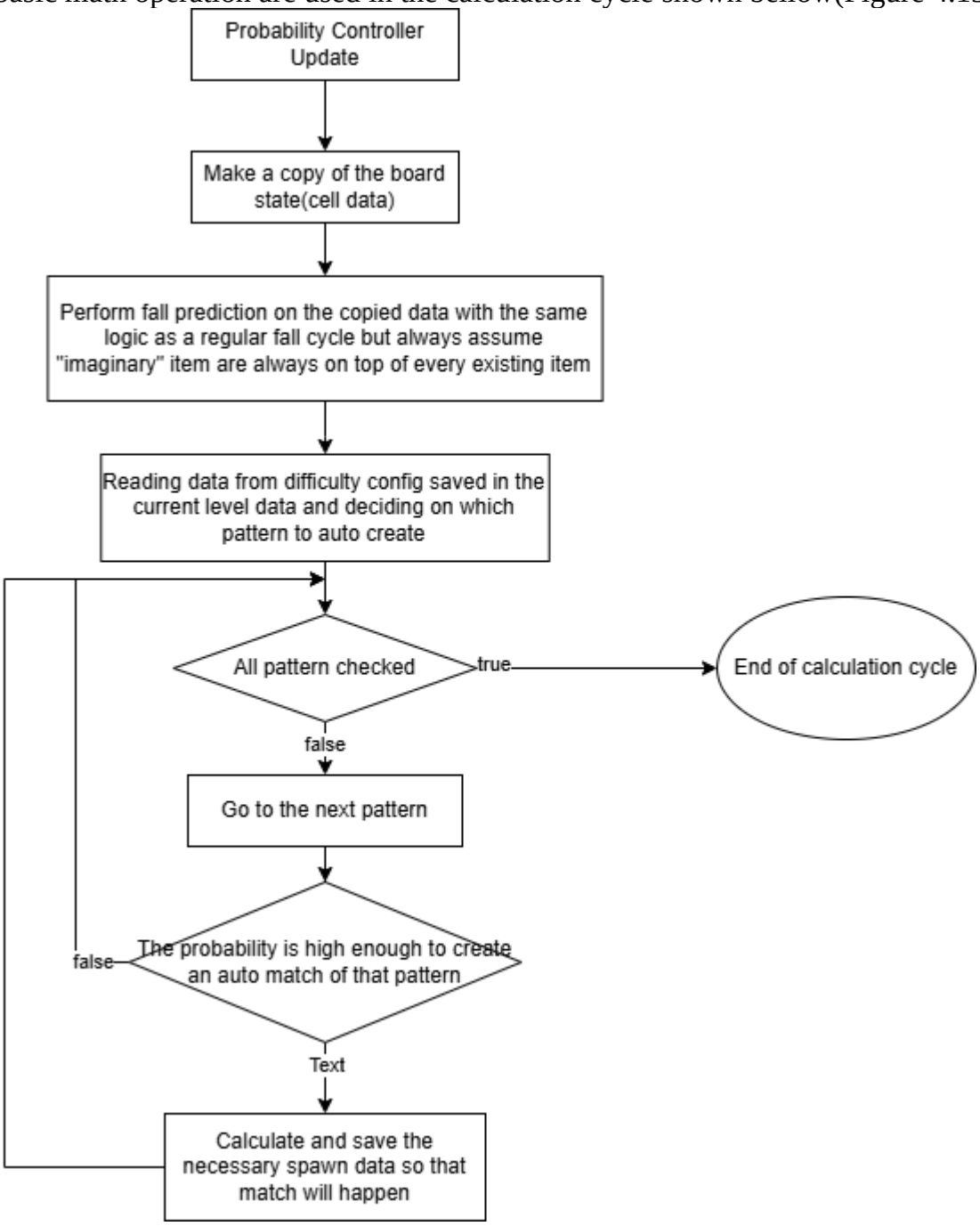

Figure 4.15: Probability calculation cycle

The resulting spawn data will be passed to the FallController system for the next fall cycle.

Chapter 5: Implementation and result

5.1. Introduction

This is one of the interesting chapter that contain lots of image showing game feature in action and how many of the game design look in the final product

All image in this section are capture on the following environment:

Device model: Samsung galaxy S20

Android version: android 13

Build Configuration: Debug

Target Framerate: 120Fps

Texture Compression level: Medium

Note: All game asset used in this project are available publicly for free, otherwise they are replaced by place holder asset from an asset pack *"Royal Match Game Assets"* available on *freegameassets.net*

5.2. Main menu

Main menu UI and Play level popup

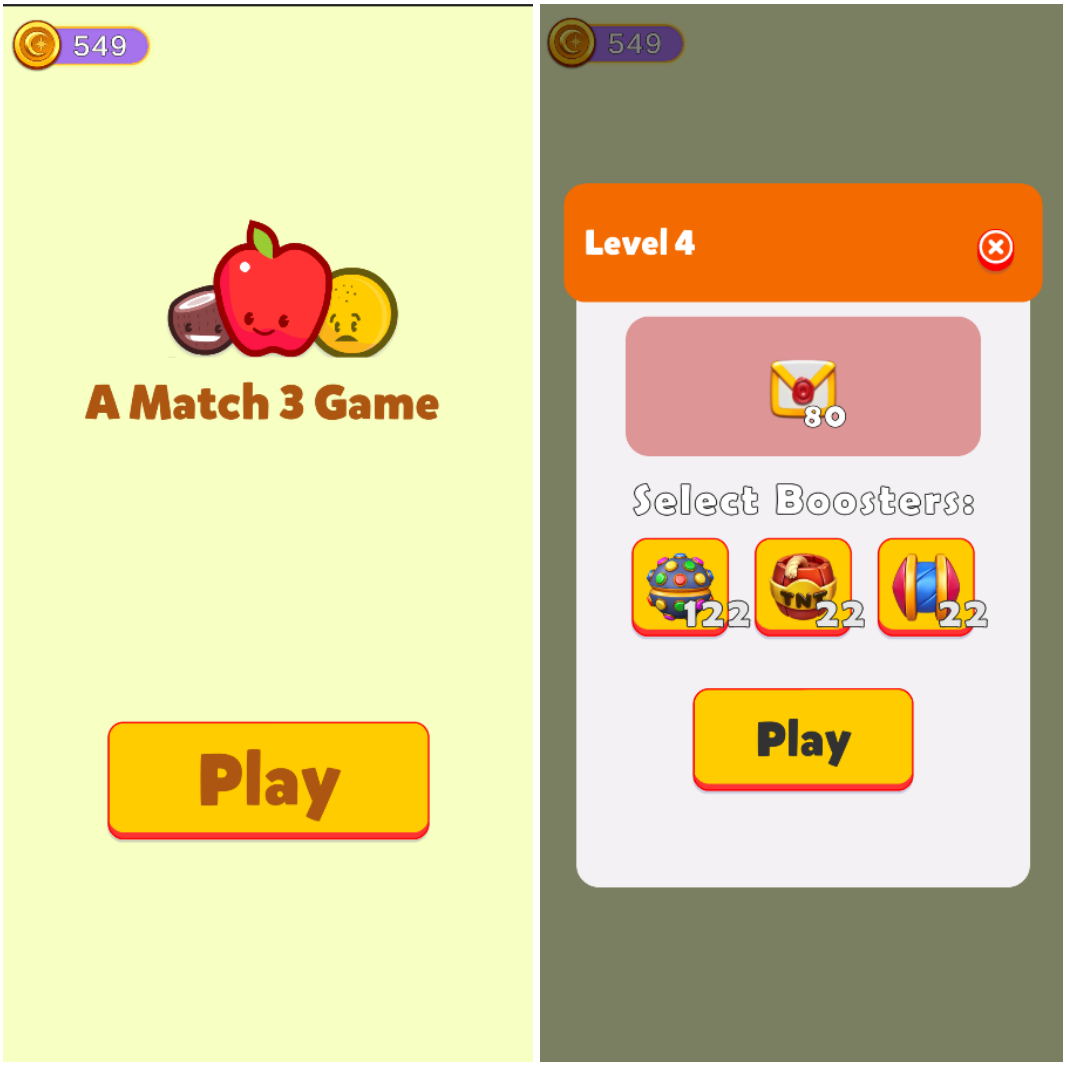

Figure 5.1: Main menu

5.3. Main game

Main game UI and some game play screen shot:

Main game(Figure 5.2)

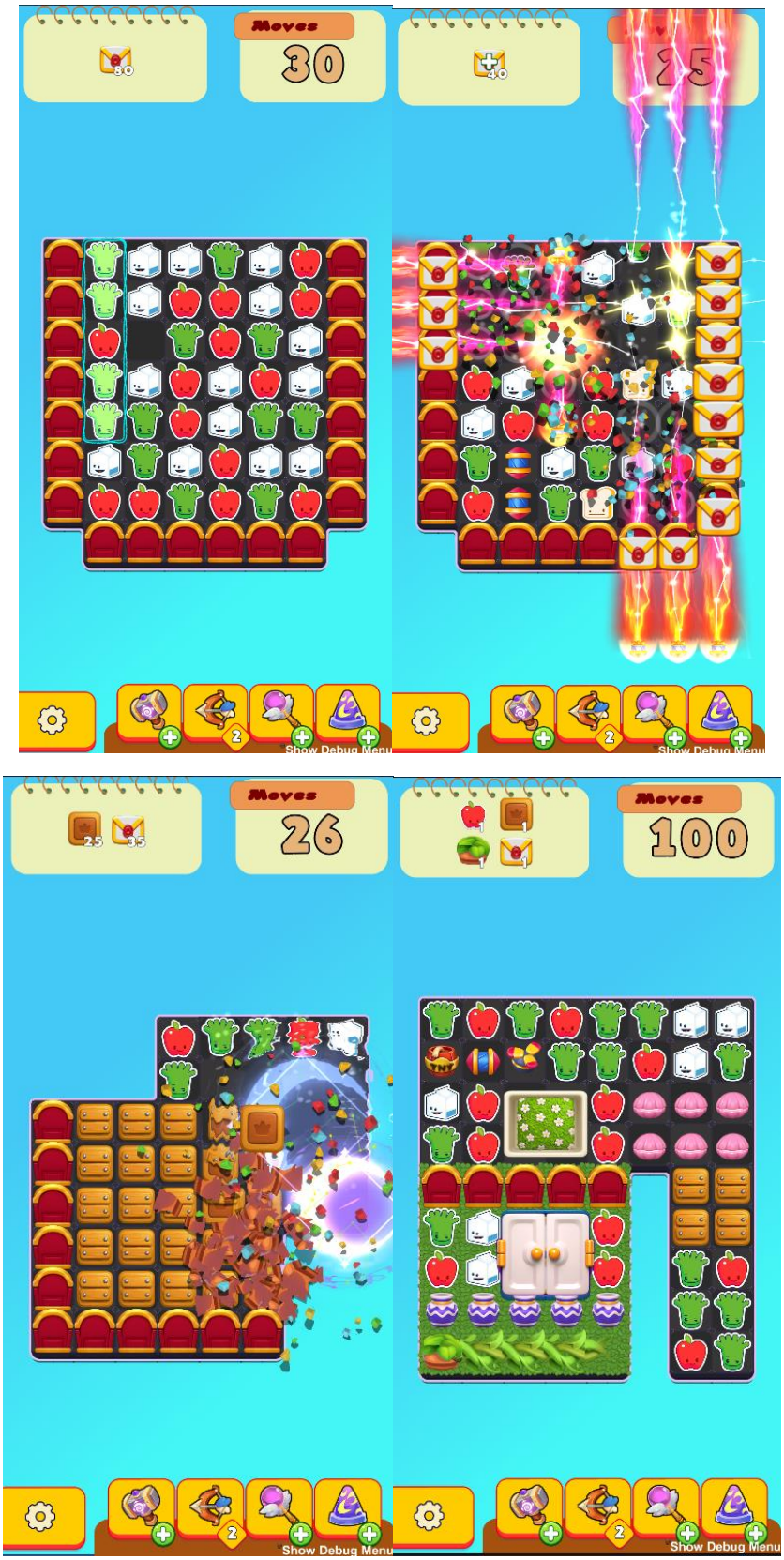

Figure 5.2: Main game

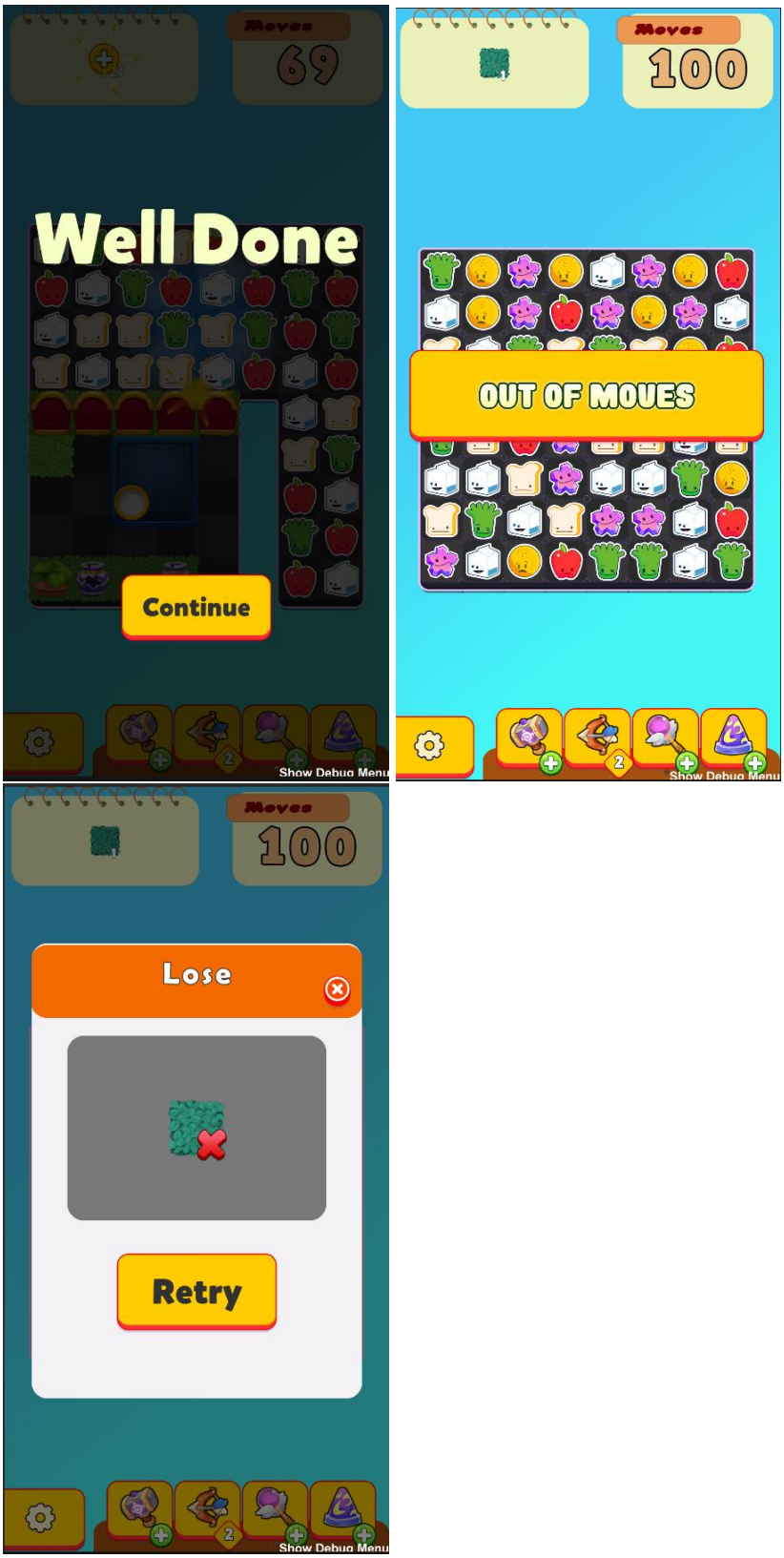

Figure 5.3: Main game popup

Setting menu and setting popup (Figure 5.4)

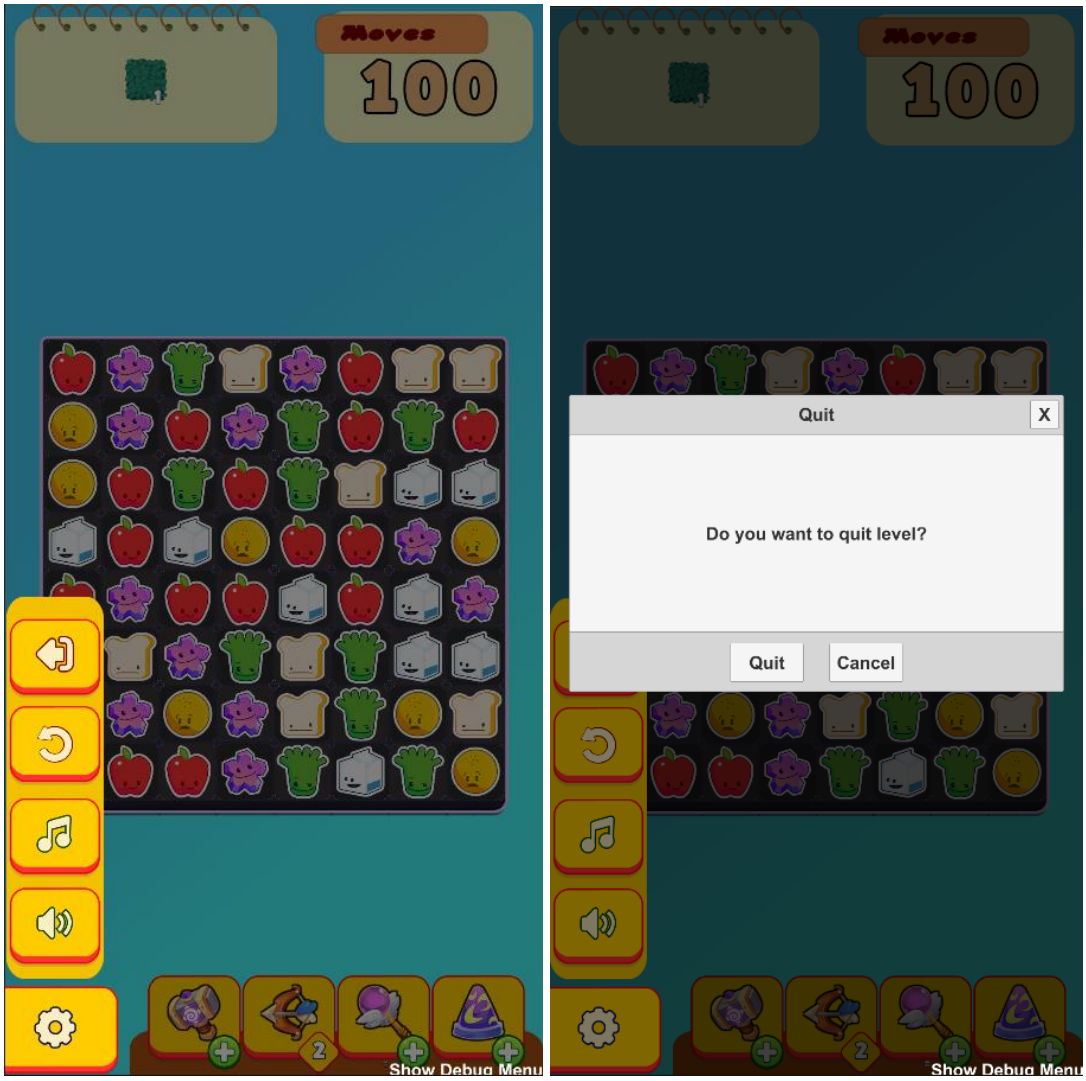

Figure 5.4: Setting menu

In game Booster (Figure 5.5)

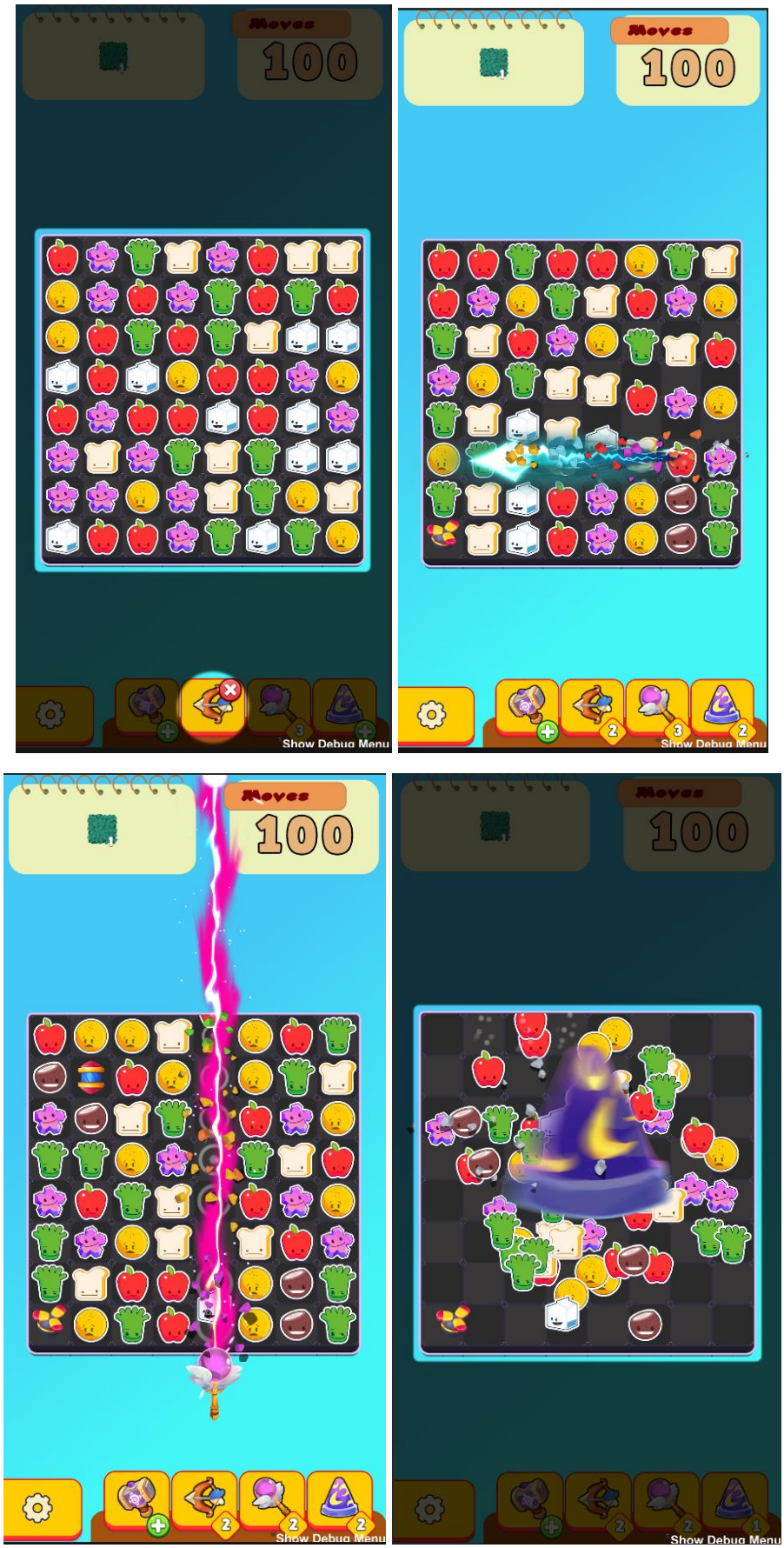

Figure 5.5: Booster

Debug menus (Figure 5.6)

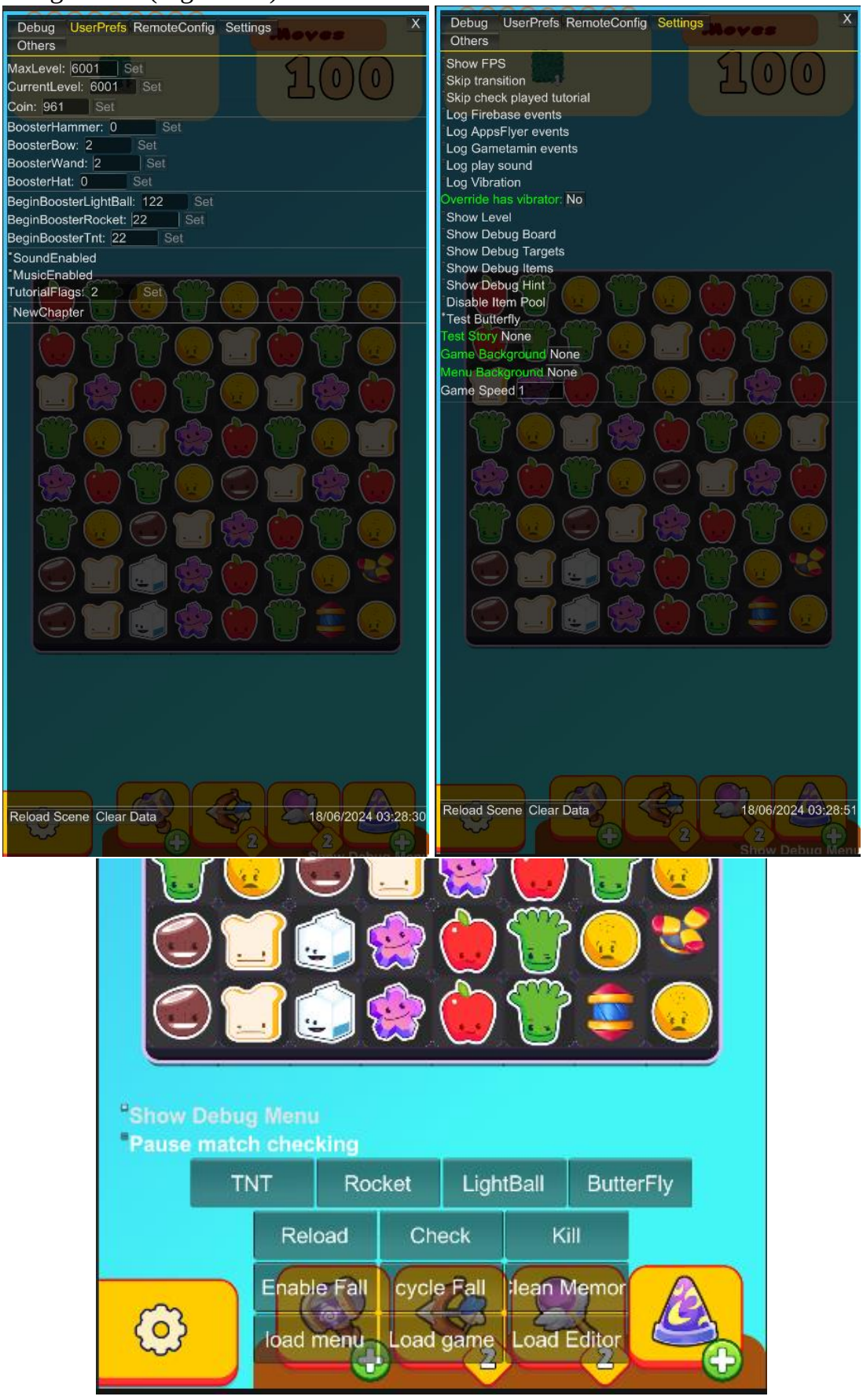

Figure 5.6: Debug Menu

5.4. Level editor

This is a unity application developed to design level for this game

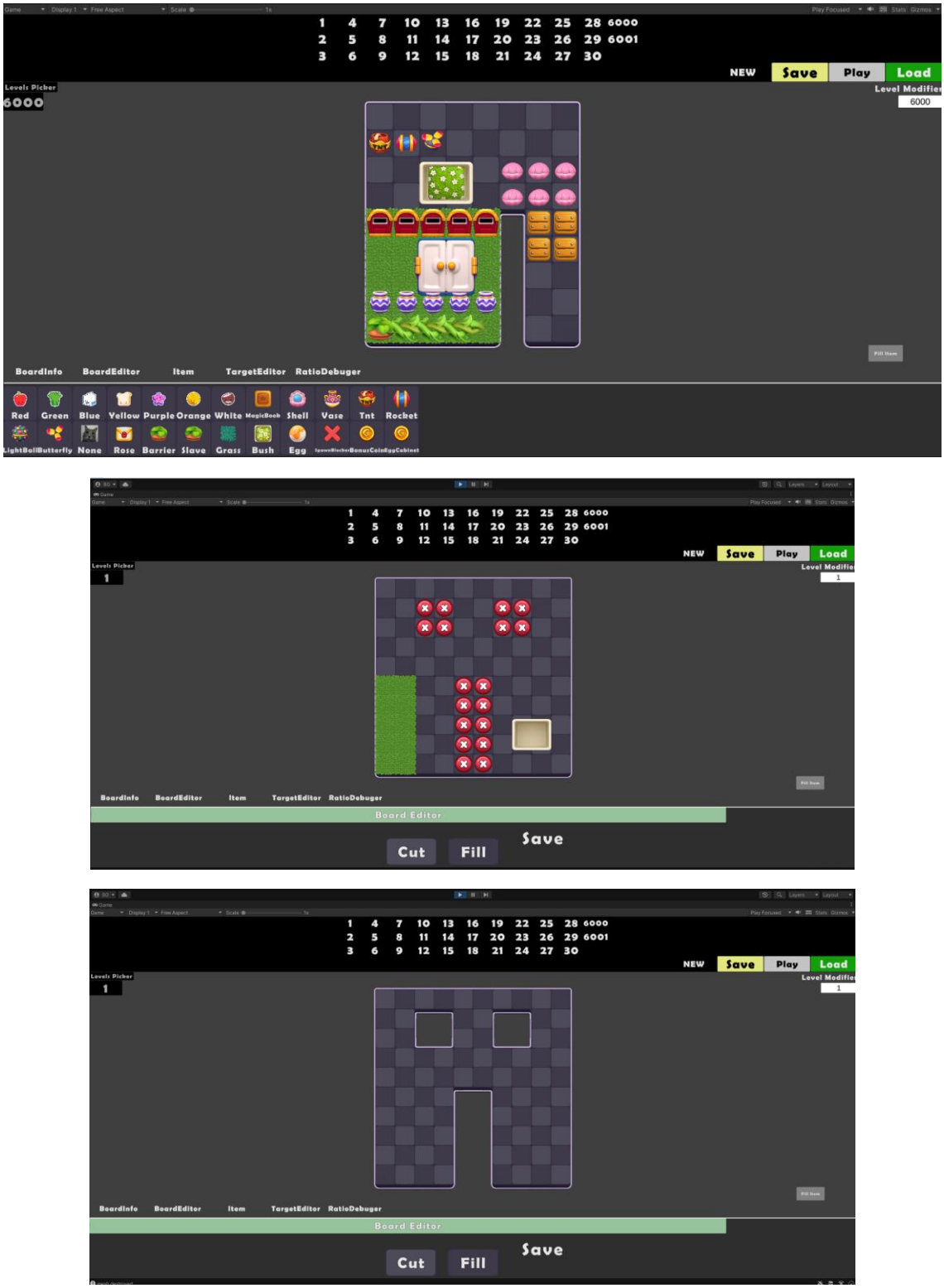

Figure 5.7: Level editor

Here are all the functionality the level editor provide:

Board Creation Tool(Figure 5.8)

Used for creating a new game board.

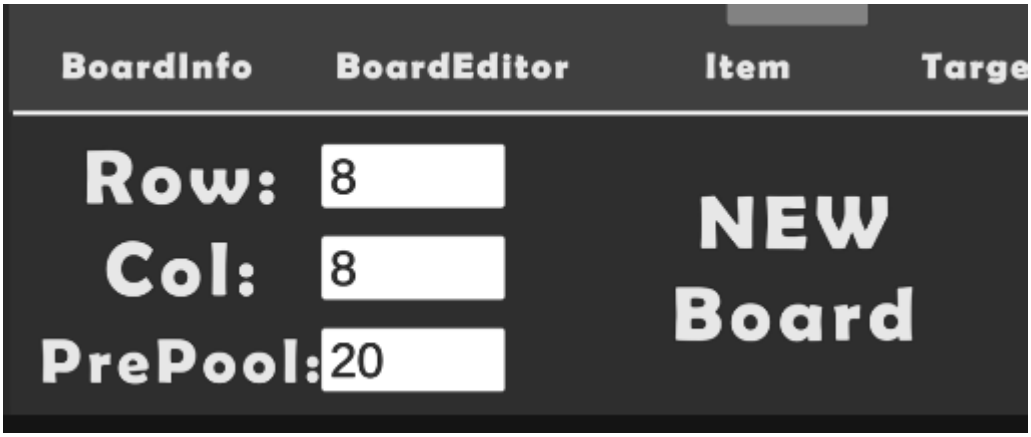

Figure 5.8: Board creation tool

Description:

- Row/Col: the amount of row the new board will have
- PrePool: The amount of item preloaded into the item pool for performance optimization, use full for big board.
-

Board Editor Tool(Figure 5.9)

Used for cutting or filing the board shape.

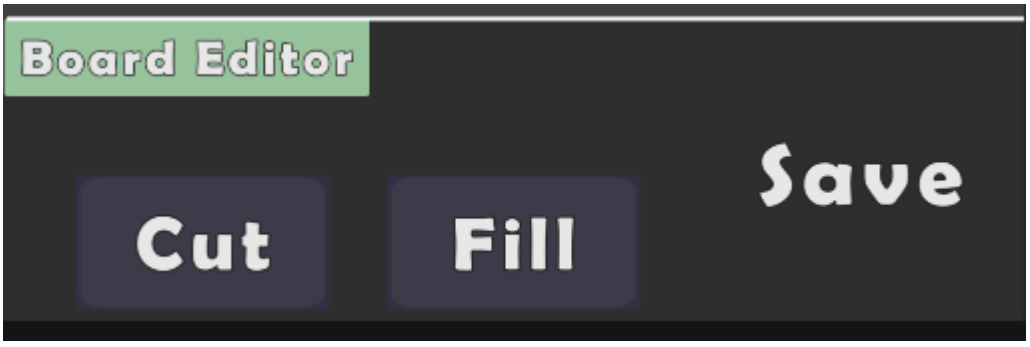

Figure 5.9: Board editor tool

Description:

- Cut/Fill: select an option and click on a cell to perform cut/fill

Item Tool(Figure 5.10)

Used for placing item onto the board

Figure 5.10: Item tool

Target Editor Tool(Figure 5.11)

Used for setting target for current level

Figure 5.11: Target Editor Tool

Ratio Debuger Tool(Figure 5.12)

Used for debugging the probability calculation cycle

Figure 5.12: Ratio Debugger Tool

Description:

- Update view: update the current editor board to display the data calculated
- Update Simple Data: clear and make a new copy of board data
- Calculate result: calculate the spawn data
- Next Fall Step: Perform fall on the copied data
- Calculate Fall: repeating Next Fall Step until fall is complete

CONCLUSION

1. The Achievements

- Growing creative thinking and problem solving capability.
- The game has achieved a playable and enjoyable state.
- All the core system has been implemented and with a carefully designed code base and structure allow for easy update/extension.
- Learned to design and implement a complete game system.
- Learned new coding practice, algorithm and improved optimization skill.

2. The Obstacles

- The time budget is tight for a game developed by one person so the level of bugs and glitch are still high.
- Finding and editing game asset are time consuming. So the game still used a lot of placeholder asset, resulting in a disconnected art style.
- The quality of UI/UX are low.

3. Future plans

- Design more level for the game
- Improve the consistency of the art style
- Bug fixes

REFERENCES

Internet

[1] From Bejeweled to Candy Crush: Finding the key to match-3

https://www.polygon.com/2014/2/26/5428104/from-bejeweled-to-candy-crush-finding-the-key-to-match-3

[2] Royal Match Wiki**:** https://royalmatch.fandom.com/wiki/Royal_Match_Wiki

[3] Royal Match Game Asset: https://freegameassets.net/product/royal-match-game-assets/

[4] The Devil's Work.shop Match 3 Game asset : https://devilsworkshop.itch.io/match-3-free-2d-sprites-game-art-and-ui

[5] The Good and the Bad of C#: https://www.altexsoft.com/blog/c-sharp-pros-and-cons/

[6] MonoBehaviour: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html

[7] The Advantages and Disadvantages of Unity Game Development: https://medium.com/@devstree.au/the-advantages-and-disadvantages-of-unity-game-development-ff3d8b177dd0

APPENDIX1

APPENDIX 2