Technical Design Document: Push Sushi

Table Of Content

1.0 Introduction........ 3
2.0 Game engine........ 3
2.1 Main components used........ 3
3.0 Programming language........ 4
3.1 Naming convention........ 4
4.0 Features & Systems........ 5
4.1 Pawns Movement System........ 5
4.2 Undo Moves........ 5
4.3 Hint........ 5
4.4 Score System........ 5
4.5 Save System........ 5
4.6 Switch Theme System........ 5
5.0 Code overview........ 6
5.1 Managers........ 6
5.1.1 HUD Manager........ 6
5.1.2 Level Manager........ 6
5.1.3 Undo Manager........ 6
5.1.4 Sound Manager........ 7
5.1.5 Tutorial Manager........ 7
5.1.6 Score Manager........ 7
5.1.7 Save Manager........ 7
5.2 Components........ 8
5.2.1 Slidable Components........ 8
5.2.2 Undo Components........ 9
5.2.3 Skin UI Component........ 9
5.2.4 Level UI Component........ 9
5.2.5 Coin........ 9
5.3 User Interfaces (UI)........ 10
5.3.1 UI Window........ 10
5.3.2 Theme Setter........ 11
5.3.3 Levels Grid UI Handler........ 11
5.3.4 Game Over Tab........ 11
5.4 Datas........ 12
5.4.1 HUD Data........ 12
5.4.2 Level Data........ 12
5.4.3 Menu Data........ 12
5.5 Utilities........ 13
5.5.1 Grid<TGridObject>........ 13
5.5.2 Tile........ 13
5.5.3 Constants........ 13

5.5.4 Level Loader........13
5.5.5 Coin Spawner........13
5.6 Tool........14
5.6.1 Level Editor Tool........14
5.6.2 Level Tester........14
5.6.3 Toggle........14
6.0 Level Editor Tool Usage........15
6.1 How To Create a New Level........15
6.2 Test Created Levels........16
6.3 Additional Information........16

1.0 Introduction

This document provides an overview of the architecture and features associated with the development of "Push Sushi."
This TDD also aims to guide the development team through the implementation process of the game, ensuring a clear understanding of the system components, technical choices, and fundamental interactions that characterize the game. It also serves as a reference resource for developers, designers, and stakeholders involved in the project.
Project repository at this link.

2.0 Game engine

"Push Sushi" will leverage the Unity game engine for its development because it provides a versatile and powerful platform, well-suited for 2D puzzle games.
The Unity version used is the 2022.3.10.f1

2.1 Main components used

- Collider;
- Camera;
- Physics Raycaster;
- Event System;
- Sprite Renderer;
- Mesh Renderer;
- Image;
- Button;
- Toggle;
- Canvas;
- Canvas Scaler;
- Graphic Raycaster;
- Animator;

3.0 Programming language

The primary programming language used is c#

3.1 Naming convention

Naming Convention Scripts		
Script	Visibility	Convention
Class Member variable	Private	_camelCase
Class Member variable	Public	PascalCase
Class Member variable	Protected	_camelCase
Class Member variable	Internal	camelCase
Local Variable	///	camelCase
Properties	Public	PascalCase
Methods	Private	PascalCase
Methods	Public	PascalCase
Methods	Protected	PascalCase
Methods	Internal	PascalCase
Constants	Private	SNAKE_CASE
Constants	Public	SNAKE_CASE
Constants	Protected	SNAKE_CASE
Constants	Internal	SNAKE_CASE

4.0 Features & Systems

Below are the main mechanics of the game. For more detailed information, see the GDD

4.1 Pawns Movement System

The game pieces can be dragged by sliding the finger along the direction in which they are oriented while remaining within the limits of the map or within those determined by the other pieces on the field or in other words the pieces must not penetrate each other.
More detailed information here.

4.2 Undo Moves

In the HUD there is a button that cancels the last move.
It is possible to go back in the moves up to the initial configuration of the playing field.
More detailed information here.

4.3 Hint

When the player clicks on the hint button in the hud, he is guided in solving the level through UI elements that indicate which pawn to move and in which position to place it.
To do this it is necessary to develop a pathfinding algorithm capable of automatically solving the level.
the team's choice was to use A* to implement this feature.
More information on hint feature here.

4.4 Score System

System that calculates the score at the end of each game based on the number of moves made and on the hints and undos used to solve the level.
More detailed information here.

4.5 Save System

Saving system for all the information that keeps track of the player's progression through the levels, scores achieved and coins collected.
More detailed information here.

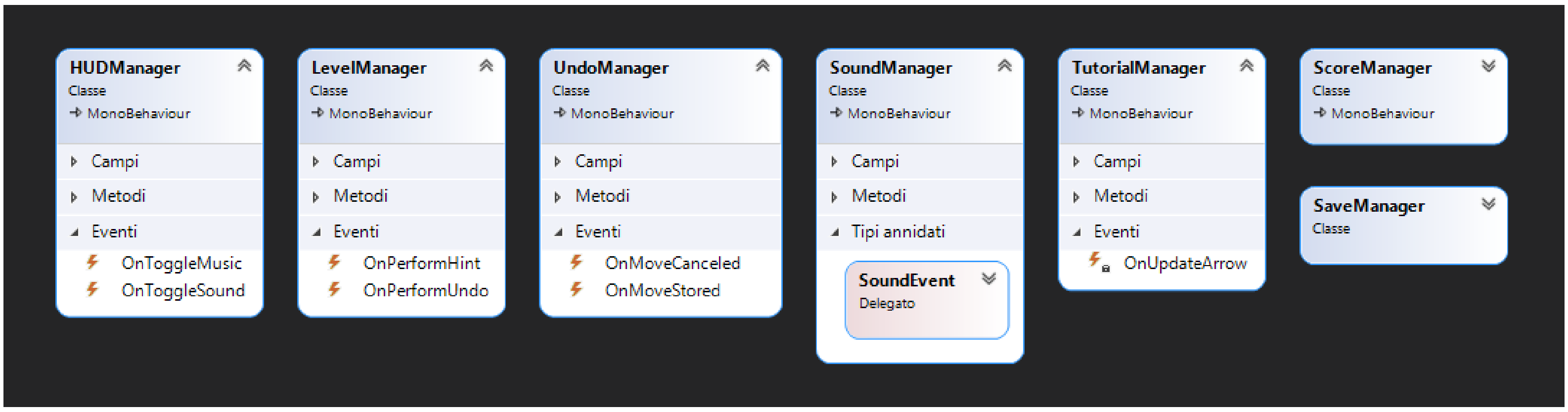
4.6 Switch Theme System

Each theme has a different HUD. The switch theme system will have to manage the change of the hud theme based on the theme choice made by the player in the menu.

5.0 Code overview

This section provides a general overview of the code structure.

5.1 Managers



5.1.1 HUD Manager

Responsible for managing all HUD elements present in the game scene, including UI elements in the pause window.

Events:

- **OnToggleMusic**: is called when you click the music toggle in the pause menu;
- **OnToggleSound**: is called when you click the sound toggle in the pause menu;

5.1.2 Level Manager

Responsible for managing the game state, loading the level, executing the Undo, Restart and Hint buttons and quickly switching between the next and previous levels.

Events:

- **OnPerformHint**: is called when you click the hint button;
- **OnPerformUndo**: is called when you click the undo button;

5.1.3 Undo Manager

Responsible for managing undo, the feature that allows you to go back in the moves you have performed.

Events:

- **OnMoveCanceled**: is called after the last move was canceled;
- **OnMoveStored**: is called after the last move was performed;

5.1.4 Sound Manager

responsible for managing and playing all sounds in the game.

Events:

- **SoundEvent**: each sound event is a delegate that can be called from anywhere in the code

5.1.5 Tutorial Manager

This manager is activated only during level one of each theme and is responsible for managing the tutorial.

Events:

- **OnUpdateArrow**: private event called when the player move the suggested pawn in the indicated position;

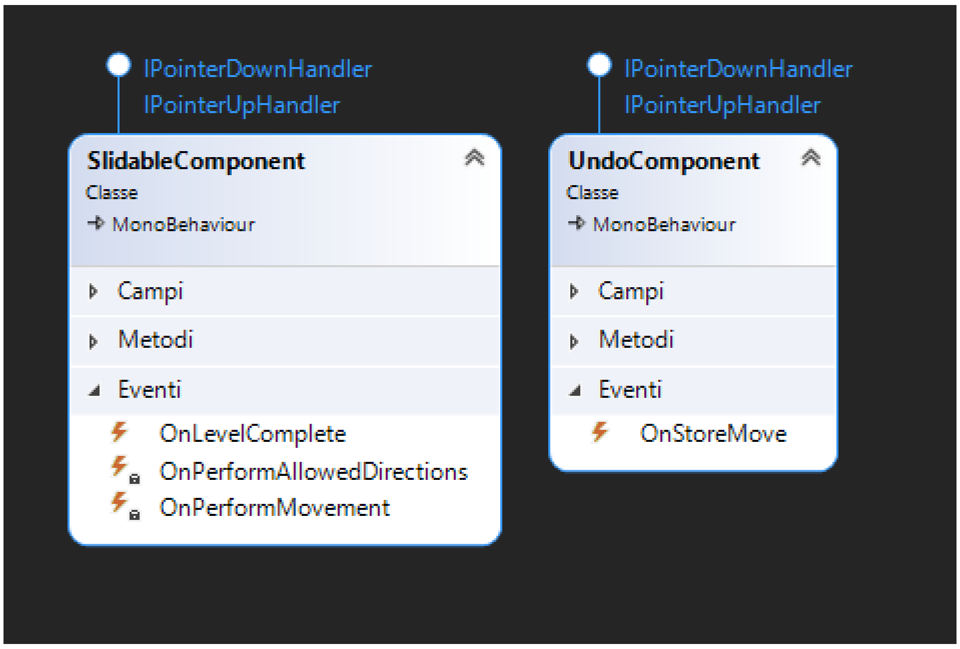
5.1.6 Score Manager

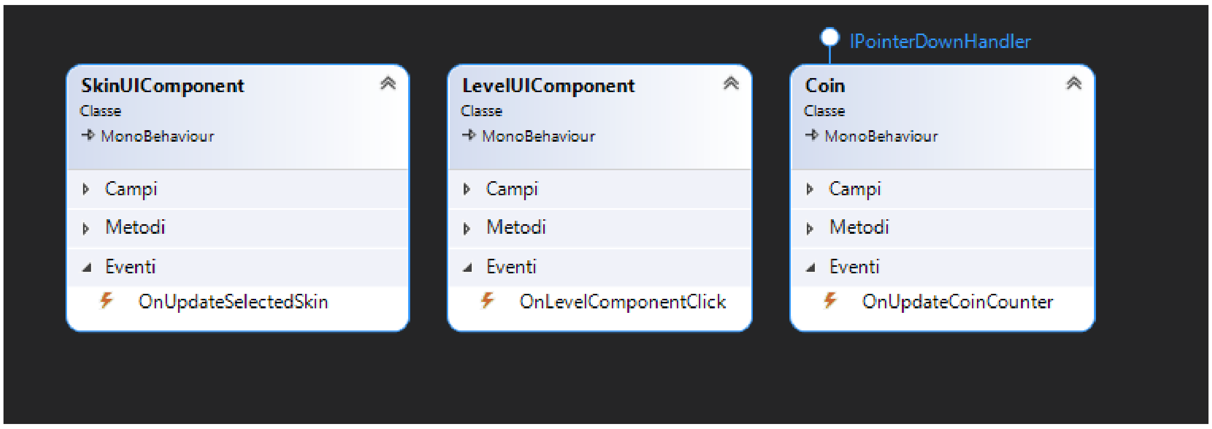
Responsible for counting the moves and calculating the score at the end of the game, it also saves the level data using the SaveManager after it has calculated the score.

5.1.7 Save Manager

contains static methods that allow you to save the data that requires it to disk, to do this it uses unity playerprefs.

5.2 Components





5.2.1 Slidable Components

Component that makes a pawn draggable within the game grid along the direction in which the pawn is oriented

Events:

- **OnLevelComplete**: is called when you click on the main pawn if the exit way is free from obstacles

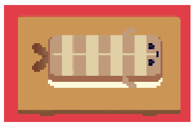
5.2.2 Undo Components

Component needed for the undo feature, checks whether the pawn has been moved and released to a new position on the grid

Events:

- **OnStoreMove**: is called when the pawn is released on a new position;

5.2.3 Skin UI Component

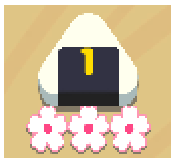


Component attached to skin icons (prefab) in the skin selection window

Events:

- **OnUpdateSelectedSkin**: is called when you click on the skin icon;

5.2.4 Level UI Component



Component attached to level icons (prefab) in the level grid UI menu window

Events:

- **OnLevelComponentClick**: is called when you click on the level icon;

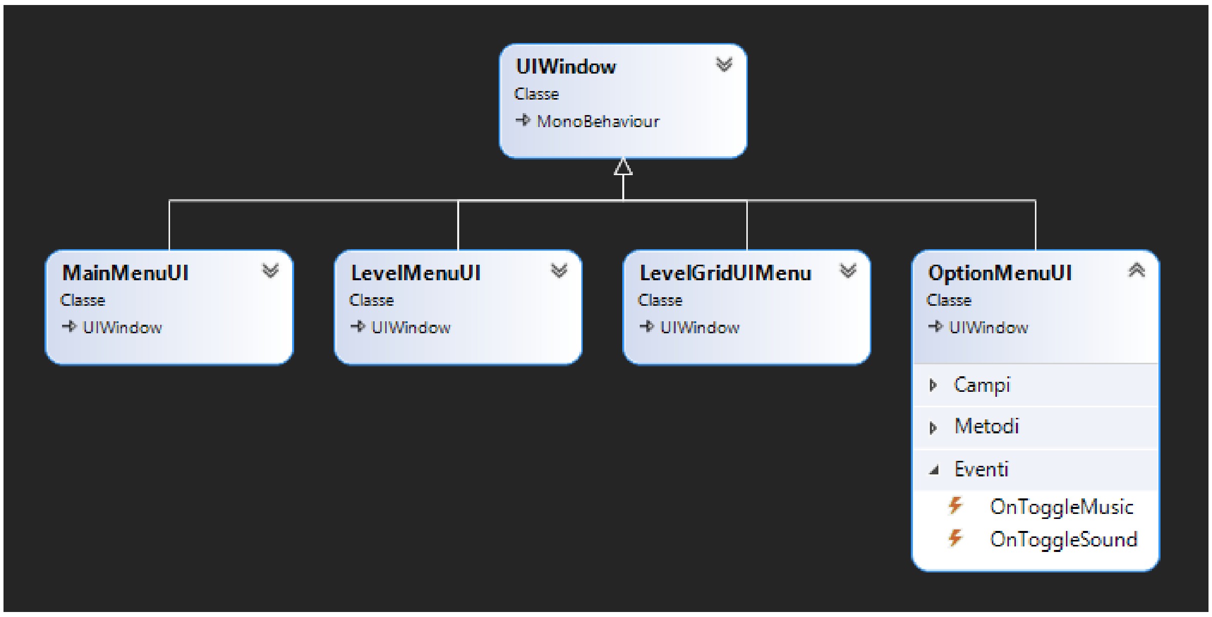
5.2.5 Coin

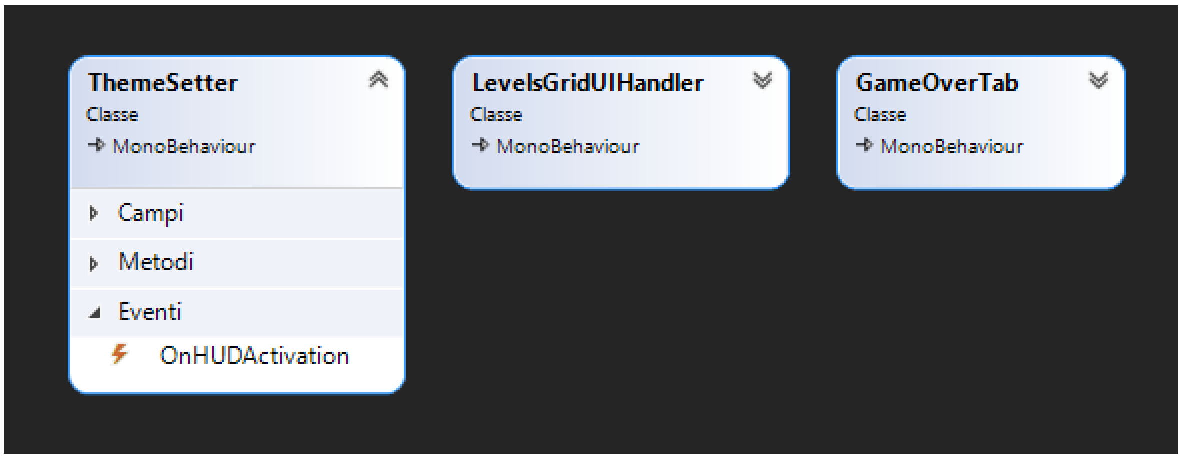
Component attached to the coin prefab.

Events:

- **OnUpdateCoinCounter**: is called when you click on a coin during the game;

5.3 User Interfaces (UI)

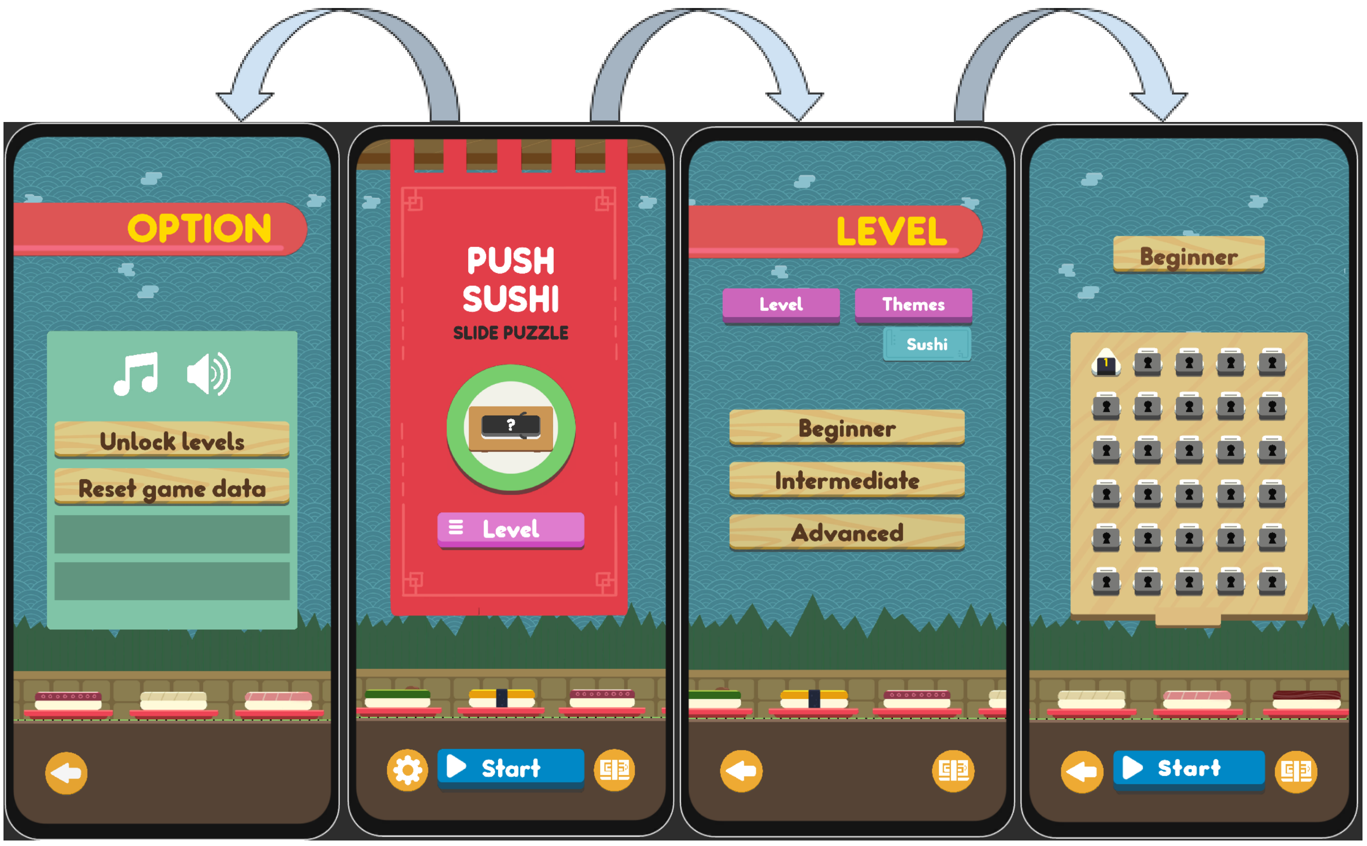




5.3.1 UI Window

There are 4 windows that make up the menu interface, and each of them has a script that contains the references of its UI elements (Buttons, Texts etc...) and manages the related events.
The 4 windows that inherit from the UIWindow base class are:

- **MainMenuUI**: First window displayed when opening the game;
- **LevelMenuUI**: Accessible from the main menu by pressing the Level button;
- **LevelGridUIMenu**: You get there after selecting the difficulty you want to play;
- **OptionMenuUI**: Accessible from the main menu by pressing the settings icon;



OptionMenuUI MainMenuUI LevelMenuUI LevelGridUIMenu

5.3.2 Theme Setter

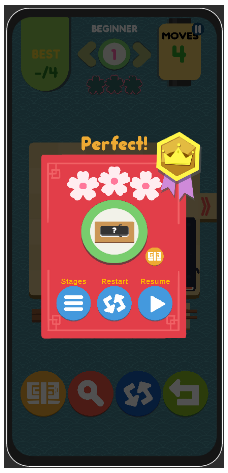
There are 3 different themes (sushi, penguins, sweets) and each of them corresponds to a different HUD, the ThemeSetter class takes care of activating the correct HUD based on the choice selected by the player in the LevelMenu.

5.3.3 Levels Grid UI Handler

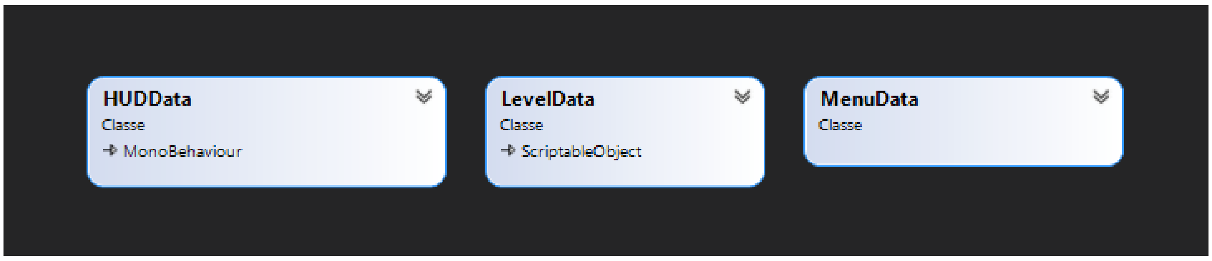
This class manages the grid of level icons present in the LevelGridUIMenu

5.3.4 Game Over Tab

Script attached to the game over tab, manages the UI elements present in the tab and shows the correct icons (flowers, fish or candies) based on the score achieved by the player at the end of each level



5.4 Datas



5.4.1 HUD Data

Contains the references of all the UI elements present in the hud, passed through the inspector.

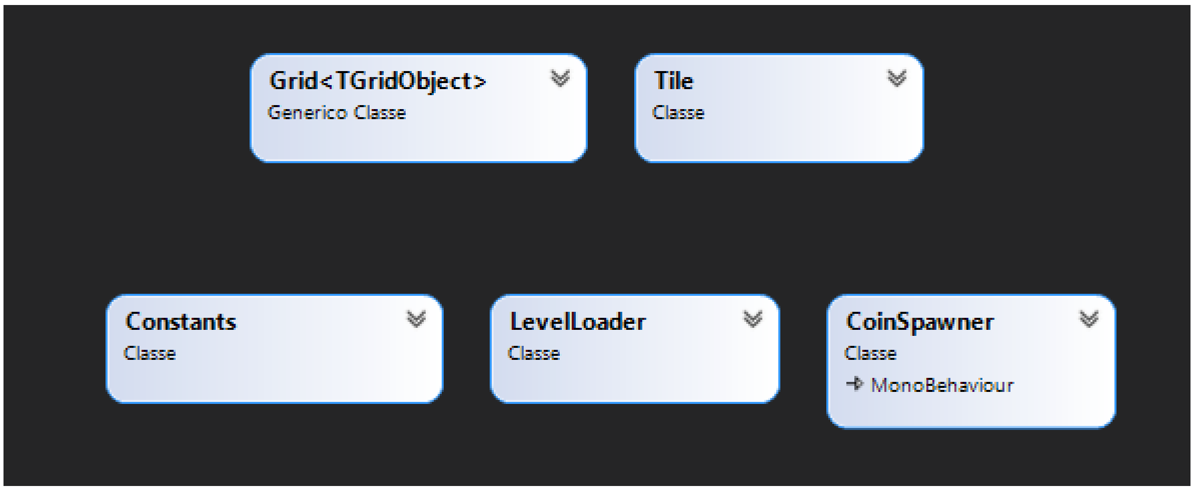
5.4.2 Level Data

Scriptable object that contains data on the position, rotation and type of pieces present in the level, theme, difficulty, level index and minimum number of moves to obtain the maximum score.
Each level is a scriptableobject that is loaded via Resources.Load and its path in the **Assets/Scriptableobject/Levels/Resources** folder is determined by the triad **Theme/Difficulty/Level Index**

5.4.3 Menu Data

Contains data relating to the selections made by the player in the menu.

5.5 Utilities



5.5.1 Grid<TGridObject>

This class represents a grid of generics TGridObject, mainly used to represent the playing field.

5.5.2 Tile

Class containing two integer coordinates x and y and a constructor, used as a representation of a grid tile.

5.5.3 Constants

This class contains only static members used for the constants.

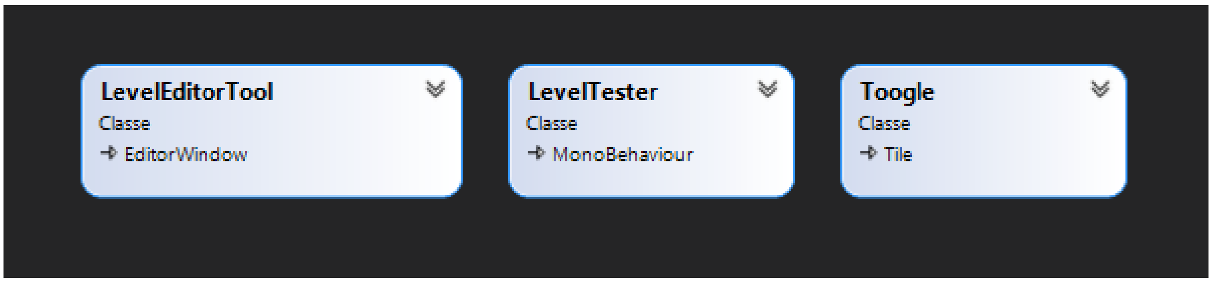
5.5.4 Level Loader

This class contains static methods necessary for loading levels (ScriptableObject).

5.5.5 Coin Spawner

Manages the spawn of coins at the start of each game

5.6 Tool



5.6.1 Level Editor Tool

EditorWindow class relating to the tool for creating levels.

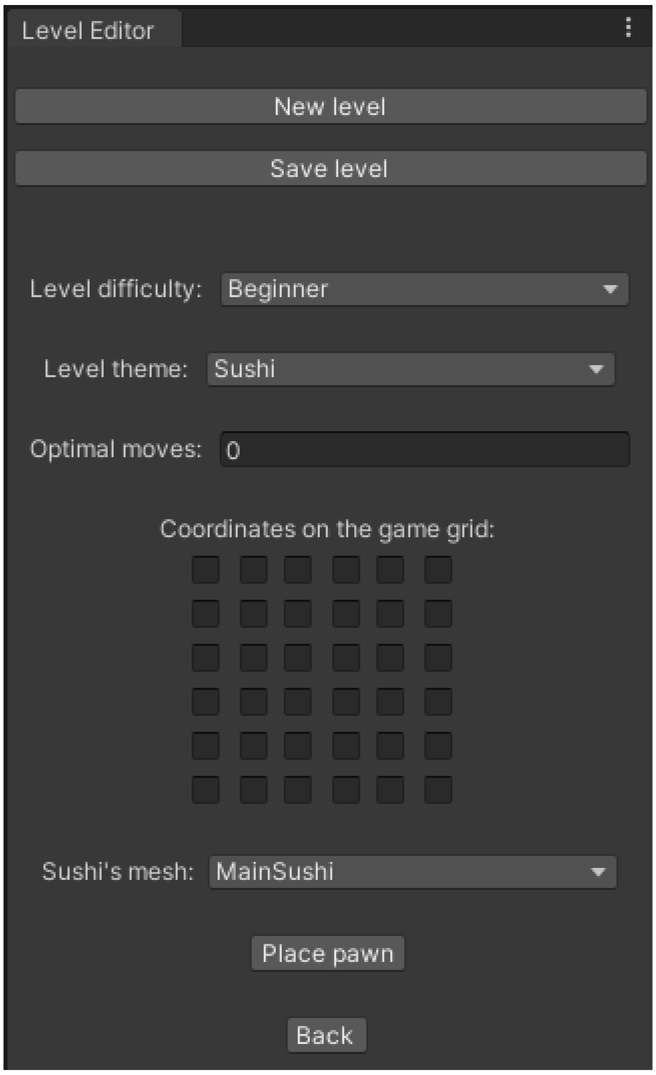
5.6.2 Level Tester

Used only in the "EmptyLevelTemplate" scene to test the level that is passed to it via inspector in the LevelToTest member.

5.6.3 Toggle

Inherits from Tile and represents the single tile of the toggle grid in the level editor tool UI

6.0 Level Editor Tool Usage


Level Editor

New level

Save level

Level difficulty: Beginner

Level theme: Sushi

Optimal moves: 0

Coordinates on the game grid:

Sushi's mesh: MainSushi

Place pawn

Back

6.1 How To Create a New Level

1. Open the tool window from Tools > Level Editor;
2. Click on the New Level Button;
3. Select the level difficulty;
4. Select the level theme;
5. Enters the optimal moves for obtaining the highest score;
6. Start to place the pawn by selecting the pawn mesh and then selecting on the grid's toggles the head and tail position of the pawn;
7. Once all the pawns are positioned correctly click on Save level to generate the level's scriptableobject;

6.2 Test Created Levels

If you want to test the newly created level, select the LevelTester in the Hierarchy and in the Inspector drag the scriptable object of the level you want to test into LevelToTest.
You can find all the levels scriptable object in the folder:
Assets/**ScriptableObjects**/**Levels**/**Resources**/{**LevelTheme**}/{**LevelDifficulty**}

6.3 Additional Information

1. Always click on the New Level button before starting to compose a new level;
2. Never edit or save changes in the "EmptyLevelTemplate" scene;
3. If you make a mistake in placing a pawn use the Back button in the level editor UI to delete the last placed pawn;
4. There are no controls on the selected position where you want to place a pawn, it's up to you to avoid placing pawns in the same point;