About

The Tegridy MatchTwo Controller is designed so you can easily implement a fun puzzle based mini game to your project with ease where the goal is to combine the matching symbols until either the maximum tile is reached or the board is full.

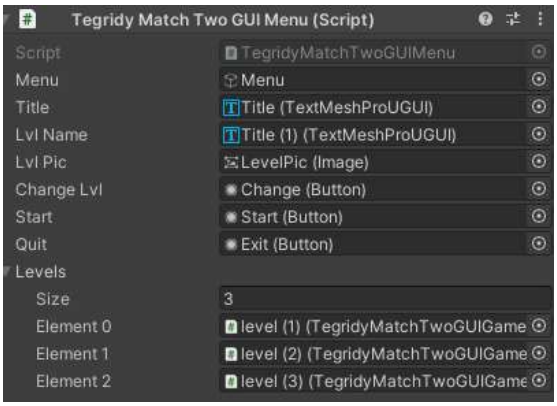
Usage

The controller can be used in stand alone mode with the TegridyMatchTwoMenu component this will provide a basic menu with a level select screen for your project and is demonstrated in the example scene. Levels can also be called directly for use as in game puzzles, door locks etc by using the TegridyMatchTwoInterface and passing it a level GUI and *AudioSource*. You can also implement your own interface by interacting with the TegridyMatchTwoController by defining a *GameGrid*, setting the win conditions and monitoring the *gameState*.

Support

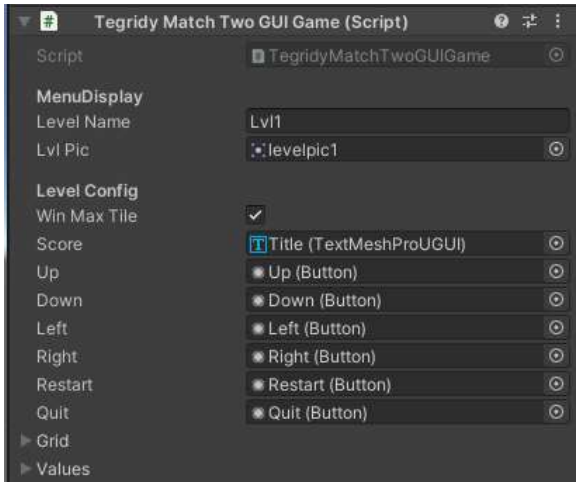
Support@tegridygames.co.uk

Menu Configuration



- **Menu** – This should be set to the GameObject that holds your menu and level GUI's
- **Title / LvlName** – If the *title* TextMeshPro object is set the application will set the text field to the *gameName* variable found in the language class and *lvlName* will be set if the number of *levels* matches *levelnames* length in the language class.
- **LvlPic** – This will be changed when changeLvl is pressed to the image set in that levels configuration.
- If *start*, *quit* or *change* have TextMeshPro text fields as children these will be set to what is defined in the language controller class.
- Levels – Once you have configured your levels GUI add the game object here.

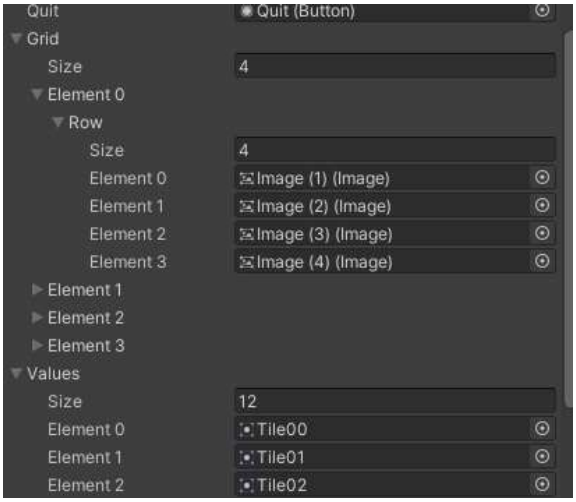
Level Configuration



- **LevelName** – Displayed on the main menu, if the number of levels set in the menu matches what is found in the language class then it will use that, otherwise it will default to the editor setting.
- **LvlPic** – Displayer on the main menu and should represent the level to be player.
- **WinMaxTile** – If this is set to true once the player gets a tile equale to the maximum number of Values then the game will end, otherwise the game will

go on until the board is full.

- **Score** – If set this will display the current score.
- **Up, Down, Left, Right** – These buttons control the direction of the game board moves, if they have a TextMeshPro text object as a child that will be set to the corresponding variable found in the language class along with restart and quit.



- **Grid** – You will need to set the length to the desired number of rows, and then set *row* to the desired number of columns. For each *row* add your image game objects in the inspector. You will need to define atleast a 2x2 grid and ensure all image fields are complete.
- **Values** – these images will be displayed as the game tiles, the first one should represent a blank tile and they should increment in the desired order you want them to progress. You will need to set at least 3 tiles for this array.

Class/Script Descriptions

TegridyMatchTwoController

- This class is used for Monitoring the game state and working out the game moves. It can be used in your own interface by handing it a configured TegridyMatchTwoGUIGame component when *StartGameGUI* is called.

TegridyMatchTwoInterface

- This class starts a game instance and requires a configured TegridyMatchTwoGUIGame component when *StartGame()* is called. If a *host* has been set once the level is complete that gameObject will be enabled when the menu is closed and if a *AudioSource* has been defined it will be used.

TegridyMatchTwoMenu

- Provides basic menu functionality, requires the *gui* field be filled with a configured TegridyMatchTwoGUIMenu component.

TegridyMatchTwoLanguage

- All strings used in the GUI will be set from here, if the *levelName* count does

not match the number of levels in the scene the program will use the names set in the editor for the levels instead.

TegridyMatchTwoGUI

- These scripts are used to define the GUI components In the editor and configure the levels settings.