Memory Game in Unity 2D

High Level Overview

This is a memory matching game built in unity. Upon startup, the player is presented with a 4x4 grid of cards (16 cards in total. At the start of the game, each card is flipped face up for a total of 7 seconds (as defined by TIME_TO_MEMORIZE in Controller.cs) to give the player a chance to memorize them. The cards are completely randomized in terms of order each time the player plays the game. After the 7 seconds are up, the cards are all flipped face down. The player must click on a card, and then try to click on another card which they think is the matching pair. If the two cards match, the player is allowed to continue. In this case, the player will continue with this process until all 8 cards are matched and the player is presented with a You Won! screen. However, if the player's second card flipped does not match the first, the game is over, and the player will be presented with a screen saying that they lost. Either way, if the player wins or loses, they have a chance to click the 'play again' button, where the game will reset.

GameScene

In the hierarchy of this game, there is a MainCamera which is centered at the center of the game (and the 16 cards) the camera is 800px by 800px. There is also a Controller object, which contains the Controller.cs component, which is used to control the game logic and will be explained in more detail later on. There is an EventSystem, which is not used due to the use of the controller instead. There is also a canvas, where each object is drawn onto the screen. Within the Canvas, we have the Card object. This is the main object which represents each of the 16 cards on the screen. The card object contains the Card.cs component, which is used to control the card behavior and will be explained in more detail further on. The canvas also contains a Background object, which was used to build the Game Over Screen. By default, the background is not visible, but then becomes activated when the player wins or loses the game as determined by GameOver.cs. Within the background, we have the Game over text, which simply states "Game Over", and the won lost text, which will change to "You Won!" if the player wins, or "You Lose!" if the player loses. Finally, we have a button which is used as the play again button and will restart the game when pressed

GameOver.cs

Gameover.cs is used to control the game over screen. When the GameIsOver() function is called from Controller.cs, the Setup() function is called where the game over screen is set to active and becomes visible. The player is no longer able to click on any of the cards, and is only able to press the play again button to play again. Also in the Setup() function, it is determined whether the text will display that the player won or lost, based on the result passed (0 or 1).

There is also a function to handle the pressing of the Play Again button, which would reload the game scene.

Controller.cs

Here is where most of the logic of the app takes place. In the start() function, we first clone the card class 16 times in 4 columns and 4 rows. The faces of the cards are randomized using a List and System.random. Once all the cards are cloned/created, they are added to a list to track all of them. There is a function startOfGame() which is called at the beginning of the game, and essentially flips the cards face up for TIME_TO_MEMORIZE seconds (default 7), and then flips the cards back face down so the player can begin to play. There is then a function called TwoFaces() which checks whether two faces have been turned up or not by checking the visibleFaces array, which indicates -1 for each card if there is no card face up. There is then a function to add faces to the visible faces array in order to track which cards have been flipped. Of course, there is a RemoveFaces() function that will remove a card from the visibleFaces array if it has been flipped back face down. There is then a CardsMatch() function, which is only called when TwoFaces() is true (two cards are face up). This card checks whether the two face cards are matching. If they are, they will be declared as matched, and the function will check whether the player has won the game yet or not (all cards face up). There is a gameWon() function which is called from CardsMatch() to check for these game wins. Then, there is a function GameIsOver() which initiates the game over screen.

MainCard.cs

This is the class that represents each card. This class will track whether or not the card has been matched, and whether the card is face up or face down, along with determining which sprite to render in the scene based on the faceIndex variable. There is a function showFace() and showBack() which is called from Controller classes startOfGame method, which will flip the face of the card up or down respectively.