DTT
APPS WEB GAMES

SINCE 2010

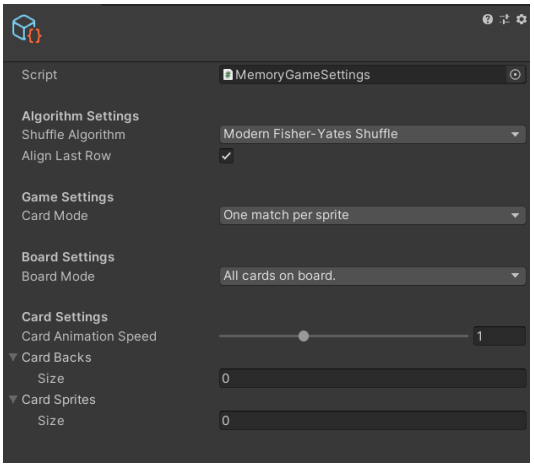
DTT Memory

Documentation | V1.0.0 | 20-01-22

Table of Contents

1. Get started quickly

To get started, create a game object and add the **MemoryGameManager** component to it, this will function as the manager of the minigame. For the minigame to work correctly a **Board** is needed. A prefab for the board can be found in the prefabs in the Demo folder.



Now that you have the manager setup you need to make a **MemoryGameSettings**. This will create a scriptable object on which you can define the rules of your game.

To create one, go to the project tab in Unity, right-click, and go to 'Create/DTT/MiniGame/Memory/GameSettings'. To be able to use the settings you must add at least one sprite for the card backs and one sprite for the card sprites.

To start the minigame, call the function StartGame(game settings) on the **MemoryGameManager** and provide it with an instance of the **MemoryGameSettings**.

When the game finishes, you can make use of the **Finished<MemoryGameResults>** event to gain insight into how the player performed in the game.

2. Introduction

DTT Memory is an asset for the Unity engine that allows you to easily implement a game where the player has to select two cards and match their images. The asset allows you to configure the difficulty aspects of the game and is open for extension. Using a Scriptable Object you can change the difficulty of the game, allowing you to have different difficulty settings per level.

3. Changelog

1. version 1.0.0 – Initial release

4. Set-Up

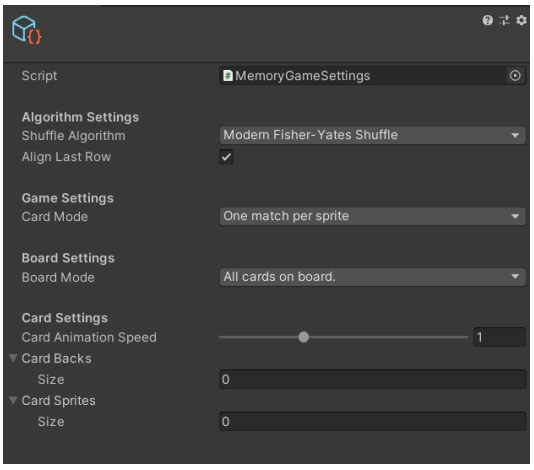
First, add the MemoryGameManager component to a game object in your scene. The MemoryGameManager holds a reference to the Board. The board has a **GridLayoutGroup** component and a prefab to create the Cards from. Prefabs for both the Board and a prefab for the Cards can be found in the 'Demo/Prefabs' folder.

If you want to make your own prefab for the Cards you can. A Card needs to have a content game object. Within the content, add another game object with a **button** component. The Card component holds a reference to both the content and the button component.

To set up the UI you can make use of the prefabs provided in the 'Demo/Prefabs' folder or you can create your own user interface. The MemoryUIManager script expects to have a paused and finished menu as well as one display text for the game results.
If you wish, you can make your own and use the API from MemoryGameManager to get the needed information.

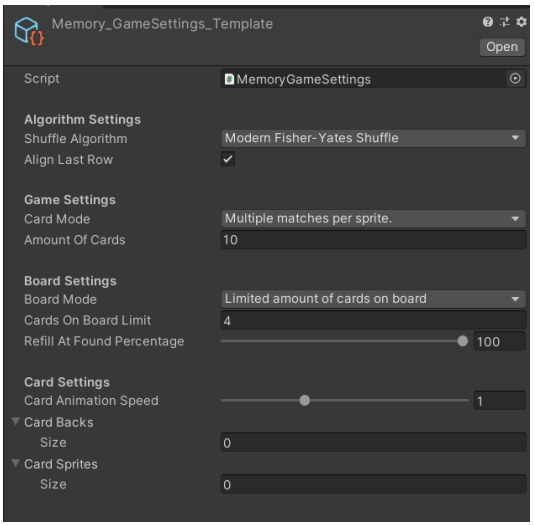
With the UI and the manager of the game setup, you can start to create your MemoryGameSettings. MemoryGameSettings is a ScriptableObject that contains the game rules you can configure.

You can decide what algorithm is used to shuffle the cards and whether or not the last row of cards is placed in the center. You can choose to align the last row of cards to the center. Enabling this will center the cards in the lowest row if the amount of cards in that row is less than the rows above.



There are two card options to choose from, this will determine how many cards are in the game. The setting ‘one match per sprite’ will create two cards for each card sprite. making every match unique. The other option ‘reuse cards’ will create multiple matches with the sprites provided. The amount of matches made is determined by the ‘amount of cards’ setting.

For the board, there are also two options. ‘All cards on board’ will place all the cards in the game on the board, whereas ‘limited amount of cards on board’ will place the same amount of cards on the board as the player has decided. The refill at percentage option is used to decide when the board is refilled with new card pairs.



Required is at least one sprite for both the card backs and card sprites. for each sprite in the card sprites, two cards will be made.
so for each match, only one sprite is needed.

Now you have everything you need to start a Memory minigame. Reference the **MemoryGameManager** and use the **StartGame** Method. The method will need the **MemoryGameSettings** you have created.

5. API

Memory Game Manager

```
// Reference to the SetTheClockManager in the scene.
private MemoryGameManager _manager;

// Reference to your custom SetTheClock GameSettings.
private MemoryGameSettings _mySettings;

private void Start()
{
    // Starts an Memory game with the given GameSettings.
    _manager.StartGame(_mySettings);

    // Pauses the game.
    _manager.Pause();

    // Continues the game from the paused state.
    _manager.Continue();

    // Forces the game to end.
    _manager.ForceFinish();

    // Restarts the game.
    _manager.Restart();

    // Whether the game is currently active.
    bool isGameActive = _manager.IsGameActive;

    // Wheter the game is currently paused.
    bool isGamePaused = _manager.IsPaused;
}
```

```
private void OnEnable()
    {
        // This event is invoked when the game is started.
        _manager.Started += MyStartMethod;

        // This event is invoked when the game is paused.
        // The event provides a bool indicating if the game is paused or resumed.
        // True, the game is paused.
        _manager.Paused += MyPausedMethod;

        // This event is invoked when the game is finished.
        // The event provides MemoryGameResults which contain the time played,
        // the amount a player has tried to match two cards.
        _manager.Finish += MyFinishedMethod;
    }

    private void MyStartMethod()
    {
        // My start behaviour ...
    }

    private void MyPausedMethod(bool isPaused)
    {
        // My start behaviour ...
    }

    private void MyFinishedMethod(MemoryGameResults results)
    {
        // Time the player took to match all the cards.
        TimeSpan time = results.timeTaken;

        // Amount of times the player has tried to match two cards.
        int levelRestarts = results.amountOfTurns;
    }
```

6. Known Limitations

- **No known limitations.**

7. Support and feedback

If you have any questions regarding the use of this asset, we are happy to help you out. Always feel free to contact us at:

unity-support@d-tt.nl

(We typically respond within 1-2 business days)

We are actively developing this asset, with many future updates and extensions already planned. We are eager to include feedback from our users in future updates, be they 'quality of life' improvements, new features, bug fixes or anything else that can help you improve your experience with this asset. You can reach us at the email above.

Reviews and ratings are very much appreciated as they help us raise awareness and to improve our assets.

DTT stands for Doing Things Together

DTT is an app, web and game development agency based in the center of Amsterdam. Established in 2010, DTT has over a decade of experience in mobile, game, and web based technology.

Our game department primarily works in Unity where we put significant emphasis on the development of internal packages, allowing us to efficiently reuse code between projects. To support the Unity community, we are publishing a selection of our internal packages on the Asset Store, including this one.

More information about DTT (including our clients, projects and vacancies) can be found here:

https://www.d-tt.nl/en/