Match-3 Starter Kit by DFT Games

Release Notes - Version 2.5

New Features:

- Side sliding option (Use Side Sliding flag in the inspector)
- Points are on a per piece basis (optional)
- Bomb pieces option (Use Bombs flag in the inspector)
- Full Game Execution flag to tell the script if to use or not the static values to initialise the game

New additions

- Demo project demonstrating
    - how to use delegates
    - how to build a multilevel game with just two scenes
    - how to manage pause
    - how to manage the progress saving
    - how to automate time limit computation based on the level number

Breaking Changes

- The Pause Management delegate has been flagged as Obsolete. It'll be removed in version 3.0

Structural Changes

- Now all the project's parts are in the Match3StarterKit folder. Make suse to have your own Resources folder in the root Asset folder or just move the provided folder in the root to be able to edit the provided game sample boards.

Release Notes - Version 2.4.1

Built with Unity 3.5 to keep the compatibility with that version.

Release Notes - Version 2.4

Fixed an unnoticed memory leak. No changes.

Release Notes - Version 2.3

Breaking changes

- OnGUI has been dismissed favouring NGUI implementation as this is the common way nowadays and because the upcoming new Unity GUI system will be very much NGUI-like as its author is the one hired by Unity to work on that new GUI system.

New features

- Added a new delegate (CheckBonusDelegate) to allow you to manage multipliers and bonuses
- Added four static counters that are updated at every user's move to be used in the CheckBonusDelegate code (see instructions)
- Compacted the start board so that you can restart/reset/reload any board without having to load a new scene

Other changes

- The code has been completely revised and optimized for even better performances
- The Board class now sets the Target Frame Rate to 30 as for this type of game there is no need for more than that, so this way we save battery life on mobile.
- The code has been widely commented in the most complete way.
- Refactored a few variable names that weren't intuitive.

Release Notes - Version 2

Breaking changes
None

New features
- It’s now possible to make a Match-4 game simply checking the newly added flag “Is Match 4”in the Inspector.
- Hints: now you can add (in the Inspector) an effect and an audio to be give moves hints to the player. In the Inspector you can also define after how much inactivity time (in seconds) the hint is shown.
- You can now decide to make the new pieces fall from the top instead of flying in from behind the camera simply checking in the Inspector the flag “New Pieces From Top”.

Release Notes - Version 1.2

Breaking changes
- The board’s file name has been changed from “board[BoardNumber]” to “Board[BoardNumber].[rows].[columns]”, so upgrading from previous version you have to rename the boards accordingly in Unity project view. For instance, the file **board1** becomes **board1.10.10** (this because in previous version the board was fixed to be 10 by 10.
- The points are now defined as Board’s parameters.

New features
New parameters added to the Board script as follows:

1. Mobile input is now supported out-of-the-box.
2. Delegates for Pause, Level Cleared and Time Out called in the OnGUI.
3. The Play Area Editor allows now to define and edit any board size.
4. Rows: amount of rows on the board.
5. Columns: amount of columns on the board.
6. Fill On X: tell the script to fit the board on the screen. This is mostly used on mobile phones. The script will not deform the layout so to have the board actually filling the screen you have to design it having the phone resolution in mind. The board will be resized to fit the screen as much as possible having the starting point set on the upper left corner.
7. Centre on X: This is considered only if Fill On X is True.
8. Points to be awarded are now Board’s parameters as follows:
   a. Points Normal
   b. Points Strong
   c. Points Extra Strong
   d. Points Super Strong

Type of gameplay
In this script we also give you an original gameplay: Marina’s Style (you set the gameplay style in the Board script’s “Game Style” parameter).

Marina’s Style is played by selecting the starting piece when the game starts. The selected piece will be highlighted (using the effect defined in the “Active Effect” parameter). Once the piece has been selected there is no way back, so the player is supposed to choose the start position carefully. Having the selected piece active the player can click on any other piece on the board to swap position provided that there is a match on at least one side of the swapping pieces. The piece on the clicked position becomes the new selected piece and **cannot be changed**, so the player has to think carefully before to move to the next match. As usual the goal is to clear the board. In case of clearing under blocked pieces the new selected position is empty: in that case the user has to select a new position to continue.

If you are going to implement Marina's Style gameplay in your game it would be really nice of you to credit it to DFT Games in your game's credit screen ;)

Just a note about the blocked tiles: if on the board there are blocked tiles the script spawns the special piece as well. Matching three special pieces unblocks one blocked tile (a demo of this behaviour is in the board2 definition: see the How to define the board section).

Both gameplay styles can be played in Match-3 and Match-4 mode.

Kit's Content

The Kit contains the full logic and a simple serialization script to implement a leaders' board. All the code has been designed to allow the fastest and lightest execution possible. We have successfully tested it on very slow Atom based netbooks and on mobile using a Sony Ericson Xperia Mini Pro (really slow phone).

How to define a board

In the menu bar select **Match-3 SK** and execute the **Play Area Editor**. This will bring up the board definition editor which allows you to properly generate the board's file in the Resource/Data folder.

In the Resource/Data folder there are example files. The files contain the definition of the board coded as follows:

1 = Normal tile
2 = Strong piece on normal tile
3 = Extra Strong piece on normal tile
4 = Super Strong piece on normal tile
5 = Locked tile
X = No tile

The file name must be "board" followed by a number, a dot, the number of rows, a dot and the number of columns; its file extension is TXT and is ASCII encoded. The board number is the one used in the Board component's parameter "Board Number", rows and columns are the Board parameters Rows and Columns.

Creating a Match-3 level

Just drop a board on screen and define the empty objects Left and Right to mark the board's limit and add to your board the Board script, then feed all its parameters (also add an Audio Source!). You can define as many pieces as you want feeding their prefabs to the list parameters for the 4 classes: Pieces Normal, Pieces Strong, Pieces Extra Strong and Pieces Super Strong. Setting the Max Pieces parameter you decide how many pieces should be actually put into the game picking from those lists. On a 10 by 10 board, six pieces for the Standard Style and Five for Marina's Style are our advice to avoid too many reshuffles. **Remember to change the PieceColour structure (in Definitions.cs) to match the number of pieces you actually put into your game!**

Make sure to have the same amount of pieces in each category and when you set the board parameters along with the columns and rows make sure they match because the file name and the size of the internal arrays are built on that.

Pieces

Have a look to the Prefabs folder. There are four different subfolders in the Pieces folder: Normal, Strong, ExtraStrong and SuperStrong. Each piece is a GameObject containing an AudioSource and the PieceScript component along with a suitable collider to allow mouse events detection. The actual mesh is a child of the Piece.

Adding your code

The Board script is the actual game manager. Once the time is up or the level is cleared you have to do something. In the Update we check the timeout, pause and winning states and we execute the methods assigned to their delegates.

In your code you are supposed to feed the delegates to manage the three situations. The delegates are accessible as follows:

OBSOLETE ~~**Board. GamePaused** to signal the pause to the game~~

OBSOLETE ~~**Board.Instance.GamePausedMethod** to set the method to manage the pause~~

Board.Instance.TimedOutMethod to set the method to manage the Time Out (player loses)

Board.Instance. LevelClearedMethod to set the method to manage the Game Cleared state (player wins)

In addition, when we compute the matching we call the related delegate to allow you to reward the player:

Board.Instance. CheckBonusMethod to set the method to manage the bonuses and multipliers.

In the Check Bonus delegate you should access to the following variable to see what's just happened:

Board. totalDestroyedPieces to know how many pieces have been destroyed during last move;

Board.totalNormalPiecesDestroyedto know how many normal pieces have been destroyed during last move;

Board.totalStrongPiecesDestroyed to know how many strong pieces have been destroyed during last move;

Board.totalSuperStrongPiecesDestroyed to know how many super strong pieces have been destroyed during last move;

Board.totalExtraStrongPiecesDestroyed to know how many extra strong pieces have been destroyed during last move;

All you have to do to restart/reload your game from your code is to set 3 variable: rows, columns and boardNumber, then just call the StartBoard method. That can also be done from other components via the static Instance exposed by the Board class.

Support and Help

In case you need any help implementing our Kit do not hesitate to write at pino@dftgames.com

Thank you for using Match-3 Starter Kit! **Don't forget to review it on the Asset Store!**