T3Dmake

Mobile Puzzle Game Kit

Unity Asset Store

Thank you for purchasing the Mobile Puzzle Game Kit

CONTENTS

INTRODUCTION

The mobile puzzle game kit is a collection of 5 fun mobile puzzle games. I started with the idea to make a sudoku generator and I didn't think it would become such a big project. The sudoku generator took a little bit more time than expected since it had to only generate puzzles with a single solution. After the sudoku game was done I made a list of the games and the kit slowly started to come together. The word search generator was quite a challenge as well, but all scripts are commented so that should make it easier to understand. In this document I'll try to explain the games and provide some useful information.

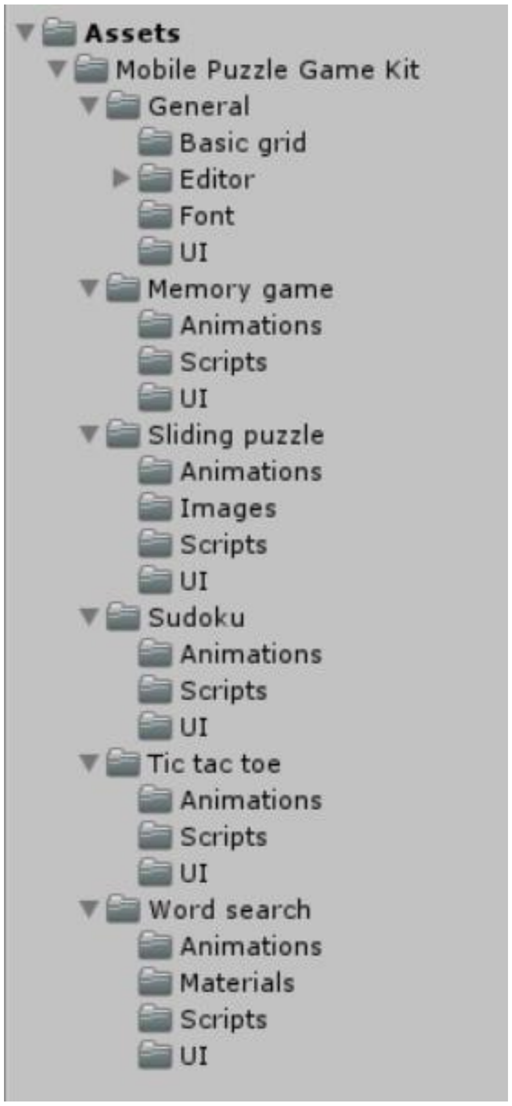
GENERAL

All puzzles use just one scene (per puzzle). Menu's like categories and start level panels are on top of the other stuff. Most scenes contain just a camera, a canvas and an event system. In the canvas, the most important thing is the board object. In all games, the board object is the main grid that will contain all cells and it also has the scripts that you can use to tweak the games. Except for the board object there's only some UI panels and effects.

Tags and layers are barely used, the only object in the games that uses a tag is the cell object in the sudoku game. As long as you make sure the sudoku cell prefab (not the cell grid) is tagged 'cell' it's fine.

If you want to delete data like the saved levels, you can go to *Window – Delete PlayerPrefs*.

PROJECT STRUCTURE



The project contains 6 main folders. If you don't want one of the puzzle games, you can just remove the folder. **Please don't remove the 'General' folder**, this folder contains all the editor scripts, the fonts and some general UI backgrounds.

One less important part of the 'General' folder is the 'Basic grid'. In this folder you'll find a grid prefab that you can add to your canvas which creates a grid of cells. You can tweak some settings to change the grid. I left this in the package in case you want to make a similar game that's not included.

Basically, each game has 3 folders, 'Animations', 'Scripts' and 'UI'. The materials and images folders are only useful for those specific games.

The 'UI' is actually a combination of the prefabs folder and the UI images/backgrounds folder. This contains both the cells, buttons and effects and the UI images. The 'Scripts' folders contain scripts (some contain just one script) and the 'Animations' folders contains animations and animators.

SUDOKU

GENERAL

“**Sudoku** (数独 *sūdoku*, digit-single) (originally called **Number Place**)[1] is a logic-based,[2][3] combinatorial[4] number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

Completed games are always a type of Latin square with an additional constraint on the contents of individual regions. For example, the same single integer may not appear twice in the same row, column, or any of the nine 3×3 subregions of the 9x9 playing board.”

https://en.wikipedia.org/wiki/Sudoku

ALGORITHM

Like the definition above says, the script has to fill the puzzle with numbers so that each column, row and sub-grid contains the numbers 1 to 9. After that, it has to remove the unnecessary clues and make sure only one solution is left.

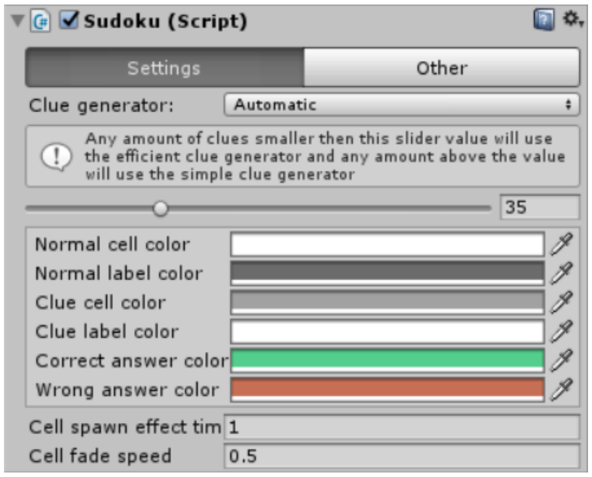
Basically, the algorithm works as follows:

- First, it creates a valid grid of numbers from 1 to 9:
    - For each of the 81 cells, it will find a fitting number.
    - To find a fitting number, it randomly tries the numbers 1 to 9 and checks if they fit horizontally, vertically and in the sub-grid.
    - If it finds a cell where none of the 9 numbers fit, it will start again, but now with different numbers. I first did this using backtracking since that seemed more efficient, but it turned out the same puzzles were generated over and over again.
- After generating the base grid, it will start with 81 clues
- It then creates the actual grid with 81 cells, but the numbers are covered until we've found the clues
- For each of the cells it will check if the clue can be removed (depending on the simple or efficient generator, if will either go through the clues randomly or find the clues that have most chance of being removed):
    - To check this, it will try to find possible solutions for the current puzzle and if the removed clues leads to multiple solutions, the clue needs to be enabled. If the clue is removed and there's still one solution for the puzzle, we don't have to enable the clue.

    - In order to find the actual solutions, it’ll create a list of the cells that are not clues, and it will go through all these cells using backtracking (it goes back and forward through the puzzle to find valid numbers for all cells).
    - When it reaches the end of the puzzle, we have a new solution and it continues to find a second solution. If we reach the start of the puzzle, we know there’s no solution left. We know the clue needs to be enabled when we find more than one solution, so after it reaches the end of the puzzle a second time, the loop breaks and we enable the clue.
- After checking all cells, we have a certain number of clues left. If this number is small enough, we can continue to the next step. If there are too much clues left, we’ll need to clear the clues and try again.
- Now that we have a complete grid with all clues in the right place, the cell covers fade out and the player can start filling in the sudoku puzzle.

EDITOR

The editor for the sudoku generator is quite straight forward:

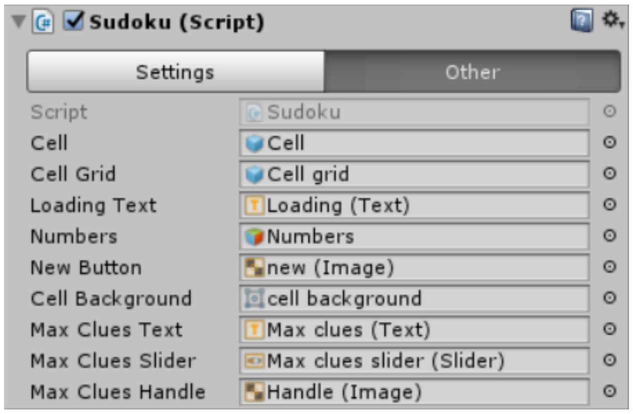


Clue generator:	Automatic

Any amount of clues smaller then this slider value will use the efficient clue generator and any amount above the value will use the simple clue generator

35

Normal cell color	⬜
Normal label color	⬛
Clue cell color	▨
Clue label color	⬜
Correct answer color	🟩
Wrong answer color	🟥
Cell spawn effect tim	1
Cell fade speed	0.5



Script	Sudoku
Cell	Cell
Cell Grid	Cell grid
Loading Text	Loading (Text)
Numbers	Numbers
New Button	new (Image)
Cell Background	cell background
Max Clues Text	Max clues (Text)
Max Clues Slider	Max clues slider (Slider)
Max Clues Handle	Handle (Image)

Clue generator: The type of clue generator you want to use. With a small number of clues, the efficient generator takes a few more steps than the simple generator, but the simple generator will probably need more tries to create a puzzle with less clues. Automatic means it automatically switches between the two based on the maximum number of clues.

Cell spawn effect time: The time it takes to spawn all cells. It’s not really necessary but it’s a nice effect when the game starts.

Cell fade speed: The speed at which the cells fade in when the puzzle has been generated.

In the ‘other’ panel, there’s some additional objects and fields, but most of these don’t have to be changed. For example, if you want to change the look of the grid you can change the grid component on the board object. If you want to change the look of the cells, you can open the UI folder, find the cell and change some settings like sprite and color on the cell itself.

MEMORY GAME

GENERAL

“**Concentration**, also known as **Match Match**, **Match Up**, **Memory**, **Pelmanism**, **Shinkei-suijaku**, **Pexeso** or simply **Pairs**, is a card game in which all of the cards are laid face down on a surface and two cards are flipped face up over each turn. The object of the game is to turn over pairs of matching cards. Concentration can be played with any number of players or as solitaire. It is a particularly good game for young children, though adults may find it challenging and stimulating as well. The scheme is often used in quiz shows and can be employed as an educational game.”

https://en.wikipedia.org/wiki/Concentration_(game)

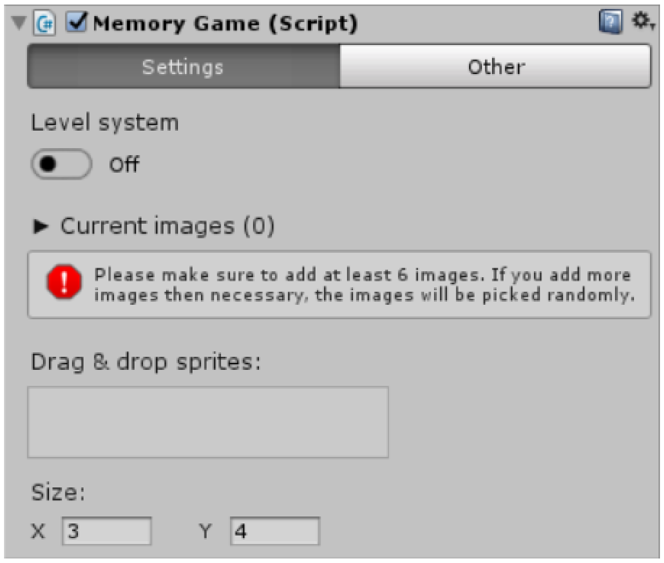
ALGORITHM

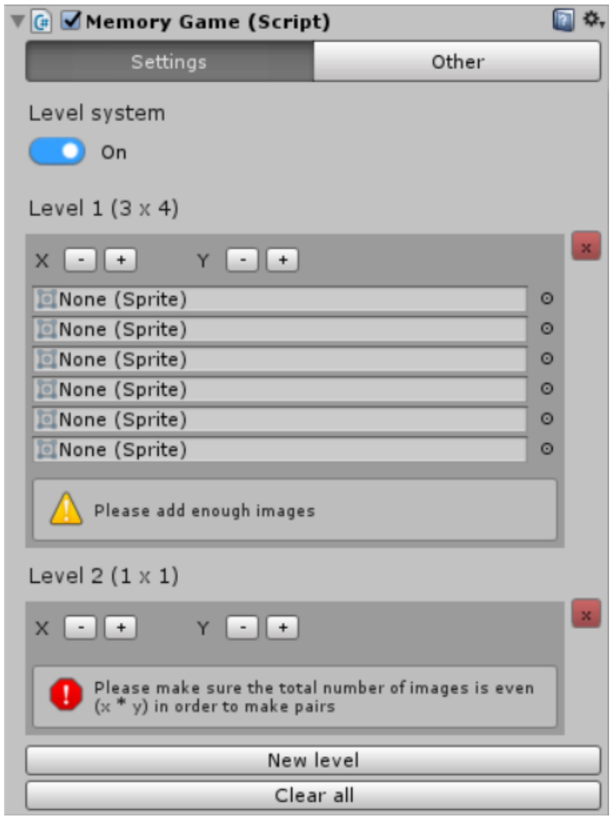
The algorithm for a memory game is quite simple. All it needs to do is shuffle pairs of images on a grid. Then each tile has a pair index to check if the player flipped two cards with the same image and end the game when no pairs are left.

- It checks if the number of tiles is even
- It creates the grid:
    - For each pair, it’ll create two tiles with the same image and index, but it doesn’t yet add the tiles to the grid (then all pairs would be together)
    - Then it loops through all tiles and randomly enables them and adds them to the grid

EDITOR

For the memory game, you have two options. You can enable levels or just add some images and create the puzzle:


Memory Game (Script)
Settings | Other
Level system
Off
► Current images (0)
Please make sure to add at least 6 images. If you add more images then necessary, the images will be picked randomly.
Drag & drop sprites:
Size:
X 3 Y 4


Memory Game (Script)
Settings | Other
Level system
On
Level 1 (3 x 4)
X - + Y - +
None (Sprite)
None (Sprite)
None (Sprite)
None (Sprite)
None (Sprite)
None (Sprite)
Please add enough images
Level 2 (1 x 1)
X - + Y - +
Please make sure the total number of images is even (x * y) in order to make pairs
New level
Clear all

For the first option (no levels), all you have to do is drop some images to use and set a size for the puzzle. Please make sure your images are squares with the type sprite.

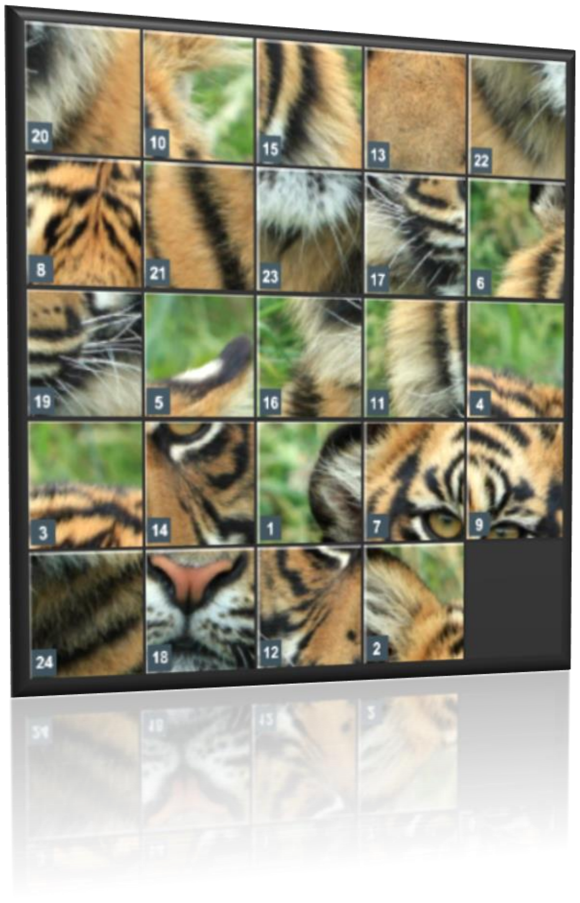
If you do want to add some levels, the process is very similar. You can add new levels and for each level set its size. Then add your images and play the memory game.

SLIDING PUZZLE

GENERAL

“A **sliding puzzle**, **sliding block puzzle**, or **sliding tile puzzle** is a combination puzzle that challenges a player to slide (frequently flat) pieces along certain routes (usually on a board) to establish a certain end-configuration. The pieces to be moved may consist of simple shapes, or they may be imprinted with colors, patterns, sections of a larger picture (like a jigsaw puzzle), numbers, or letters.”

https://en.wikipedia.org/wiki/Sliding_puzzle



ALGORITHM

The sliding puzzle algorithm will need to do a few things. It must split the start image into a number of tiles, shuffle the tiles and check if the puzzle has a solution.

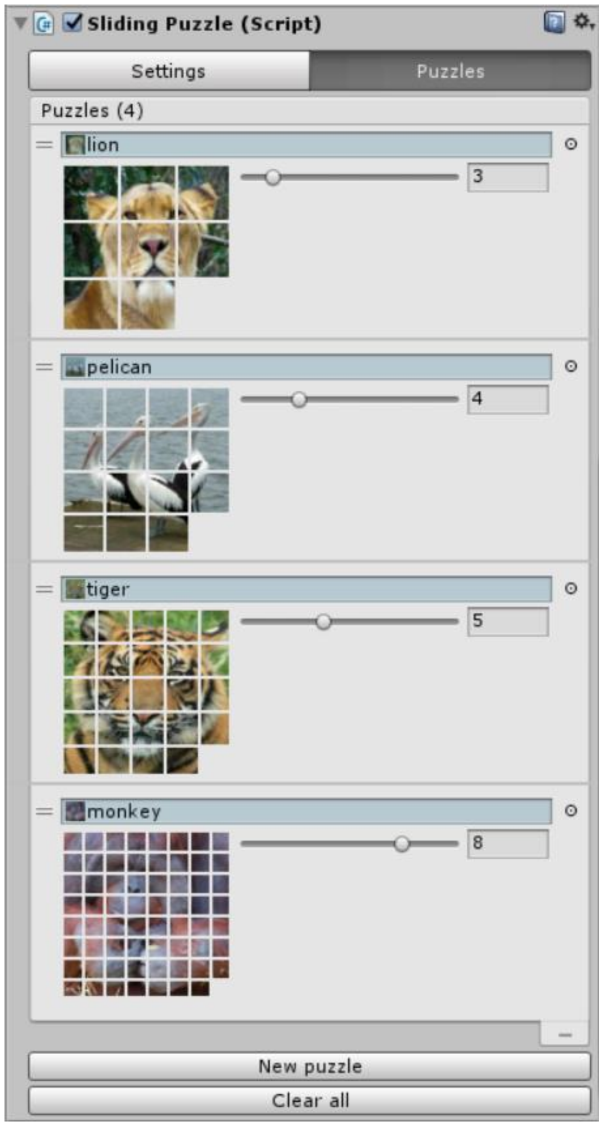
Before creating the tiles, we need to make sure we can move the tiles through the grid later. Since the grid component makes it a bit hard to move tiles, I decided to add a fixed base tile on each tile position. Then we can add the real tiles on top and move those.

- First, it adds the base tiles to the grid
- Then it adds the main tiles:
    - It creates a list of shuffled grid positions
    - For each cell in the grid, it adds a tile and assigns a sprite that contains a part of the original image
- It checks if the puzzle is solvable with the tiles we just added
- If it is not solvable, it adds the main tiles again
- If it is solvable, it'll go through all tiles and enable them to start the puzzle

EDITOR

The ‘puzzles’ tab is the most important part of the sliding puzzle editor. You can add new puzzles with the button, add your image in the blue field and use the slider to set the size/difficulty for your puzzle.

If you want, you can drag and reorder your puzzles.



Sliding Puzzle (Script)

Settings | Puzzles

Puzzles (4)

lion 3

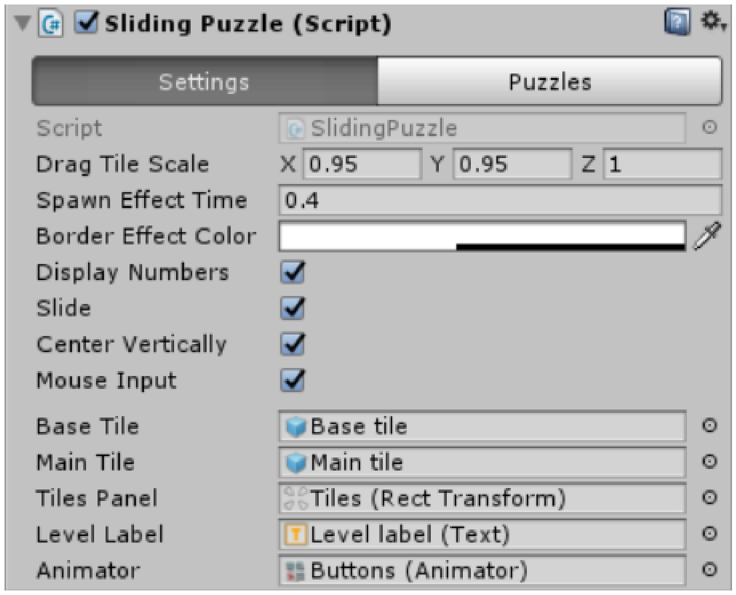
pelican 4

tiger 5

monkey 8

New puzzle

Clear all



Sliding Puzzle (Script)

Settings | Puzzles

Script	SlidingPuzzle
Drag Tile Scale	X 0.95 Y 0.95 Z 1
Spawn Effect Time	0.4
Border Effect Color	
Display Numbers	✓
Slide	✓
Center Vertically	✓
Mouse Input	✓
Base Tile	Base tile
Main Tile	Main tile
Tiles Panel	Tiles (Rect Transform)
Level Label	Level label (Text)
Animator	Buttons (Animator)

The settings tab contains the following options:

- *Drag tile scale*: The scale of tiles that are being dragged by the player. This is not necessary but it adds a nice effect.
- *Spawn effect time*: The time it takes to spawn all tiles to the grid.

- *Border effect color*: The tiles have a colored border to create some depth and you could increase or decrease the border alpha or change its color.
- *Display numbers*: Should it display the hints in the bottom-left corner? So players know where the tiles should go.
- *Slide*: Do we want to slide the tiles or just tap the screen to move them?
- *Center vertically*: Do we want to center the grid vertically?
- *Mouse input*: When trying the game on a pc, you can enable mouse input. It’s better to disable this when building for mobile devices.

WORD SEARCH

GENERAL

“A **word search**, **word find**, **word seek**, **word sleuth** or **mystery word** puzzle is a word game that consists of the letters of words placed in a grid, which usually has a rectangular or square shape. The objective of this puzzle is to find and mark all the words hidden inside the box. The words may be placed horizontally, vertically, or diagonally. Often a list of the hidden words is provided, but more challenging puzzles may let the player figure them out. Many word search puzzles have a theme to which all the hidden words are related. The puzzles have, like crosswords and arrowords, been very popular in the United Kingdom, and - also in common with these latter puzzles - have had complete magazines devoted to them.

Word searches are commonly found in daily newspapers and puzzle books. Some teachers use them as educational tools for children, the benefit being that young minds can learn new words and their spellings by intensively searching for them, letter by letter, in the puzzle.”

https://en.wikipedia.org/wiki/Word_search

➔ Please notice that the board object contains two other scripts except for the main word search script. Currently the ‘Draw lines mouse input’ script is enabled so you can play the game on a pc, but if you want to play on a mobile device, you should disable the ‘Draw lines mouse input’ script and enable the ‘Draw lines touch input’ script.

ALGORITHM

Like the sudoku algorithm, the word search algorithm was a bit more complicated:

- First, it checks if we’re using a text file and it selects some random words to use in the puzzle
- Then it starts trying to add all words into the grid
- For each word, it first tries to find a corresponding character in the grid and add it at the corresponding position:
    - It creates a list with all corresponding positions
    - For each position, it’ll go through all directions and see if the word can be placed:
        - It creates a list to store all the changed character positions and another list to change the original characters of those positions
        - It goes through all characters in the word and for each character it finds the grid position by multiplying the index of the character in the word by the current word direction and adding the word start position
        - Then it checks if the new character can be added at that position by checking if the position exists in the grid and if the position is in the right direction relative to the other characters
        - If the character can be placed, it’ll continue to the next character. If the character can’t be placed, it resets the previous character positions using the lists of positions and original characters and it goes to the next direction.
- If the entire word was placed, it’ll continue to the next word

- If it didn't find a solution to place the word, it will pick some random positions and checks if it can find a valid one
- After a certain number of tries, it'll reset the grid and start over
- To make sure the puzzle won't get stuck in infinite loops (for example if we're trying to add a 6-character word into a 4x4 grid) the puzzle will stop and show an error after resetting the puzzle several times
- If it did add all words into the grid, it loads the actual cells with the characters and it show the labels so players can see which words were found and which words to look for

EDITOR

The word search editor might seem a bit more complicated at first. Let's start with the 'other' tab, which contains a few important options:

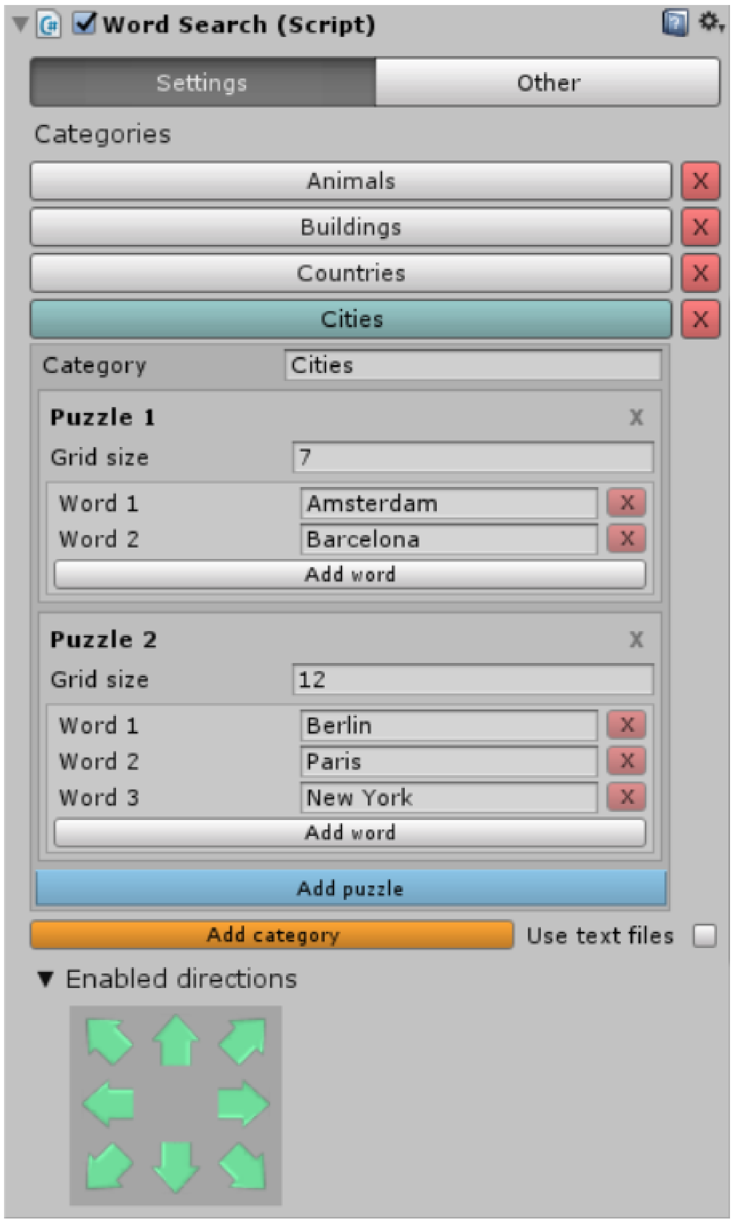
- *Effect time*: The time it takes to create the category buttons and word labels.
- *Center*: Should the puzzle be centered vertically?
- *Tries before resetting*: The number of times it will try to fit a word into the grid before resetting all words in the grid.
- *Word effects*: Do we want an effect that shows the word after the player found a new word?
- *Include words with same length*: In case we're using text files to get words, do we want it to select words with the exact same length as the grid size? This would make generating a new puzzle harder.

Then there's also the main tab where you can add categories and puzzles.

To create a new category, please press the orange 'add category' button. You can name the new category and add a puzzle using the blue button. For each puzzle, you set a grid size and then you can use the white button to add new words to your puzzle.

You can control the word directions for each of your puzzles by clicking the arrows under 'enabled directions'.

If you want to get the words from a text file, it looks a bit different. You can then add a text file with words for each category and for the puzzles, you only have to set the grid size and the number of words.



Category: Cities

Puzzle 1
Grid size: 7
Word 1: Amsterdam
Word 2: Barcelona

Puzzle 2
Grid size: 12
Word 1: Berlin
Word 2: Paris
Word 3: New York

Use text files: ☐

Currently, the words in your textfile should be placed with only one space in between each word. Like this: *bird ant pig cat dog fly*

If you want to change this, please go to the main word search script and at line 167, change *Split(' ')* to for example *Split('\n').*

TIC TAC TOE

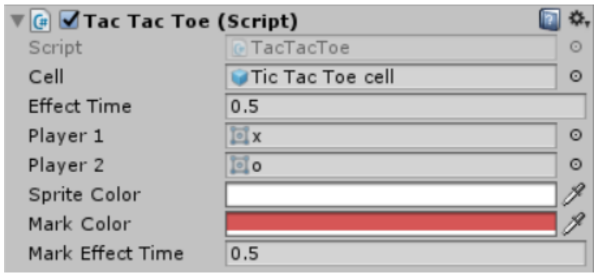
GENERAL

“**Tic-tac-toe** (also known as **noughts and crosses** or **Xs and Os**) is a paper-and-pencil game for two players, *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.”

https://en.wikipedia.org/wiki/Tic-tac-toe

Of course, Tic Tac Toe is a very basic game. It wouldn’t make much sense to explain the algorithm since there isn’t really one. The script adds nine cells to a grid and when the player presses one of them, the x or o is placed and the other player continues. The only interesting part would be the way that it checks all cells to find out if there is a winner.

EDITOR



Tac Tac Toe (Script)

Script	TacTacToe
Cell	Tic Tac Toe cell
Effect Time	0.5
Player 1	x
Player 2	o
Sprite Color	⬜
Mark Color	🟥
Mark Effect Time	0.5

Effect time: The time it takes to add the nine cells.

Player 1/Player 2: The sprite images to show when a player claims a cell.

Sprite color: The default color for the images.

Mark Color/Mark effect time: The image color and duration for the effect that marks your x’s and o’s.

CONCLUSION

In this document I’ve tried my best to explain the Mobile Puzzle Game Kit. I hope you like it and it will be useful. If you like the kit, I would really appreciate a review in the Asset Store.

If you have any questions or suggestions, you can always contact me via:

T3Dmake@gmail.com