UNITY ASSET

Matching Pair Game

Overview

This two of a kind, matching pair, game template, is a ready to deploy solution. That also features a menu, endscreen, Unity Ads, sounds, credits screen, levels and level unlocking management system.

Click on the cards to reveal them, match all card pairs within a level to win! Beat your previous scores and keep your brain fit!

Example Scene

- 1. Example Pair Game Portrait

Setup Video

- https://youtu.be/jFTi-sH48nY

Quick Tutorial/Setup

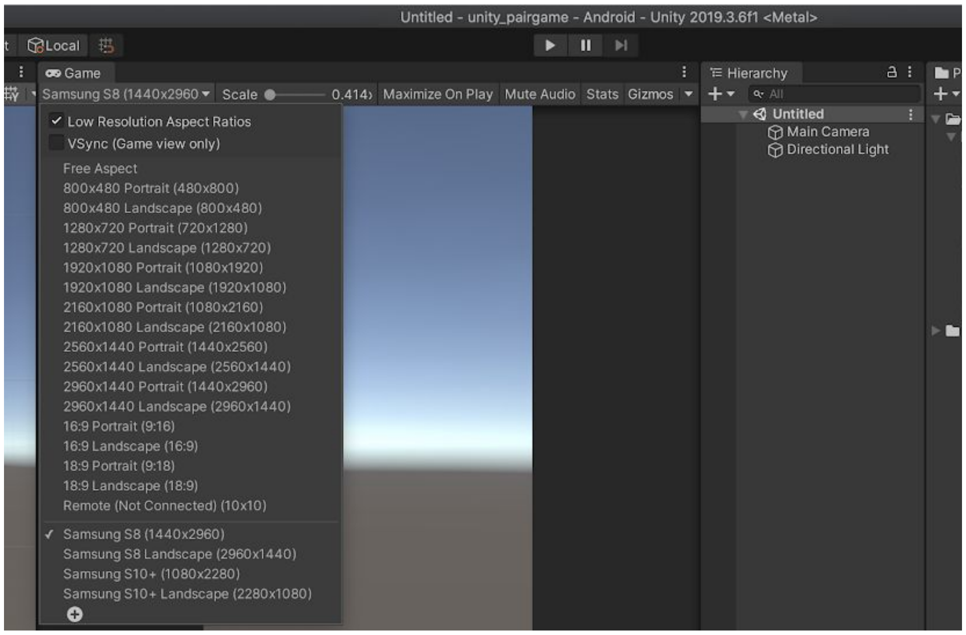
Setup is really easy!

Setup Screen resolution

The examples of this asset have been developed for mobile portrait resolutions/orientations however, can easily be modified to fit landscape orientations, simply by modifying the grid to have more columns than rows and editing the UI for a more landscape oriented approach.

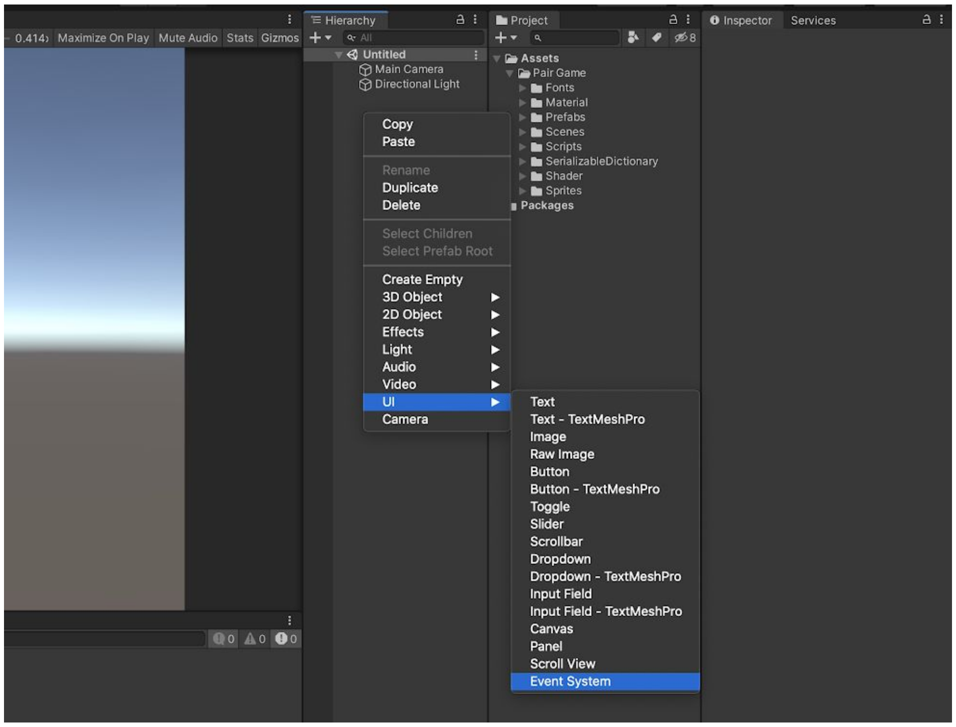
For the purpose of this setup, we’re going to assume that you want to develop your game for mobile portrait resolution/orientation.

Go ahead setup your target resolution - in this example, we are specifically targeting Samsung S8 (1440x2980)


Untitled - unity_pairgame - Android - Unity 2019.3.6f1 <Metal>
Local
Game
Samsung S8 (1440x2960)
Scale
0.414x
Maximize On Play
Mute Audio
Stats
Gizmos
Hierarchy
All
Untitled
Main Camera
Directional Light
✓ Low Resolution Aspect Ratios
VSync (Game view only)
Free Aspect
800x480 Portrait (480x800)
800x480 Landscape (800x480)
1280x720 Portrait (720x1280)
1280x720 Landscape (1280x720)
1920x1080 Portrait (1080x1920)
1920x1080 Landscape (1920x1080)
2160x1080 Portrait (1080x2160)
2160x1080 Landscape (2160x1080)
2560x1440 Portrait (1440x2560)
2560x1440 Landscape (2560x1440)
2960x1440 Portrait (1440x2960)
2960x1440 Landscape (2960x1440)
16:9 Portrait (9:16)
16:9 Landscape (16:9)
18:9 Portrait (9:18)
18:9 Landscape (18:9)
Remote (Not Connected) (10x10)
✓ Samsung S8 (1440x2960)
Samsung S8 Landscape (2960x1440)
Samsung S10+ (1080x2280)
Samsung S10+ Landscape (2280x1080)

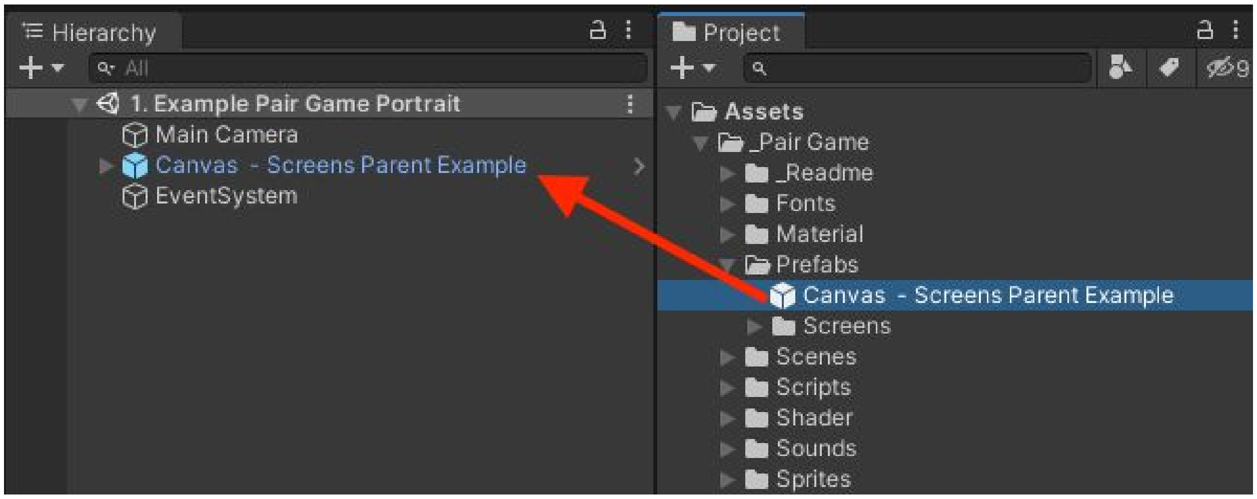
Add Event system

If it doesn't already exist, add an event system. Doing so by right-clicking in the "hierarchy window" > UI > Event System



Drag and drop Canvas - Screens Parent Example into your scene

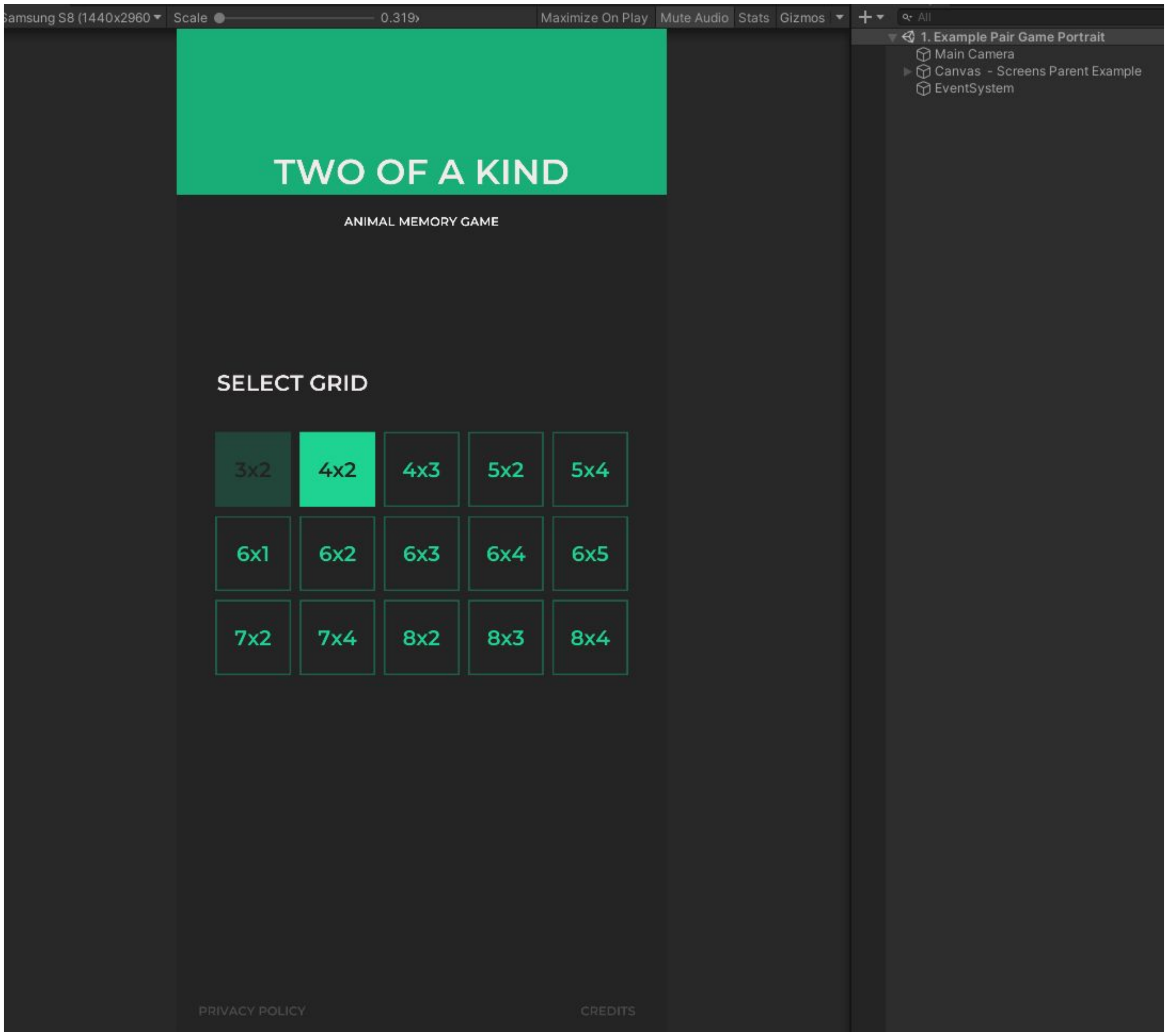
Select the Canvas - Screens Parent Example from the project window and drag and drop in into the hierarchy window/scene.



Attached as a child gameobject you should see a gameobject called Managers that also has child gameobjects. These child gameobjects are managers and have names that relate to the scripts attached to them - which will all help you with the configuration/modification of the game such as modifying sprites.

Hit Play!

As simple as that! You now have a working game! Although there is additional work required for ads to work.

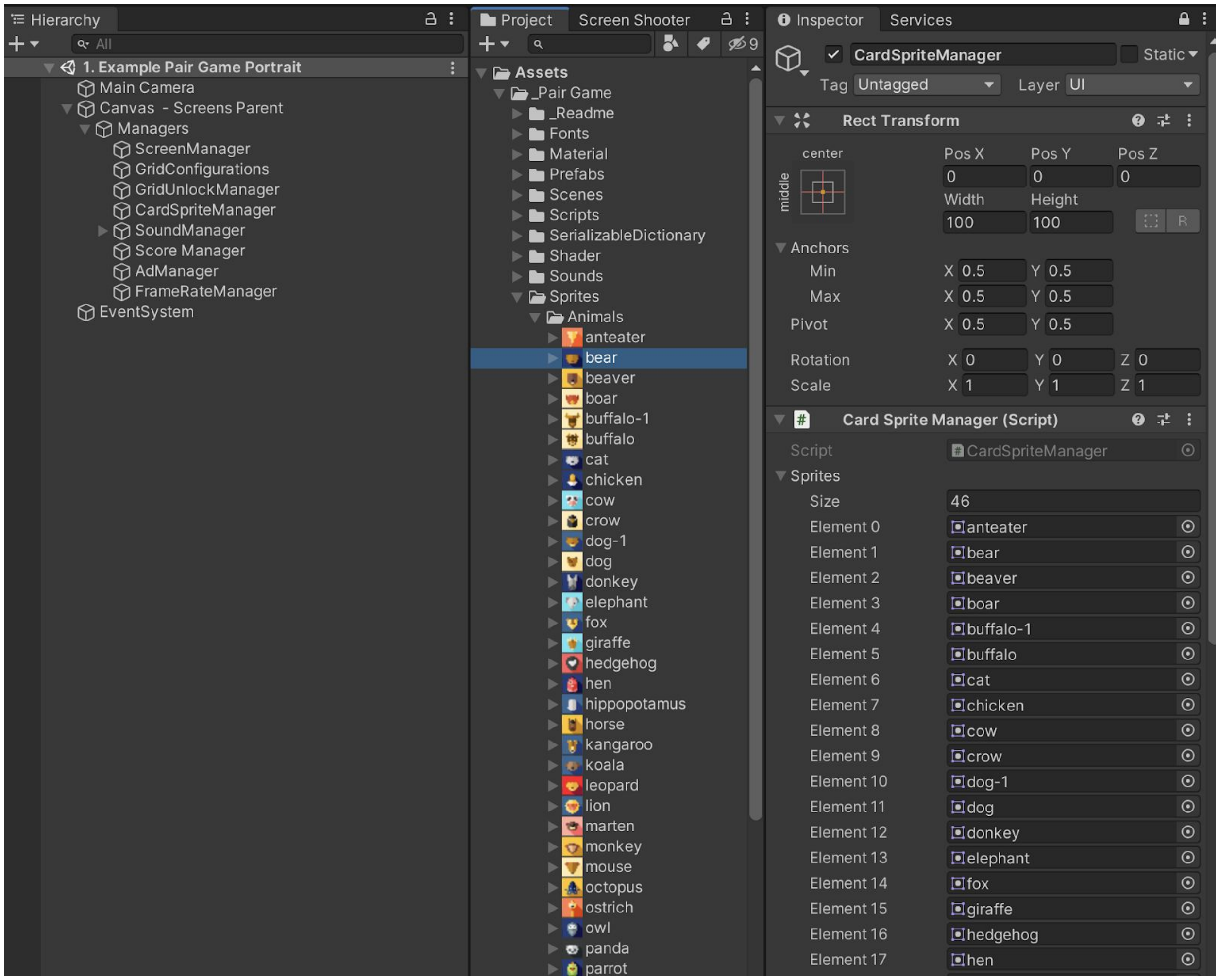


Add/Remove Sprites/Modify Cards

`Locate Canvas - Screen Parent > Managers > CardSpriteManager gameobject`

`Select it > within the inspector window CardSpriteManager.cs is attached.`

`Here you can add/remove sprites that will be used on cards`


Hierarchy

1. Example Pair Game Portrait
- Main Camera
- Canvas - Screens Parent
  - Managers
    - ScreenManager
    - GridConfigurations
    - GridUnlockManager
    - CardSpriteManager
    - SoundManager
    - Score Manager
    - AdManager
    - FrameRateManager
- EventSystem

Project

Assets
- _Pair Game
  - _Readme
  - Fonts
  - Material
  - Prefabs
  - Scenes
  - Scripts
  - SerializableDictionary
  - Shader
  - Sounds
  - Sprites
    - Animals
      - anteater
      - bear
      - beaver
      - boar
      - buffalo-1
      - buffalo
      - cat
      - chicken
      - cow
      - crow
      - dog-1
      - dog
      - donkey
      - elephant
      - fox
      - giraffe
      - hedgehog
      - hen
      - hippopotamus
      - horse
      - kangaroo
      - koala
      - leopard
      - lion
      - marten
      - monkey
      - mouse
      - octopus
      - ostrich
      - owl
      - panda
      - parrot

Inspector

CardSpriteManager
Tag Untagged Layer UI

Rect Transform
Pos X 0 Pos Y 0 Pos Z 0
Width 100 Height 100
Anchors
Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Card Sprite Manager (Script)
Script CardSpriteManager
Sprites
Size 46

Element 0	anteater
Element 1	bear
Element 2	beaver
Element 3	boar
Element 4	buffalo-1
Element 5	buffalo
Element 6	cat
Element 7	chicken
Element 8	cow
Element 9	crow
Element 10	dog-1
Element 11	dog
Element 12	donkey
Element 13	elephant
Element 14	fox
Element 15	giraffe
Element 16	hedgehog
Element 17	hen

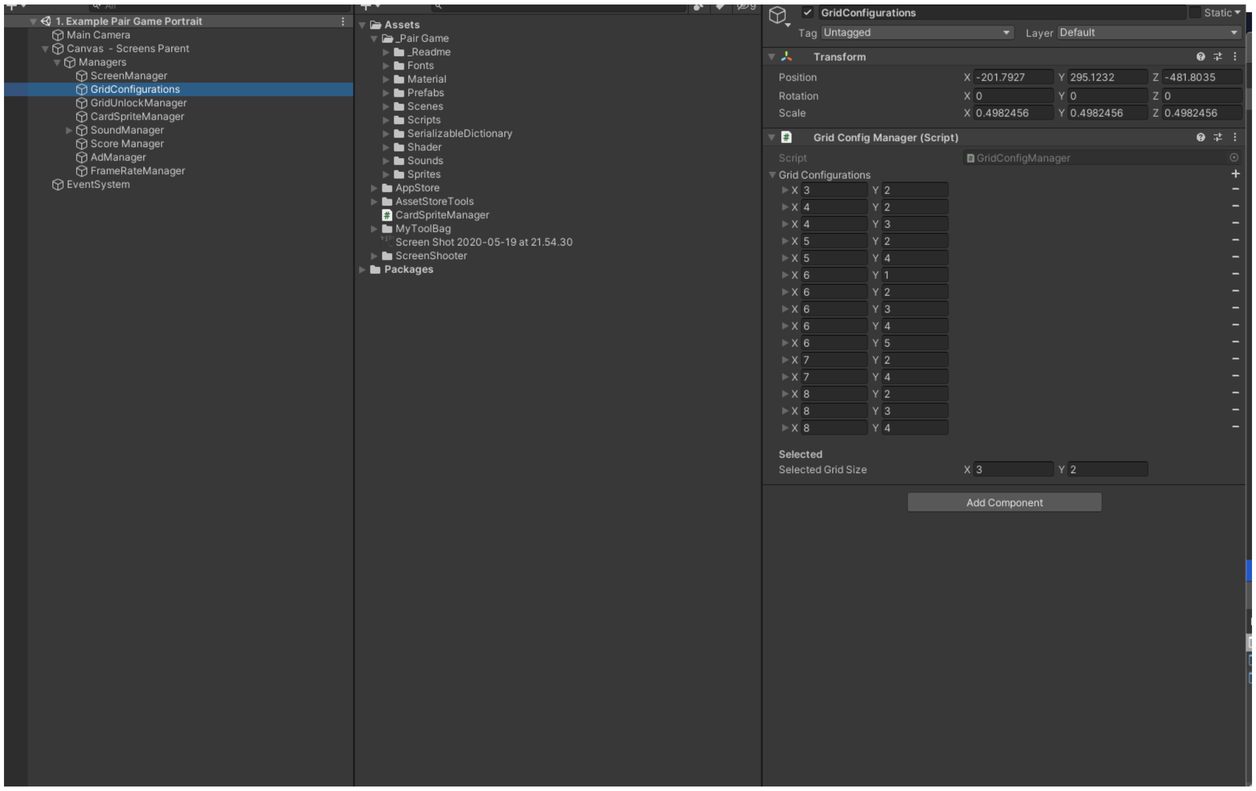
Add / Remove levels

Locate Canvas - Screen Parent gameobject > Managers gameobject > GridConfigurations gameobject within the hierarchy window.

Select it, attached will be the GridConfigManager.cs here, you can add (+) and (-)remove levels and change whether or not they are locked by default.

Note GridUnlockManager.cs and HomeScreen.cs use the list in ascending order(top to bottom) as a way to display levels in a sequential order.

To setup the next grid to be shown, in code, simply call GridConfigManager.Instance.SelectedGridSize = new Vector2(x, y); and change x with rowCount and y with columnCount for example (3, 2) and then call ScreenManager.Instance.ChangeScreen(Screen.Game);



```
1. Example Pair Game Portrait
  Main Camera
  Canvas  - Screens Parent
    Managers
      ScreenManager
      GridConfigurations
      GridUnlockManager
      CardSpriteManager
      SoundManager
      Score Manager
      AdManager
      FrameRateManager
  EventSystem
```

```
Assets
  _Pair Game
    _Readme
    Fonts
    Material
    Prefabs
    Scenes
    Scripts
    SerializableDictionary
    Shader
    Sounds
    Sprites
  AppStore
  AssetStoreTools
  CardSpriteManager
  MyToolBag
  Screen Shot 2020-05-19 at 21.54.30
  ScreenShooter
Packages
```

GridConfigurations — Tag: Untagged — Layer: Default

Transform

	X	Y	Z
Position	-201.7927	295.1232	-481.8035
Rotation	0	0	0
Scale	0.4982456	0.4982456	0.4982456

Grid Config Manager (Script)

Script: GridConfigManager

Grid Configurations

X	Y
3	2
4	2
4	3
5	2
5	4
6	1
6	2
6	3
6	4
6	5
7	2
7	4
8	2
8	3
8	4

Selected

Selected Grid Size: X 3, Y 2

Reset level data

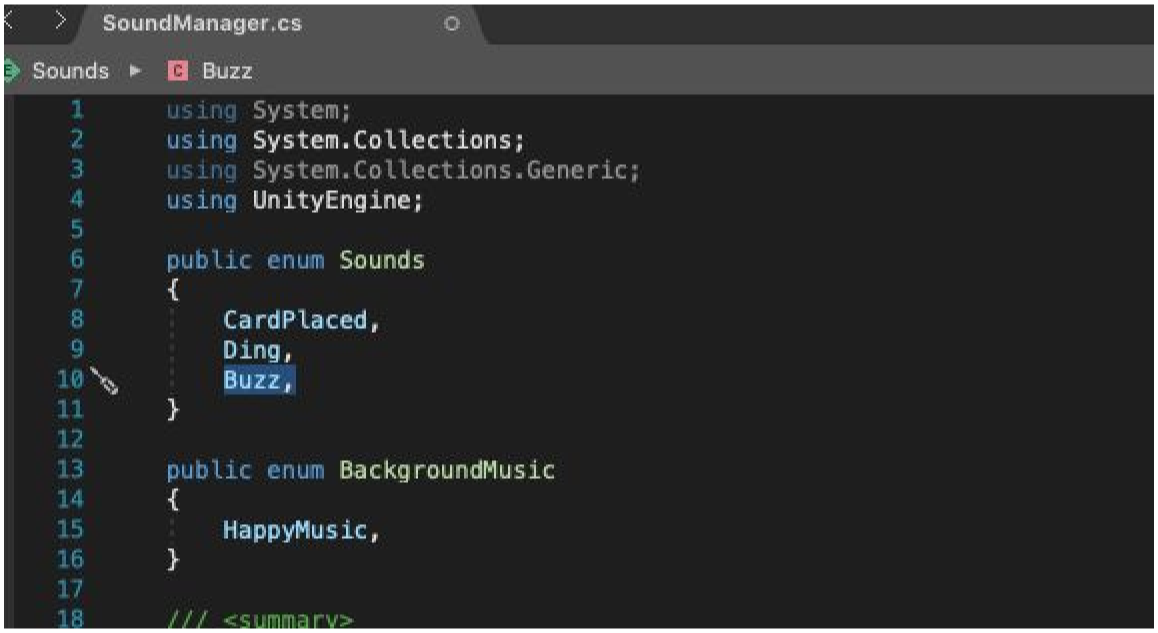
Locate Canvas - Screen Parent gameobject > Managers > GridUnlockManager GameObject.

Select it > during runtime > tick DeleteData within the inspector of GridUnlockManager.cs.

Adding/Removing Sounds

Sounds in this asset are stored/referenced by <enum, audioclip> dictionary making it easy to read/access and call sounds from external scripts for example PlaySound(Sounds.Ding).

To add sounds, you first have to add the enum. Doing so by locating SoundManager.cscript Open the SoundManager.cs script. Located at the top of the script you will find the Sounds enum. Add a new sounds enum value, for example "Buzz" and make sure to hit save! :)

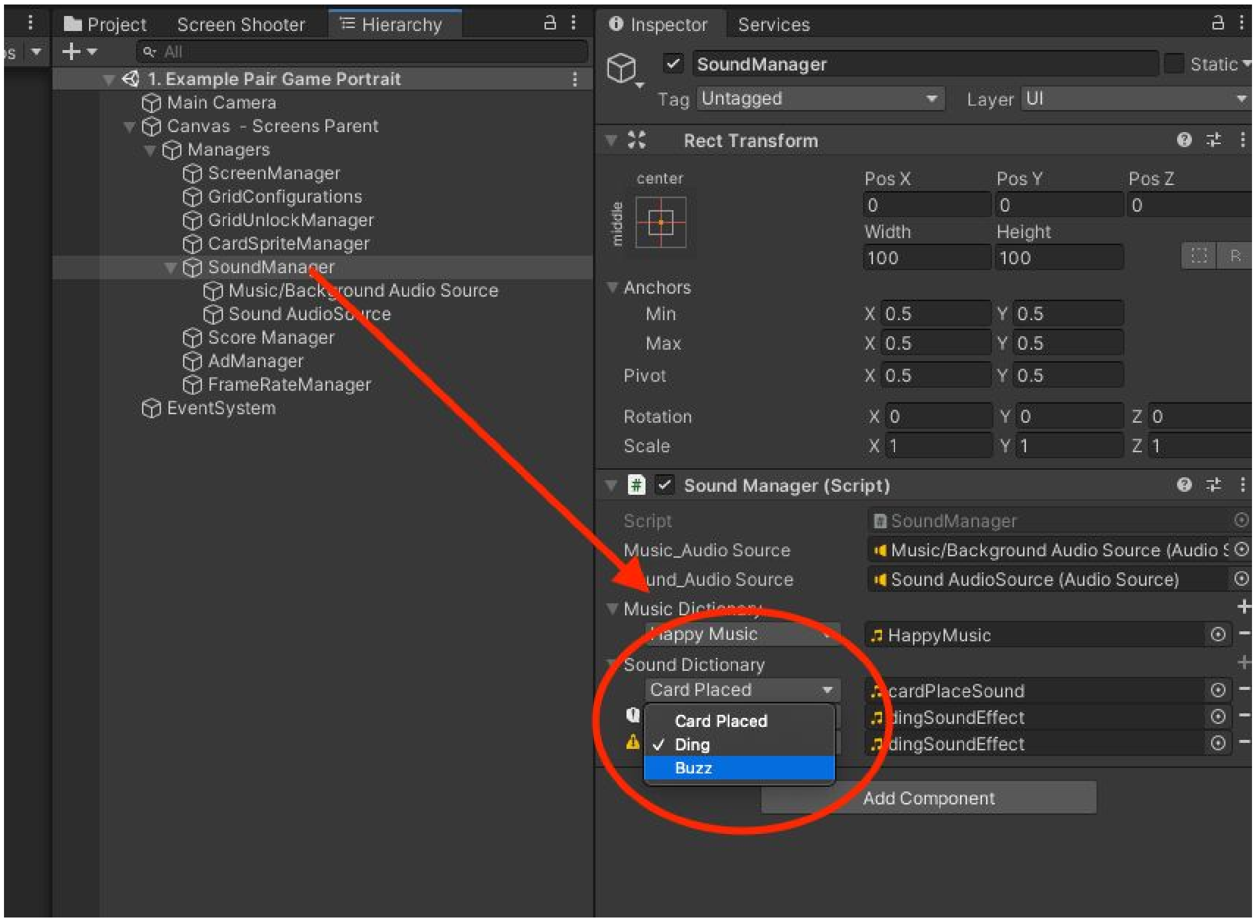


```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public enum Sounds
{
    CardPlaced,
    Ding,
    Buzz,
}

public enum BackgroundMusic
{
    HappyMusic,
}

/// <summary>
```

Locate the SoundManager gameobject in the inspector window or project window.

Locate the SoundDictionary within the SoundManager.cs inspector window and hit +.

Select the enum drop down on the newly created field> you should now see the enum you just created in the drop down e.g. “Buzz”. Select it and add the sound audio clip of your choosing into the property to the right of the selected enum.



```
1. Example Pair Game Portrait
  Main Camera
  Canvas - Screens Parent
    Managers
      ScreenManager
      GridConfigurations
      GridUnlockManager
      CardSpriteManager
      SoundManager
        Music/Background Audio Source
        Sound AudioSource
      Score Manager
      AdManager
      FrameRateManager
  EventSystem
```

SoundManager — Tag: Untagged — Layer: UI

Rect Transform: center / middle; Pos X 0, Pos Y 0, Pos Z 0; Width 100, Height 100; Anchors Min X 0.5 Y 0.5, Max X 0.5 Y 0.5; Pivot X 0.5 Y 0.5; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1

Sound Manager (Script)

Script	SoundManager
Music_Audio Source	Music/Background Audio Source (Audio S
Sound_Audio Source	Sound AudioSource (Audio Source)
Music Dictionary	
Happy Music	HappyMusic
Sound Dictionary	
Card Placed	cardPlaceSound
	dingSoundEffect
	dingSoundEffect

To remove, just do the reverse.

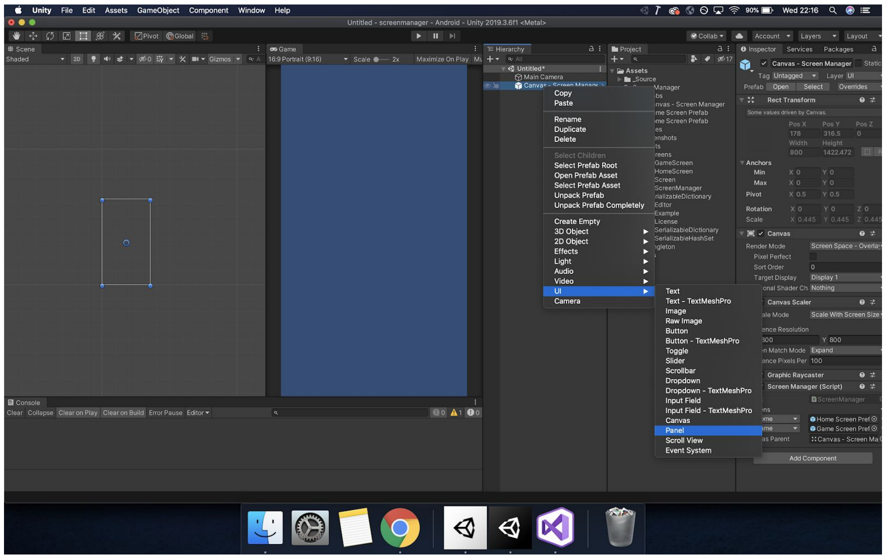
Adding background music is the same but with using the BackgroundMusic enum and using the MusicDictionary.

Note: background music uses Unity AudioSource.Play() as opposed to AudioSource.PlayOneShot().

Adding more UI screens

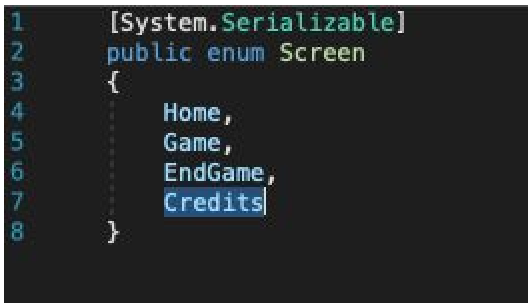
This asset uses a ui statemanager, which makes it really easy to change/manage ui screens during runtime.

In the hierarchy window > right-click Canvas - Screens Parent Example > then select UI > Panel.



Rename the Panel to the name of your screen for example Credits > create a prefab by placing it into the project window then delete it from the hierarchy.

Find Screen.cs and add the name of your screen, for example Credits, to the enum list.

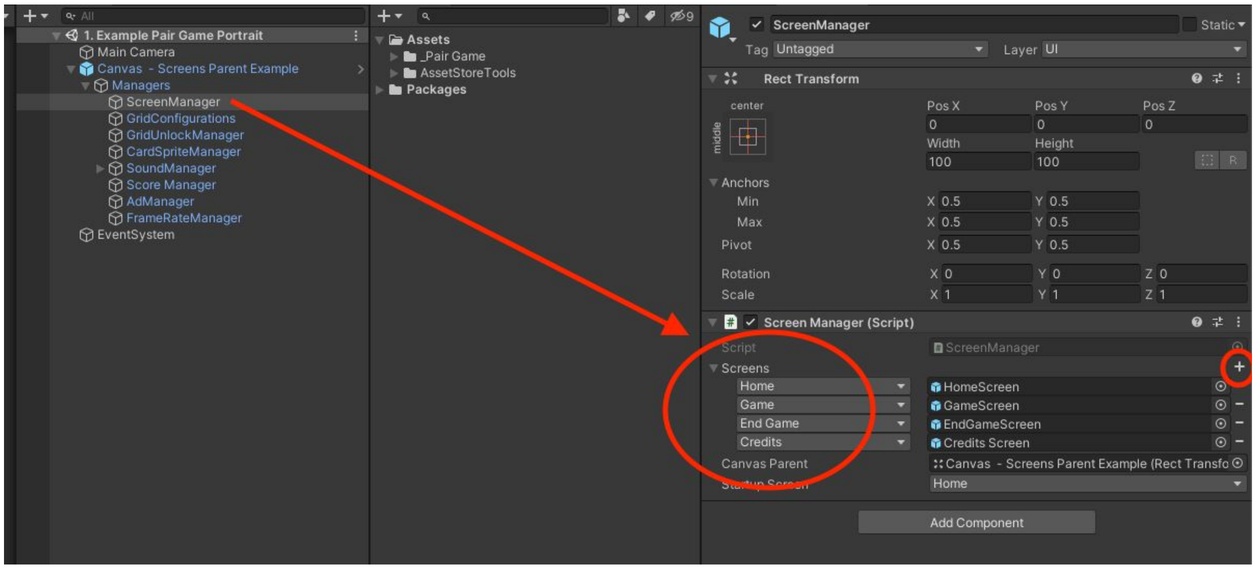


```
[System.Serializable]
public enum Screen
{
    Home,
    Game,
    EndGame,
    Credits
}
```

Next locate “ScreenManager” child gameobject of Canvas - Screen Parent Example gameobject > Managers within the hierarchy window.

Select and locate the Screens dictionary on ScreenManager.cs within the inspector window.

Select the + sign then select the enum drop down on the newly created field > you should now see the enum you just created in the drop down e.g. “Credits”. Select it and add the UI screen prefab for example the Credits Panel Prefab.



ScreenManager
Tag Untagged Layer UI
Rect Transform
center middle
Pos X 0 Pos Y 0 Pos Z 0
Width 100 Height 100
Anchors
Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Screen Manager (Script)
Script ScreenManager
Screens
Home HomeScreen
Game GameScreen
End Game EndGameScreen
Credits Credits Screen
Canvas Parent Canvas - Screens Parent Example (Rect Transfo
Startup Screen Home

Scoring

Scoring uses a <vector2(gridsize), Key<tries,value=time>>dictionary. You can find the logic for this stored in ScoreManager.cs

- Saving and loading logic uses playerprefs.
- EndGameScreen.cs uses ScoreManager.GetTopFiveScores() which returns the top five scores based upon tries count.

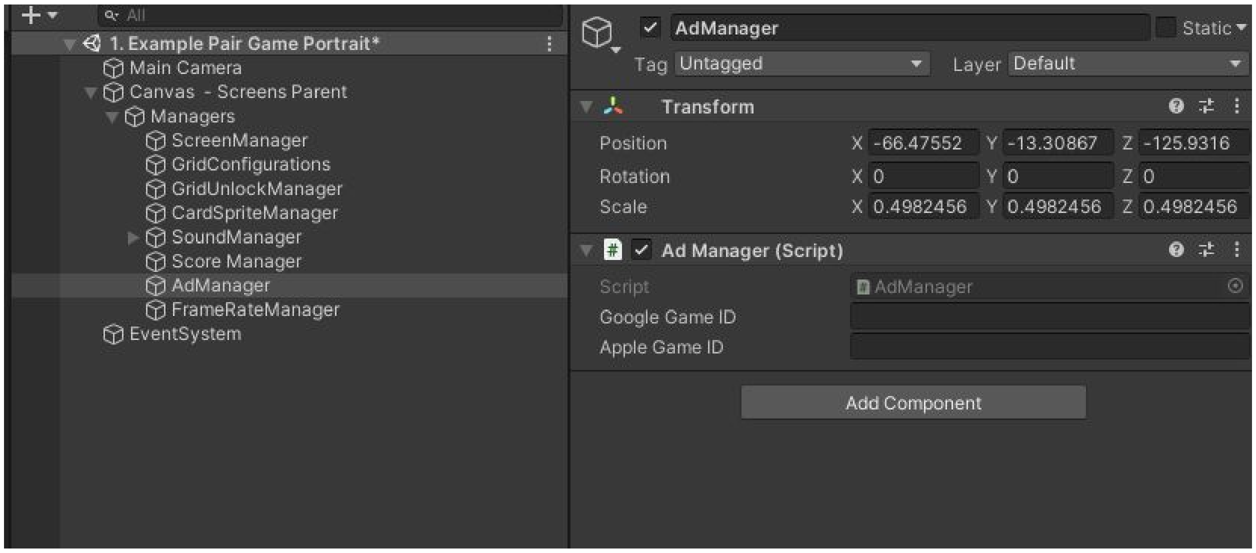
AdManager

AdManager.cs is a simple script that controls Unity Ads initialization and triggering of a video ad placement.

Setup

If you haven’t already - integrate Unity Ads SDK through the Package Manager or asset store - https://unityads.unity3d.com/help/unity/integration-guide-unity.

Locate the AdManager.cs in the hierarchy or project window and then enter your google gameid or apple gameid. You can find this in the operate section of your dashboard. https://dashboard.unity3d.com/



Within the script, you will see ShowVideo(), which is called when a user selects on a level.

NOTE: The asset is designed to show an ad every two times the user selects a new level. To lessen the spam of Unity Ads.

NOTE: You need to change the placementid, within the script to match your placement id's on https://dashboard.unity3d.com/

OverrideFrameRate.cs

By default, Unity IOS caps the framerate to 30FPS, this makes the card animations appear jittery. This simple script overrides the cap to 60FPS.

Other Things to note:

UI sprites shader

The cards require the UI Card Shader, this shader prevents UI from showing if the card is not facing the camera.

Error: Sprite limit

Error Sprite limitation will be thrown if there aren't enough sprites to fill all cards.

Random Sprites

Random Sprites chosen from the GameManager item list are chosen from random for each game, making each game not always the same.

Shuffling

Shuffling of cards make the game more random and hard, this functionality is built within GameManager.cs

Cards.cs

Responsible for hiding and showing/display specified sprites/cards.

Communicating with GameManager externally doing by calling GameManager.Instance then .Method() or .Function()

GameManager

GameManager.cs is the brains/manager of the game, controlling Cards.cs, TriesManager.cs, TimerManager.cs

Change level

You can change the level programmatically by accessing GameManager.Instance.Setup(int _rowCount, int _columnCount)

Restarting the game

You can restart the level by calling GameManager.Instance.Restart()

GameStarted(), CardStarted(), GameOver

You can add additional functionality to GameStarted(), CardsMatched(), GameOver() methods

TriesManager.cs

Responsible for counting the amount of tries a user makes and displays the value using a text component.

TimerManager.cs

Responsible for counting/showing the duration of a session/game.

Contact

Have any questions? Contact me hunter.glen@gmail.com