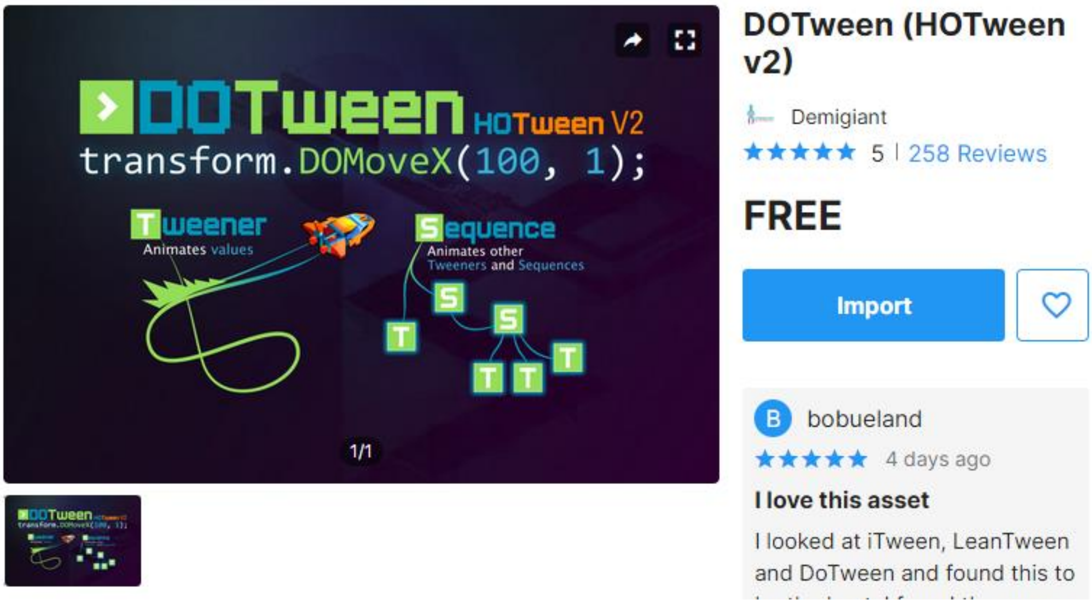
Hello. Thank you for buying my asset.

How to setup?

1) Before building this game to your phone you need to download the latest version of Admob plugin for Unity from there: https://github.com/googleads/googleads-mobile-unity/releases/tag/v5.4.0
2) After downloading complete import **GoogleMobileAds.unitypackage** to project. And finish steps from guide here: https://developers.google.com/admob/unity/start
3) In the Unity editor click **for Android**: Menu→Assets→External Dependency Manager→Android Resolver→Force Resolve.

   For iOS:

   Menu→Assets→External Dependency Manager→ iOS Resolver → Install Cocoapods
4) Install free **DOTween (HOTween v2)** plugin from Unity Asset Store for animations.


DOTween HOTween V2
transform.DOMoveX(100, 1);
Tweener
Animates values
Sequence
Animates other Tweeners and Sequences
S
S
T
T
T
T
1/1
Import

DOTween (HOTween v2)

Demigiant

★★★★★ 5 | 258 Reviews

FREE

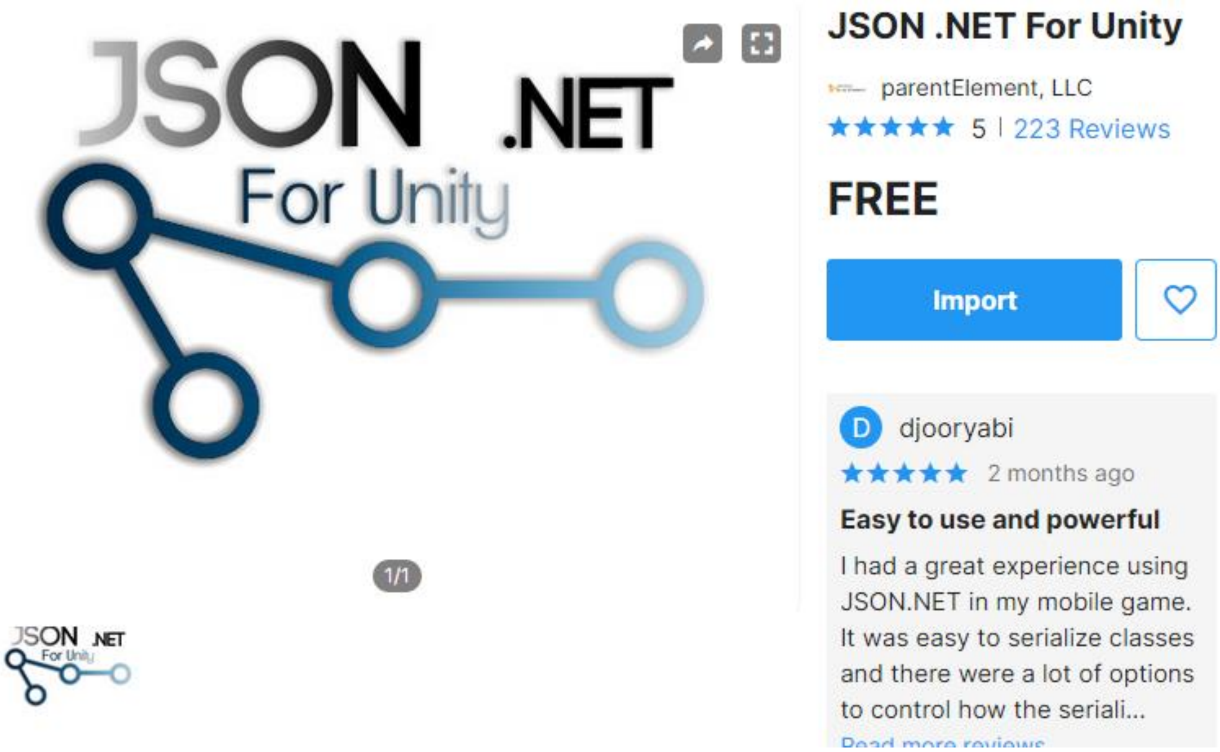
bobueland

★★★★★ 4 days ago

I love this asset

I looked at iTween, LeanTween and DoTween and found this to

5) Install free **JSON .NET For Unity** plugin from Unity Asset Store for serialize settings.



JSON .NET For Unity

parentElement, LLC

★★★★★ 5 | 223 Reviews

FREE

D djooryabi

★★★★★ 2 months ago

Easy to use and powerful

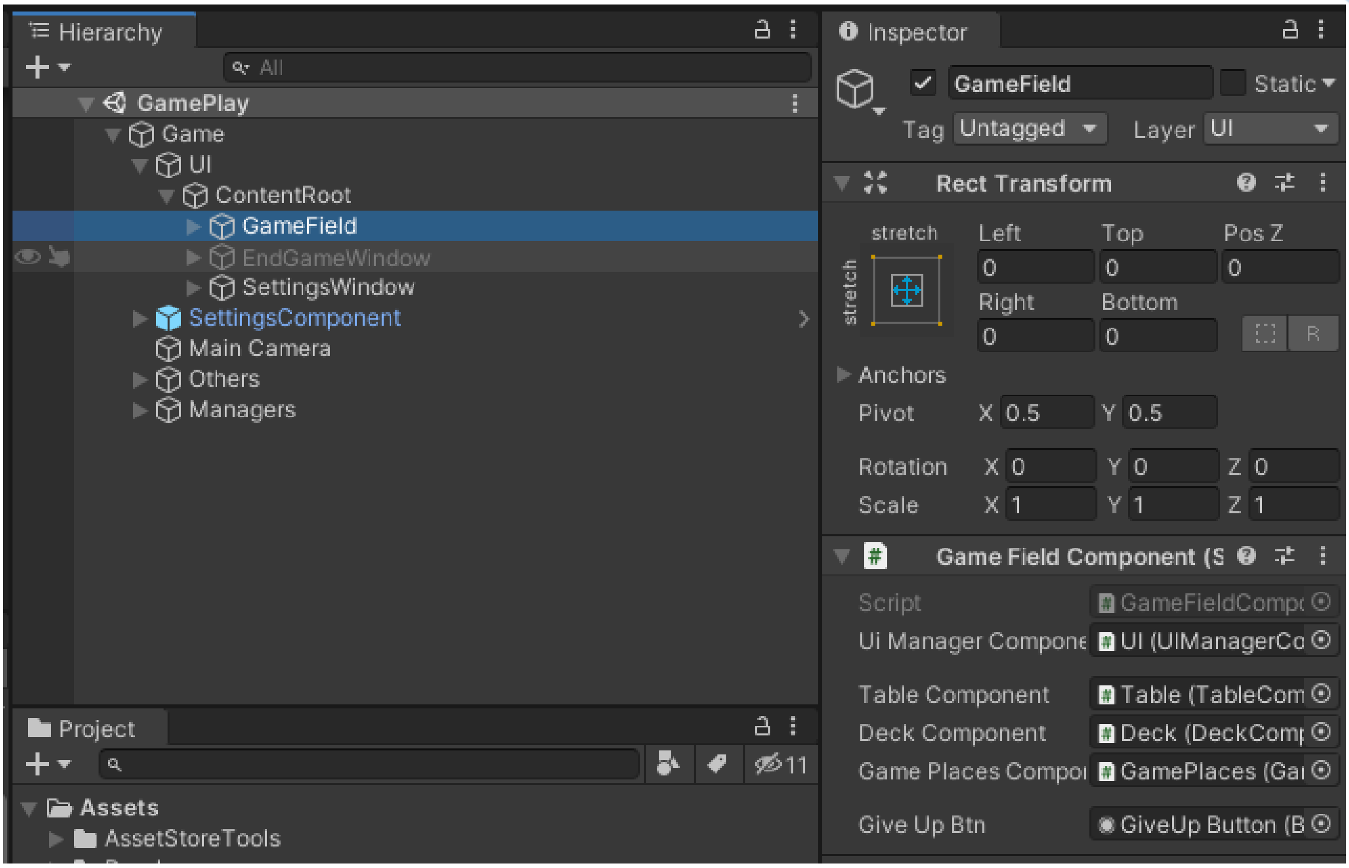
I had a great experience using JSON.NET in my mobile game. It was easy to serialize classes and there were a lot of options to control how the seriali...

Read more reviews

6) Success. Now you can use asset.

Durak Template OOP v 1.2 - SelvAssets

1) The entry point of this game is:



Hierarchy

All

- GamePlay
  - Game
    - UI
      - ContentRoot
        - GameField
        - EndGameWindow
        - SettingsWindow
    - SettingsComponent
    - Main Camera
    - Others
    - Managers

Project

Assets

- AssetStoreTools

Inspector

☑ GameField □ Static

Tag Untagged Layer UI

Rect Transform

stretch

Left	Top	Pos Z
0	0	0
Right	Bottom	
0	0	

Anchors

Pivot X 0.5 Y 0.5

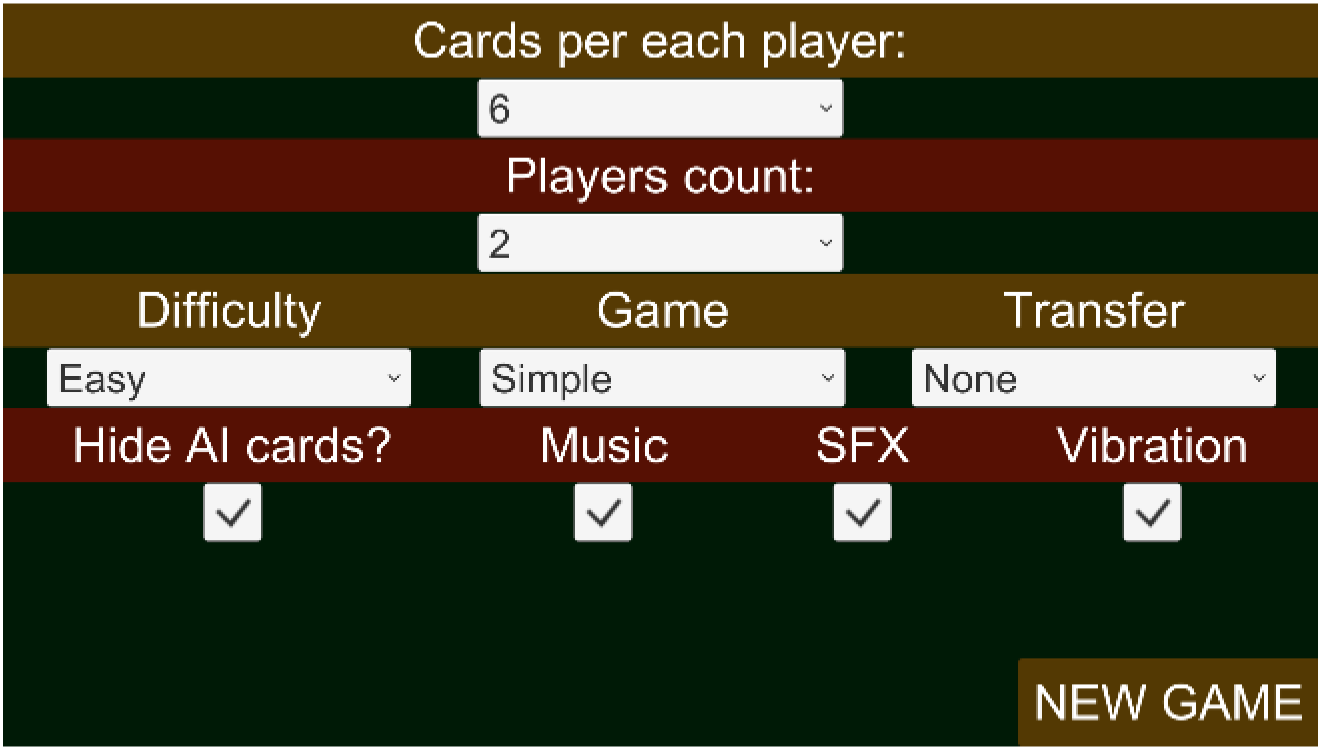
Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Game Field Component (S

Script	GameFieldCompo
Ui Manager Compone	UI (UIManagerCo
Table Component	Table (TableCom
Deck Component	Deck (DeckComp
Game Places Compo	GamePlaces (Ga
Give Up Btn	GiveUp Button (B

Settings window contains 9 type of parameters:



Cards per each player: 6

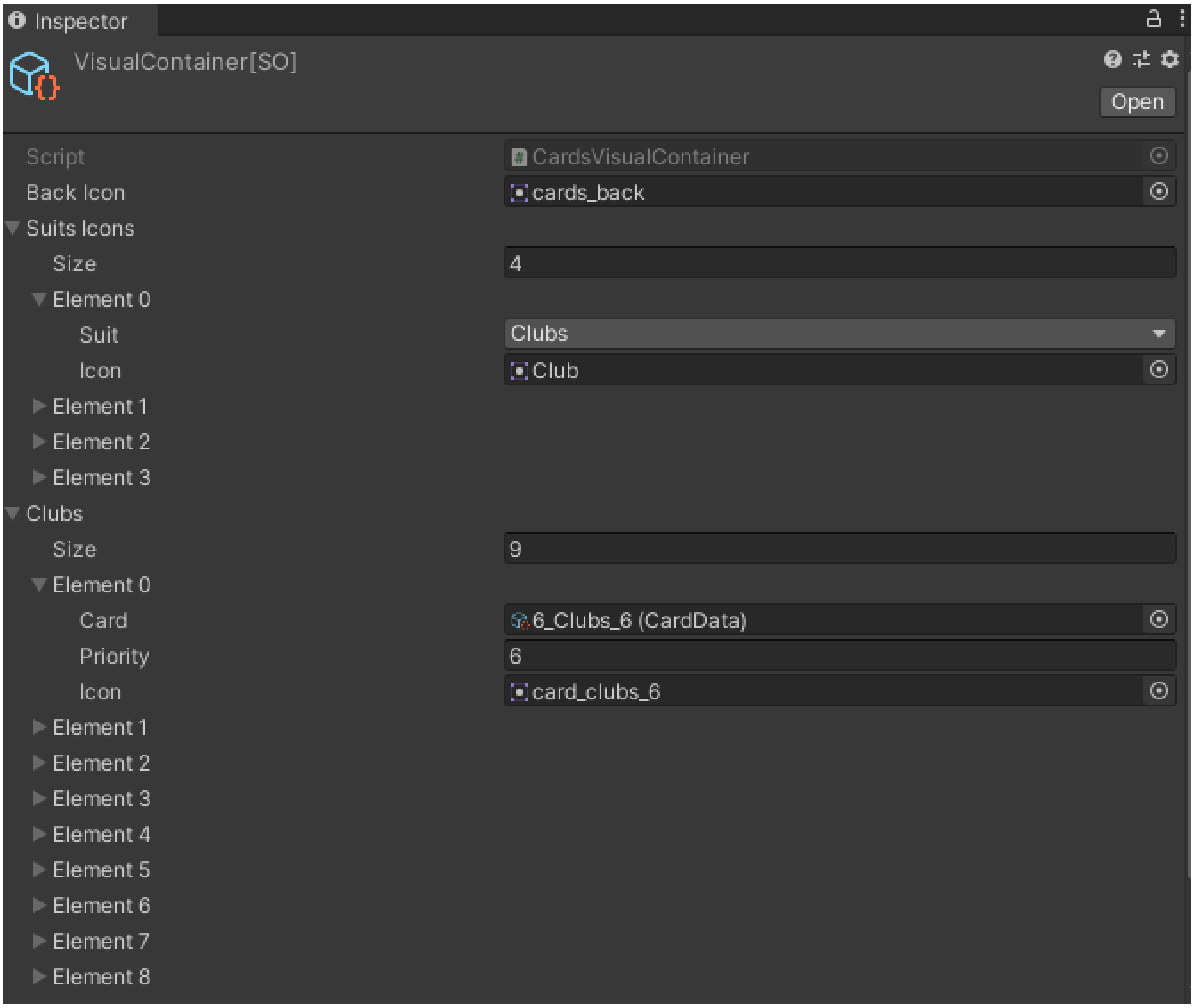
Players count: 2

Difficulty	Game	Transfer
Easy	Simple	None

Hide AI cards?	Music	SFX	Vibration
☑	☑	☑	☑

Cards per each player (2..6) – How much cards user can hold in session
Players count (2..6) – How much users will be play in session (include YOU)
Difficulty – The power of AI. Easy and Hard.
Game – The game type/mode. Simple and Throw-in.
Transfer – Will be available in the next releases.
Hide AI cards – Show/Hide cards in game window for AI players.
Music – On/Off background music. (State is saved in persistent data)
SFX – On/Off SFX music. (State is saved in persistent data)
Vibration – On/Off vibration. (State is saved in persistent data)

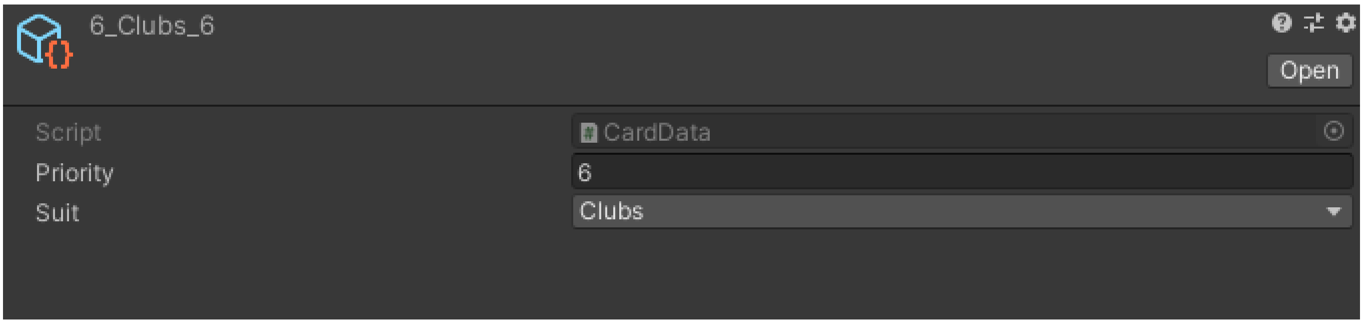
VisualContainer[SO] contains data and links to art and CardData scriptable object. In future releases I will add way to change back and face art of card.



VisualContainer[SO]

Script CardsVisualContainer
Back Icon cards_back
Suits Icons
Size 4
Element 0
Suit Clubs
Icon Club
Element 1
Element 2
Element 3
Clubs
Size 9
Element 0
Card 6_Clubs_6 (CardData)
Priority 6
Icon card_clubs_6
Element 1
Element 2
Element 3
Element 4
Element 5
Element 6
Element 7
Element 8

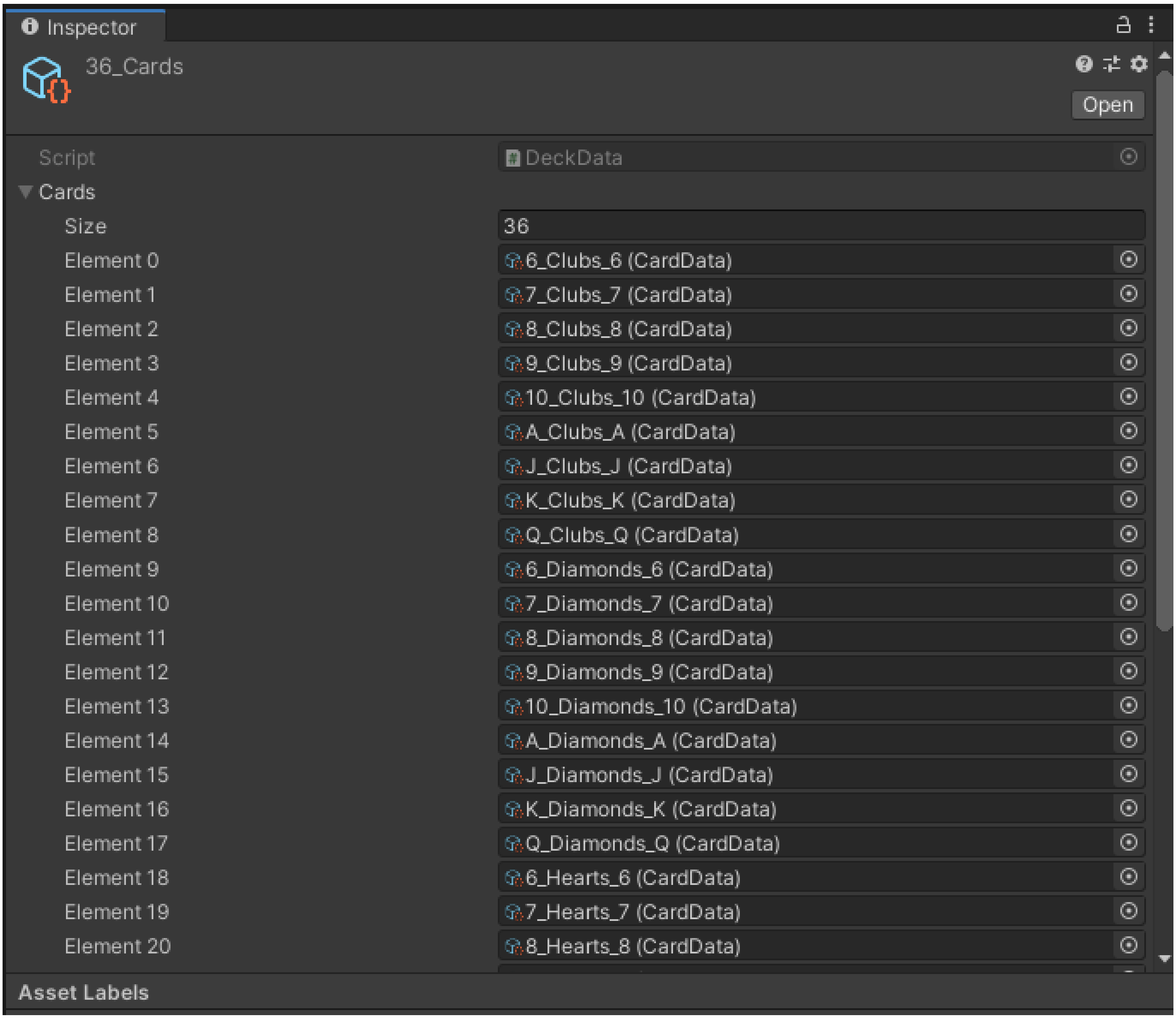
Each card is scriptable object that contains Suit and Priority.



6_Clubs_6

Script: CardData
Priority: 6
Suit: Clubs

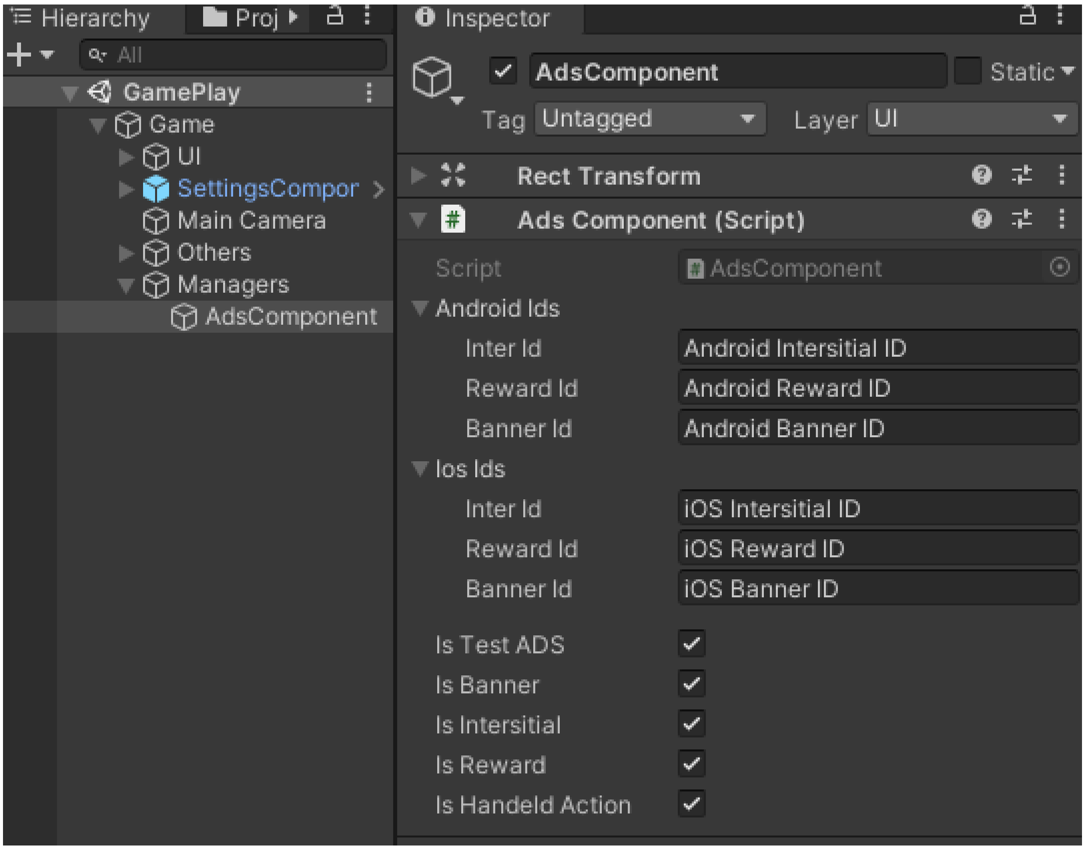
DeckData has links to all cards in Deck.



36_Cards

Script	DeckData
Cards	
Size	36
Element 0	6_Clubs_6 (CardData)
Element 1	7_Clubs_7 (CardData)
Element 2	8_Clubs_8 (CardData)
Element 3	9_Clubs_9 (CardData)
Element 4	10_Clubs_10 (CardData)
Element 5	A_Clubs_A (CardData)
Element 6	J_Clubs_J (CardData)
Element 7	K_Clubs_K (CardData)
Element 8	Q_Clubs_Q (CardData)
Element 9	6_Diamonds_6 (CardData)
Element 10	7_Diamonds_7 (CardData)
Element 11	8_Diamonds_8 (CardData)
Element 12	9_Diamonds_9 (CardData)
Element 13	10_Diamonds_10 (CardData)
Element 14	A_Diamonds_A (CardData)
Element 15	J_Diamonds_J (CardData)
Element 16	K_Diamonds_K (CardData)
Element 17	Q_Diamonds_Q (CardData)
Element 18	6_Hearts_6 (CardData)
Element 19	7_Hearts_7 (CardData)
Element 20	8_Hearts_8 (CardData)

To setup ***Admob*** settings you should fill AdsComponent:



☑ AdsComponent ☐ Static

Tag Untagged Layer UI

Script AdsComponent

Android Ids

Inter Id Android Intersitial ID

Reward Id Android Reward ID

Banner Id Android Banner ID

Ios Ids

Inter Id iOS Intersitial ID

Reward Id iOS Reward ID

Banner Id iOS Banner ID

Is Test ADS ☑

Is Banner ☑

Is Intersitial ☑

Is Reward ☑

Is Handeld Action ☑

And **GoogleMobileAdsSettings.asset**

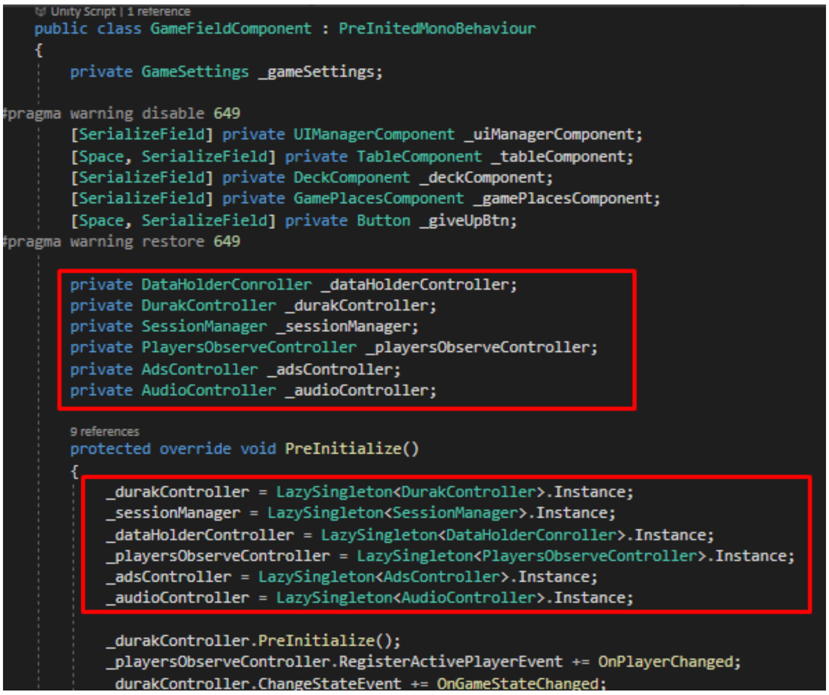
For activating test ADS you should set **IsTestADS** variable to **TRUE** before play (like on screen). For deactivating test ADS you should set **IsTestADS** variable to **FALSE** before play.

This asset represent the correct way of development game on Unity and OOP. So, the controllers responsible for logic of game, components and views responsible for visual and provides data for controllers. If logic is simple and very local you can write it in component. But for large logics you should create controller and write logic there.

To simplify access to controllers and there is no need to create new instances of controllers.

I commented each method so you can easy understand how it works.

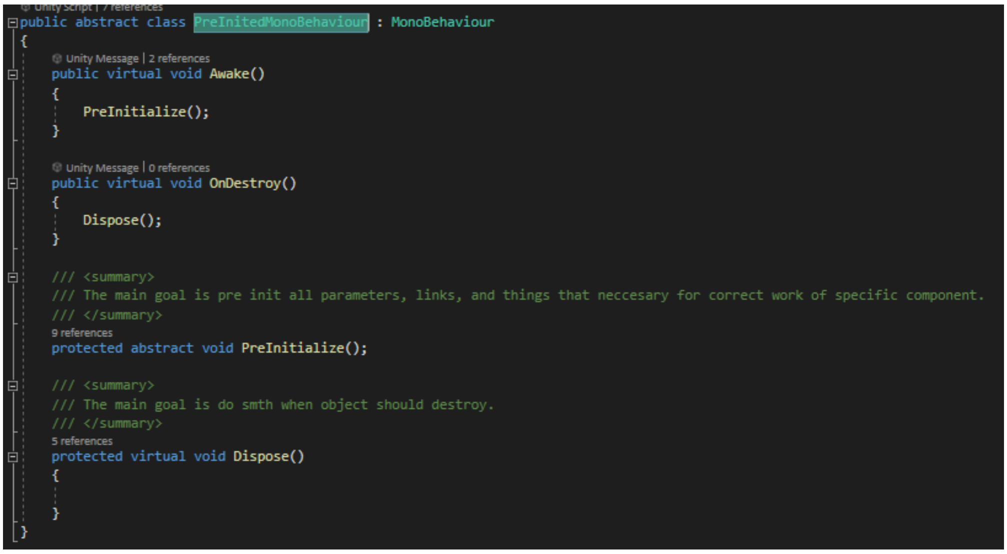
I provide in this asset LazySingleton example:



```
    Unity Script | 1 reference
    public class GameFieldComponent : PreInitedMonoBehaviour
    {
        private GameSettings _gameSettings;

#pragma warning disable 649
        [SerializeField] private UIManagerComponent _uiManagerComponent;
        [Space, SerializeField] private TableComponent _tableComponent;
        [SerializeField] private DeckComponent _deckComponent;
        [SerializeField] private GamePlacesComponent _gamePlacesComponent;
        [Space, SerializeField] private Button _giveUpBtn;
#pragma warning restore 649

        private DataHolderConroller _dataHolderController;
        private DurakController _durakController;
        private SessionManager _sessionManager;
        private PlayersObserveController _playersObserveController;
        private AdsController _adsController;
        private AudioController _audioController;

        9 references
        protected override void PreInitialize()
        {
            _durakController = LazySingleton<DurakController>.Instance;
            _sessionManager = LazySingleton<SessionManager>.Instance;
            _dataHolderController = LazySingleton<DataHolderConroller>.Instance;
            _playersObserveController = LazySingleton<PlayersObserveController>.Instance;
            _adsController = LazySingleton<AdsController>.Instance;
            _audioController = LazySingleton<AudioController>.Instance;

            _durakController.PreInitialize();
            _playersObserveController.RegisterActivePlayerEvent += OnPlayerChanged;
            _durakController.ChangeStateEvent += OnGameStateChanged;
```

This way will get rid of multiple links in inspector and streamline process of accessing to controllers in game components. So if you has MonoBehaviour component on scene and you need to get access to this component in other MonoBehaviour you should link them via field in inspector. If you need link to controller you should use the example above.

I provide abstract component **PreInitedMonoBehaviour** that will be initialized in Awake function (This need to correctly provides all links to controllers for MonoComponents, like on screen above).



```
  Unity Script | 7 references
public abstract class PreInitedMonoBehaviour : MonoBehaviour
{
    Unity Message | 2 references
    public virtual void Awake()
    {
        PreInitialize();
    }

    Unity Message | 0 references
    public virtual void OnDestroy()
    {
        Dispose();
    }

    /// <summary>
    /// The main goal is pre init all parameters, links, and things that neccesary for correct work of specific component.
    /// </summary>
    9 references
    protected abstract void PreInitialize();

    /// <summary>
    /// The main goal is do smth when object should destroy.
    /// </summary>
    5 references
    protected virtual void Dispose()
    {

    }
}
```

For other questions write me an email: *support@selvassets.ltd.ua*

Thank you for advance!