# Unity Version: 2021.3.14f1 LTS

# System Setup:

**Flow**

* The game starts with GameInstaller.cs (I used Zenject to manage the bindings and for dependency injection)
* SlotMachineInstance - initializes the reels and controls the spin/stop on the reels
  + ReelInstance
    - Each reels are built at runtime, the symbols are based on the ReelDataAsset scriptable object.
    - Reel has its own controller that is responsible for the spin animation and stopping the symbol on the center of the rows.
    - Reels are dynamic and symbols can be replaced from the reel data asset.
* Combinations - responsible for displaying and consuming the payout lines combinations, it has its own view and controller.
* GameCurrency - manages the currency in the game. It accepts observers to notify when there is a change in the currency amount.
* Bet - controls the bet amount for each spin.

**Services**

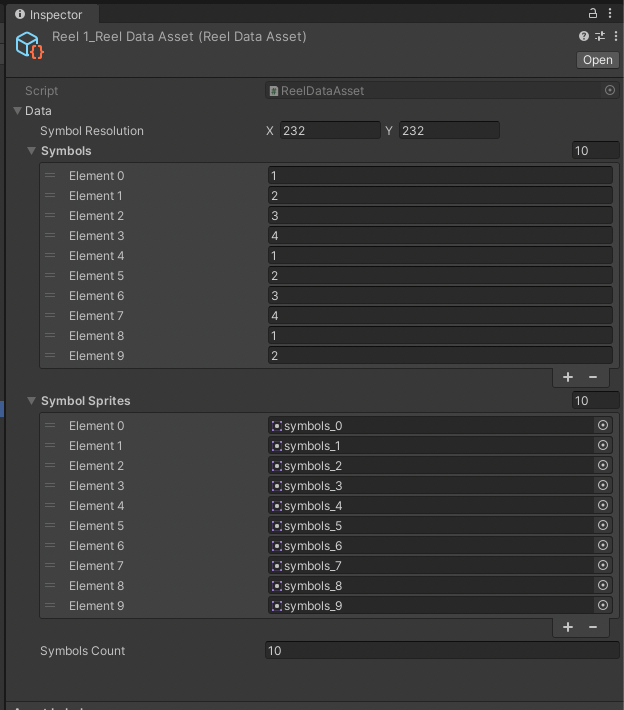
* PayoutService - calculates the payout when there is a winning line from the spin result, and notifies the subscribers for the reward.
* CurrencyService - controls the flow of currency for the game, also an observer like payout service and notifies subscribers for changes in currency amount.

**Architecture:**

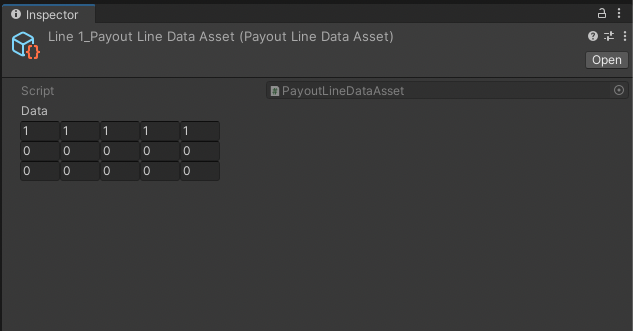
* The game has an architecture under Assets/Scripts/Anino
* It consists of Framework, Implementation with separate assembly definitions for decoupled implementation and testability.
* The Tests folder contains UnitTests.

# Data Sources:

* Data sources are under Assets/Data
* There are two types of data
* **ReelData** to modify reels symbol. It is a scriptable object you can create an instance from context menu->Create->Anino->Data->ReelData
  + This is consumed by the ReelInstance.



* **PayoutLineData** to add or modify payout line combinations.
  + Values can be modified from the Inspector. 1 values are the shaded/marked cells and the 0 values are the blank cells.
  + The data is being consumed by Combinations class and PayoutService.
  + You can create an instance from context menu -> Create -> Anino -> Data -> PayoutLineData



# Scalability:

The data of the game can be improved to be changed remotely either via assetbundle or remote config. And since there is a simple architecture already set up and unit tested, we can add new features without worrying too much of affecting other implementations.

# Flexibility:

Since the data sources are on scriptable objects. One can easily replace reel symbols and payout line combinations visually from the Inspector.

# MVC Usage:

The design of the system is MVC. For example the reels visual and logics are on separate classes that are decoupled from each other. The view doesn’t know the data that are being processed by the controllers and the controllers are the one supplying what visuals to be displayed on the views.

# Improvements:

* Remote config setup
* Selecting number of reels and number of lines to bet.
* Implementing data id for symbols so players can also win from individual symbols not just from the payout lines.
* Implementing winning streaks

# Output:

**Github Link:** <https://github.com/k3nsalazar2012/anino_exam>