Samet Burak Turk
FortuneMineGamesProjectReport

MVP Architecture:

The Model, View and Presenter classes are the first classes initialized by Initializer. Both View and Presenter depend on Model at construction because Model has Slot array (data class that holds reward info) and ResourcePaths. Model also holds Wallet and loads previous wallet.json if one exists.
Presenter Initializes RouletteLogic and listens to button push. When the button is pushed calls RouletteLogic to select a slot asynchronously and triggers the roulette animation based on the selected slot's index.
Lastly, View loads assets from resources and builds the scene using assets.
Due to time constraints I couldn't implement some features. Ideally I would have interfaces for each MVP class to allow flexibility, but my implementation does not use inheritance. AssetConfig class is not comprehensive enough since the UI elements are pre build in the scene. Also it could be ScriptableObject instead of instantiated by code in Model.

RouletteLogic:

RouletteLogic maintains Slots and an ActiveSlotList to track active slots and remove selected ones. When choosing a random slot it calculates the weight of ActiveSlotList and uses System.Random to get a random number and iterates over the list returns the current slot. It should have been an interface so it could be swapped with a more complex implementation, such as pseudo-random distributions.

Datas:

RewardDataBase holds an enum. Even though enums are static and not scalable, they were sufficient for this project.
Slot holds reward, amount, weight and index data representation of CircleRows.
Wallet holds a reward int dictionary.

Prefabs:

Loaded with Resource.Load and not Addressable. CuisinePartyCircleRow and SceneBuilder are only prefabs, other objects are in the scene at load. AssetConfig is very small and simple. I could have used the factory and changed the sprite for different themes and animations on BaseCircleRow and BaseUi however as mentioned above time constraints .

Animation:

CircleRows have ICircleRowAnimations interface implemented with CuisinePartyCircleRowAnimations which uses Coroutines to change alpha values gradually. Since it uses an Interface the animation can be swapped with different implementations. For each sprite there is a child object in CircleRowObject. CuisinePartyCircleRowAnimations class sequentially plays CircleRow animation to achieve Roulette animation. After 2 full turns it lands on the selected one.The animation starts before Roulette selects a slot. If no slot is selected after two full turns, an additional turn is added until RouletteLogic determines the slot. This is why SlotSelectEvent has a Task<Slot>.

EventSystem:

EventSystem is an implementation of the event bus pattern. Events are centralized. GlobalEventSystem is a singleton EventSystem and this is what all classes use to communicate. Except for the sequential animations, classes are mostly decoupled.