**BUILD**

To be able to test the game with an emulator, .APK built has been edited:   
 “Optimized Frame Pacing = False” **Project Settings**

**PROJECT**

**Classes:**

**Spin Game Manager:** It has accessibility with static instance. It holds references to other needed to provide references. It holds “LOSE, RESTART, STOP GAME, INITALISE NEXT ROUND” functions.

**Zone Initialiser:** Vital script initialises the zones and spin values. Since spin values are highly **dependent** on zone level, spin is initiated by this script.

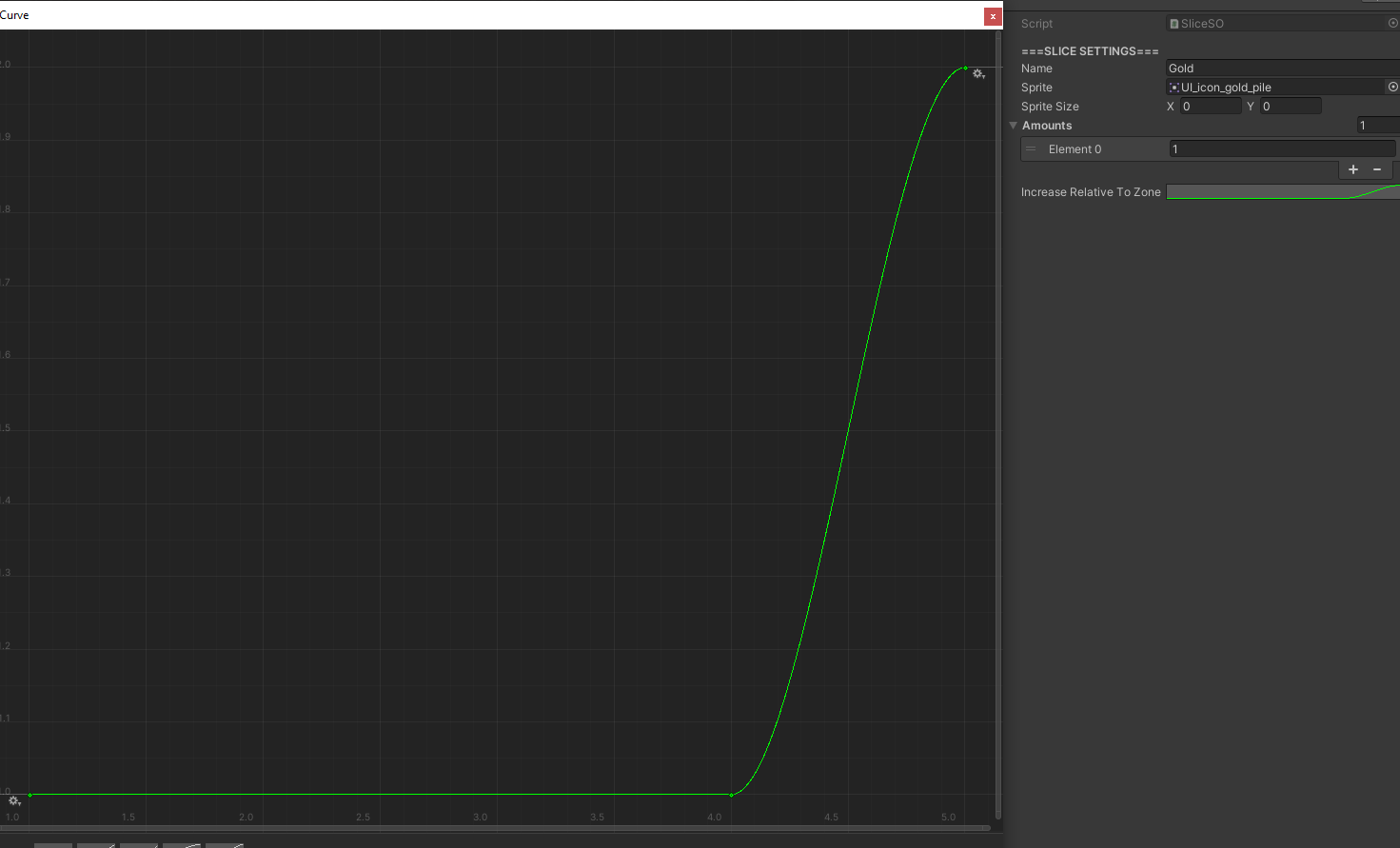
**Zone Tracker:** Keeps track of the zone and provides info such as “CurrentZone()”. It is responsible of setting the **NextZone** and **Managing Zone Levels**. This script is holder of event **ContinueNextZone.**

**Spin Displayer:** Holder of Scriptable Object **SpinSO**. It holds method of setting references of spins and **slices**, as well randomises the **slices. Also,** this script decides on the winner slides through **ClosestSliceToIndicator()**

**Spin Content Randomiser:**  Randomly sets all the slices and places a bomb including their values.

**SpinSO:** Scriptable Object script, hold info of name, sprite, indicator and slices. It has a method to set references for slices called by the **Spin Displayer.**

**Slice Displayer:** It holds reference of **SliceSO**, **Images** and **RewardNotifier’s Animator**. This script represents one slice piece. This piece sets itself relative to **SliceSO**.

**SliceSO:** It holds Info of name, sprite and possible amounts. It also has an important designer friendly value **AnimationCurve** to increase amounts relative to zone levels:

X value means time. And, if time is fed with **zone level**, corresponding value can be **amount to be given as reward**.

**Wheel Spinner:** It deals animation of spinning the wheel and randomises the spin values to be able to increase the chance factor. It is the holder of **StoppingWheel** event.

**Reward Click Handler:** This is used as a pioneer to initiate next round. It rechecks if the round is fail or not and executes functions relative to that. It claims the reward and sends it to Inventory or accepts it as a bomb and triggers **Lose** function.

**MyButton:** To be used by other scripts, it could be used anywhere.

**Inventory:** Responsible of holding objects and setting their parents. It’s best to use with layout groups.

**Rect Scroller:** Animation for background image by repeating background tiles and moving UV.

**Dynamically Rect Mover:** It is used to move object according to need. This script used to set zone level pieces locations.

**EVENT TIMELINE**

**STOPPING THE WHEEL EVENT ->** Is Called When Wheel Stops Spinning (1)  
**CONTINUENEXTZONE EVENT ->** Is Called When Player Clicks On The Card (2)

**PACKAGES USED** -> SIMPLEFOLDERICON: Folder Segregation,  
 HIERARCHY 2 (EDITED) : Hierarchy segregation and utility.  
RECORDER: Recording Gameplay

**Points of Interests as Method**

**ZoneInitialiser.SetSpritesAndInitialiseSpin()** => Main method to randomise spin values relative to zone level.

**SpinDisplayer.ClosestSliceToIndicator()** => Finds the one that is closest to indicator to determine the reward.

**Spotted Issues**

**Issue**: Tight-Coupled Classes are issued to avoid null-references and proper timing of calling the methods. Some other Unity-wise programming patterns must be searched to avoid similar issues.

**Issue**: Class-native methods are called in other classes which results complexity. However, provides one method to be called to initiate everything and avoiding possible null references since spin values are deeply related to zone values. Decoupling and OOP practices are needed to find the correct pattern.

**Common Issue**: Complexity caused by lack of proper planning increased bug-fixing time drastically. Possible temporary solution would be proper documentation and commenting.

**Visual Bug (Stretching)**: Due to sprites’ native sizes are not 1:1, visual bugs due to unwanted stretches caused disturbance.

**Self-Criticism**: Lack of experience in proper and planned programming causing issues listed below. Decision making process is taking time before deciding on classes and architecture. Current experience was highly **influenced** by fast-paced prototype programming without planning the entire project beforehand.

As well as that, more experience in UI and 2D programming required to polish programming and UI programming concepts.