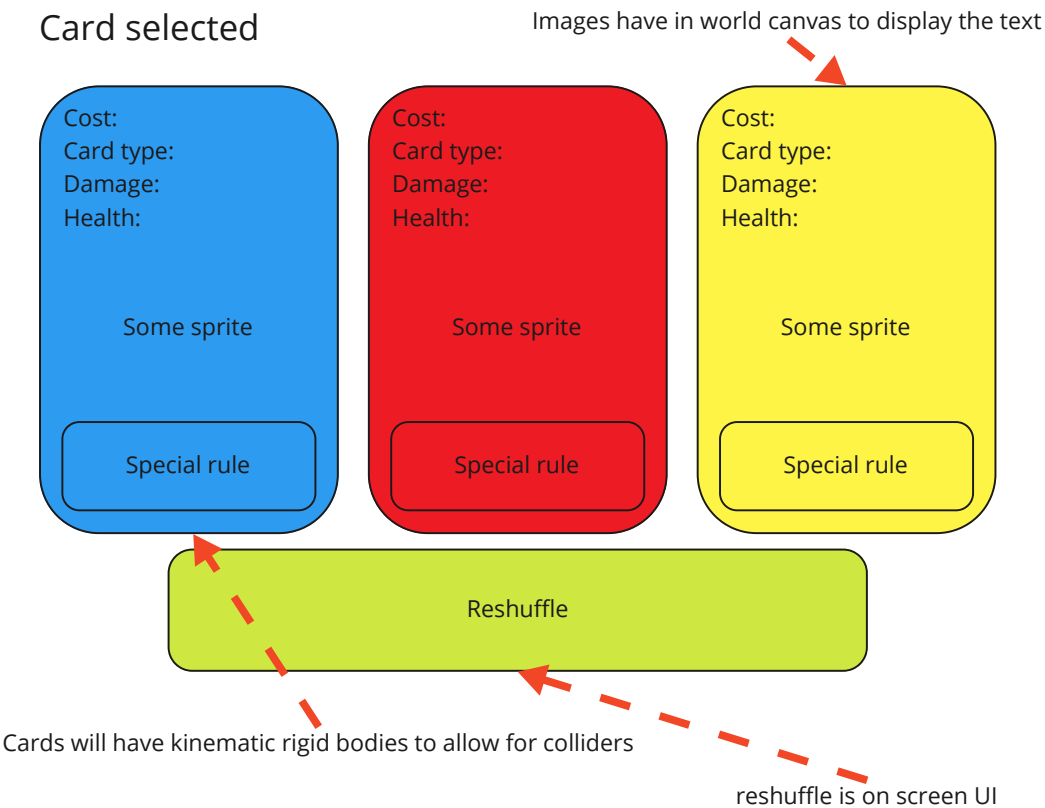
Programming Systems project - Card System

Joe Place 991717262

The system will be a card randomization and selection system. The system will display 3 cards for an army strategy game that represent units. The cards will display their different values on text objects on a greater object resembling a card. Their will be a Reshuffle button that allows the player to randomize the cards again.

The player will select one of the cards which will then move up and down to signal to the player the card was selected. There will be a UI element that communicates to the player what was selected.



Each of the special rules have no impact but change the visuals
Infantry: +1 movement
Armor: -1 to all incoming damage
Artillery: range of 5

Deconstruction:

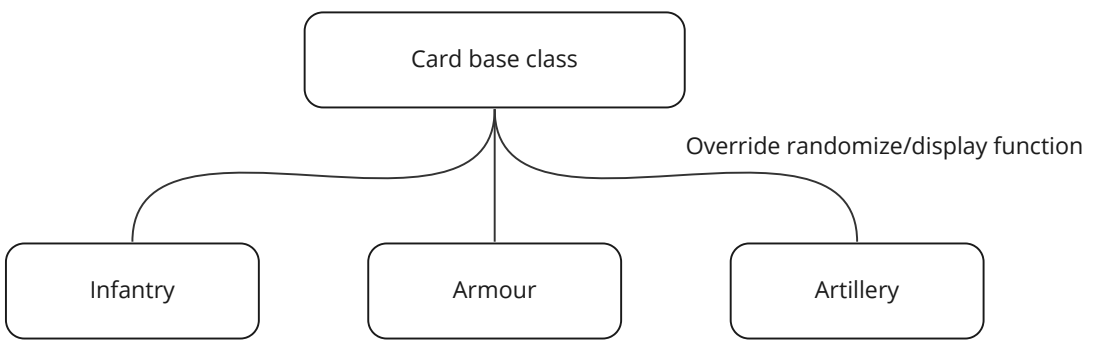
a) The system solves the problem of giving the player options on a reward as well as gives a chance of randomness to make things unpredictable

b) It displays three card options that are each a different unit type with randomized attributes. This functionality can be achieved by having a base card class that has all the parameters and values that also has the main functions for displaying the information.

c) Composition can be used for the interaction as OnMouseDown() needs a rigidbody and collider to operate which can trigger the select event. As well composition can be used for the texts and sprites to display the information in a cohesive manner. The reshuffle button will be a UI button that triggers the function on start up that randomizes the values. Inheritance will be used to accomplish the different values and card types as each can have different values displayed to the card as well as a special rule. There will be three types of cards each with a different unit type.

d) The base class functionality will control the base values for Cost, Damage, Health, and card type. It will contain the object references for the text objects and SpriteRenderers as well as have a function that randomizes the values and displays them.

e) The three inheriting classes are Infantry, Armour, and Artillery that will each have different value ranges, types, sprites and special rule text.



each has different information and values on the cards

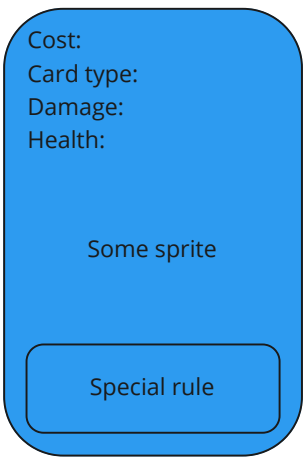
Cards

The class will have a randomizer static class that each can access to randomize its attributes. This will be controlled by a CardController empty object that will have a static function that will randomize the attributes with a static object variable that can be accessed by the cards.

Each of the attributes will be randomized in ranges. The ranges will be overwritten with each of the children to give them their identity.

There will be a text object on the screen that will display what card was selected.

The cards will have a OnMouseDown() function that triggers the coroutine Selection Animation. Each card will have a funciton that sets the values to the text objects but each have functions that override the values so they are displayed differently.



This will be the base layout for the cards. The cards will then have child objects that will inherit from this parent object. The different children will be Infantry, Armor, and Artillery. Each will have randomized damage in different ranges as well as a special rule. There will be one card of each displayed to the scene.

Problems:

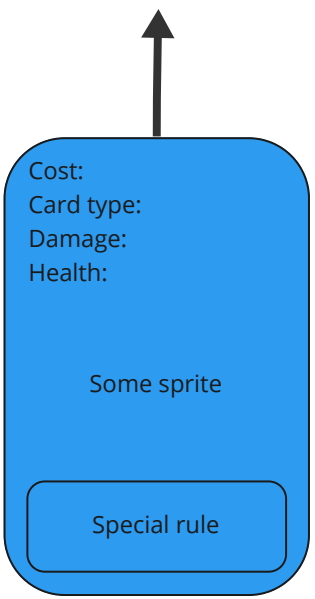
- The card GetComponent are not detecting any components on the object. Solution was to simply make it public and drag in the proper things as with lots of text objects this could prove unreliable
- The card object themselves when using a horizontal layout group and elements keep disappearing. Solution was to have three hard displayed cards to the screen

Selection Animation

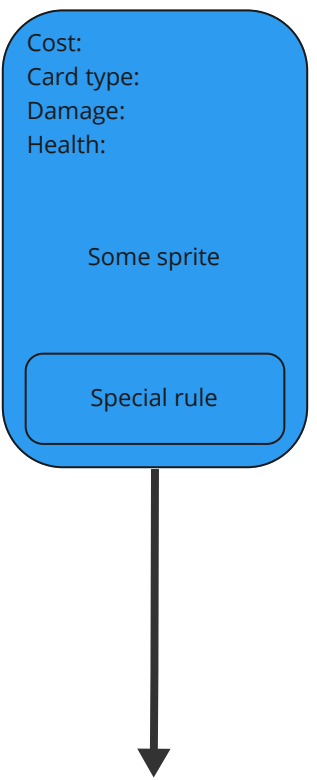
The coroutine will be a animation that will give feedback to the player and move the card off screen to showcase they have been selected. Using a coroutine will allow the object to move over a period of frames giving it a more fluid visual experience.

The animation will be triggered once it has been clicked to be selected. The animation will have the card move up slightly then down to the bottom of the screen. These will be contained with while loops and yield return null; with the second one activating after it reaches the top location

First section of coroutine



Second section of coroutine



Problem:
The animation seemed very blocky. Changed it to oeprate on animation curves. However, this change would not give the desired elastic effect using lerp as the values are clamped between 0 and 1. So using lerpUnclamped() allows me to overshoot and have the elastic effect

Card Controller

The card controller will be able to identify which was selected and nullify the player pressing more than 1 card

Identifying the card allows it to control the text displaying what was selected. It will also be able to go through a for loop and randomize ALL the cards so that the button can function properly. This means this object will make use of static variables so each object can check if the player can select anymore cards. It will have a static function that saves which card was selected and if anymore can be selected.

The static function sets a card object to a static variable to be referenced as for the selected text. Each card during their OnMouseDown() they set the variable to themselves and on reshuffle it becomes null. The cards will check if the variable is null to determine if they can be selected, else they are unable to be pressed.

CardController uses a list of all the cards and calls their randomization function. Since each of the items override their respective functions they keep their parameters and identity but still randomize and reset position.

Problem:
Having the position reset to zero is still affected by the animation coroutine. Solution is to add stopallcoroutines() during randomization as it will interrupt and be called only when the reshuffle button is pressed.
This solution would be awkward if implemented into a game as it would stop all coroutines. So a variable was set to contain the coroutine to reference it specifically.
As well a if null statement was added to prevent this change from causing the randomize function to fail to display