Weekly breakdown - Week 12

Weekly objective: Students will understand testing procedures to identify issues in their code, and external assets, using collection data structures to record initial parameters of objects, and standardization practices for their code and external assets.

Goals:
Students will be able to:

- Unit test code and stress test UI elements for bug hunting and code fixing
- Record initial parameters settings on objects to allow for runtime parameter changes
- Structure complex object and component organizations
- Create reusable components and animations for quick asset variety
- Use the list data structure to store and access initial color settings on objects
- Enable relative changes to object parameters with UI, not overwriting previous values
- Standardize assets for accessibility ease with code

Lecture Topics:

- Stress testing UI to find failing points for code fixes
- Ensuring color information is stored even if user pulls all color down to zero
- Ensuring changing speed factor is relative to existing speed settings
- Using temporary prefabs in scene for testing, then hiding them during runtime
- Ensuring scale consistency among prefabs by viewing them together in scene
- Naming and structure organization of objects and components in prefabs
- Rebuilding prefabs to ensure organization structure amongst prefabs
- Copying and pasting components to move them across objects
- Testing UI features across all prefabs to ensure consistency
- Creating animations that are reusable across different prefabs
- Copying and pasting animation keys to quickly modify or create new animations
- Proper methods for updating and saving prefabs as files in the project view
- Repurposing prefabs and animations for creating a variety of new ones
- Understanding the prefabs, animators, animations, and materials are external files
- To continue developing our features, we need a way for a prefab to remember its color and animation information, this means creating a new class that gets its data structure set when the object is spawned
- Each prefab will need to have a component script that records its initial data settings
- Float variable is needed to record the initial animation speed
- Introduction to creating a list variable as we need to create a list of unknown size of color variables to record color information for each child object in the prefab which is unknown
- A list is a collection of elements similar to an array but easier to use with more overhead
- View spawned objects during runtime to see how each prefabData script is populated
- Show how the prefabData objects connect to the ClickPositionManager object

- ClickPositionManager sets the data structures of the prefabData object connected to the spawned object when it is first spawned
- List class has an Add method to add elements to initiated list objects
- Use the Add method to pass in the current color when assigning it to the spawned object
- Set the initialAnimSpeed float of prefabData when you set the speed of spawned object
- Use UI to change color of painted objects to show that prefabData still holds initial color
- Use the initialAnimSpeed float to relatively adjust object speed as UI changes speed
- Discussion that lists, arrays, collections all start counting their elements at 0, not 1
- Implementing a count interator with our forEach loop to identify elements in our list
- Updating the change color methods to act as relative changes based on prefabData color
- Prefabs must have animator and prefabData components on root, and all children objects must have a material component for script compatibility