Weekly breakdown - Week 10

Weekly objective: Students will understand how to make effective use of the animator and animation windows to create animations, understand how to break down complex parts of code or animation into simpler parts for creation and modification ease, and understand how to use forEach loops to process lists of unknown size.

Goals:
Students will be able to:

- Procedures for testing features and UI elements
- Refactor code as features get larger and classes need to be modularized
- Understand use cases for isometric and perspective camera modes
- Handle animation and code complexities by isolating and focusing on single elements
- Understand use cases for dopesheet and curves view in animation window
- Create smooth loopable animation by editing animations in dopesheet and curves view
- Understand connection between time and value changes between keys creates speed
- Understand use cases for global and local orientation for parented animation
- Understand use cases of for and forEach loops to update elements in unknown list sizes
- Make use of underutilized variables instead of creating new ones
- Create methods for changing parameters of objects based on user interactions with UI
- Create random variables for randomization effects during object spawning

Lecture Topics:

- Animation Topics
- Testing features and UI elements connected to features
- Updating UI elements with relevant information for user
- Refactoring scripts when they get too large, one feature area per class
- Restricting spawn areas to stop creating objects over UI elements
- Isometric vs perspective modes in the viewport
- Isometric view is useful to remove perspective distortion for editing purposes
- Animation techniques for prefabs with multiple objects
- Focus animation ideas on one element at a time to build up more complex animations
- Use the curves view in animation window to remove unnecessary keys, and to modify tangent handles at start and end of animation to create clean loopable animation
- Making use of the infinity curves to understand the loopable shape of the animation
- Focus on the min and max values of an animation and remove middle-value keys
- If a start or end key is a middle value, you will need to keep the key because it defines a start or end value, but you will need to modify its tangent handle for a smooth transition
- Playback in isometric view to ensure smooth and correct animations
- Understand the connection between an animation key in dopesheet vs curves view
- Use dope sheet view to reposition keys and understand time values between keys
- Understand connection between time and value changes between keys creates speed

- Value changes of children objects are relative, not absolute to parent values
- New animations will require building new transitions in animator
- Two ways to create property keys in animation window: app property button, and record button, record button is quicker but you have to be aware to move time position before adding key from using gizmos in viewport
- Estimating time placement of keys based on desired value changes
- Getting used to jumping between dopesheet and curves as creating/modifying animation
- Idea of using a primary actor as the user focus and secondary actors to support primary
- Selecting keys with click or drag select, reusing keys with copy and paste shortcut keys
- If too much is moving in your animation and you cannot diagnosis the issues, then delete some animations and focus on only 1-2 changes at a time, then add back the others
- Always think about the user's camera perspective as you build the animations
- Animator states reference animation files which can be changed easily
- Testing animation state changes in play mode with shortcut keys and dragging in a prefab that we delete later on as its not part of the application experience
- Combining value changes for new effects like position and scale changes
- Uniform vs non-uniform scaling
- Global vs local orientation
- Position changes in local orientation can be very useful when an object is rotated
- Rationale for parenting the daylight system with rotation offset is good connection here
- Coding Topics
- Checking for keyboard input in Update loop to call ChangeAnimationState method
- Calling the ChangeAnimationState method from UI dropdown element
- Updating material change methods to have a loop within a loop to check if prefab materials need to be changed and also if prefab's children material needs to be changed
- Using forEach loops to feed in lists of objects with unknown sizes
- Reusing variables in methods when not in use instead of creating new variables
- Updating one color channel of a color data structures leaves the other channels as is
- Color channels range from 0-1f so multiply by 100 and truncate decimal to send to UI
- When generating a new random color you have to get the emission color as well since its based off the characteristics of the main color and they mix together in lighting system
- Pausing in play mode to view dynamically created objects in inspector
- Recreating emission color in ChangeEmissionStrength from the current object color
- ChangeOpacityStrength only changes the alpha color channel in color data structure
- ChangeAnimationSpeed method goes through all active objects, checks if they have an animator component, updates their animation speed, updates the class' float that remembers the animation speed and updates the UI element for animation speed
- UI toggles for randomizing anim type and speed set bools in click detector class, those bools are used in if statements when generating new objects to see if we should randomize those attributes using Random.Range method
- When using RandomRange, between x and y, call method from x to (y + 0.9999) to give full chance of calling each value yet not having chance to get the number y + 1