Weekly breakdown - Week 9

Weekly objective: Students will understand how to create more advanced prefabs with multiple objects, and with multiple animations. Students will understand setting up an animator to accept multiple animations, transitions between animations, and how to switch between different animation states. Students will learn how to connect UI elements to animators to allow users to change animation states.

Goals:
Students will be able to:

- Create advanced prefabs with multiple objects and multiple animations
- Build animators with multiple animation states and proper transition points between them
- Connect UI elements to animators to allow users to change animation states
- Use keyboard input to create shortcut keys to test out UI features
- Understand difference of inherited and separate animation parameters for multiple objects
- Check is reference variables are null before attempting to access them
- Understand difference between absolute and relative value changes in animation

Lecture Topics:

- Prefab with multiple objects
- Prefab with multiple animations
- Parameters connected to UI elements or dynamically changed by other code must be animated with code and not the animation system
- Swap out prefabs on script for quick testing of content
- Animation on one prefab will look different when played back on multiple prefabs, this is emergent animation or behavior that was not intended with the initial animation or code
- Motion design is about a balance of movement and patterns that are pleasing and informational but not distracting and noisy
- Hard coded parameters changes vs dynamic parameter changes, ie absolute vs relative value changes
- Workflows between animator and animation windows
- Each animation is a state in the animator state machine
- Properly designed prefabs will allow for using both animation by code and by animator to get the best of both worlds, and allows dynamic changes of parameters via code
- Prefabs can have multiple gameobjects each with their own parameter changes
- Prefabs with multiple gameobjects will inherit parameters changes from parent
- Use if statements to check if references are pointing to something before trying to access or modify that data structure or parameter
- Checking all child of a prefab, check if it has a renderer, and update its material if needed
- Connecting new animations to the correct animator

- Connecting multiple animations together in an animator and setting the transitions
- Setting parameter conditions per transition
- UI dropdowns are enumerations, enums are a list of two value elements that map one value to another, usually, a value thats appropriate for the computer mapped to a value appropriate for a user
- Transitions can be time or condition based
- Testing, viewing, and debugging animations, animator states and transitions
- Setting parameters on animator components with code
- Input.GetKey, Up, Down for keyboard inputs
- KeyCode enumeration to reference any keyboard key with a human-legible label
- Updating UI elements when we use keyboard shortcuts to keep it up to date with current states of objects
- Ensuring both current and future generated objects have the same animation state
- Creating UI elements to allow user to change animation states
- Dynamic vs static parameter changes with On Value Changed on an UI element
- Sending information back to UI to ensure its updated if system changes happen outside of UI
- Using UnityEngine.UI to create variables of type UI elements like dropDrown