Use Case

The program starts by loading the ControlManager. This manager has a collection of States which is filled on start. It loads the first state on the list and changes to a different state based on return values.

Each state has a Boolean stating which indicates whether it replaces the previous state or is temporarily put in place over top of a previous one. Each state also has an integer that is used as an identifier for when it should be started up and a collection of Controller. The State initializes all its Controllers and calls their build-in functions to monitor input. If a Controller returns positive for input, it may report a return value for the State to pass to its Manager.

Each button has a list of Key inputs and a RECT. It has a function that checks if a Key has been pressed or a RECT button has been clicked. This monitor function can optionally report a return value to containing State which it will pass to the Manager. If it detects a key or button has been pressed, it calls run, which executes whichever Event is supposed to happen on pressing that particular button.

Psuedocode

# Controller

## CheckController()

* Checks if the mouse has clicked on the button location if and only if the BUTTON is not NULL
* Checks if the player has pressed the key that is being listened for if and only if the key is not NULL
* Returns ReturnValue on confirmation of either of the above. Returns 0 if neither of the above has occurred. Negative numbers are used for error handling.
* Buttons are managed using the external getCalledButton() function which takes the button's ID
* Keys are managed internally

## RunCommand()

* Runs the command (Read: Preloaded function call, usually to an EVENT wrapper; See ControlEvents.h) stored by this controller.
* Should only be called internally, on a successful CheckController() call.
* As this is the main point of divergence between Control instances, it can be implemented using inheritance or a function pointer (at the preference of the c oder).

## LoadGUI()

* Loads the stored BUTTON into the Graphics-side Render Manager
* The above should be done using a function provided by the graphics team

## UnloadGUI()

* Unloads the stored BUTTON from the Graphics-side Render Manager
* The above should be done using a function provided by the graphics team

# ControlState

## LoadGUIs()

* Calls the LoadGUI() function of each Controller stored in this State.

## UnloadGUIs()

* Calls the UnloadGUI() function of each Controller stored in this State.

## CheckControllers()

* Calls the CheckController() function of each Controller stored in this state
* Returns any return values passed by the CheckController() Function(s)

## AddController()

* Inserts the passed in controller to the ControllerCollection

# ControlManager

## AddNewState()

* Inserts the passed in State to the StateCollection

## CheckState()

* Calls the CheckControllers() function of the currently active state
* If the CheckControllers() function returns a positive, non-0 value the Control Manager calls the getIdentifier() function of each stored State in order to compare each Identifier with the returned value
* If the above finds a match it calls SetNewActiveState(), passing in a pointer to the new state
* This function is tended to be called in a loop. As such it returns 0 until such a time as the program is terminated.

## SetNewActiveState()

* Calls GetOverride() on the passed in State and the Current ActiveState
* If Override is set to true on the new State OR set to false on the old one, it calls UnloadGUIs()
* Regardless of the above, it then loads a pointer to the new State into the ActiveState field and calls LoadGUIs() on that State

## Various

* Event Interfaces will be defined by me and there is expected to be a Controller for each one.
* RECTs will be defined in a separate header file that has not yet been finished. Until this is finished I will be providing a dummy header with placeholder RECTs defined
* Names are important. I expect all names defined in the ControlOOD.vsd to be follow \*EXACTLY\*
* Functions for adding and removing RECTs from the RenderManager have not yet been defined. I will be providing a dummy function [AddGraphic(RECT) and RemoveGraphic(RECT)] for this purpose until the proper function has been fully defined
* Controllers check if a button has been pressed by checking the a global PressedCollection, which has a list IDs of buttons that have been pressed