Complex Game Systems

I have chosen to do the Complex User Interface segment of this assignment. To prove my understanding I will develop a list of classes that can be used together to make basic ingame menus, display simple information and control elements within the game itself.

Elements:

- BitFont
    - This class has no drawing features
    - Used to generate BitString
- BitString
    - Draws a given string
- Button
    - Simple button
    - Activated on mouse release inside its bounds
- Console
    - Acts like a Window
    - Contains
        - ElementList
        - Textbox
    - Functions can be added to a console
    - When certain strings are entered, Functions will be called.
- Container
    - No drawing features
    - Contains other UI Elements
    - Used as a base element
- ElementList
    - Contains child elements
    - Used to order and manage
- Graph
    - Displays a Graph from the given data
    - Data can be set to any numeric value
    - A max can be set
- Progress Bar
    - Max can be set
    - Displays percentage from current value to max value
- Slider
    - Max can be set
    - Returns given value
- Textbox
    - Uses keyboard
    - Returns a given string

- UIElement
    - Base of all elements
    - Controls Position, Scale, Rotation, Size and Colour
    - Has a depth
    - Can have children and a parent
    - Can have a texture
    - Can have a shader
    - Has default Animations. Translate, Scale, Rotation and Colour
    - Can be activated
    - Has bounds
    - Has a pivot location
- UITexture
    - Simple rectangle to draw a texture on
- Window
    - Can be moved
    - Can be resized(bottom corners)
    - Has a title

Techniques

- UI
    - Elements will inherit from the same super class, This will insure all elements can be used together in a dynamic fashion.
- Parental control
    - All child elements inherit their parents final model matrix. This enables elements to be moved easily within its respected containers. Parental control is also present when adding a child to an element, if the child does not have a texture or shader, it will use the same as the parent. As always this can still be changed at a later date with no side effects.
- Inheritance
    - If the user chooses to, they can setup the Elements in a parent/child fashion. When doing so the user can call “ElementGet(‘name’)” to be given a pointer to a given child, This function is also templated to receive a specific element instead of the base type. The main advantage of this is self cleaning on removal, and the Parental control feature labeled above.
- Local Origin
    - All elements will have a local origin, or pivot point from where they are drawn. This can be used to creating snapping boundaries which can be seen in the main menu state.
- Animation
    - Elements have an animation feature, this can be seen on a buttons hover or in the splash state with the scaling text. When a user sets one up, Anything can be animated as it is setup with function pointers. All Elements will have a TranslateTo, RotateTo, ScaleTo and BlendTo animation built in.
- Offset
    - When a element has a parent, it will be offset by the given amount, this can be used in conjunction with 3D menus to create a sense of depth seen in games like Dead Space 3 and Borderlands 2.
- Override
    - Elements can be overridden by passing a new Element to the same key, An example of this being used is when one item is being replaced by another in an inventory system.
- HasKeyboard
    - Some elements need the use of the keyboard, and it would be annoying to have the game still control you while typing. The use of this feature can remove that annoyance. It simply returns a boolean if any of its children are currently using the keyboard.