**Description**

This library is designed to be an extension to Henning Karlsen's UTFT and UTouch libraries. Essentially it is quite similar to his newer UTFT\_Buttons library, because it itself is focused on a few different button types. These types also have two ways of getting the touch data and that is either from the Raw touch points which needs **myTouch.TP\_X**  / **myTouch.TP\_Y**, or the calculated touch points from **getX()** and **getY()** functions. The reason I implemented these two different ways is because under certain circumstances, one way is faster than the other.

These types are:

* Touch button (Rectangle/ Circle /Triangle)
* Latching button (Rectangle/ Circle/ Triangle)
* Touch Delay button (Rectangle/ Circle/ Triangle)
* Text button (Rectangle, 90% completed)
* Radio buttons (Rectangle and Circle)

Using the raw *myTouch.TP\_X/Y* is technically faster because it does not require the user to calibrate the screen or worry about touch precision, but it does require the user to manually map the boundaries of the screen and the use of the map function. However the *getX/Y* functions do not require such hassling work, but they do require the screen to be calibrated and certain variables in the .cpp file to be changed. The getX/Y functions also need to be set to a precision setting, as either Low, Medium, Hi or Extreme. Low is the fastest of the four settings, because it requires minimal calculations to be done per touch, but it is not as precise; whereas Extreme has the highest touch precision but it takes a lot more time to calculate per touch thereby slowing down the processor from other tasks.

**TouchButton / TouchCircle / TouchTriangle:**

- The Touch button is your normal box shaped button. It needs two sets of X and Y coordinates to work, top left and bottom right. It can also be any size or orientation and accept either the touch data from myTouch.TP\_X/Y or getX/Y.

-TouchCircle is nearly identical, but it only requires one X/Y coordinate and a radius. The X/Y coordinates you enter will become the center of the circle, and the radius will determine how big the circle is.

- TouchTriangle is just like TouchCircle, but instead of a radius, you have a base. The triangle you make is an equilateral, no other triangle shapes are implemented yet. The TouchTriangle does require two other variables to be entered and those are direction and degree. Direction is either "up, down, left or right" all lower case, and the degree is 0 - 359.

**Latchbutton / LatchCircle / latchTriange:**

Just like the touch button and touch circle functions, they too can be any size and orientation, but this type button will need its own identification number. The current maximum allowed is 10 but this can be changed in the TFT\_Extension.h file, *#define Num\_Of\_Buttons 10.*

**TouchDelayButton / TouchDelayCircle / TouchDelayTriangle:**

What this function does is once you press the button it will look at the X/Y pairs and start a timer. If the button is held down for a set time, the button will either return true if that time has passed, or false if it has not.

**TextButton:** (currently only applied to boxes)

This allows a button to have text in the center of it at two different font sizes, Big and Small. Right now this will center the text even if the text is large than the button, so this function is still a work in progress. I also plan on making the button be able to rotate with the text and still work as a normal button. This will also be implemented to the other buttons as well.

**RadioButton / RadioCircleButton:**

If you are familiar with the old style car radios, then you will know how these particular buttons work. Only one button can be on at a time and if another is pressed, it goes on and turns off the previous button. To get the radio buttons to work, you first need to tell the compiler how many groups you want and the number of buttons per group. This can be done with the functions *TotalRadioButtons* and *TotalRadioCircleButtons.* An array is needed to set the different number of buttons per group.

You can have a total of 5 groups with 10 buttons in each group and both button types work independently from one another, so both can have 5 groups and 10 buttons.

There are quite few functions dedicated to the radio buttons, in which they can change the appearance of the buttons either individually, as a group or all as one. *See the radio button example sketch*.

In the event you want to use menus with the radio buttons, the functions *ResetRadioButton* and *ResetRadioCircleButton* can clean/reset the buttons so the screen can be changed. NOTE: This does not reset the states of the buttons. Currently it only does one group per function. (5 groups, 5 functions.)

\*\*Both types of radio buttons now are able to have text in the center of them. **Please note** the text will not be cut off, nor will the buttons size be altered if the text is larger than the button.

**TouchButton \_Draw/ TouchCircle\_Draw / TouchTriangle\_Draw:**

All three are nearly identical to their counterparts TouchButton, TouchCircle and TouchTriangle, but these require an ID number to work.

**These type of buttons also need to be colored, told whether they should be rounded or not, and filled or not. This is done with the corresponding functions, *SetTouchButtonColors*, *SetTouchCircleColors and SetTouchTriangleColors.***

**Note: Circles do not have the option to be rounded, because they are already round XD**

**LatchButton \_Draw/ LatchCircle\_Draw / LatchTriangle\_Draw:**

-See "Latchbutton / LatchCircle / latchTriangle:" for description

**Again, these type of buttons also need to be colored, told whether they should be rounded or not, and filled or not. This is done with the corresponding functions, *SetLatchButtonColors*, *SetTouchCircleColors and SetTouchTriangleColors.***