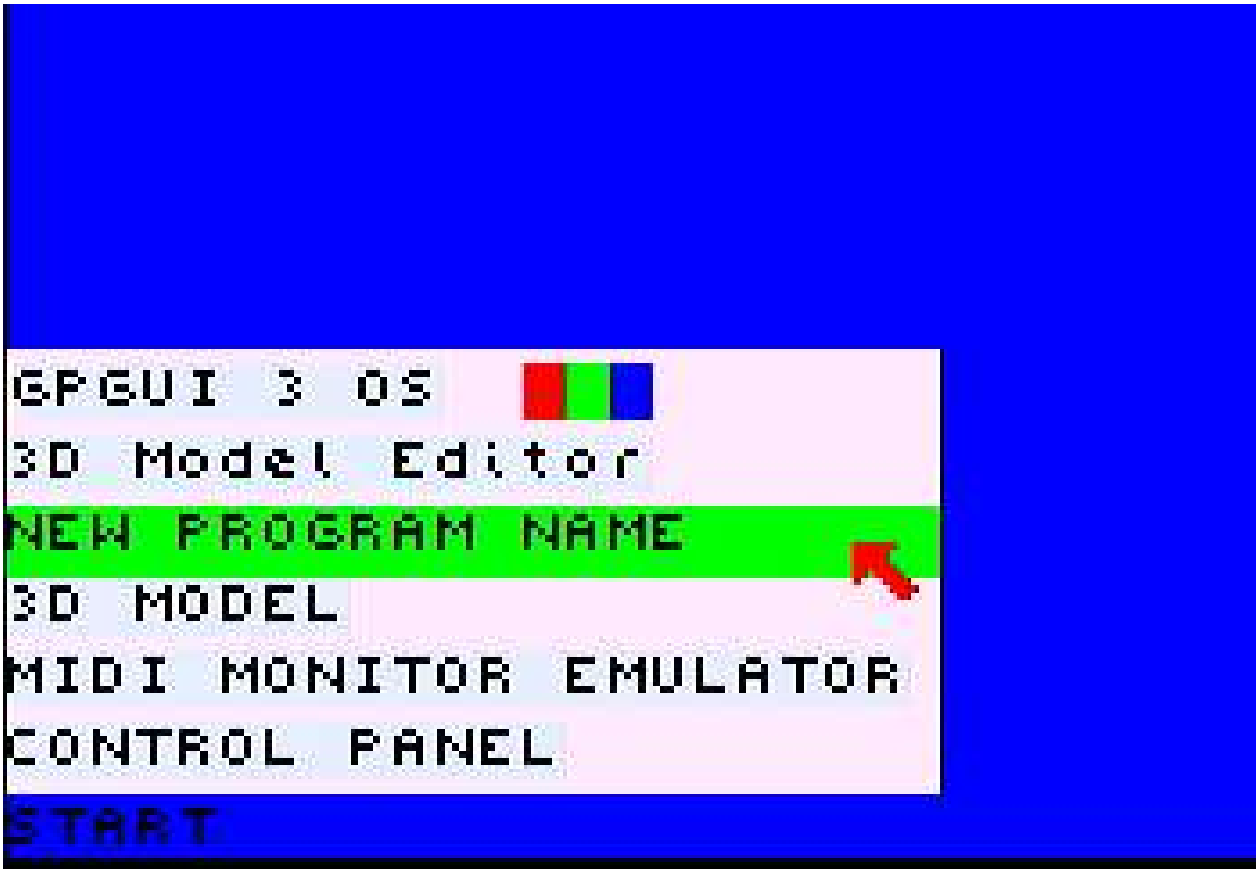
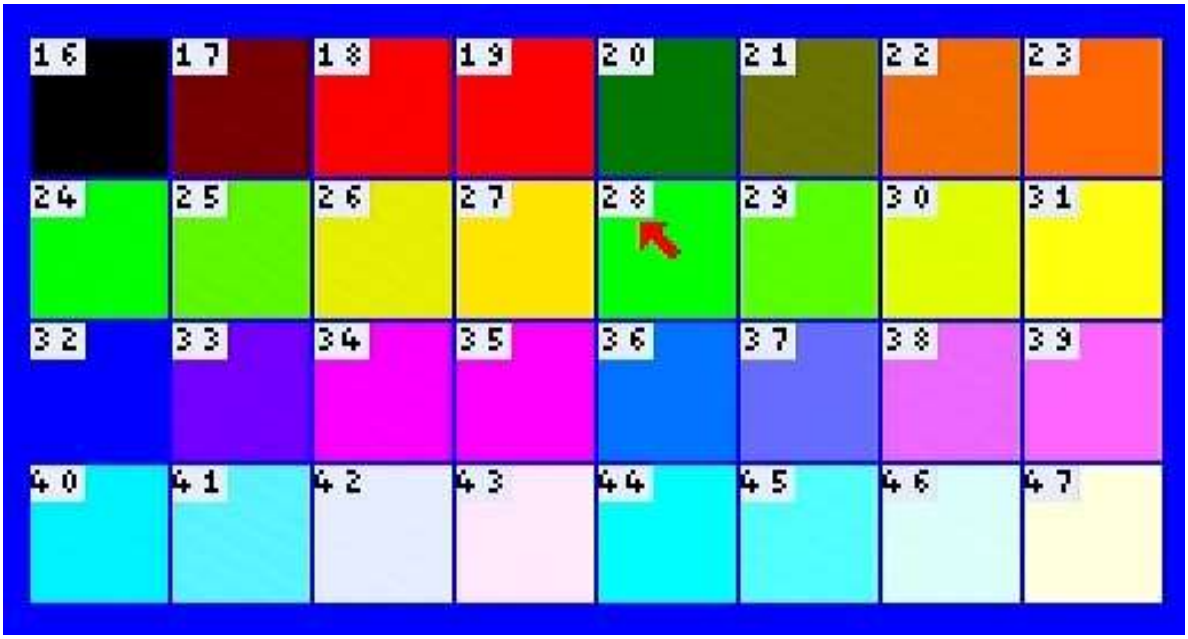
GPGUI 3 - HOW TO CREATE MOUSE BUTTONS



Summary:

How to create basic buttons on the screen that can be clicked on using the mouse cursor. The buttons can be flat shaded ON/OFF and highlight's when the cursor is over it, or text that highlight's when the cursor is over it, or an icon referenced to a color texture map byte array[16][16].

Contents



- **Color Values.**

16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47

- Color texture map, **NOTE**: Icons that are smaller than 16 are drawn in the top left corner.

```
 byte exit_Sprite[16][16] =
{
 {1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
 {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
};
```

- **This is the button data structure**.

```
struct button
{
    bool state;
    int xStart;
    int yStart;
    int xSize;
    int ySize;
    int onColor;
    int offColor;
    int id;
    char* text;
    int icon[16][16];
    bool fill_Type;
    int texSize;
    bool text_Enable;
    int icon_Index_X;
    int icon_Index_Y;
};
```

STEP 1 - Create a buffer to hold the buttons.

- Create - active_Button_Id memory address .
- Create – button_Buff vector< button > .

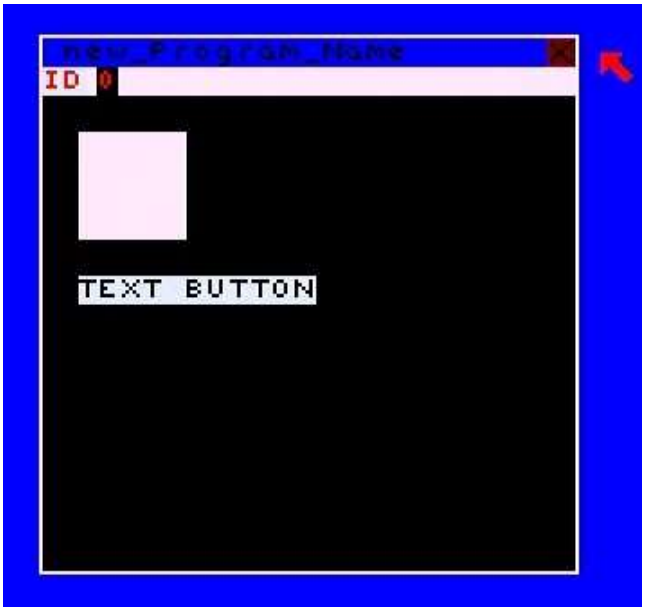
```
    vector<button> buttons_Buff;
    static int   active_Button_Id = 0;
```

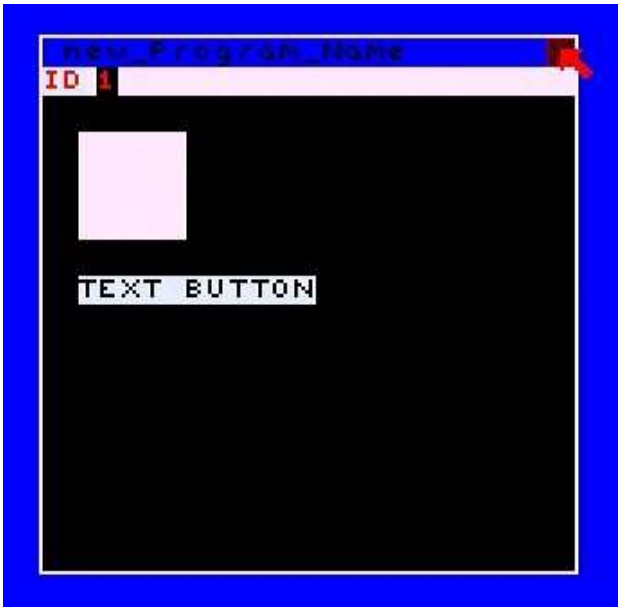
STEP 2 - Create a button object and add it to the button buffer.

STEP 2.1 - Create a button object and add it to the button buffer (ICON).

- Set button start (Top left).
- Set button area size.
- Set the button ID.
- Set fill Type to 1.
- set the texture size (this is a square size i.e. 16x16 = 16). 16 is the max size.
- Set text Enable too false.
- set the texture index (**Only if the texture is smaller than the button area and offset is required**).
- Use the write_Icon_To_Button(**button***, byte*[16][16]) function.
- Push the button object into the button buffer.

```
button exit;
exit.xStart = (w1.xStart + w1.xSize) - 9;
exit.yStart = w1.yStart + 1;
exit.xSize = 8;
exit.ySize = 8;
exit.id = 1;
exit.fill_Type = 1;
exit.texSize = 8;
exit.text_Enable = false;
exit.icon_Index_X = 0;
exit.icon_Index_Y = 0;
write_Icon_To_Button(exit, exit_Sprite);
buttons_Buff.push_back(exit);
```

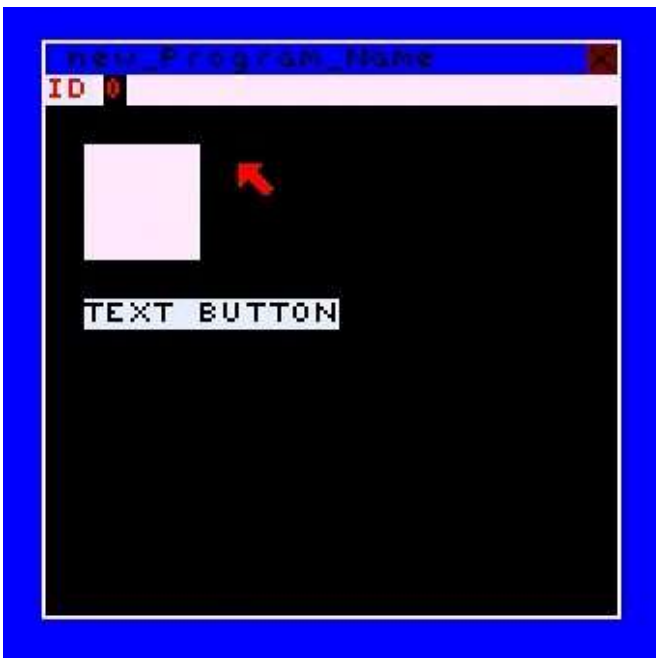


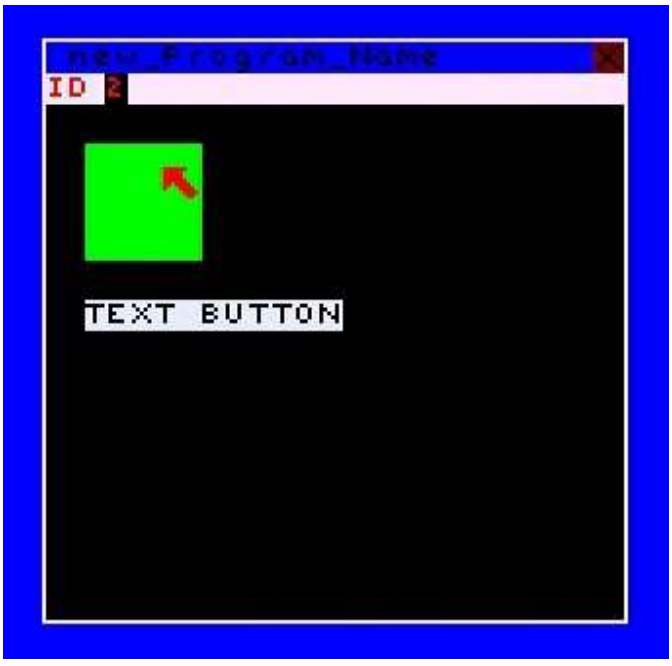


STEP 2.2 - Create a button object and add it to the button buffer (FLAT SHADED).

- **Set the active and inactive color.**
- **Set button start (Top left).**
- **Set the button ID.**
- **Set button area size.**
- **Set text Enable too false.**
- **Set fill Type to 0.**
- **Push the button object into the button buffer.**

```
button b1;
b1.onColor = green;
b1.offColor = grey;
b1.xStart = w1.index_X + 10;
b1.yStart = w1.index_Y + 10;
b1.xSize = 30;
b1.ySize = 30;
b1.id = 2;
b1.text_Enable = false;
b1.fill_Type = 0;
buttons_Buff.push_back(b1);
```

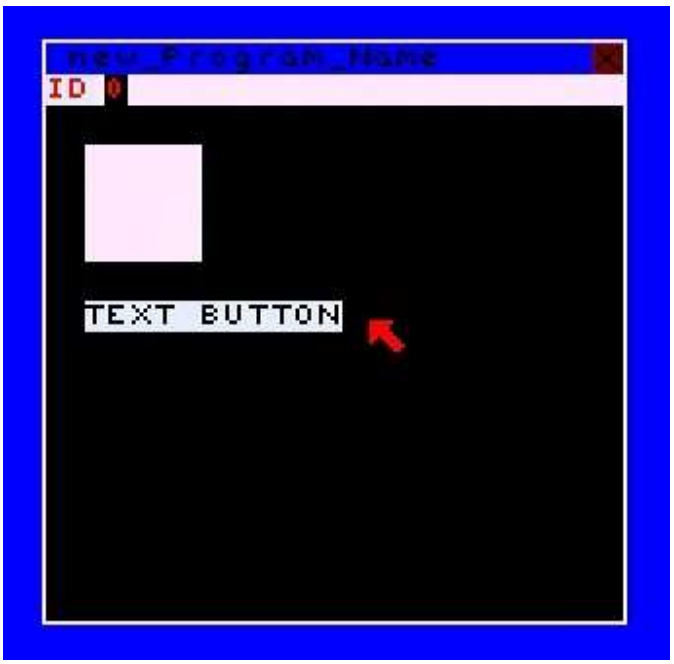


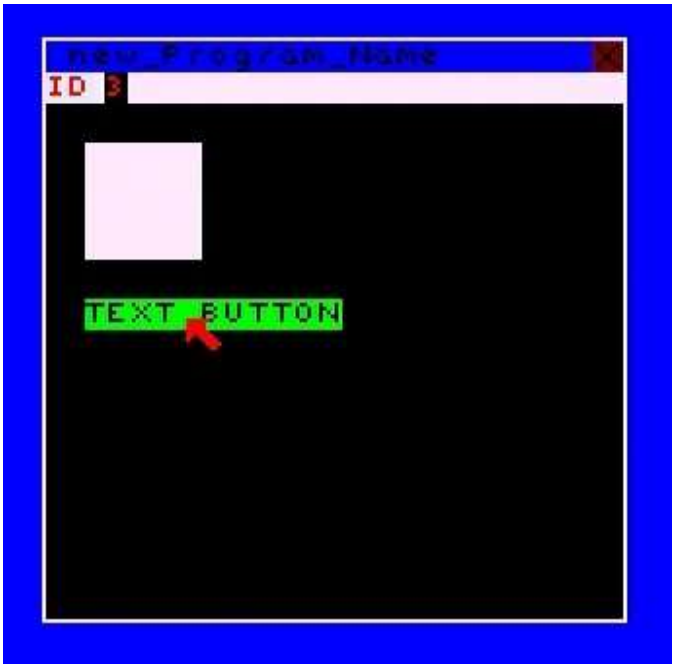


STEP 2.3 - Create a button object and add it to the button buffer (TEXT BOX).

- **Set button start (Top left).**
- **Set button area size(The width is the number of chars in the TEXT * 6).**
- **Set the button ID.**
- **Set the text string.**
- **Set text Enable too true.**
- **Set fill Type to 0.**
- **Push the button object into the button buffer.**

```
button b2;
b2.xStart = w1.index_X + 10;
b2.yStart = w1.index_Y + 50;
b2.xSize = 66;
b2.ySize = 8;
b2.id = 3;
b2.text = "TEXT BUTTON";
b2.text_Enable = true;
b2.fill_Type = 0;
buttons_Buff.push_back(b2);
```





STEP 3- How to link a button's id to flags and functions.

- **Now that a buffer of buttons has been created, the** `mouse_Icon_List_Handle()` **function must be called.**
- **This function checks the current mouse position against all the buttons in the buffer.**
- **As it checks the buttons it draws either the inactive state or the active state.**
- **If there is a button match, that button's id tag will be returned to the** active_Button_Id **address.**

```
mouse_Icon_List_Handle(buttons_Buff, &active_Button_Id, mouseX, mouseY);

print("ID", w1.index_X, w1.index_Y - 8, red, bar_Color);
printNumberInt(active_Button_Id, w1.index_X+15, w1.index_Y - 8, red, bar_Color);
```

- **If a mouse left, click is registered.**
- **A switch case statement can be used on (active_Button_Id).**
- **Each case is linked to a button, either a flag can be set, or a function can be run.**
- **The mouse_Left_Make flag must be cleared.**

```
//MOUSE BUTTON  CLICK FLAGS
if (mouse_Left_Make)
{
    //PROGRAM BUTTONS
    switch (active_Button_Id)
    {
    case 1://ICON
        close_All_Programs();
        button_Click_Close_Window_Tone();
        beep(exit_Beep_Pitch, 200);
        mouse_Left_Make = false;
        break;

    case 2://FLAT SHADED
        button_Click_Enter_Tone();
        mouse_Left_Make = false;
        break;

    case 3://TEXT BOX
        button_Click_Close_Window_Tone();
        beep(exit_Beep_Pitch, 200);
        mouse_Left_Make = false;
        break;
    }

}
```

Interactive Icons.

Vertical slide bar.

- **This is the data structure of a slide bar.**

```
struct slide_Bar
{
    int xStart;
    int yStart;
    int output_Min;
    int output_Max;
    int back_Color;
    int slide_Color;
    int bar_Pos;
    bool scroll_Enable;
    char on_Color;
    char off_Color;
};
```

STEP 1 – Create slide bar variables.

- **Initialize a static int bar_Pos_b1 memory address with the initial value of the bar.**
- **Initialize a static float output memory address to store the output from the slide bar.**
- **Initialize a static bool en memory address for enabling or disabling the slide bar.**
- **Initialize a slide bar object.**

```
static int bar_Pos_b1 = 70;
static float output;
static bool en = false;

slide_Bar bar;
bar.xStart = w1.index_X + 10;
bar.yStart = w1.index_Y + 70;
bar.back_Color = grey;
bar.slide_Color = red;
bar.off_Color = blue;
bar.on_Color = green;
bar.bar_Pos = bar_Pos_b1;
bar.output_Max = 127;
bar.scroll_Enable = en;
```

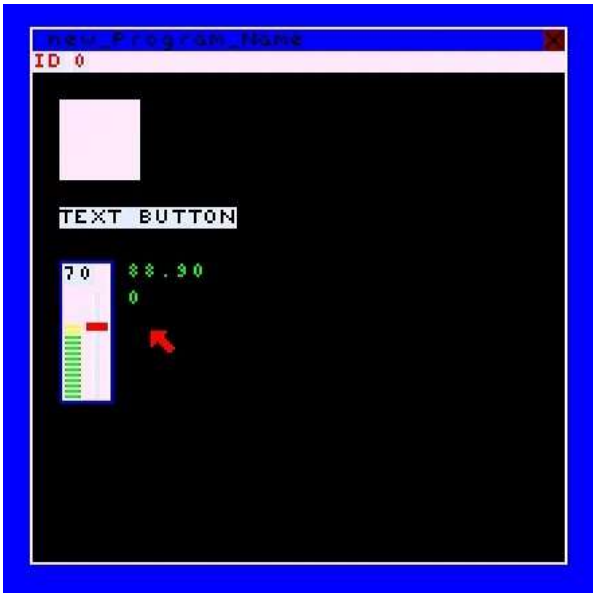
STEP 2 – Call the vertical_Slide_Bar() function.

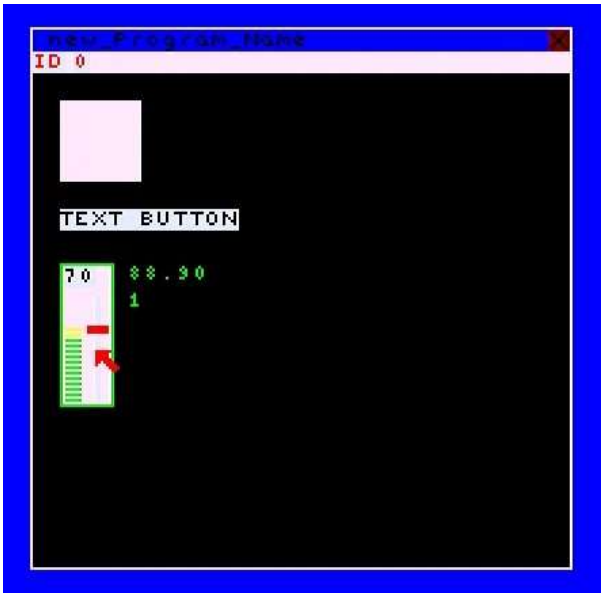
- **This function reads and handles all the mouse data and draws the slide bar to the frame buffer.**
- **The &en and &output variables can then be used to set program values.**

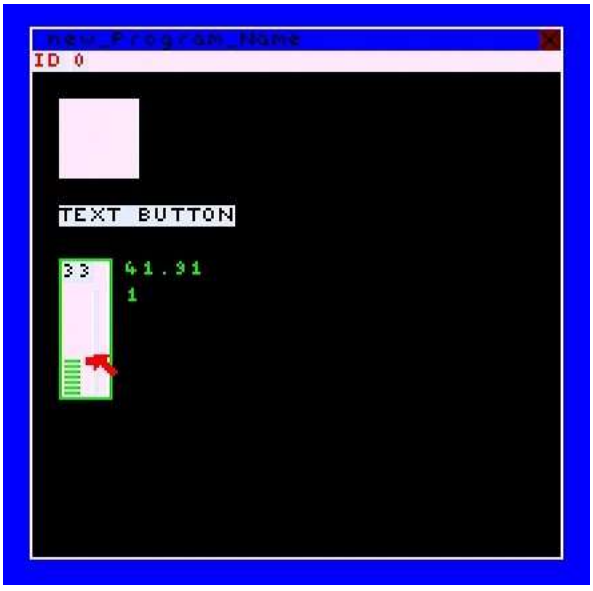
```
vertical_Slide_Bar(bar, bar.output_Max,&output,&en,&bar_Pos_b1);

printNumberDouble(output, w1.index_X + 35, w1.index_Y + 70, green, black);
printNumberInt((int)en, w1.index_X + 35, w1.index_Y + 80, green, black);
```

- **Click on the slider with the mouse scroll click to enable or disable.**
- **The boarder of the box will be set to the on color when enabled, off color when disabled.**
- **Scrolling the scroll wheel up/down changes output.**
- **Left clicking on the slider will set the output to the clicked position.**
- **Left click held on the slider will drag the slider up/down(the mouse will be clamped to the slider).**







Latching buttons.