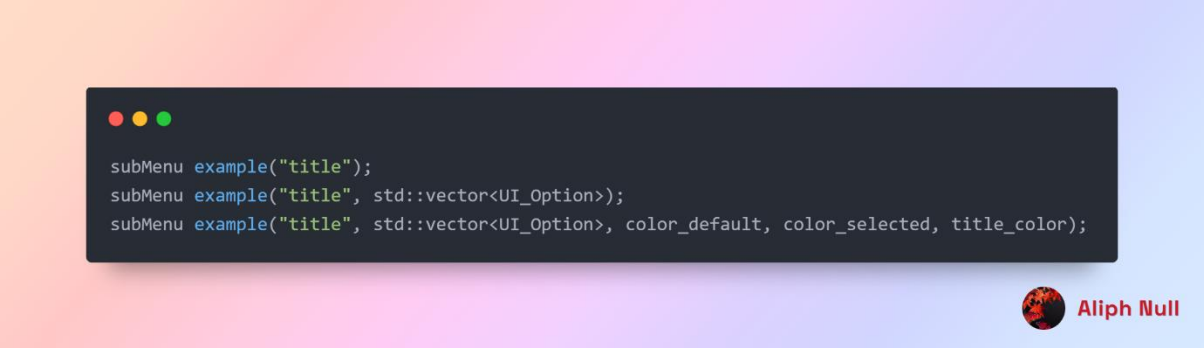
How to use **menuLight.h**

Include the library and create a global pointer.

In the main function just declare a **cli_menu** object with a vector of **subMenu** and assign the global pointer to it.

Then you can just start the main loop.

How to declare a subMenu



At the very least you need a title, but it's a lot easier to create the **subMenu** with all the options. Additionally, you can specify some colors to use. The "color_default" is the color an **UI_Option** will have when it's **NOT** selected, just being there. When an **UI_Option** is selected it wil have the "color_selected". Lastly the title_color dictates the color of the title. If you make it black it will use an external function to create a gradient onto the title.

This is how you declare a color

You always have the ability to change anything about the **subMenu**.

There are 3 text alignments “**Aligments**” because I cant spell 😊 :

- LEFT
- CENTER
- RIGHT

The **UI_Option_Bar** dictates how decorated the options will be. The first 2 strings are for idle options, while the last 2 strings are for the one that is selected. The color is optional, but if written will overwrite the color of the selected option defined in the **subMenu**, so the selected option and it’s decorations can be different colors. The strings may be empty.

The gradient function is any user defined function that takes a double between 0 and 1 (in proportion with the percentage of the position of the character from left to right in relation the length of the title) and returns a color.

Here is an example

```
color rainbowColor(double x){
    return HSLtoRGB(x * 720, 1.0, 0.5);
}
```

Aliph Null

It uses the provided HSLtoRGB function.

```
void f_start() { std::cout << "Start pressed\n"; }
void f_settings() { std::cout << "Settings pressed\n"; }
void f_exit() { std::cout << "Exit pressed\n"; }

    UI_Option myOption("Start Adventure");
    UI_Option myOption("Start Adventure", f_start);

    myOption.Subscribe(f_settings);
    myOption.Subscribe(f_exit);

    myOption.overwriteColor = {0, 255, 0};

    myOption.Unsubscribe(f_settings);
    myOption.Call();
```

Aliph Null

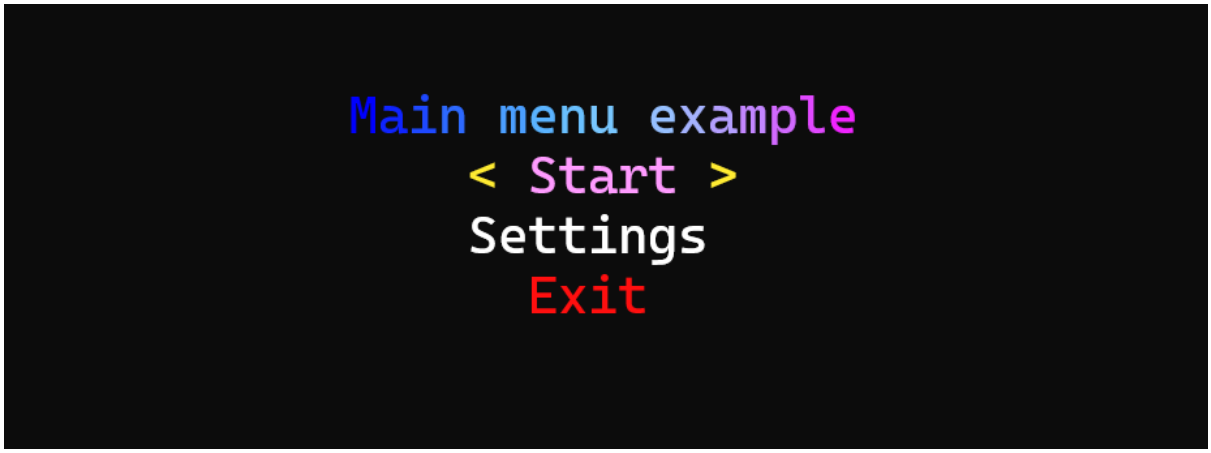
Lastly the **UI_Option** or the button. It needs a name, optionally a color to be special that will overwrite the **subMenu**'s color, and optionally a function to call when pressed (by pressed it is meant when it's called, not clicked by

the mouse, and under the hood it's implemented to use the ENTER key). It may have multiple functions which it calls. All functions MUST be void functions and take no parameters. This is an example using a shorter notation covering most of the functionality

```cpp
#include "menuLight.h"
cli_menu* global_menu = nullptr;

void f_start(){
    global_menu->selectSubMenu("Select a world to start your adventure");
}
void f_settings(){
  cout << "nothing implemented, press enter to go back";
  getch();
}
void f_exit(){
    global_menu->exit();
}

void f_back(){
    global_menu->selectSubMenu("Main menu example");
}

cli_menu menu({
        subMenu("Main menu example", {
                    UI_Option("Start", f_start),
                    UI_Option("Settings", f_settings),
                    UI_Option("Exit", {255, 15, 15} ,f_exit)
                    }, {255, 255, 255}, {255, 155, 255}),
        subMenu("Select a world to start your adventure", {
                    UI_Option("The Lord of The Rings"),
                    UI_Option("Starwars"),
                    UI_Option("Minecraft universe"),
                    UI_Option("Fantasy World"),
                    UI_Option("Fantasy World #2"),
                    UI_Option("A DnD campaign"),
                    UI_Option("Your favourite book"),
                    UI_Option("Dreamworld"),
                    UI_Option("Sky Castle"),
                    UI_Option("Back", {255, 15, 15} ,f_back)
                    }, {155, 155, 155}, {155, 155, 255}),
    });
```

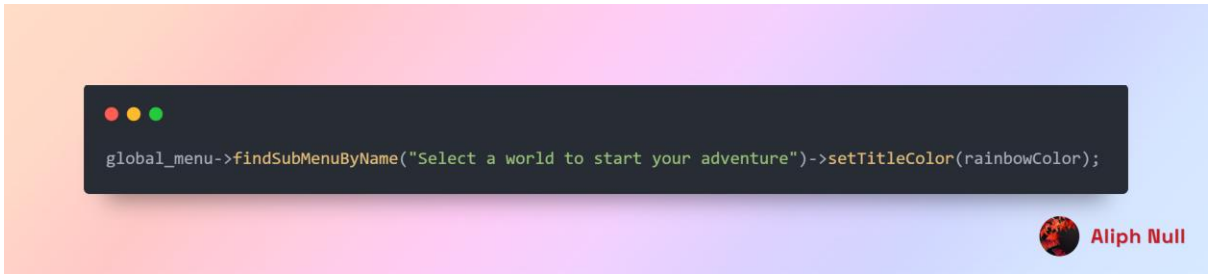
Aliph Null

And here is how that looks like:



*There is a default gradient function
*by default all text is aligned to center

And here is the second menu that uses the previous gradient function

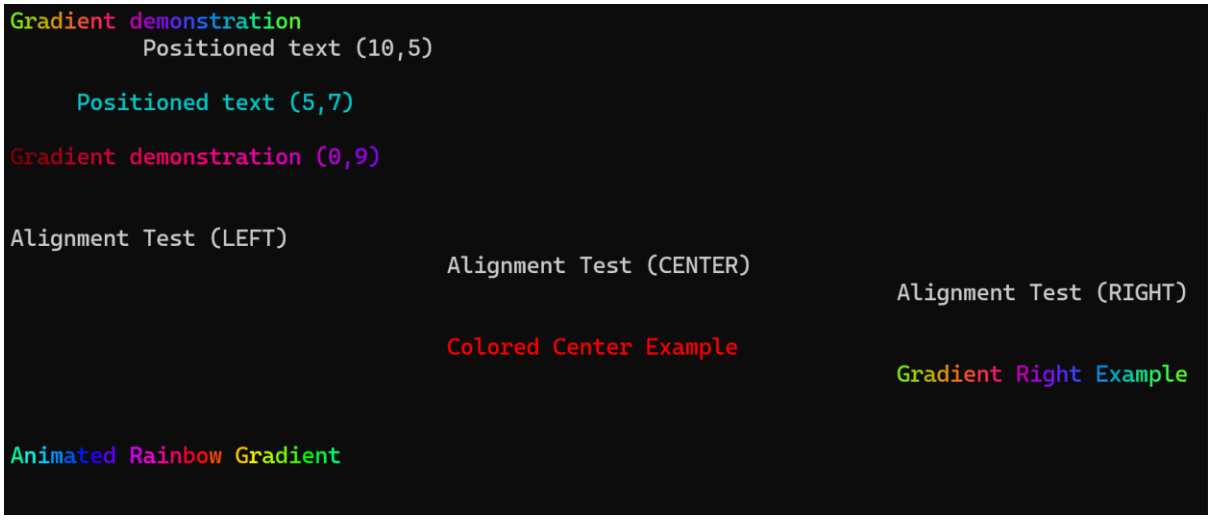


Don’t forget to make use of the better print function, which offers basic but powerful text color and alignment tools.

```
cli_menu menu;
menu.clearConsole();

using namespace beautyPrint;
using namespace AvailableAlignments;

// Basic prints
print("Hello from beautyPrint!");
std::cerr << "\n\n";

print("Hello from beautyPrint!", {255, 50, 50});
std::cerr << "\n\n";

print("Gradient demonstration", rainbowGradient);
std::cerr << "\n\n";

// Absolute position prints
print({10, 5}, "Positioned text (10,5)");
print({5, 7}, "Positioned text (5,7)", {0, 200, 200});
print({0, 9}, "Gradient demonstration (0,9)", blueToPurple);
std::cerr << "\n\n";

// Alignment examples
print({menu.getWidth(), 12}, "Alignment Test (LEFT)", LEFT);
print({menu.getWidth(), 13}, "Alignment Test (CENTER)", CENTER);
print({menu.getWidth(), 14}, "Alignment Test (RIGHT)", RIGHT);
std::cerr << "\n\n";

// Alignment + color / gradient
print({menu.getWidth(), 16}, "Colored Center Example", CENTER, {255, 0, 0});
print({menu.getWidth(), 17}, "Gradient Right Example", RIGHT, rainbowGradient);
std::cerr << "\n\n";

// Animated gradient demo
std::string anim = "Animated Rainbow Gradient";
for (int frame = 0; frame < 50; ++frame) {
    cursor(0, 20);
    print(anim, [&](double x) {
        return HSLtoRGB(fmod((x + frame * 0.05) * 360.0, 360.0), 1.0, 0.5);
    });
    std::this_thread::sleep_for(std::chrono::milliseconds(60));
}

cursor(0, menu.getHeight() - 1);
std::cerr << RESET_ALL << "\n";
```

Aliph Null

And here is the result of that (The animated rainbow gradient really is animated. You have to just trust me on this one.



Props to [Code to Image Converter: Code Screenshots Online | 10015 Tools]

That helped create those beautiful code snippets.