```cpp
#ifndef MENU_H
#define MENU_H
#include "Menu.h"
#endif
#include <stdio.h>
#include "Window.h"
#include "string"
#include "array"
#include "Mixer.h"

Menu::Menu() {
    running = false;
    firstMenuOptionPickedFlag = false;
    curr_cursor_pos = 1;
    distance_between_menu_option = 50;
    menu_items.clear();
    unused_menu_items.clear();
};

void Menu::render() {
    //Set Renderer
    SDL_Renderer* renderer = Window::getInstance()->getRenderer();

    //Clear Screen, Set Background Color
    SDL_RenderClear(renderer);
    SDL_SetRenderDrawColor(renderer, 0x11, 0x11, 0x11, 0xFF);

    //render menu items
    for (int i = 0; i < menu_items.size(); i++) {
        menu_items.at(i)->render();
    }

    //Present Renderer
    SDL_RenderPresent(renderer);
}

void Menu::tick() {
    for (int i = 0; i < menu_items.size(); i++) {
        menu_items.at(i)->tick();
    }
};

void Menu::moveCursor(std::string input) {
    if (menu_items.size() < 2) {
        printf("Currsor could not be placed\n");
        return;
    }

    //Find Cursor
    int cursor = menu_items.size();
    int last_el = menu_items.size()-2;
    for (int i = 0; i < menu_items.size(); i++) {
        if (menu_items.at(i)->getType().compare("CURSOR") == 0) {
            cursor = i;
        }
        if (menu_items.at(i)->getType().compare("CURSOR") == 1) {
            last_el = i;
        }
    }

    //Check if
    if (input.compare("UP") == 0) {
        //If Cursor is not already on the first menu option element
        if (curr_cursor_pos > 1) {
            //set cursor one more up
            menu_items.at(cursor)->setY(menu_items.at(cursor)->getY() - 
              menu_items.at(1)->getH() - distance_between_menu_option);
            curr_cursor_pos -= 1;
        }
        else if(curr_cursor_pos == 1){
            //set cursor to lowest element
            menu_items.at(cursor)->setY(menu_items.at(last_el)->getY() + 
              (menu_items.at(last_el)->getH() / 4));
            curr_cursor_pos = menu_items.size() - 4;
        }
    }
    else if (input.compare("DOWN") == 0) {
        //If Cursor is not at the last element
        if (curr_cursor_pos < (menu_items.size() - 4)) {
            //set cursor one more up
            menu_items.at(cursor)->setY(menu_items.at(cursor)->getY() + 
              menu_items.at(1)->getH() + distance_between_menu_option);
            curr_cursor_pos += 1;
        }
        else if (curr_cursor_pos >= (menu_items.size() - 4)) {
            //set cursor to first element
            menu_items.at(cursor)->setY(menu_items.at(1)->getY() + 
              (menu_items.at(1)->getH() / 4));
            curr_cursor_pos = 1;
        }
    }

    //Play Sound
    Mixer::getInstance()->play(Mixer::SWORDDRAWN2);
};

void Menu::input() {
    SDL_Event e;

    //Exit if quit pressed
    while (SDL_PollEvent(&e)) {

        if (e.type == SDL_QUIT) {
            running = false;
        }

```

```cpp
        if (e.type == SDL_KEYDOWN) {
            if (e.key.keysym.sym == SDLK_RETURN) {
                switch (curr_cursor_pos)
                {
                case 1:
                    running = false;
                    firstMenuOptionPickedFlag = true;
                    Mixer::getInstance()->playMusic(Mixer::FIGHTSONG);
                    break;
                case 2:
                    running = false;
                    firstMenuOptionPickedFlag = false;
                    break;
                default:
                    break;
                }
            }

            if (e.key.keysym.sym == SDLK_ESCAPE) {
                running = false;
                firstMenuOptionPickedFlag = true;
                Mixer::getInstance()->playMusic(Mixer::FIGHTSONG);
            }

            if (e.key.keysym.sym == SDLK_DOWN) {
                moveCursor("DOWN");
            }

            if (e.key.keysym.sym == SDLK_UP) {
                moveCursor("UP");
            }
        }
    }
};

bool Menu::show(int flag) { //"false" for [exit], "true" for [start game 
  or continue or restart]
    //Check what menu item shall be shown
    if ((flag >= 0) && (flag<=2))
        {changeFirstMenuItemTo(flag);}
    else
        {changeFirstMenuItemTo(CONTINUE);}

    //Set flags
    firstMenuOptionPickedFlag = false;
    running = true;

    //Start menu loop
    while (running) {
        input();
        tick();
        render();
    }

    return firstMenuOptionPickedFlag;
};

bool Menu::loadMedia() {
    bool success = true;

    for (int i = 0; i < menu_items.size(); i++) {
        if (!menu_items.at(i)->loadMedia()) {
            printf("Failed to load menu item Number: %d\n", i+1);
            success = false;
            break;
        }

        //PLACE MENU ITEMS
        //Set y-coordinate of menu options
        if (i == 1) {
            menu_items.at(i)->setY(menu_items.at(0)->getH() + 2 * distance_between_menu_option);
        }
        else if (i > 1) {
            menu_items.at(i)->setY(menu_items.at(i - 1)->getY() + menu_items.at(i - 1)->getH() + distance_between_menu_option);
        }

        //Set x-coordinate
        if (i <= 1) {
            //Place every menu option in the middle of the screen width
            menu_items.at(i)->setX((Window::getInstance()->getWindowSizeW() - menu_items.at(i)->getW()) / 2);
        }
        else if (i > 1) {
            //Place every menu option after first one to the same x-coordinate as the first menu option
            menu_items.at(i)->setX(menu_items.at(i - 1)->getX());
        }

        //Place Cursor - if .compare(...) == 0 it means the strings are equal
        if ((menu_items.at(i)->getType().compare("CURSOR") == 0)) {
            menu_items.at(i)->setY(menu_items.at(1)->getY() + (menu_items.at(1)->getH() / 4));
            menu_items.at(i)->setX(menu_items.at(1)->getX() - menu_items.at(i)->getW() - 50);
            curr_cursor_pos = 1;
        }
        //Place Control textures
        if ((menu_items.at(i)->getType().compare("CONTROLSTX") == 0)) {
            menu_items.at(i)->setX(((Window::getInstance()->getWindowSizeW() / 2) - menu_items.at(i)->getW())/2);
            menu_items.at(i)->setY(menu_items.at(2)->getY() + menu_items.at(2)->getH());
            if (i >= menu_items.size() - 2) {
                int tmp = (Window::getInstance()->getWindowSizeW() / 2) + menu_items.at(i)->getW();
                tmp = (Window::getInstance()->getWindowSizeW() - tmp)/2;
                tmp += Window::getInstance()->getWindowSizeW() / 2;
                menu_items.at(i)->setX(tmp);
            }
        }
    }

    //Load later used menu items
    for (int i = 0; i < unused_menu_items.size(); i++) {
        if (!unused_menu_items.at(i)->loadMedia()) {
            printf("Failed to load menu item.\n");
            success = false;
        }
        else {
            //Set Coordinates for later used menu items
            unused_menu_items.at(i)->setY(menu_items.at(1)->getY());
            unused_menu_items.at(i)->setX(menu_items.at(1)->getX());
        }
    }
    return success;
};

bool Menu::init() {
    bool success = true;

    //Set up values for Menu Items
    std::string fontpath1 = "assets/fonts/PlayfairDisplay-BlackItalic.ttf";
    std::string fontpath2 = "assets/fonts/PlayfairDisplay-Italic.ttf";
    MenuTexture* menutx = NULL;
    SDL_Color color1 = { 213, 0, 28, 255 };
    SDL_Color color2 = { 255, 184, 81, 255 };
    std::array<std::string, 3> titles = { "Big City Knights", "Start Game", "Exit" };

    for (int i = 0; i < titles.size() + 1; i++) {
        if (i == 0) {
            menu_items.push_back(new MenuTexture(titles[i], fontpath1.c_str(), 112/*112*/, color1));
        }
        else if(i < titles.size()){
            menu_items.push_back(new MenuTexture(titles[i], fontpath2.c_str(), 48, color2));
        }
        else {
            menu_items.push_back(new MenuControlsTexture("assets/sprite_sheets/menu/MENULEFTPLAYER.png"));
            menu_items.push_back(new MenuControlsTexture("assets/sprite_sheets/menu/MENURIGHTPLAYER.png"));
            menu_items.push_back(new MenuCursor());
        }
    }

    //Menu Items that are used later
    unused_menu_items.push_back(new MenuTexture("Resume", fontpath2.c_str(), 48, color2));
    unused_menu_items.push_back(new MenuTexture("Restart", fontpath2.c_str(), 48, color2));
    unused_menu_items.push_back(new MenuTexture("Start Game", fontpath2.c_str(), 48, color2));

    return success;
};

void Menu::changeFirstMenuItemTo(int x) {
    menu_items.erase(menu_items.begin() + 1);
    menu_items.emplace(menu_items.begin() + 1, unused_menu_items.at(x));
};

void Menu::close() {
    //Free menu items
    for (int i = 0; i < menu_items.size(); i++) {
        menu_items.at(i)->free();
    }
    menu_items.clear();

    //Free unused menu items
    for (int i = 0; i < unused_menu_items.size(); i++) {
        unused_menu_items.at(i)->free();
    }
    unused_menu_items.clear();
};
```