SDL2 Course Day-2
Moving to Classes

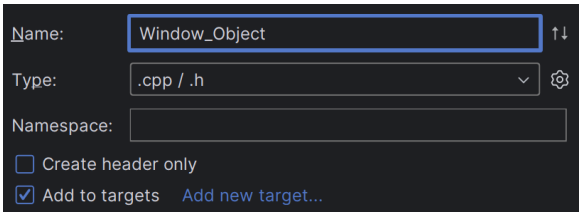
Currently all our code is in the main.cpp file. In the future when we have other objects with their own behaviour, it will be a huge pain to correct any mistakes or extend to any new feature. So, we will move to classes which will make our job later way easier.
Our main.cpp file currently:

```cpp
int main(int argc, char **argv) {

    // initializes SDL2
    SDL_Init(SDL_INIT_VIDEO);

    // Creating window and renderer
    SDL_Window *Window = SDL_CreateWindow("My window", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, 900, 600, 0);
    SDL_Renderer *Renderer = SDL_CreateRenderer(Window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);

    // Creating a rectangle
    SDL_Rect rect1 = {50, 50, 30, 30};

    bool quit = false;
    SDL_Event event;
    while (!quit) {
        // taking input from user
        SDL_PollEvent(&event);

        // handling input
        if (event.type == SDL_QUIT) {
            quit = true;
        }
        if (event.type == SDL_KEYDOWN) {
            if (event.key.keysym.scancode == SDL_SCANCODE_RIGHT) {
                rect1.x += 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_LEFT) {
                rect1.x -= 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_UP) {
                rect1.y -= 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_DOWN) {
                rect1.y += 10;
            }
        }

        // drawing
        SDL_SetRenderDrawColor(Renderer, 0, 0, 0, 0);
        SDL_RenderClear(Renderer);          // clears background

        SDL_SetRenderDrawColor(Renderer, 255, 0, 0, 255);
        SDL_RenderFillRect(Renderer, &rect1);   // draws rectangle

        SDL_RenderPresent(Renderer);        // swaps buffers
    }

    // destroys everything
    SDL_DestroyWindow(Window);
    SDL_DestroyRenderer(Renderer);
    SDL_Quit();
    return 0;
}
```

Red boxes are properties of the window.
Green boxes are properties of the player.

To create a new class in clion, right click on the project name in the left and do the following:

Click on new C++ class:



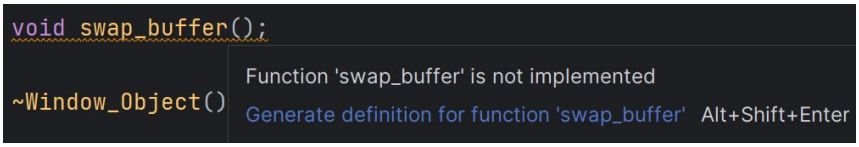
Now in our Window_Object.h file, we will place all our declarations:

Notice that in our main.cpp file, our Window_Object should have a window and a renderer and it should create them. It should then be able to handle input based on an SDL_Event and also be able to clear the background and swap buffers. In our handle input function, we will also need to be able to update a variable quit which should be accessed in the main function to quit the while loop. To do this we can either pass a reference to the quit variable to the handle_input function OR we can create a member inside the class called quit. We have followed the latter.

Hence, we create the following functions in our header file:

```
#ifndef SDL2_TEST_WINDOW_OBJECT_H
#define SDL2_TEST_WINDOW_OBJECT_H

#include <SDL2/SDL.h>

class Window_Object {
public:
    SDL_Window *Window;
    SDL_Renderer *Renderer;
    bool quit;

    Window_Object(char *TITLE, int width, int height);

    void handle_input(SDL_Event event);

    void clear_background();

    void swap_buffer();

    ~Window_Object();
};

#endif //SDL2_TEST_WINDOW_OBJECT_H
```

To now create definitions, hover your mouse over the function, and you should get the following option:



Create definitions for all of the functions in the header file and you should now see the following in Window_Object.cpp:

```cpp
#include "Window_Object.h"

Window_Object::Window_Object(char *TITLE, int width, int height) {

}

void Window_Object::handle_input(SDL_Event event) {

}

void Window_Object::clear_background() {

}

void Window_Object::swap_buffer() {

}

Window_Object::~Window_Object() {

}
```

Now we copy paste the code from our main function which we think belongs to the window object in their respective functions (refer to the red boxes in the main.cpp picture on page 1):

```cpp
#include "Window_Object.h"

Window_Object::Window_Object(char *TITLE, int width, int height) {
    Window = SDL_CreateWindow(TITLE, SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, width, height, 0);
    Renderer = SDL_CreateRenderer(Window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
    quit = false;
}

void Window_Object::handle_input(SDL_Event event) {
    if (event.type == SDL_QUIT) {
        quit = true;
    }
}

void Window_Object::clear_background() {
    SDL_SetRenderDrawColor(Renderer, 0, 0, 0, 255);
    SDL_RenderClear(Renderer);
}

void Window_Object::swap_buffer() {
    SDL_RenderPresent(Renderer);
}

Window_Object::~Window_Object() {
    SDL_DestroyWindow(Window);
    SDL_DestroyRenderer(Renderer);
}
```

Now in our main.cpp file, we include our header file and delete the original code which corresponded to our window and then create a new object and use the functions:

```
#include "Window_Object.h"

int main(int argc, char **argv) {

    // initializes SDL2
    SDL_Init(SDL_INIT_VIDEO);

    Window_Object *win_obj = new Window_Object("My Game", 900, 600);

    // Creating a rectangle
    SDL_Rect rect1 = {50, 50, 30, 30};

    SDL_Event event;
    while (!win_obj->quit) {
        // taking input from user
        SDL_PollEvent(&event);

        // handling input
        win_obj->handle_input(event);
        if (event.type == SDL_KEYDOWN) {
            if (event.key.keysym.scancode == SDL_SCANCODE_RIGHT) {
                rect1.x += 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_LEFT) {
                rect1.x -= 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_UP) {
                rect1.y -= 10;
            }
            if (event.key.keysym.scancode == SDL_SCANCODE_DOWN) {
                rect1.y += 10;
            }
        }

        // drawing
        win_obj->clear_background();  // clears background

        SDL_SetRenderDrawColor(win_obj->Renderer, 255, 0, 0, 255);
        SDL_RenderFillRect(win_obj->Renderer, &rect1);   // draws rectangle

        win_obj->swap_buffer();      // swaps buffers
    }

    // destroys everything
    delete win_obj;
    SDL_Quit();

    return 0;
}
```

The red boxes are new code

Please note that, we havent changed any logic and the game still runs exactly the same but now we have 3 files instead of one and the code logic for the window is seperated and is in a different file;
We shall now do the same for our player; Create a new class called Player_Object. (Refer to the green boxes in page 1 which correspond to the code of the player), The player only has one property of a SDL_Rect and has the functions handle_input and draw. Note that to draw, we will need to pass in the renderer. We could either pass in the renderer as a paremeter to the function or we can save it as part of the object. We follow the latter.

We now make our Player_Object.h file as below:

```
#ifndef SDL2_TEST_PLAYER_OBJECT_H
#define SDL2_TEST_PLAYER_OBJECT_H

#include <SDL2/SDL.h>

class Player_Object {
public:
    SDL_Rect rect;
    SDL_Renderer *Renderer;

    Player_Object(int x, int y, int w, int h, SDL_Renderer *Renderer);

    void handle_input(SDL_Event event);

    void draw();
};

#endif //SDL2_TEST_PLAYER_OBJECT_H
```

We now generate our definitions which will appear in the Player_Object.cpp file:

```
#include "Player_Object.h"


Player_Object::Player_Object(int x, int y, int w, int h, SDL_Renderer *Renderer) {

}

void Player_Object::handle_input(SDL_Event event) {

}

void Player_Object::draw() {

}
```

We then copy paste code from our main function into the file (Refer to the green boxes in page 1):

```
#include "Player_Object.h"

Player_Object::Player_Object(int x, int y, int w, int h, SDL_Renderer *Renderer)
        : Renderer(Renderer) {
    rect = {x, y, w, h};
}

void Player_Object::handle_input(SDL_Event event) {
    if (event.type == SDL_KEYDOWN) {
        if (event.key.keysym.scancode == SDL_SCANCODE_RIGHT) {
            rect.x += 10;
        }
        if (event.key.keysym.scancode == SDL_SCANCODE_LEFT) {
            rect.x -= 10;
        }
        if (event.key.keysym.scancode == SDL_SCANCODE_UP) {
            rect.y -= 10;
        }
        if (event.key.keysym.scancode == SDL_SCANCODE_DOWN) {
            rect.y += 10;
        }
    }
}

void Player_Object::draw() {
    SDL_SetRenderDrawColor(Renderer, 255, 0, 0, 255);
    SDL_RenderFillRect(Renderer, &rect);
}
```

Make sure in the above code to replace rect1 with rect, since that's the name of our rectangle in our class.

Now we include our Player_Object.h, remove the code from our main.cpp and call our functions:

```cpp
#include "Window_Object.h"
#include "Player_Object.h"

int main(int argc, char **argv) {

    // initializes SDL2
    SDL_Init(SDL_INIT_VIDEO);

    Window_Object *win_obj = new Window_Object("My Game", 900, 600);
    Player_Object *player = new Player_Object(50, 50, 30, 30, win_obj->Renderer);

    SDL_Event event;
    while (!win_obj->quit) {
        // taking input from user
        SDL_PollEvent(&event);

        // handling input
        win_obj->handle_input(event);
        player->handle_input(event);

        // drawing
        win_obj->clear_background();  // clears background
        player->draw();               // draws rectangle
        win_obj->swap_buffer();       // swaps buffers
    }

    // destroys everything
    delete win_obj;
    delete player;
    SDL_Quit();

    return 0;
}
```

Notice how easy to read our main.cpp file has become due to the usage of classes, although we now have 4 files more, but the code of each has been split and is more easier to understand and debug.