Game loop :

{

Input- player interact with game

Simulation- create, update, collision

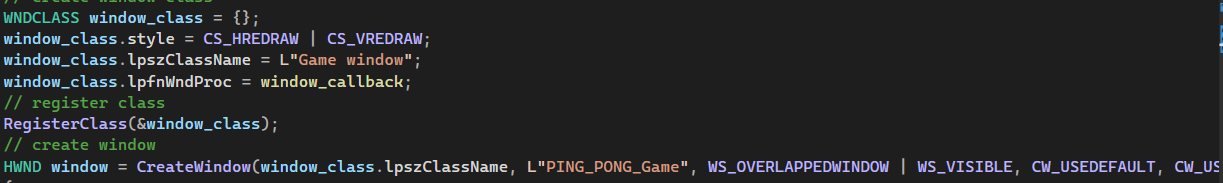
Rendering- create in window, and display

}

# Winmain is API for graphical windows based application

// just include windows for this,

// winmain will be our main



To create window, we make window class:

Set Style it, have its name , have callback function

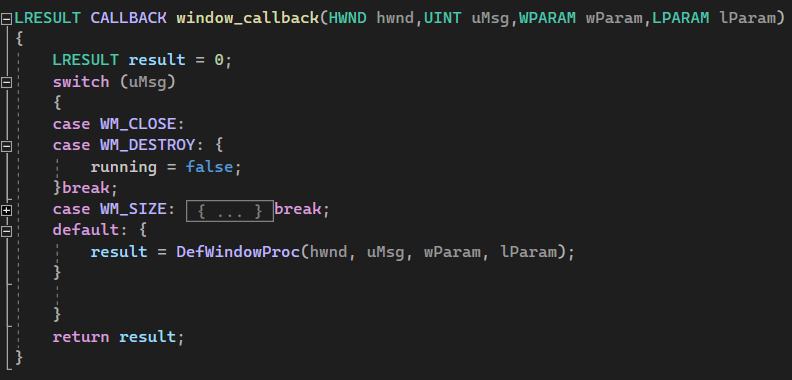
Window\_callback : is something which windows use to pass msg down to us when something important happens to us, like window is closed, resized, input from user.

Then we call function” create window” in which we pass parameters like this windows class, title of window and windows properties, width height, instance;

In this winmain function we will run a while loop infinte times uptill user wants to close the game , for that bool running is used in – while,

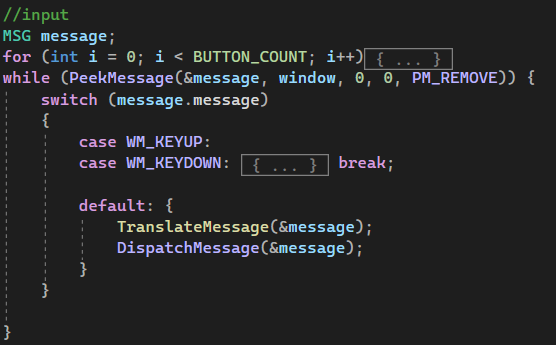
And running is always true, and become false only when user clicks on close button.

This all happens in window callback function as it will have msg (this msg will be passed from us from game loop) from user if it closes the window by clicking on x button.



The msg passed to callback function from game loop like this:

This code is in game while loop



How rendering is done

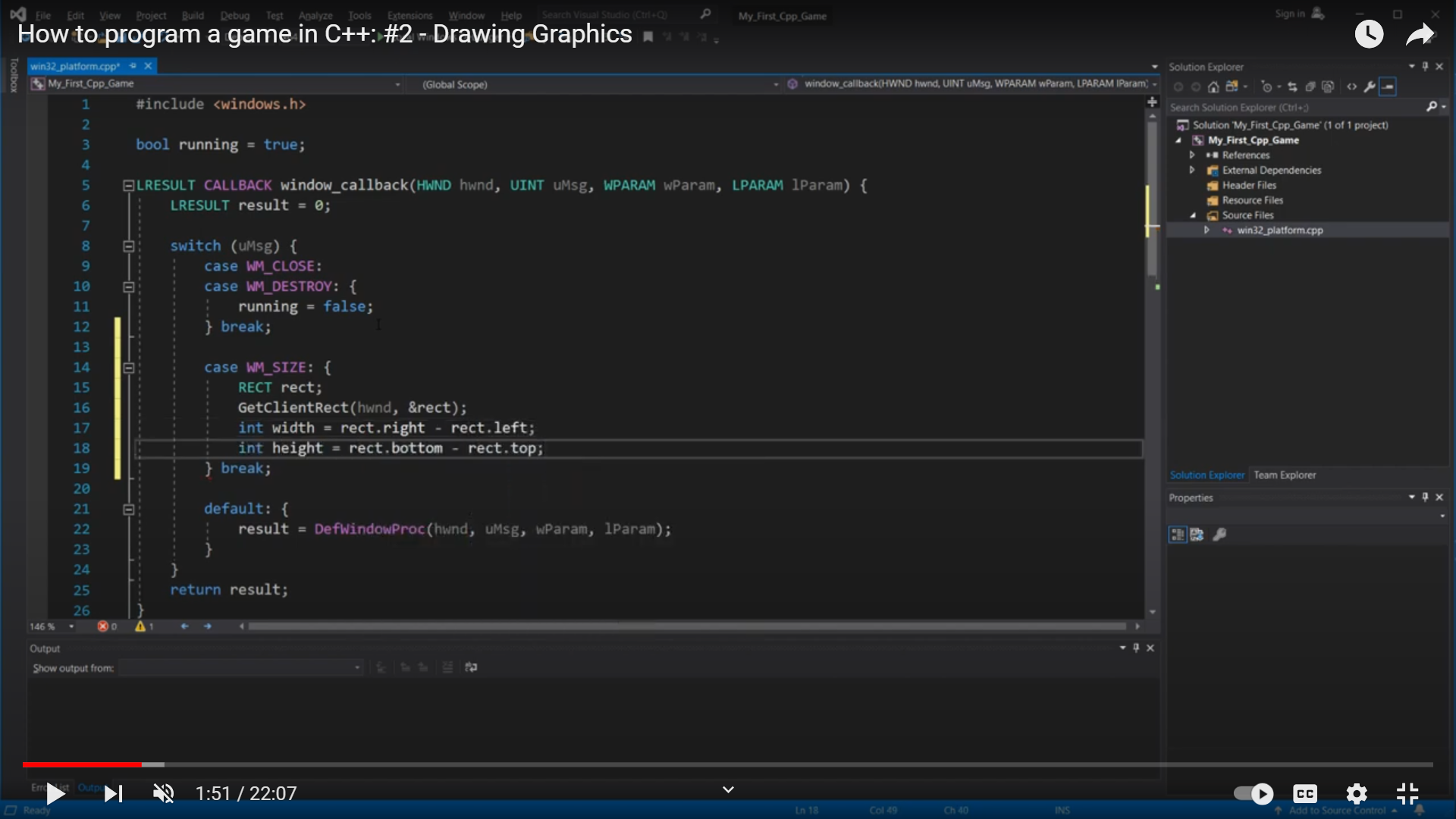
1. Get the buffer(memory)
2. Fill it
3. Send to window

This rendering will be done in loop, in game loop,

so frame 1-> position of element 1

Frame2-> position 2

How to get buffer:



GetClientRect -> gives RECT that has height and width of that window.

After getting size of window: which can be 1280x 780

So no of pixels = w\*h;

Buffer\_size = no of pixel \* size of pixel;

Size of pixel is 32 bits integer, storing 8 bits red, 8 bits green , 8 bits blue;

So now in i=this case for buffer we want it to be persisted while game goes along, so we need memory in HEAP.

This is done by “VirtualAlloc” function.

Before allocating new memory we should first free the old memory, this case will hit when window resizes by user.

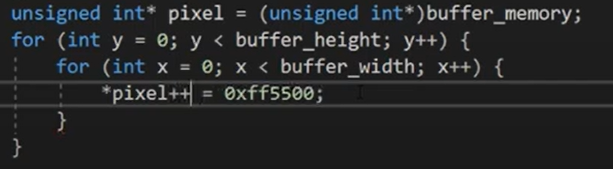
Now we have the memory now we will send it to windows and ask it to use it by using method :

StretchDIBits(…)

Things followed uptill now will give us window with black screen.

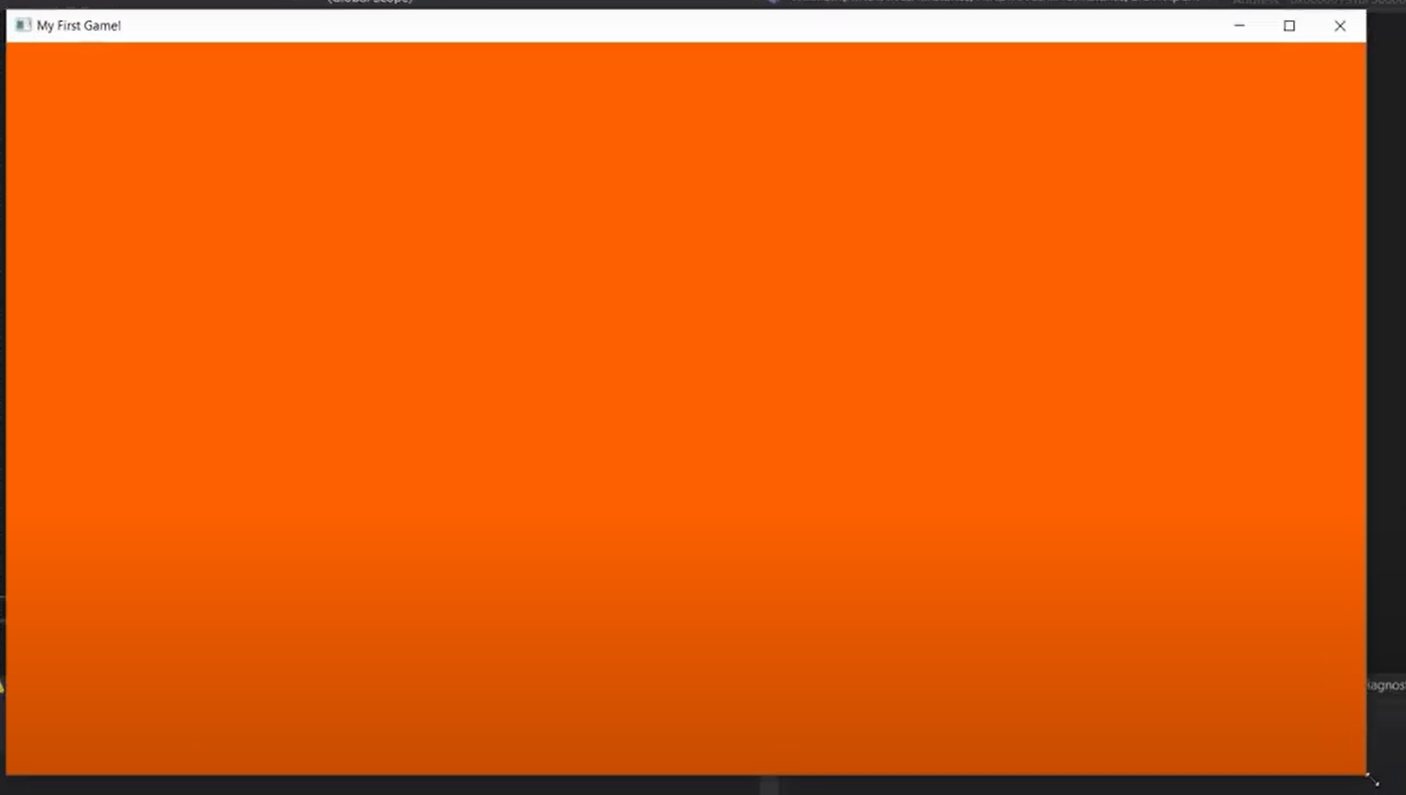
And window can get resized, still black screen remains.

Now to color the window :

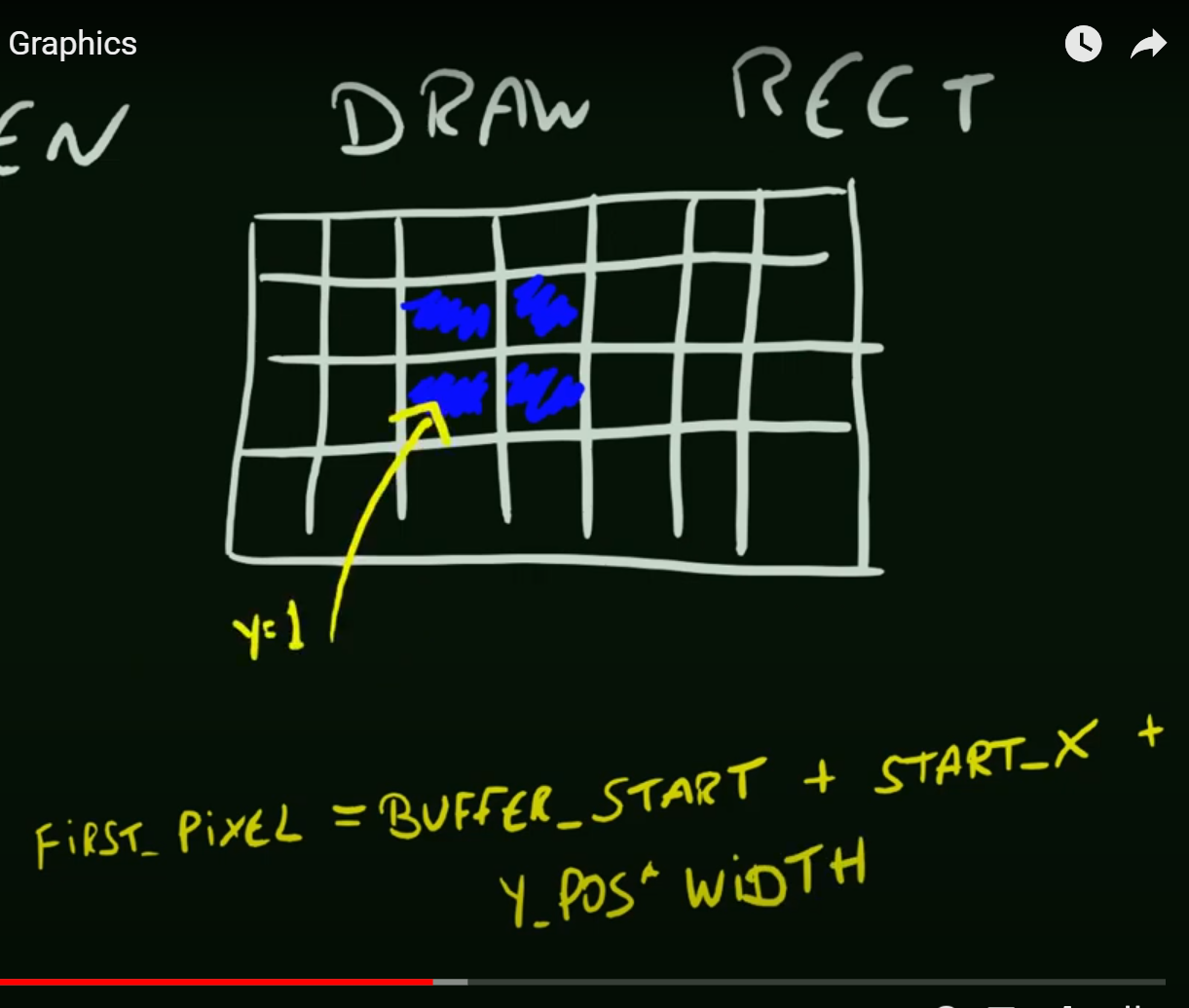


Do this in game loop, here pixel is filled with orange colour.

Our screen will look like this:



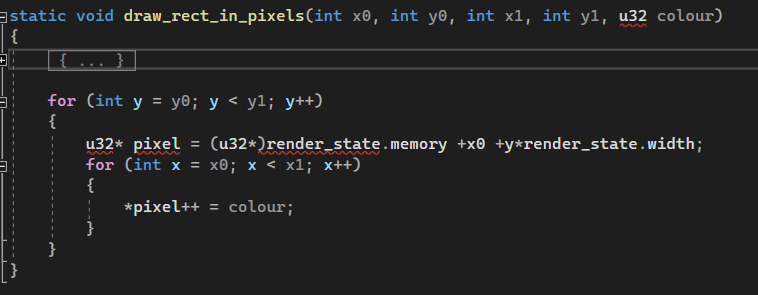
Now for creating a rectangle , we need to colour few pixels (all together who forms the rectangle) with some other colour.



Lets say rectangle to be drawn at x0 , y0 location on screen.

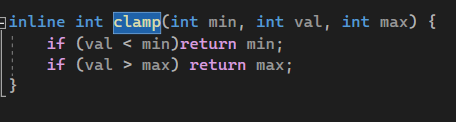
For this instead of pixel =0 , start pixel will be = buffer\_start+ x0+ y\* width of buffer;

We can just do y++, x++ to fill every pixel uptill we reach limit of our rectangle (x1, y1)

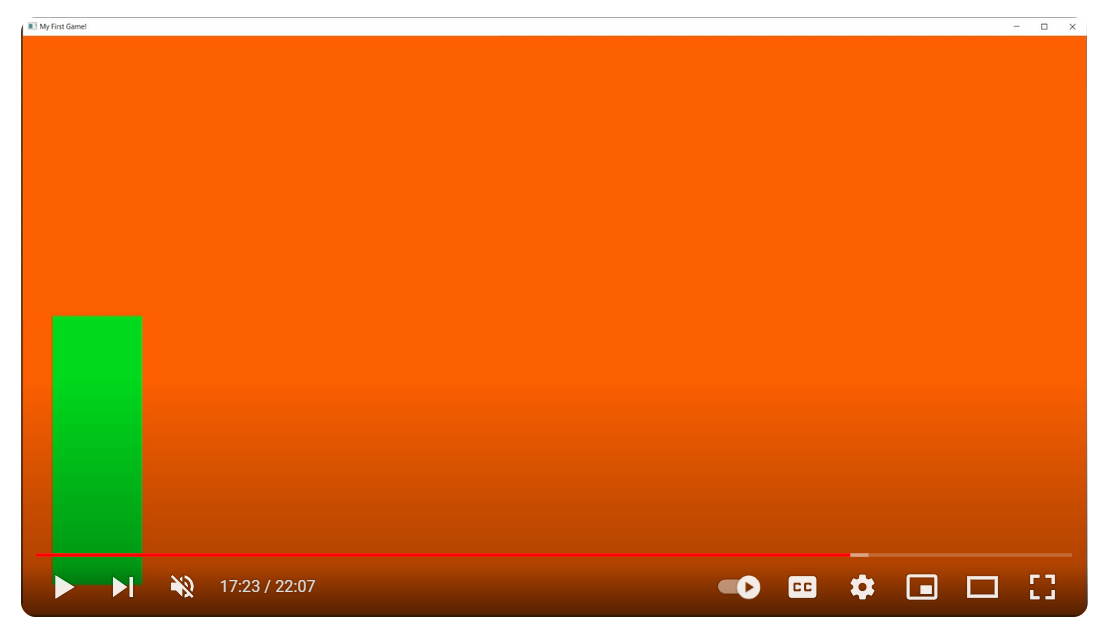
.

Now when we resize , width and heght of screen changes and our x0 can greater or smaller than screen start and end points, and it crashes our game

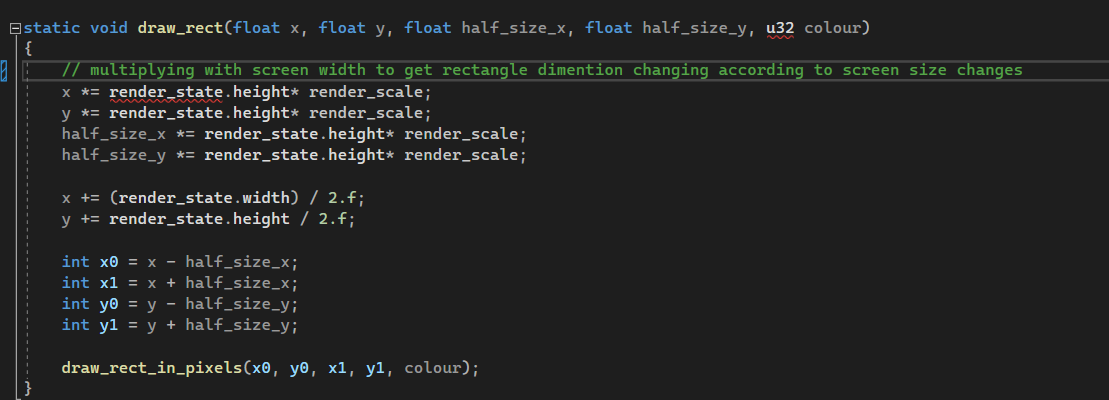
So in that case we want x0 to change according to screen size which is done by clamp function:



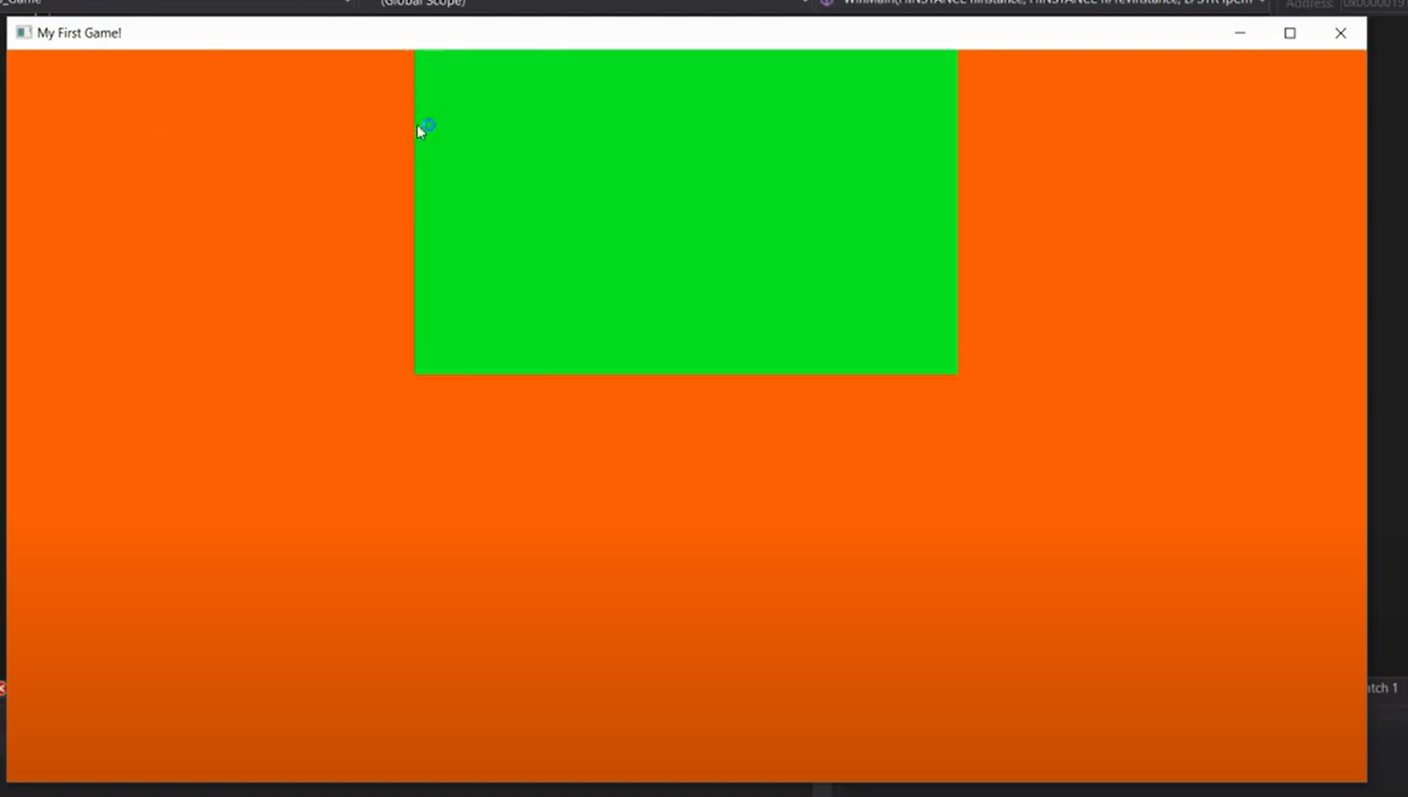
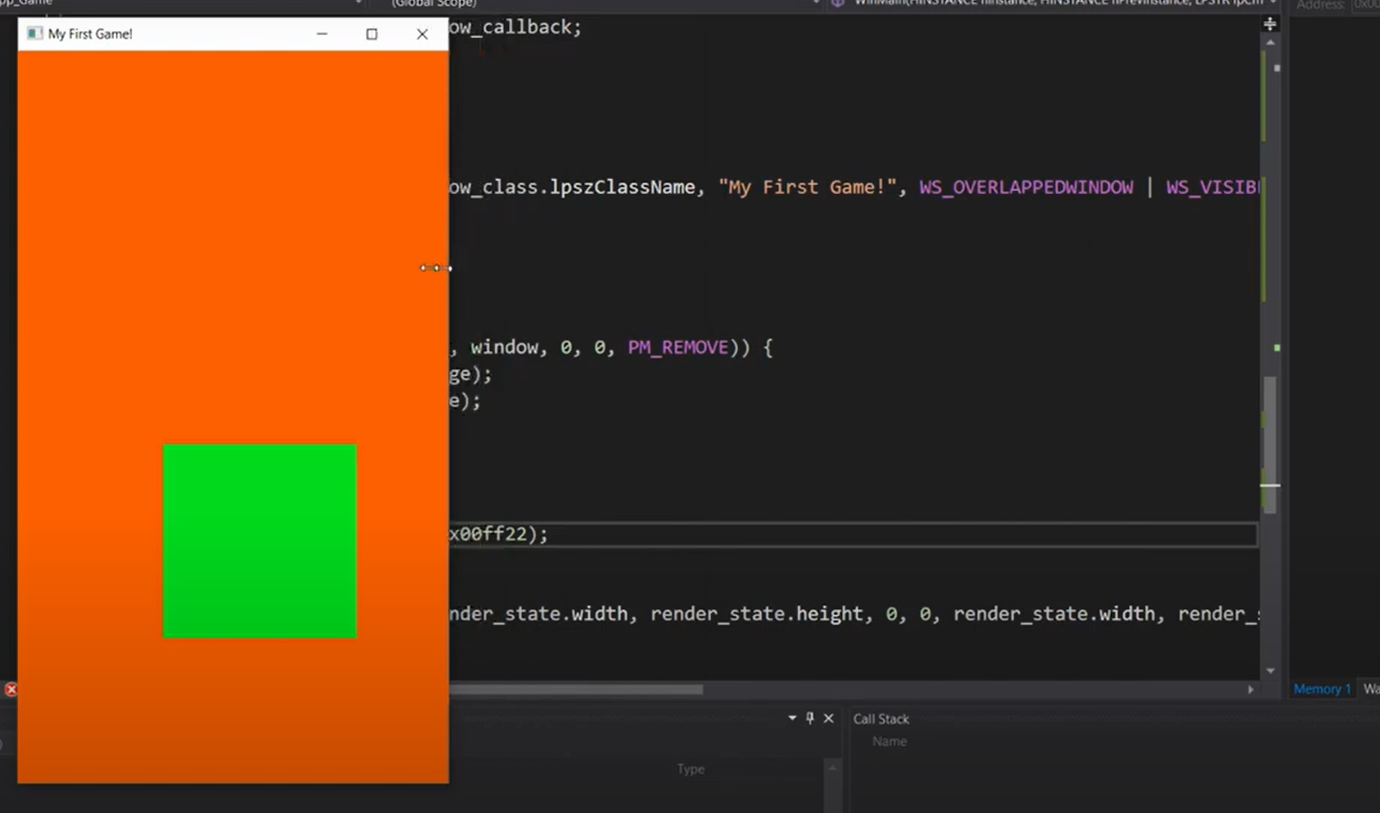
But overall width of rectangle is still constant as compare to whatever screen size we have.

So in order to make rectangle width adjusting automatically when we resize->



After this when we resize screen , rectangle also change sits dimension:

Now back to take inputs from user:

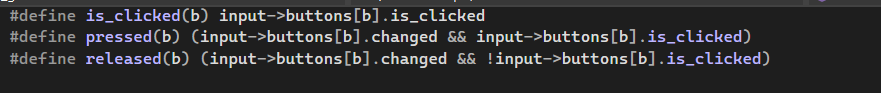
In game loop, we try to get what interaction user does with keyboard buttons:

Eigther user press once the button === called -> is\_clicked

Or user keep it pressed for more frames -==== called -> pressed

Is clicked is basically once press of keyboard button ,

But pressed is condition when : button is clicked and its state is changes from not clicked to clicked.



When user pushes one button suppose up button ,

For frame 1:

We set for that button is\_clicked = true ; changed = true;

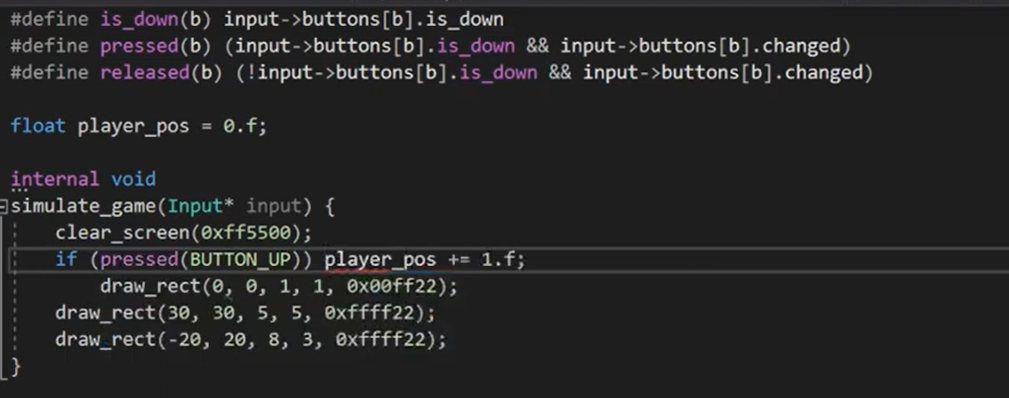
And now for frame 2:

If still button is holded still, its chnges param will be true and also is\_clicked is true,

That means it is pressed.

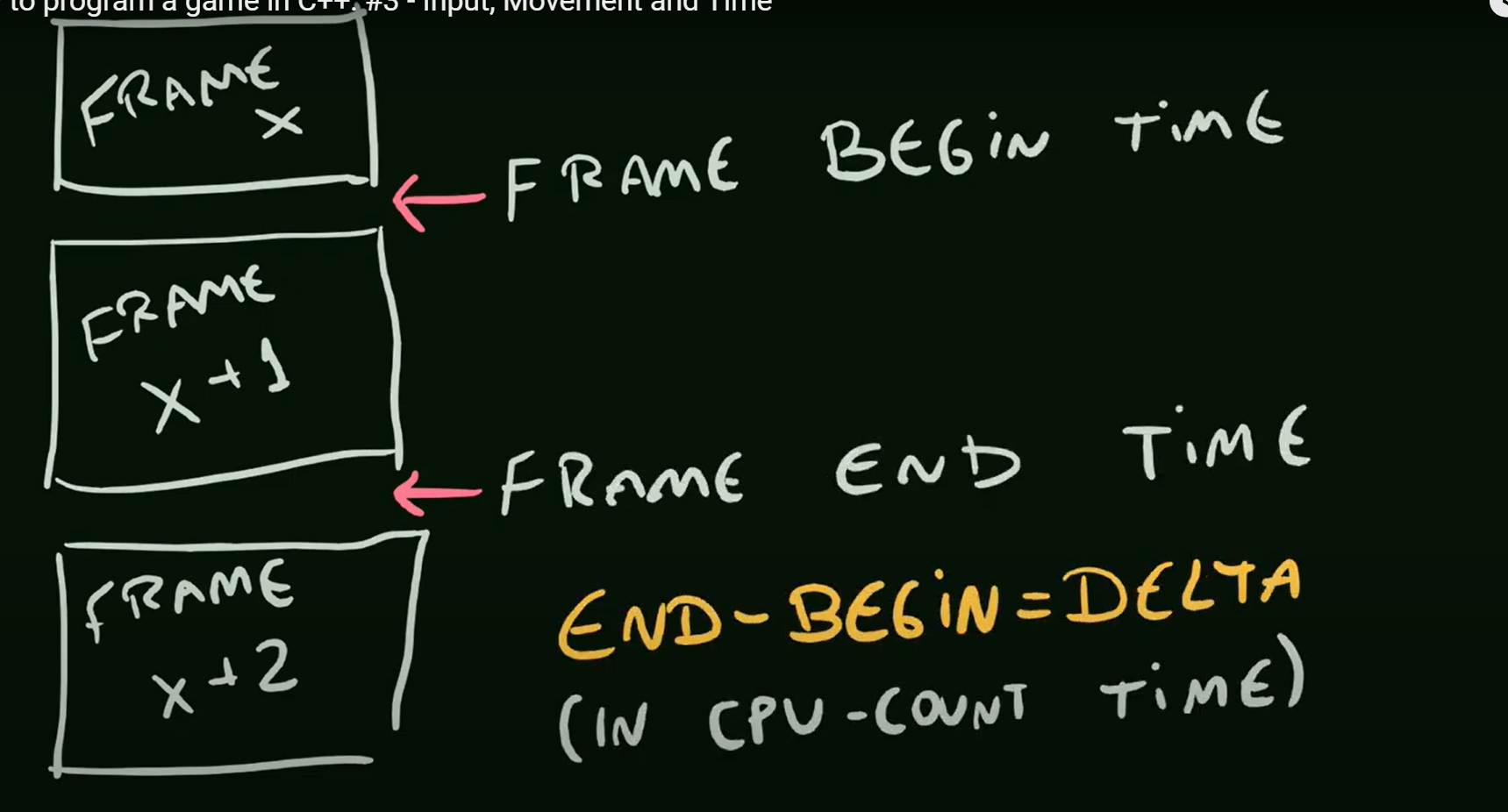
If release : is\_clicked = false && changes = true;

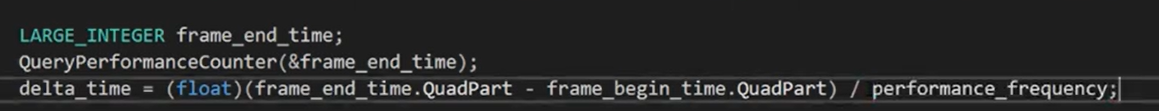
Now how to move a rectangle , this is happening in multiple frames, in each frame , all screen is colouring orange, and a rectangle is drawn at location x0, then at x0 + 1.f, then at x0 +1.f+1.f;



Now here we gave speed input by incrementing position by manual value,

But this can we done given with respect to time, which can be retived by CPU clock time





We will give this delta time(dt) to game,

And increment position of every rectangle as = pos = speed\*dt;