Graphics (GFX) framework

Mostafa Khaled

June 1, 2024

Contents

0.1 what is gfx?

gfx (which stands for graphics) is an abstract project for OpenGL. which is intented to be used to simplify usage of OpenGL and glfw. These library isn't as good as something like SFML or SDL2 BUT It's just been made for learning purpose.

0.2 about the author

I started learning OpenGL short time after getting into linux. I started with D3D9 which was a bit old. but I think it was a good experience to have.

0.3 who is this book for?

anyone with c++ intermediate experience who wants to learn graphics programming. it's worth noting that this is NOT a high quality OpenGL tutorial. it just discusses a basic framework based on OpenGL.

Chapter 1

system

1.1 gfx::rgba

In GFX we represent colors in gfx::rgb, gfx::rgba. the value of colors are in range [0, 1] not [0, 255].
you may wonder if you can use something like hex, fortunatly we provide gfx::hex function which accepts a hex unsigned int and return rgba based on its value.
NOTE: you must provide complete hex to avoid weird colors that can result if you didn't do so. for example: use 0xffffffff instead of 0xffffff

1.2 gfx::clock

gfx::clock class is a class use to measure time based on C++ "chrono" header using chrono timepoints and clocks.
it have 2 member functions which are elapsed() and restart().
elapsed() return the total time since the clock creation or the last call to restart().
restart() resets the clock.
It's worth nothing that this class is very light and it contains only 1 timepoint as a member.

1.3 debug

debug.hpp contains 2 macros: ASSERT and GLFW_ASSERT
both of them accepts a boolen value called x and an error message called y.
if x is false, then:

1. ASSERT print the error message and exit with error code -1.
2. GLFW_ASSERT print error message and terminate GLFW through glfwTerminate() and exit with error code = -1

debug have namespace test that have 3 functions:

1. pause() which pauses the program till pressing a key. NOTE: it doesn't notify you by any message, you need to output -for example "Press any key to continue.. " on your own.
2. print(std::string msg) to print a certain message as extra info.
3. dprint(std::string msg) which print msg if DEBUG or _DEBUG is defined.

NOTE: both of them are rarely -or even never- used because of the more advanced logger system that is logger.hpp which is found in the deps folder. take a look. it supports console, file, dialogs(only for windows).

1.4 gfx::callback

gfx::callback is a namespace that have two functions only (actually it's intended to add more functionality).
these two functions are meant to be used as default for gfx programs. you can create your own if you want.

The 2 functions are:

1. error
2. key

NOTE: callbacks are only some functions to notify you of certian events like errors and a certian key pressed etc..
Lookup callbacks in GLFW documentation.

1.5 gfx::vector2 and gfx::vector3

These are two template classes, which means they can accept may data types: int, float, etc...
Till now, they only supports addition and subtraction from each other.
It have a single member functions called gfx::vectorX::distance(), which accepts another vector of the same type and return the distance between them.
NOTE: vectors are intended to use with float especially in range [-1-1] which is screen coordinates in OpenGL by the way.

1.6 gfx::box

This is a very simple structure(class) that contains the centre of the box and dimentions (width and height).

gfx::vbuffer can have rectangle data based on the data in the box, by using an overload of gfx::vbuffer::append(). This overload accepts box and a vertex of the same type that the class use -as this class can use different vertcies as it's a template- to provide the extra data that these vertex need.
To simplify, box data only overwrites the x and y coordinates in the gfx::vbuffer

1.7 interpolators

functions are:

- lerp(linear interpolation): which works with 1D , 2D, 3D coordinates.
- clamp: ensure a value lies in a certian range. only 1D when writing this.
- smoothstep function.

gfx::lerp Our 1D version gfx::lerp accepts (start denoted by a, end denoted by b, progress denoted by t, between [0, 1]) and is implemeted like that

$$lerp(a, b, t) = a + (b - a) * t$$

gfx::lerp in other dimentions does the same for each axis.

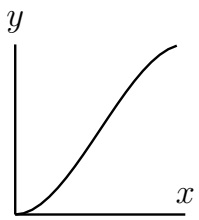
gfx::clamp gfx::clamp accepts (value v, minimum, maximum) and return max if v greater then max, and min when v smaller than min. and v if it lies in between min and max.

gfx::smoothstep This is just an ease function that gives lerp smooth motion when applied to lerp t parameter.

let smoothstep function be st for short.

$$st(t) = 3t^2 - 2t^3$$

Our smoothstep function looks like this:-

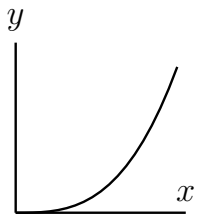


I think it looks nice. you may have noticed that this functions controls speed and by looking to the last figure you can notice that the speed ease in and ease out which results in a motion that looks like something is sliding.

1.7.1 gfx::cubic namespace

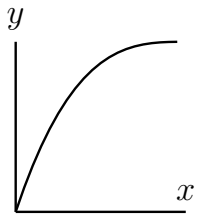
This is a very small namespace to use cubic ease functions in, out and inout (prefixed with ease_ ex: ease_in).
gfx::cubic::ease_in

$$f(x) = x^3$$



gfx::cubic::ease_out

$$f(x) = 1 - (1 - x)^3$$

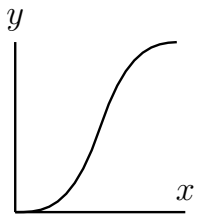


gfx::cubic::ease_inout
when x < 0.5:

$$f(x) = 4x^3$$

when x $\geq$ 0.5:

$$f(x) = 1 - \frac{(2-2x)^3}{2}$$



actually this is just a steeper version of our smoothstep function