Simple and Fast Multimedia Library (SFML) 3.0 & C++

Draw a colored/textured circle on your screen

First you must install homebrew and SFML. If you are using MacOS and vscode (Visual Studio Code) this is easily done using your terminal. I use iTerm 2 and I recommend you install it and get used to it rather than the default mac terminal (maybe even switch to linux why you're at it). Anyways, here are the installation steps:

1. Open the terminal (iTerm 2)
2. Type: /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
   - This is the command that installs homebrew from github
3. Wait for the onscreen instructions and downloads to finish
4. Type: brew –version
   - Checks to see the version of Homebrew installed.
5. Type: brew install sfml
   - Installs sfml
6. Type: brew info sfml
   - Displays info about the installed SFML package, including its version and installation path.
     i. It should be SFML 3.0
7. (Possibly Optional) If your VS Code intellisense does not automatically recognize the SFML 3.0 info, then possible using command + shift + p to open the command palette and select C/C++ Edit Configurations (UI).
   - Now locate the include path section and add the path to the SFML headers usually they are the ones below, but they could be different and you should use brew info sfml in your computer terminal to determine the install path.
     i. /opt/homebrew/include
     ii. /opt/homebrew/lib
   - If you still have error squiggles this helped me, change the default C++ standard to c++23 and it fixed my error squiggles

            i. You can changed you C standard version to c23 as well, but this won't change anything for this project

<u>Start of the actual code explanation:</u>

- #include <SFML/Graphics.hpp>
    - Includes the SFML graphics module, which provides classes and functions for rendering 2D graphics, handling windows, and managing textures, shapes, etc.
- #include <optional>
    - Includes the C++ standard library's std::optional, which is a wrapper type that can contain a value or be empty. It is used here for handling optional events
        - std::optional represents
            - A valid sf::Event object (when an event exists)
            - An empty state (when no event exists)
        - A wrapper type is a type that encapsulates another type to provide additional functionality, safety, or flexibility. It acts as a container for the underlying type, often adding features like optionality, immutability, or type safety.
        - You can check if std::optional contains a value using methods like .has_value() or by using it in a conditional statement
- sf::RenderWindow window(sf::VideoMode({800, 600}), "My window");
    - sf::RenderWindow window
        - Creates a window for rendering graphics named window.
    - sf::VideoMode({800, 600})
        - Specifies the dimensions of the window (800 pixels wide and 600 pixels tall)
    - "My window"
        - The title of the window displayed in the title bar.
- sf::CircleShape shape(200);
    - Creates a circle shape named shape with a radius of 200 pixels. This is an SFML object used for drawing a circle.
- shape.setFillColor(sf::Color(100, 250, 50));
    - This a method of the sf::Shape and its derived classes (i.e. sf::CircleShape)

    - It selects the fill color of the shape, which is the color used to fill the interior of the shape.
    - Represented using RGBA color model (Red, Green, Blue, Alpha)
    - sf::Color(r, g, b)
        - Creates a color with the specified red, green, and blue values. The alpha (transparency) defaults to 255 (fully opaque) if not specified
    - (This RGBA color combo results in a bright greenish color)
        - 128 for alpha value would set transparency to 50%
            - shape.setFillColor(sf::Color(100, 250, 50, 128));
- sf::Texture texture;
    - Declares a texture object name texture, which is used to load and manage an image file.
    - If (!texture.loadFromFile("name_of_file.png")){ return -1;}
        - If not texture load from file then return -1, so the program ends. If it does load, then the file is loaded and the program continues.
        - (I personally feel this syntax looks a little strange, but I will be always using this conditional check to load textures, images, and music from now on until I encounter problems).
    - shape.setTexture(&texture);
        - Applies the loaded texture to the circle shape created earlier. The & is used to pass the texture by reference instead of making a copy of it for the function call.
    - shape.setTextureRect(sf::IntRect({80, 80}, {400, 400}));
        - Defines the portion of the texture to use.
        - This selects a rectangular region of the texture region starting at {80, 80} for its top left corner. With a width and height 400. {400, 400}.
- While (window.isOpen()){
    - This loop runs as long as the window is open. It is the main game/rendering loop
- While (std::optional<sf::Event> event = window.pollEvent()) {

  - std::optional<sf::Event>
    - Wraps the event in an optional type.
    - If there is no event, the optional is empty.
  - window.pollEvent()
    - Checks for any events (e.g., user input, window actions) that have occurred since the last iteration of the loop
- If (event->is<sf::Event::Closed>()) {
    - event->is<sf::Event::Closed>()
      - Checks if the "event" Event variable is the sf::Event::Closed() event, which checks if the event is a "close requested" event.
      - E.g., the user clicked the close button on the window).
      - ->
        - An operator used to access a member (like a function or variable) of an object through a pointer.
        - In general, when an event contains a value, you can access the sf::Event object it wraps using the -> operator.
      - is
        - A function provided by SFML's sf::Event class
        - Used to check if the event is of a specific type
        - A templated function where you must specify the function type
          - This is why the "< >" are absolutely necessary as they are "template arguments"
            - They are used to specify a type or value for a templated function or class
  - window.close();
    - Closes the window
- window.clear(sf::Color::Black);

        - Clears the window with a black background color. This prepares the window for the next frame.
- window.draw(shape);
    - Draw the circle shape (with the applied texture) onto the window.
- window.display();
    - Displays the contents of the window on the screen. This swaps the back buffer with the front buffer, making the drawn frame visible.
        - Front Buffer
            - Part of the window that is currently being displayed on the screen.
            - You generally do not draw directly to the front buffer because doing so can cause flickering or incomplete rendering.
        - Back Buffer
            - The back buffer is an off-screen buffer where all the drawing operations are performed.
            - You draw all your shapes, textures, and other graphics to the back buffer and this is not visible to the user while rendering is happening.

(replace "main" with the file name)

<u>Compile:</u>

clang++ -std=c++23 main.cpp -o main -lsfml-graphics -lsfml-window -lsfml-system

(Runs with c++23 though this is definitely not necessary)

- clang++
    - This is the clang c++ compiler.
    - It is used to compile C++ source code into an executable program.
    - Used on MacOS
    - g++ is used to compile on windows (and on mac, I do not understand why to use clang++ instead of g++ other than it is better on mac for some reason.)
- -std=c++23

    - This flag tells the compiler to use the C++23 standard
    - It enables features and syntax introduced in the c++23 version of the language
- main.cpp
    - This is the source file that you want to compile.
    - In this case it is “main.cpp”
- -o main
    - The “-o” specifies the ***output file name*** for the compiled program
    - In this case, the compiled executable will be named “main”
- -lsfml-graphics
    - ***SFML Graphics Library***
    - Provides functionality for rendering shapes, textures, and other graphical elements.
- -lsfml-window
    - ***SFML Window Library***
    - Provides functionality for creating and managing windows, handling events, and managing OpenGL contexts.
- -lsfml-system
    - ***SFML System Library***
    - Provides low-level utilities such as time management, threads, and basic data structures.
-

<u>Run:</u>
./main
- Runs the executable “main” that was created

Congratulations, you should have a static bright green textured circle on a window!