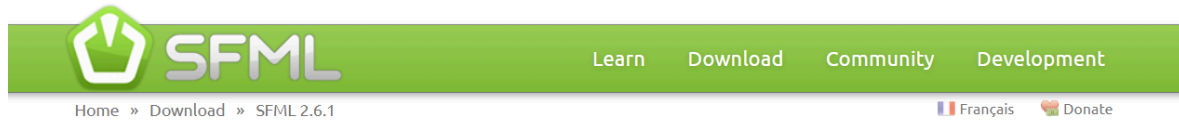


ORIENTAÇÕES SFML WINDOWS



Download SFML 2.6.1

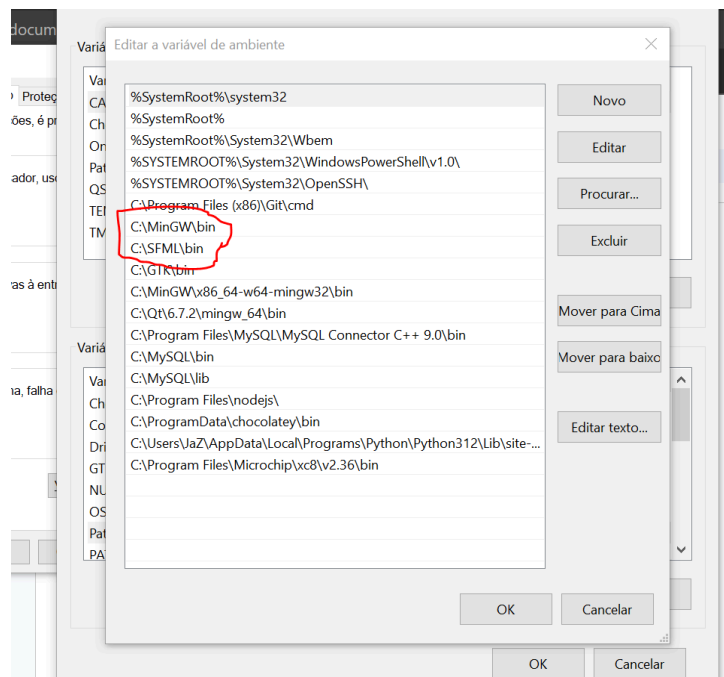
On Windows, choosing 32 or 64-bit libraries should be based on which platform you want to compile for, not which OS you have. Indeed, you can perfectly compile and run a 32-bit program on a 64-bit Windows. So you'll most likely want to target 32-bit platforms, to have the largest possible audience. Choose 64-bit packages only if you have good reasons.

Unless you are using a newer version of Visual Studio, the compiler versions have to match 100%!
Here are links to the specific MinGW compiler versions used to build the provided packages:

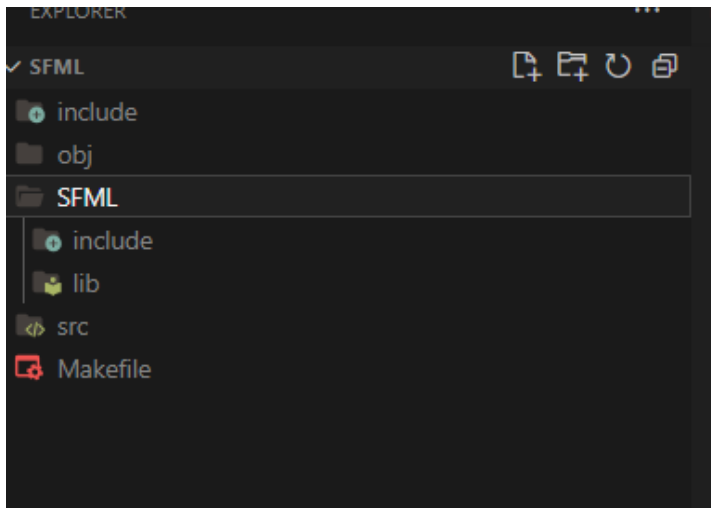
WinLibs MSVCRT 13.1.0 (32-bit), WinLibs MSVCRT 13.1.0 (64-bit)

Visual C++ 17 (2022) - 32-bit	Download 20.3 MB	Visual C++ 17 (2022) - 64-bit	Download 21.8 MB
Visual C++ 16 (2019) - 32-bit	Download 19.3 MB	Visual C++ 16 (2019) - 64-bit	Download 20.7 MB
Visual C++ 15 (2017) - 32-bit	Download 17.6 MB	Visual C++ 15 (2017) - 64-bit	Download 19.4 MB
GCC 13.1.0 MinGW (DW2) - 32-bit	Download 17.9 MB	GCC 13.1.0 MinGW (SEH) - 64-bit	Download 18.9 MB

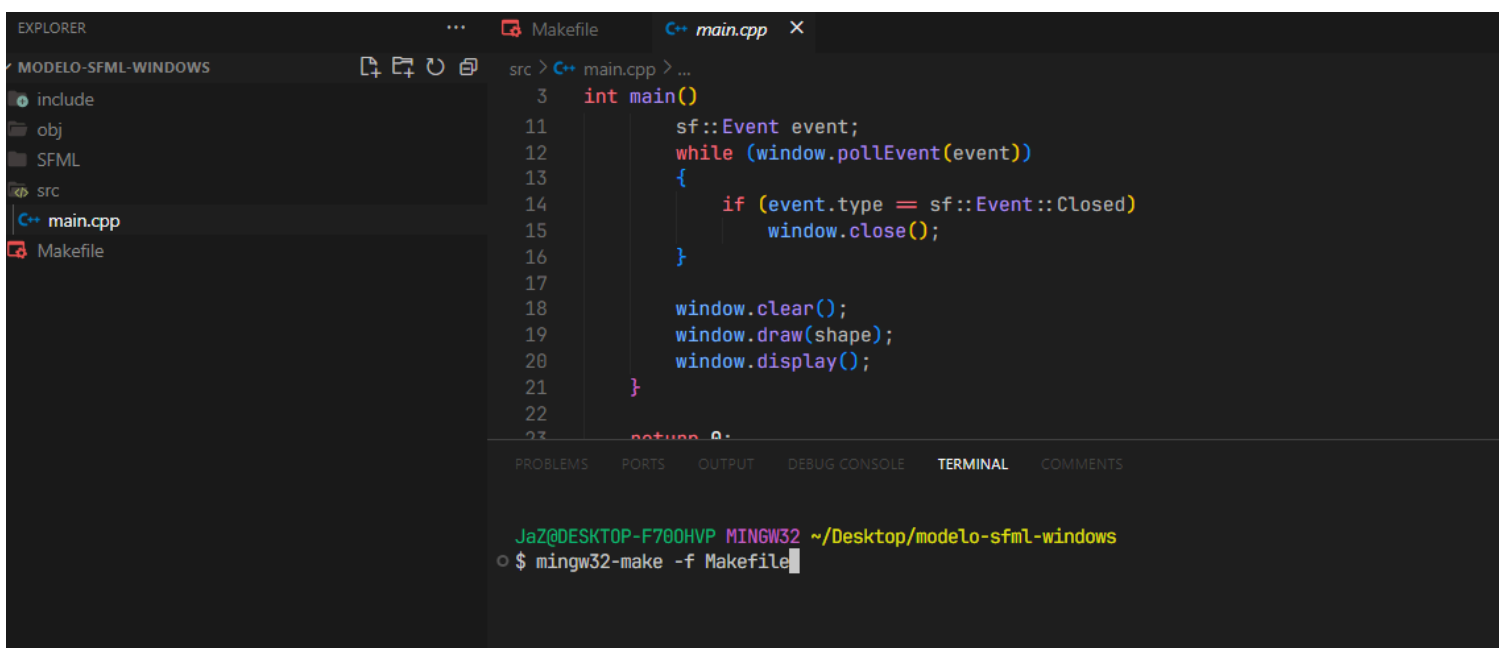
1. Instale o sfml e o compilador mingw, ambos estão no site do sfml.
2. Extraia os arquivos e os coloque em um caminho fácil para futuros acessos;
3. Depois que os extrair ambos terão uma pasta “bin” dentro deles, as coloque no path do sistema (coloquem ambos no path do sistema)



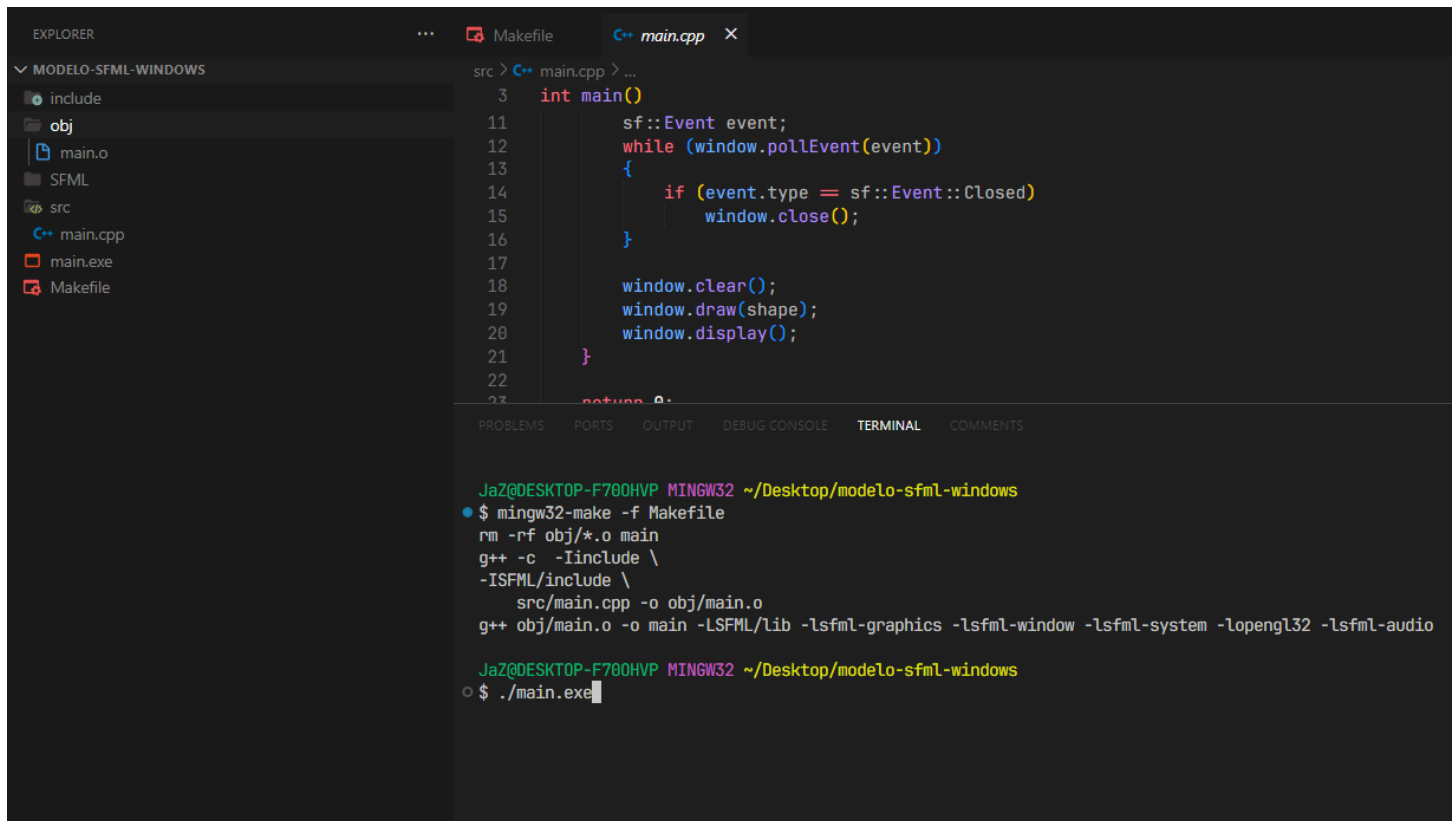
4. Abra o “modelo-sfml-window” no vscode (pode trocar o nome se quiser).
5. Na pasta “SFML” do repositório adicione a pasta “lib” e “include” do sfml que você baixou. (está junto com a pasta bin. Copie “include” e “lib” e as cole dentro do repositório).



6. Antes de adicionar o seu projeto tente compilar com o exemplo main.cpp. Para fazer isso é só dar o comando “mingw32-make -f Makefile” no diretório que contém o Makefile.



7. Depois só dar um `./main.exe` para executar o main.



```
src > C++ main.cpp > ...
3  int main()
11     sf::Event event;
12     while (window.pollEvent(event))
13     {
14         if (event.type == sf::Event::Closed)
15             window.close();
16     }
17
18     window.clear();
19     window.draw(shape);
20     window.display();
21 }
22
23 return 0;
```

```
JaZ@DESKTOP-F700HVP MINGW32 ~/Desktop/modelo-sfml-windows
$ mingw32-make -f Makefile
rm -rf obj/*.o main
g++ -c -Iinclude \
-ISFML/include \
src/main.cpp -o obj/main.o
g++ obj/main.o -o main -LSFML/lib -lsfml-graphics -lsfml-window -lsfml-system -lopengl32 -lsfml-audio

JaZ@DESKTOP-F700HVP MINGW32 ~/Desktop/modelo-sfml-windows
$ ./main.exe
```

8. Orientações finais: Os `.cpp` precisam estar todos no diretório “src”, sem ser separados por pastas. Teste antes com o exemplo, depois adicione o seu projeto, mas não esqueça de alterar o makefile com as adições. Exemplo de alteração (VERMELHO -> possível hierarquia de pastas. AZUL -> Projeto atual):

```
Makefile
8  OBJ_FILES := $(patsubst src/%.cpp,$(OBJ_DIR)/%.o,$(CPP_FILES))
9
10
11 # Regra para criar o diretório de objetos
12 # SEMPRE QUE FOR ADICIONAR MAIS UM INCLUDE LISTAR AQUI
13 # EXEMPLO:
14 # $(OBJ_DIR)/%.o: src/%.cpp
15 #     g++ -c -Iinclude \
16 #     -ISFML/include \
17 #     -Iinclude/Entidades \
18 #     -Iinclude/Entidades/Personagens \
19 #     -Iinclude/Entidades/Inimigos \
20 #     -Iinclude/Entidades/Projetoil \
21 #     -Iinclude/Entidades/Obstaculos \
22 #     $< -o $@
23
24
25 # Compilação dos arquivos .cpp para arquivos .o
26 $(OBJ_DIR)/%.o: src/%.cpp
27     g++ -c -Iinclude \
28     -ISFML/include \
29     $< -o $@
30
31 # Regra padrão para construir o executável
```

PROBLEMS PORTS OUTPUT DEBUG CONSOLE **TERMINAL** COMMENTS

```
JaZ@DESKTOP-F700HVP MINGW32 ~/Desktop/modelo-sfml-windows
$ mingw32-make -f Makefile
rm -rf obj/*.o main
```

Com isso deve ser o suficiente. Se precisar de algo só perguntar.