ONLINE TIC-TAC-TOE

Final Project- ECE 4122

Abstract

This document is a reference on how to setup and run this project.
https://github.com/sahmed85/Online-Tic-Tac-Toe-Game

Ahmed, Shadman
sahmed85@gatech.edu

Contents

0 Introduction

This documentation is a reference on how to setup and run my final project for ECE 4122 (Fall 2020) on Microsoft Visual Studio Code for **Windows**. This documentation covers how to install needed libraries and run the code to verify its functionality.

This project is an online Tic-Tac-Toe game with server and client executables. This project creates a server that can handle multiple matchmaking sessions and spawn players into matches. This project also incorporates a client side that connects to the server, waits for matchmaking to complete, and spawn the graphics for playing the game. Gameplay graphics utilizes SFML library (which uses OpenGL) to handle window, keyboard, mouse clicks and graphics on the tic-tac-toe board. Communication between players is handled by the server (as shown in the Figure 1) and utilizes UDP to communicate across client-server-client.

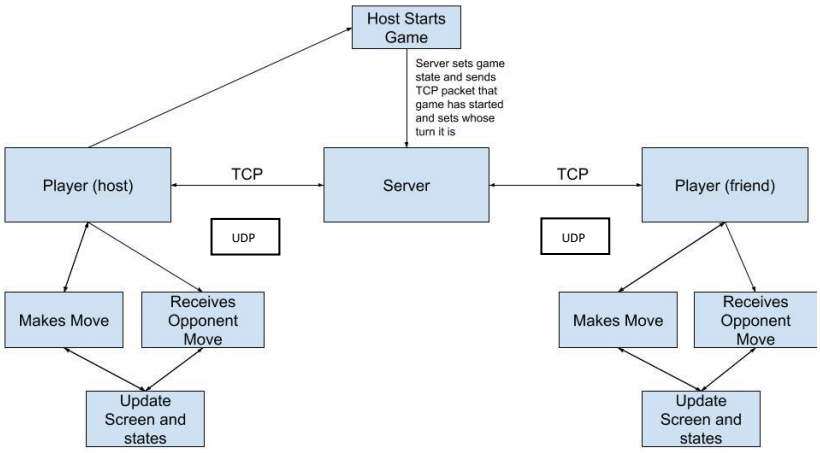

Figure 1. Shows how the player communication is handled by the server.

Communication on UDP sockets are standardized to a user-defined struct as shown below:

```
struct udpMessage {
    unsigned short nVersion = 1;
    unsigned short messageType;
    unsigned short messageLength;
    unsigned int command_pos;
    std::string message;
};
```

Varying message types is used to send/receive messages between clients and servers. Listed below is the meaning of different message types and its meaning:

- All message types will be the same version:
    - nVersion = 1

- Client (only the client will send these, and server will interpret these):
    - Matchmaking init packet:
        - messageType: 0
        - command_pos: -1
        - message: "NA"
    - Sending a move to server:
        - messageType: 1
        - command_pos: send move location on grid
        - message: 'X' or 'O'
- Server (only the server will send these, and client will interpret these):
    - Game_init packet
        - messageType: 3
        - command_pos: -1
        - message: "Player1" or "Player2"
    - Update board packet:
        - messageType: 4
        - command_pos: send move location on grid
        - message: 'X' or 'O'
    - Game won packet to appropriate player
        - messageType: 5
        - command_pos: -1
        - message: Winner
    - Game lost packet to appropriate player
        - messageType: 6
        - command_pos: -1
        - message: Loser
    - Game tied to both players
        - messageType: 7
        - command_pos: -1
        - message: Tied

1 Installing SFML and Setup in Visual Studio 2019 for Client Code

Below is a general way to setup SFML for any project. **Credit for this section.*

- Download the SFML library at this site: https://www.sfml-dev.org
- Extract to C:\SFML or where convenient (remember this path)
- Create an empty VS project, and name it HelloSFML (as an example).
- From the Visual Studio main menu select Project | HelloSFML properties.(or right click on the project name in the Solution Explorer and choose “properties” from the context menu.) In the resulting HelloSFML Property Pages window take the following steps which are numbered and can be referred to in the next image.

    1. Make sure you choose “All Configurations” in the box on the upper left.
    2. Choose the “General” option under C/C++ in the panel on the left.
    3. Add the directory where the SFML/include files are in the “Additional Include Directories” field on the upper right.

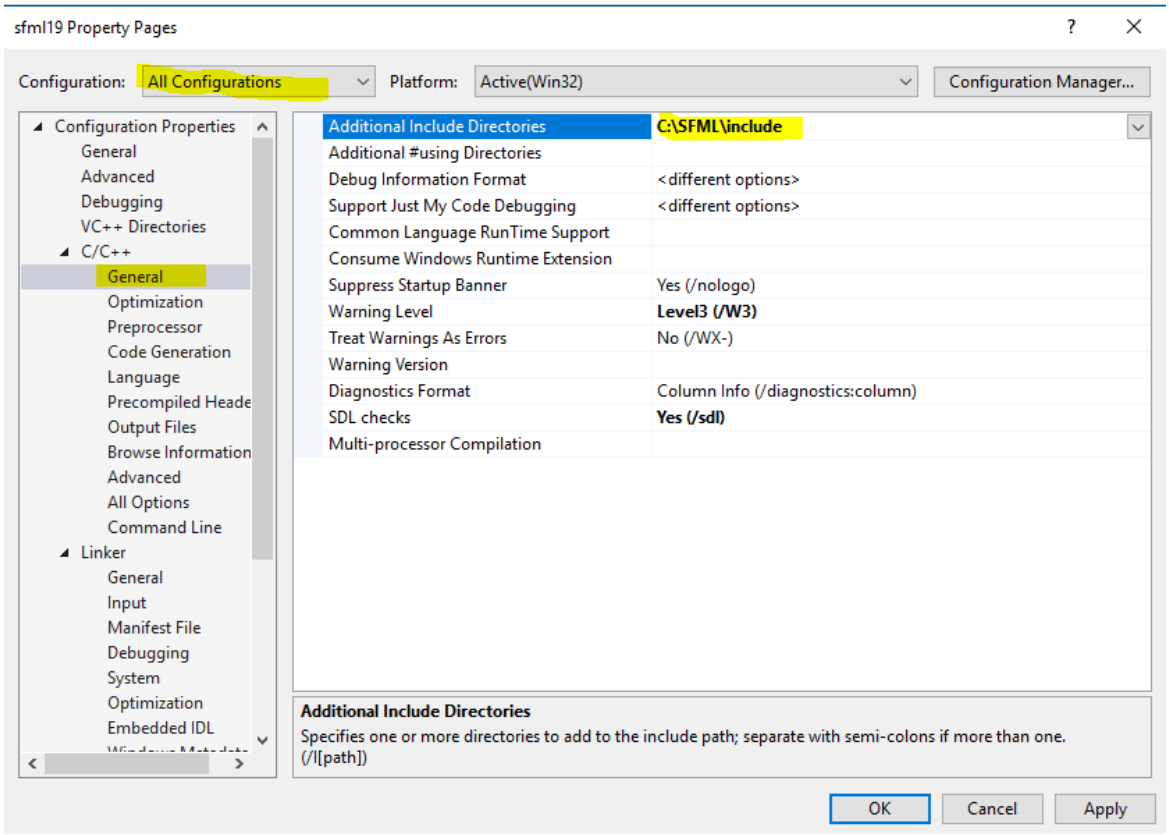

Figure 2. Shows where to add additional directories for all configurations.

In the same window perform these next numbered steps which refer to the next figure.

1. Select Linker then General.
2. Find the Additional Library Directories edit box and type the drive letter where your SFML folder is followed by \SFML\lib. So the full path to type if you located your SFML folder on your C drive is, as shown in the screen-shot is C:\SFML\lib.

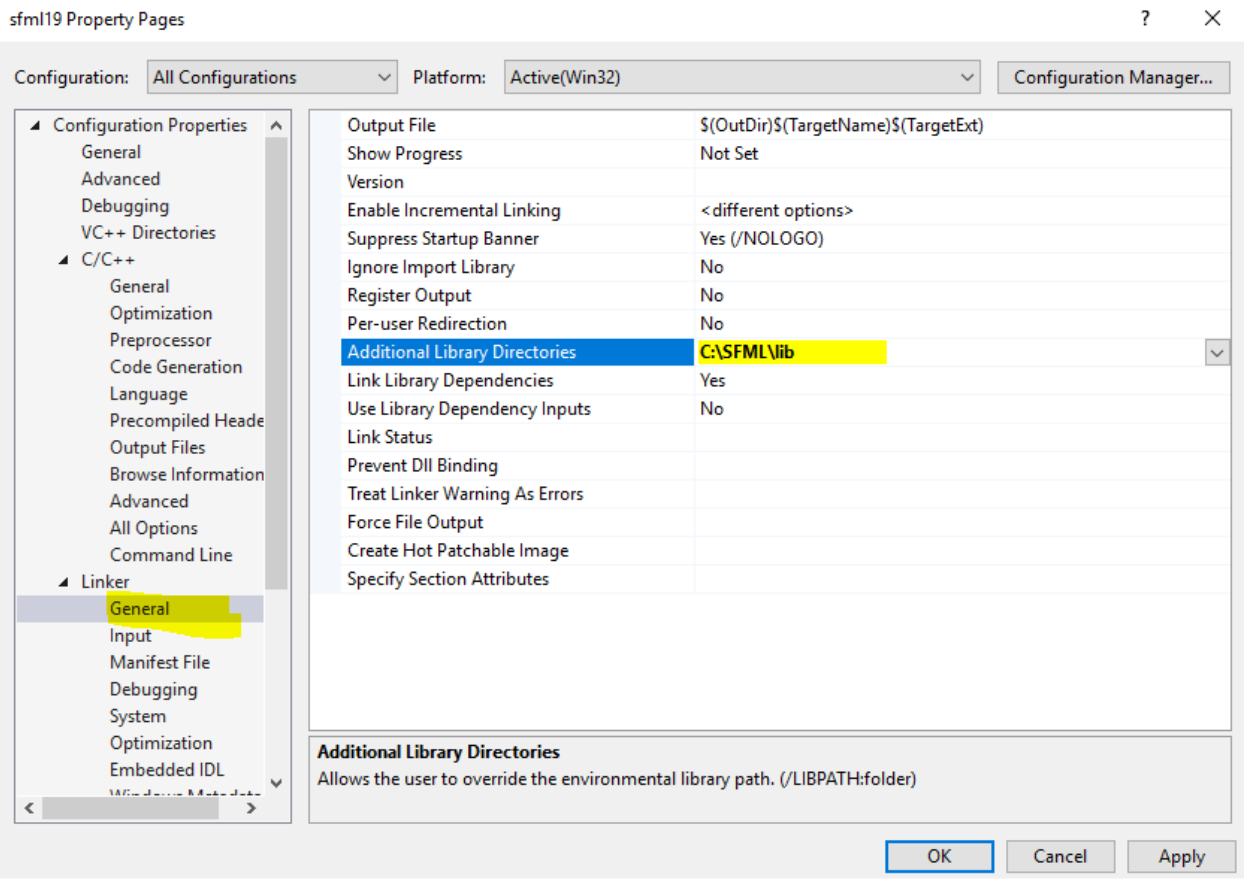

Figure 3. Shows additional library directories for linker properties (All Configs).

Finally, for this stage, still in the same “Properties” window, perform these numbered steps which refer to the next figure.

1. Switch the Configuration: drop down to Debug as we will be running and testing our games in debug mode for now.
2. Select Linker then Input.
3. Find the Additional Dependencies edit box and click it at the far left. Now copy & paste the following: sfml-graphics-d.lib;sfml-window-d.lib;sfml-system-d.lib;sfml-network-d.lib;sfml-audio-d.lib; at the beginning of that field (at right, the edit window was used). Be **REALLY** careful to place these dependencies first in the list and not to overwrite any of the text that is already there.
4. Click OK.

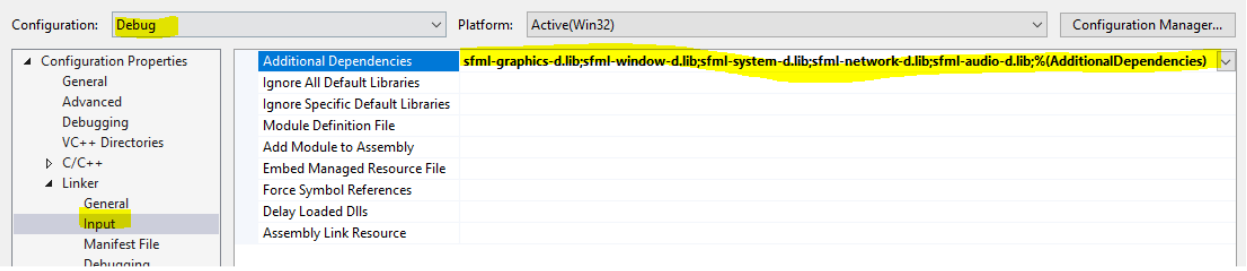

Figure 4. Shows Linker additional dependencies for debug configuration.

- The only other step is to copy the .dll files into the project folder which we will step through now.
- My project folder is C:\Visual Studio Stuff\Fall20\sfml19. The files we need to copy into there are located at C:\SFML\bin. Of course, if you installed Visual Studio and the Visual Studio Stuff folder on a different drive then replace C: from the previous paths with your appropriate drive letter. Open a window for each location and highlight the required files as shown in the next screenshot.
- Copy & paste the files in the YOUR_DRIVE:\SFML\bin to YOUR_DRIVE:\Visual Studio Stuff\Projects\HelloSFML\sfml19.
- This complete the setup for client in VS code. Lastly, insert the client code into this project. You can use the file from Canvas or pull it from https://github.com/sahmed85/Online-Tic-Tac-Toe-Game.

2 Setting up Visual Studio Code 2019 for Server Code

- Create an empty VS project under the same solution as the client code. This allows me to easily spawn debug instances of the server and client code.
- Server code does not require any additional libraries.
- This complete the setup for server in VS code. Lastly, insert the server code into this project. You can use the file from Canvas or pull it from https://github.com/sahmed85/Online-Tic-Tac-Toe-Game.

3 Running the code

- The order of code being run is important.
- The server needs to be run first then any number of clients to join a game lobby and play the game.
- For best results on localhost, run one instance of the server and one game (two client instances).
- This code is functional under Debug mode and has not been tested for release mode.

4 Future Fixes

I wanted to take some time to explain fixes that I would do if I had more time:

- Utilize TCP for matchmaking request and then UDP for all other message types.
- More graceful exit for player window:
    - Currently an error is thrown once the client game is exited. I wish I could spend more time debugging this issue (however, server is still listening for matchmaking).
- Enable a restart feature, to re-enter matchmaking or replay the person that you just played.
- Optimize a few of the helper functions.