

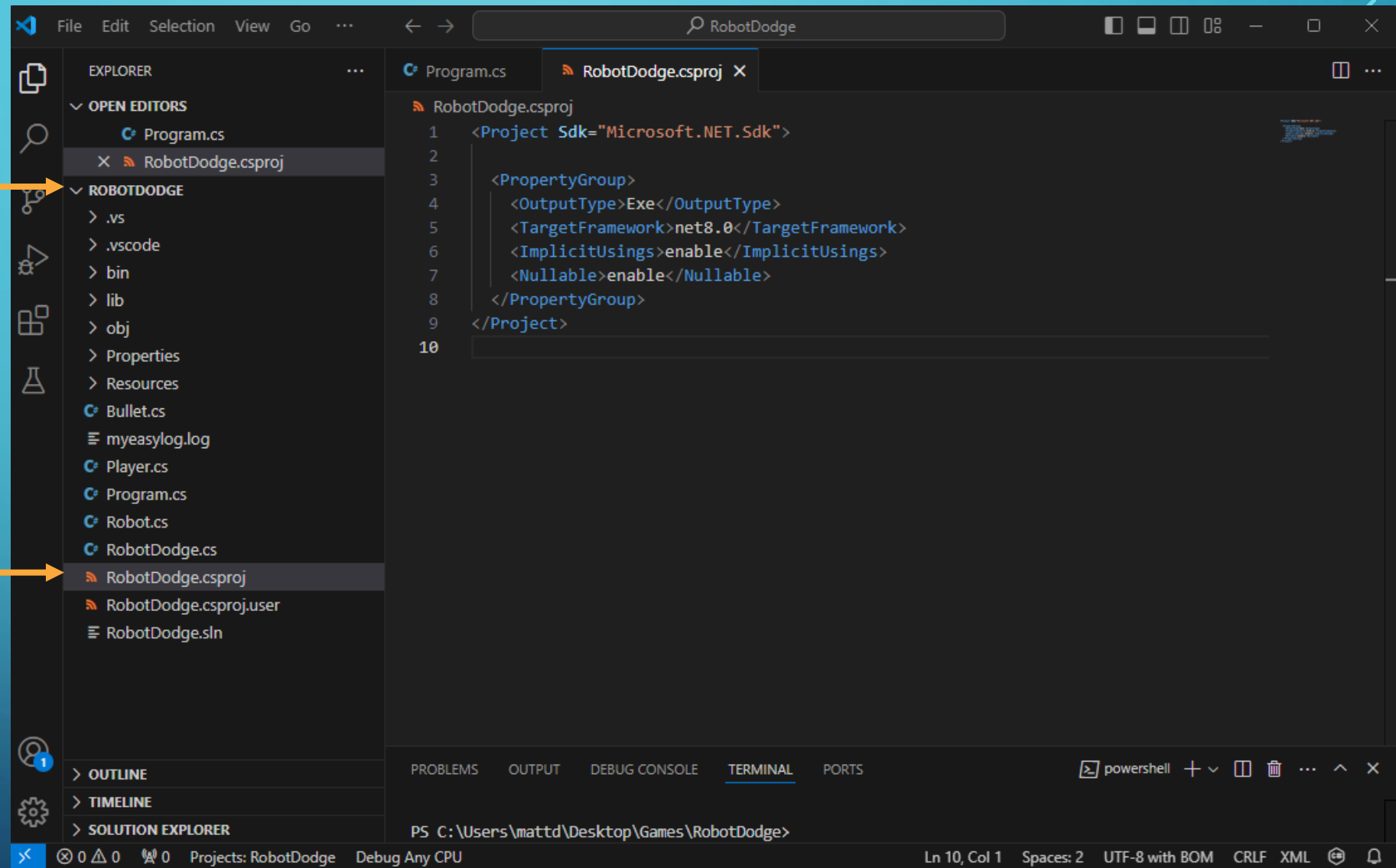


CREATING A SINGLE EXECUTABLE FILE (.EXE) USING VS CODE

Note: If resources are used in the code (i.e. .png files), this does not work. (I have not yet figured out how to accommodate this).

- Open Visual Studio Code and navigate to the folder of the program you are working on.

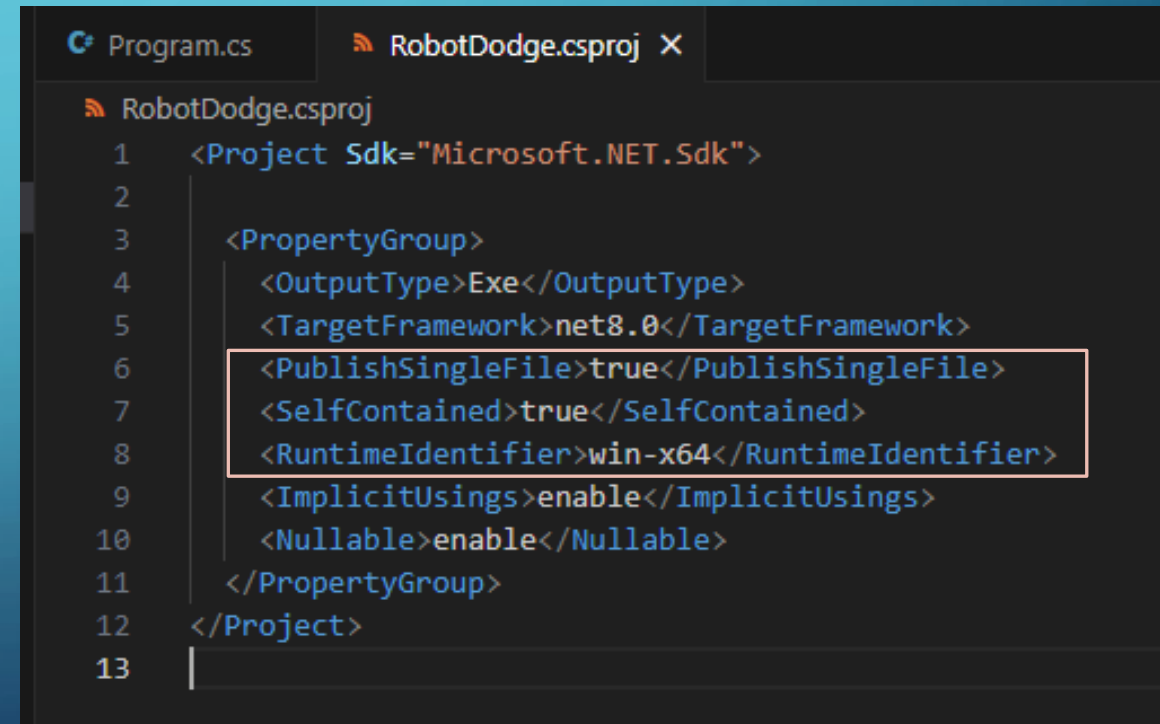
- Open the .csproj file



- Enter the following lines of code to the .csproj file:

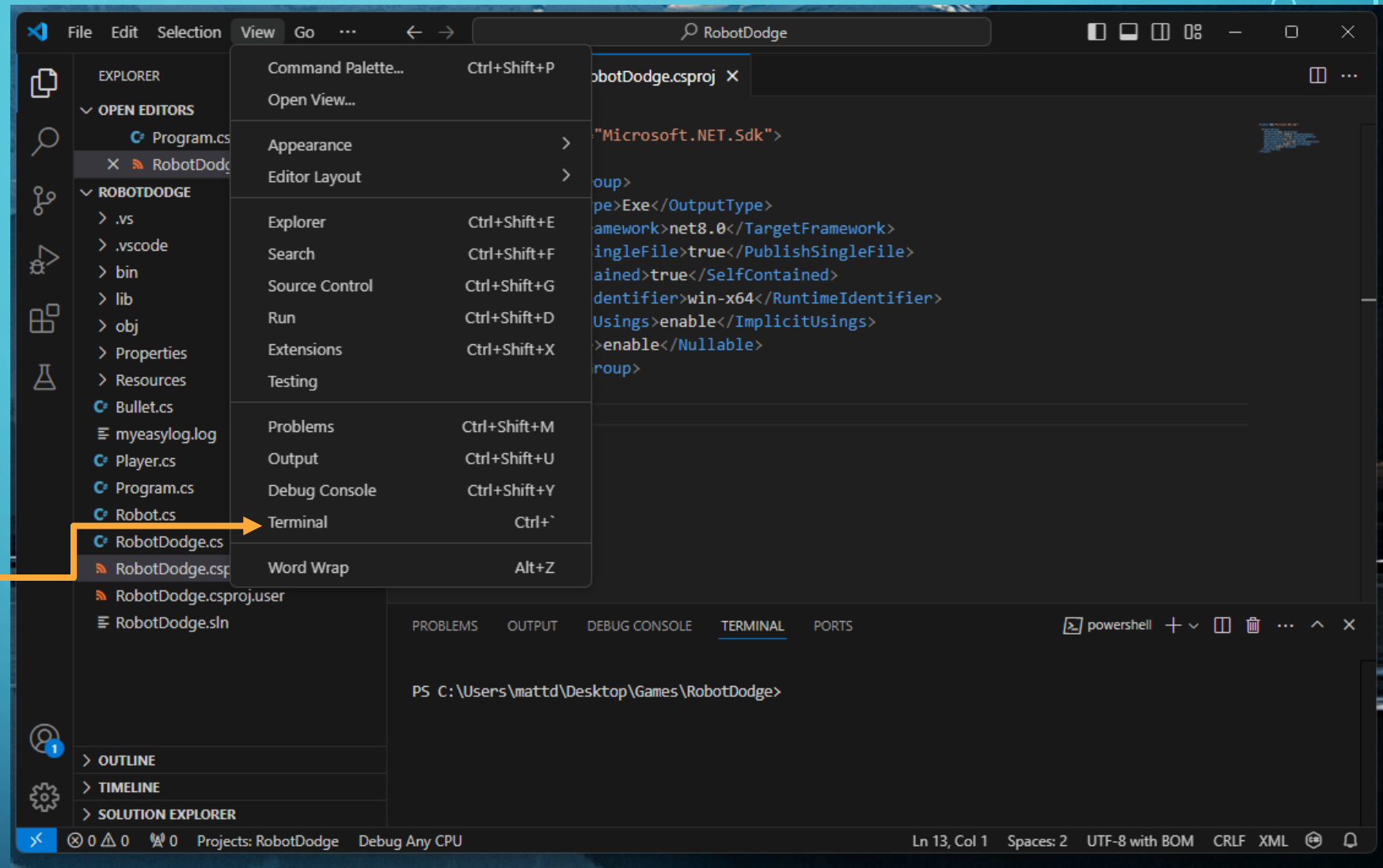
```
<PublishSingleFile>true</PublishSingleFile>  
<SelfContained>true</SelfContained>  
<RuntimeIdentifier>win-x64</RuntimeIdentifier>
```

It should look like this >>>>



```
Program.cs  RobotDodge.csproj X  
RobotDodge.csproj  
1  <Project Sdk="Microsoft.NET.Sdk">  
2  
3      <PropertyGroup>  
4          <OutputType>Exe</OutputType>  
5          <TargetFramework>net8.0</TargetFramework>  
6          <PublishSingleFile>true</PublishSingleFile>  
7          <SelfContained>true</SelfContained>  
8          <RuntimeIdentifier>win-x64</RuntimeIdentifier>  
9          <ImplicitUsings>enable</ImplicitUsings>  
10         <Nullable>enable</Nullable>  
11     </PropertyGroup>  
12 </Project>  
13
```

Open the VS Code terminal, by either selecting 'Terminal' from the 'View' drop down menu, or by using the shortcut (Ctrl + `)



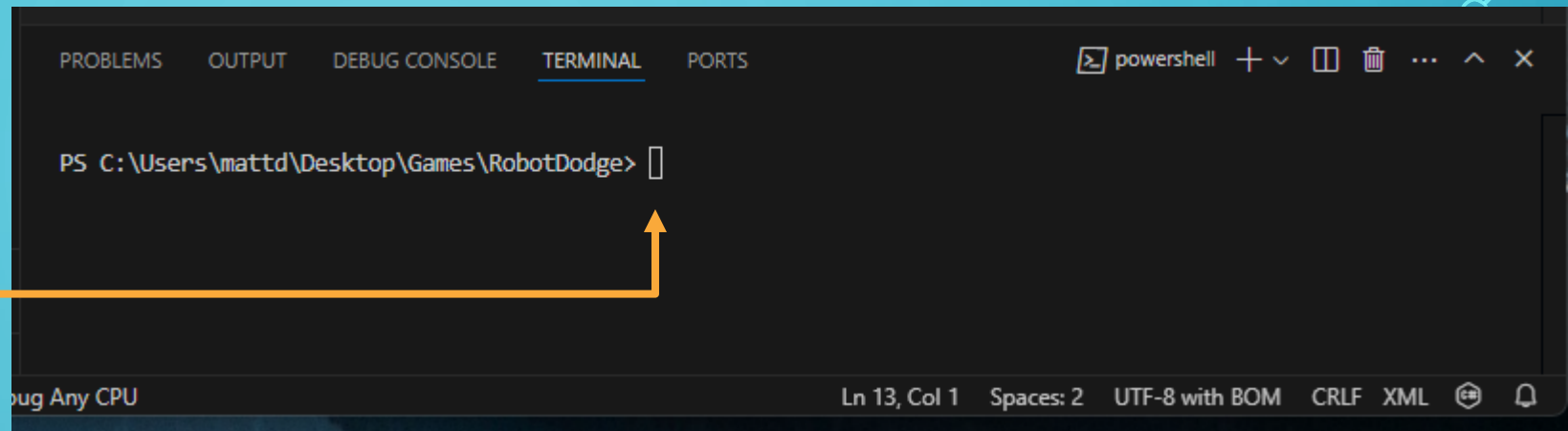
In the terminal,

Enter the following:

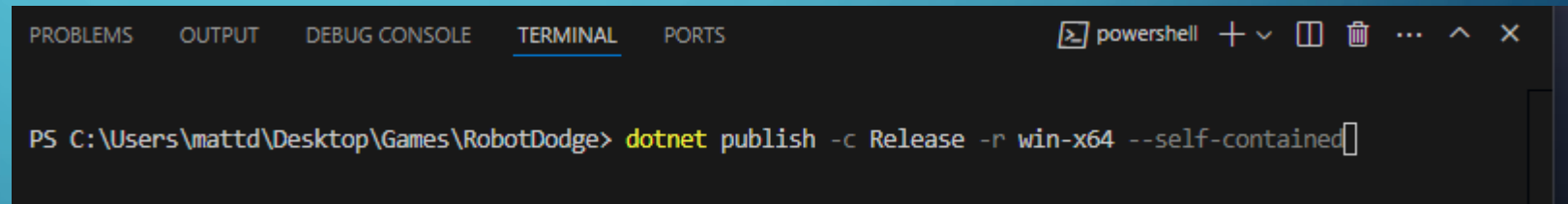
```
dotnet publish -c Release -r win-x64 --self-contained
```

It should look like this >>>>

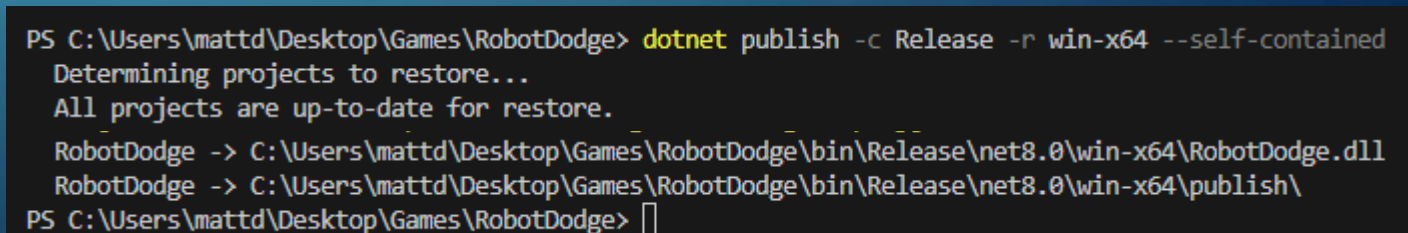
When execution has been successfully completed, you should see something similar to this.



A screenshot of a Visual Studio terminal window. The title bar shows 'powershell' and standard window controls. The terminal has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (selected), and 'PORTS'. The prompt is 'PS C:\Users\mattd\Desktop\Games\RobotDodge>' with a cursor. An orange arrow points from the text 'In the terminal,' to the prompt.



A screenshot of a Visual Studio terminal window, similar to the one above. The prompt is 'PS C:\Users\mattd\Desktop\Games\RobotDodge>' and the command 'dotnet publish -c Release -r win-x64 --self-contained' is being entered. An orange arrow points from the text 'It should look like this >>>>' to the command.



A screenshot of a Visual Studio terminal window showing the output of the 'dotnet publish' command. The output indicates that projects are up-to-date and lists the paths for the published files. An orange arrow points from the text 'When execution has been successfully completed, you should see something similar to this.' to the output.


```
PS C:\Users\mattd\Desktop\Games\RobotDodge> dotnet publish -c Release -r win-x64 --self-contained
Determining projects to restore...
All projects are up-to-date for restore.
RobotDodge -> C:\Users\mattd\Desktop\Games\RobotDodge\bin\Release\net8.0\win-x64\RobotDodge.dll
RobotDodge -> C:\Users\mattd\Desktop\Games\RobotDodge\bin\Release\net8.0\win-x64\publish\
PS C:\Users\mattd\Desktop\Games\RobotDodge>
```



Follow the folder structure as shown here.
(Note that the first few folders will be different for each of your projects, depending on where you save them).

Desktop > Games > RobotDodge > bin > Release > net8.0 > win-x64 > publish

Within that folder, there should now be two files as shown.
The application type file is an .exe file and can be shared with other people.

These files will run independently of vs code or a terminal.



Name	Date modified	Type	Size
 RobotDodge	6/09/2024 3:01 PM	Application	66,226 KB
 RobotDodge.pdb	6/09/2024 3:01 PM	Program Debug D...	115 KB