NetRumble Demo Game

with PlayFab Party and Multiplayer

(GDK UWP with Xbox Live login variant)

*This sample has been developed with Microsoft Visual Studio 2019, Microsoft GDK and PlayFab SDKs for Microsoft GDK*

# Description

This is a simple multiplayer game. It is expected that developers should be familiar with the basics of using PlayFab C++ SDKs and PlayFab API as well as creating and configuring Xbox Live game titles. They also need to create a PlayFab game title for this project in PlayFab developer portal, and know its Title ID. Additionally, they also need to create an Xbox Live game title for this project in Xbox Partner Dev Center, and know its configuration parameters: XBL Title ID, StoreID, SCID, Package Identity Name, AppID, etc. They will need to configure their Xbox Live game title to use a Sandbox accessible by the Xbox user accounts they intend to use with this app. This sample app is intended to demonstrate how developers can use PlayFab Party + Multiplayer C++ API and Microsoft GDK API to perform the following functions:

* Login to the PlayFab server using a Custom ID
  + Call PlayFab API LoginWithCustomID()
* Login to Xbox Live using an Xbox user account
  + Call Playfab API LoginWithXbox()
  + Login to the PlayFab server using Xbox Live authentication token
  + Get the player's name and player ID of the Xbox Live user
* Create a game lobby
* Find and join an existing game lobby
* Customize lobby attributes
* Use matchmaking to join arranged lobbies
* Create a Party network, have players join the Party network for message/data exchange and voice chat using PlayFab Party API
* Implement a multiplayer game

Dependencies

This example depends on the PlayFab C++ SDKs and Microsoft GDK. Developers need to download and install them to satisfy the SDK dependencies required by the example.

**Developers need to download and install the following SDK**

* PlayFab Party C++ SDK is installed with GDK, and it will be used by NetRumble game project automatically when GDK is installed.
* PlayFab Multiplayer C++ SDK download link: [PlayFab/PlayFabMultiplayer: PlayFab Multiplayer C++ SDK (github.com)](https://github.com/PlayFab/PlayFabMultiplayer) (release v1.1.0 **for GDK** was used at the moment of writing this document)
* PlayFab XPlatCppSdk (pre-installed with the game source code)
  + Its source code from GitHub was used at the moment of writing this document:   
    <https://github.com/PlayFab/XPlatCppSdk/releases/tag/3.73.220406>

If you need to use a more recent version, please follow the steps:

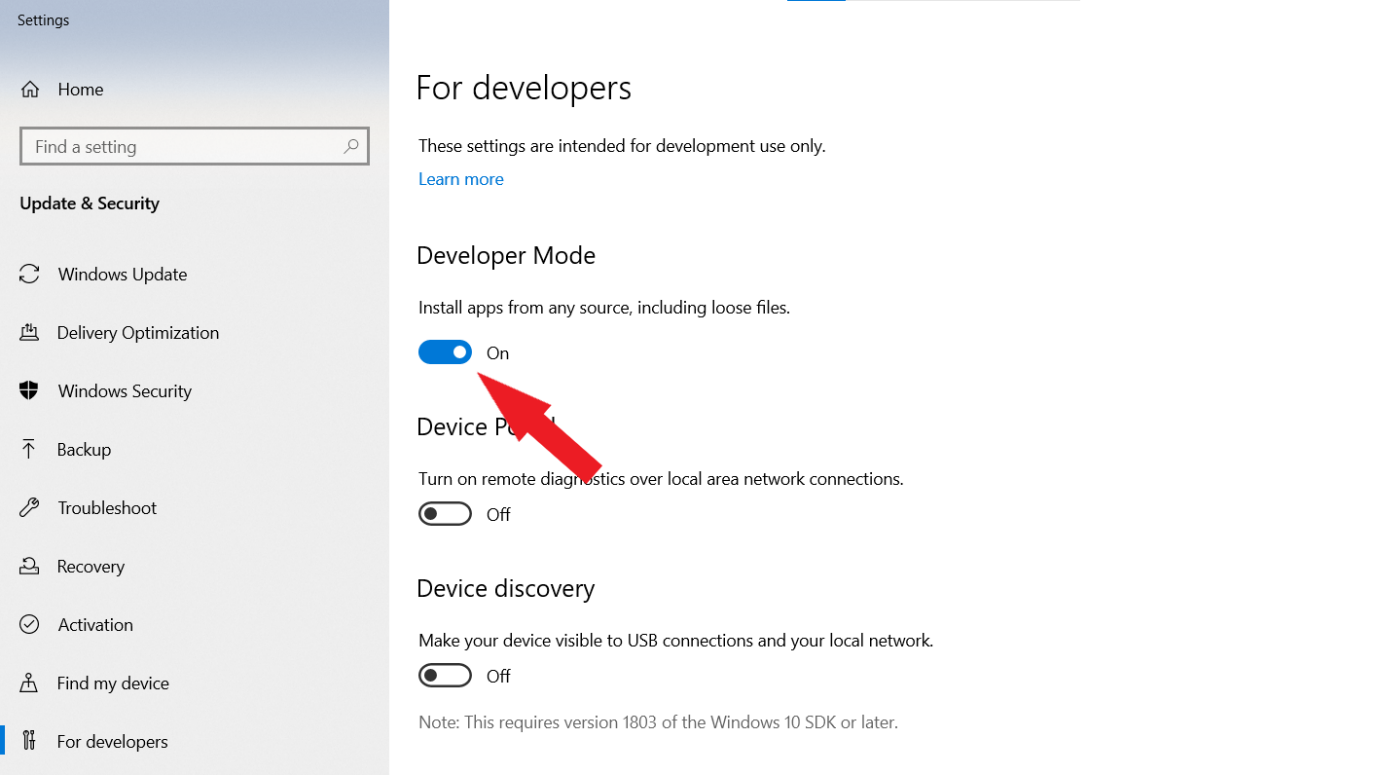
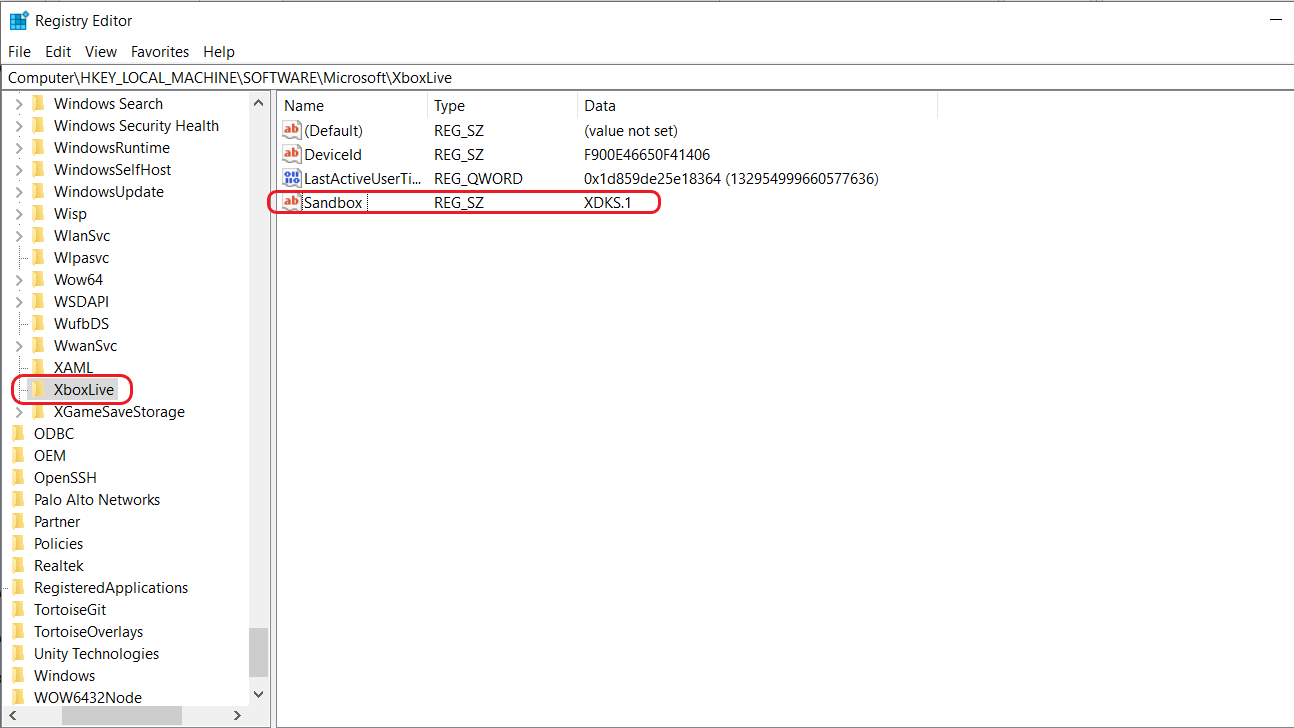
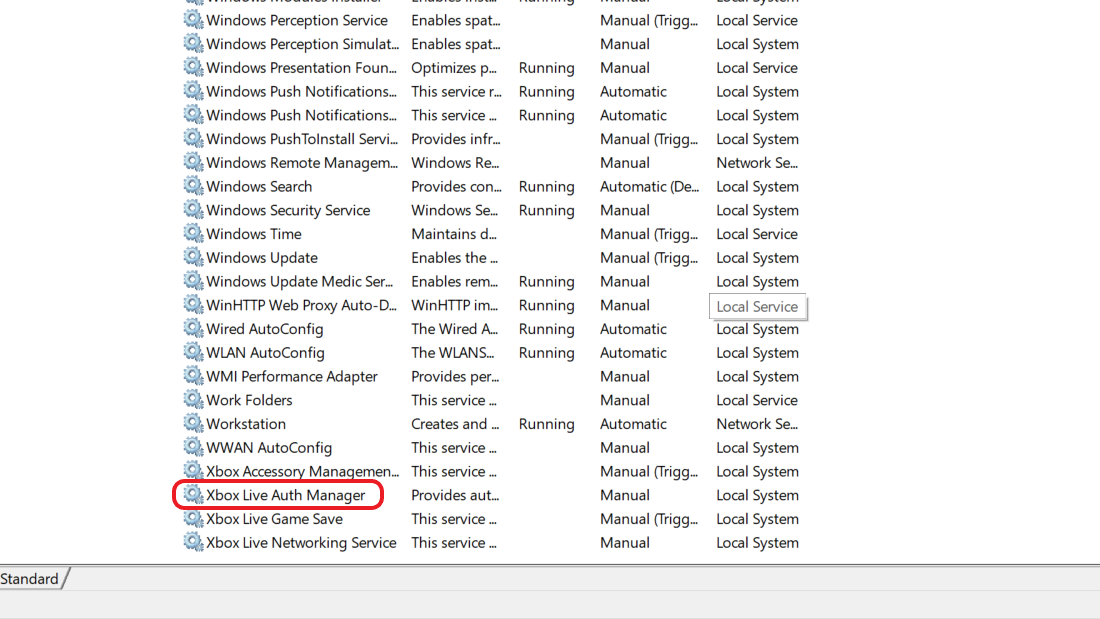
* + - Note that XPlatCppSdk relies on submodule external\jsoncpp, make sure to pull it recursively before building
    - Build Debug and Release version. This will generate Debug and Release lib files in *\XPlatCppSdk\build\Windows\x64.*
    - Compiled binaries and the headers will be in these directories:
      * \XPlatCppSdk\build\Windows\x64\Debug(Release)\lib\_json.lib
      * \XPlatCppSdk\build\Windows\x64\Debug(Release)\XPlatCppWindows\XPlatCppWindows.lib
      * .\XPlatCppSdk\code\include\playfab
      * .\XPlatCppSdk\external\jsoncpp\include\json
    - Please place them in the following project directories:
      * Put *lib\_json.lib, XPlatCppWindows.lib* in *\GDKSamples\Live\NetRumble\Dependencies\XPlatCppSdk\lib\x64\Debug(Release)*
      * Put *.\XPlatCppSdk\code\include\playfab* and *.\XPlatCppSdk\external\jsoncpp\include\json* in *.\GDKSamples\Live\NetRumble\Dependencies\XPlatCppSdk\include*
* Microsoft DirectXTK library (pre-installed with the game source code)
  + Its source code is available on GitHub if you need to use a more recent version:  
    <https://github.com/microsoft/DirectXTK>
* Open file GDKSamples/Live/NetRumble/Common/ServerConfig.h and modify the value for NETRUMBLE\_PLAYFAB\_TITLE\_ID to set your PlayFab Title ID.

Project configuration

#### Download, install and configure Microsoft GDK

Download a recent version of Microsoft GDK (for example, **2021.04 – 10.0.19041.6078** at the moment of writing this document) from Microsoft GDK portal ([GDK (microsoft.com)](https://www.microsoft.com/en-us/software-download/gdk)). Install the downloaded GDK. Make sure to select all optional components during installation. It will install the GDK and necessary VS build tools.

Before launching and testing the application, you need to configure your PC environment:

1. Make sure your Windows PC is set to Developer Mode:  
     
   
2. Make sure your Windows PC is set to the right Xbox Live Sandbox depending on the configuration of your Xbox Live game title. For example, let’s assume it is “**XDKS.1**” just for reference in the following steps:
   1. Create/Modify the registry setting Computer\HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\XboxLive\Sandbox if needed (string value):  
        
      
   2. You may restart this Windows service after updating Sandbox to make it take effect (an alternative option is to reboot your PC):  
        
      

The remaining subsections in this chapter describe how to configure a new project, they are provided for reference only. The NetRumble source code comes already pre-configured with necessary build settings and library paths. It needs only user’s custom application attributes and necessary dependencies installed. You may skip these subsections and proceed to **“Configure MicrosoftGame.config”** chapter below.

#### Download and configure dependencies on DirectX

* **Download and install DirectX**

You can click here ([Download DirectX Software Development Kit from Official Microsoft Download Center](https://www.microsoft.com/en-us/download/details.aspx?id=6812)) to download DirectX. Next, follow the default prompts to install DirectX

* **Configure DirectX in the project**

Add the path*: Properties -> Configuration Properties -> C/C++ -> Additional Include Directories* in the project:

…\DirectXTK12\Inc

Add the path*: Properties -> Configuration Properties -> Link -> General -> Additional Library Directories* in the project:

…\DirectXTK12\x64

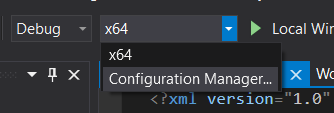
Add the path: *Properties -> Configuration Properties -> Link -> Input -> Additional Dependencies* in the project:

DirectXTK12d.lib

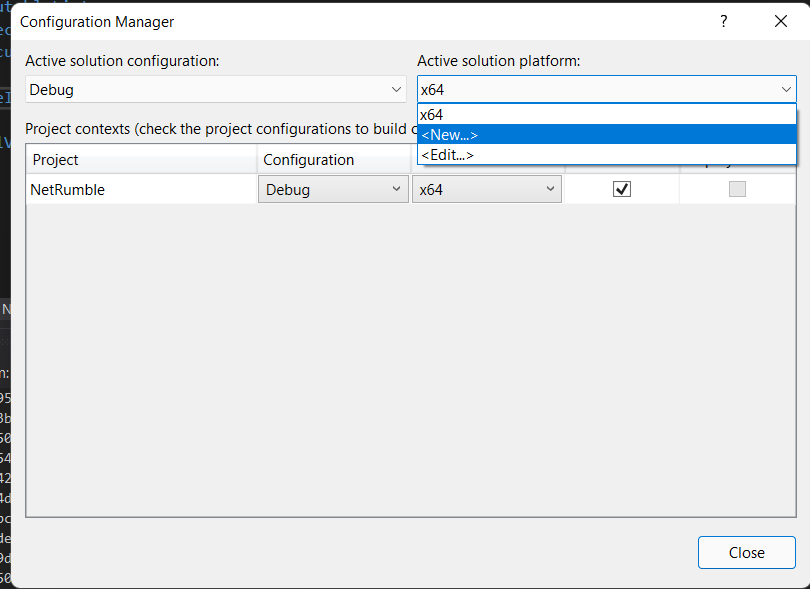
#### Configure the GDK game project

* **Create the *Gaming.Desktop.x64* solution platform**

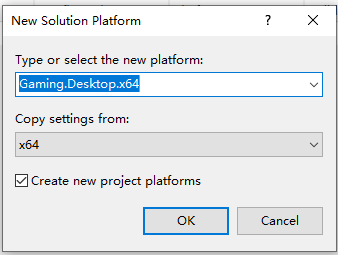
1. Click *Solution Platform*, and select *Configuration Manager...*



1. Click *<New...>* from *Active solution platform*

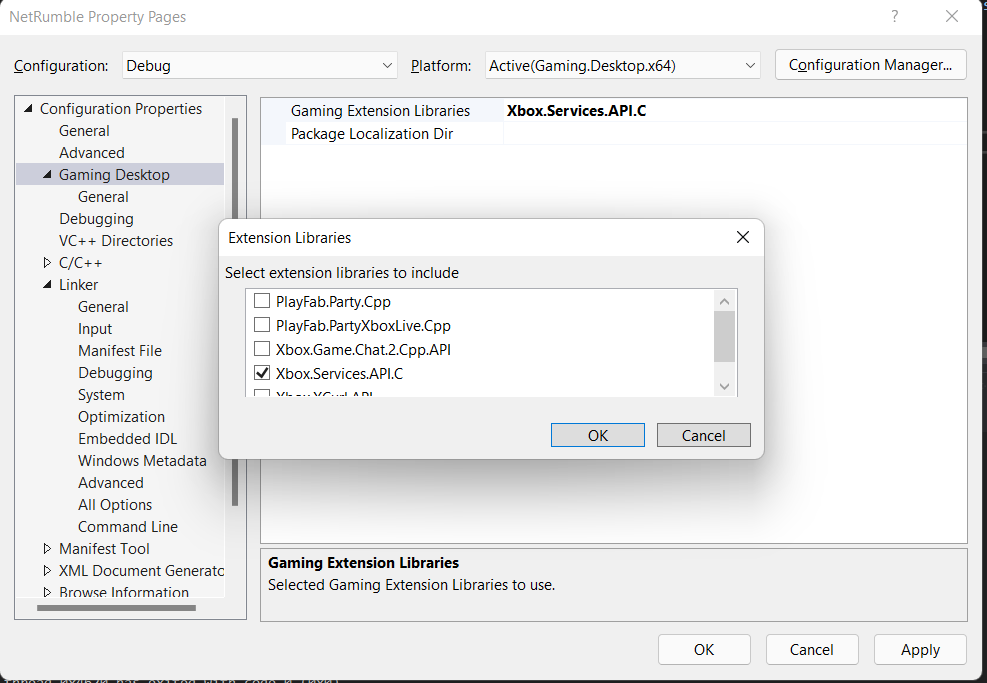


1. *Type or select the new platform* select *Gaming.Desktop.x64*. And *Copy settings from* select *x64*



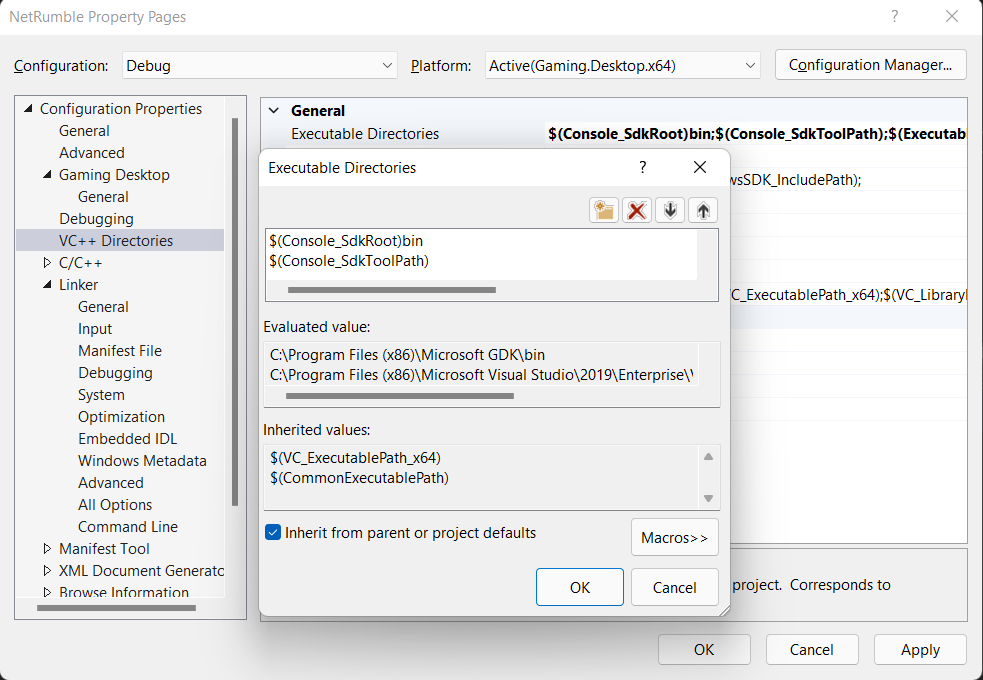
* **Add Gaming Extension Libraries**

Properties->Gaming Desktop->Gaming Extension Libraries select *Xbox.Services.API.C*

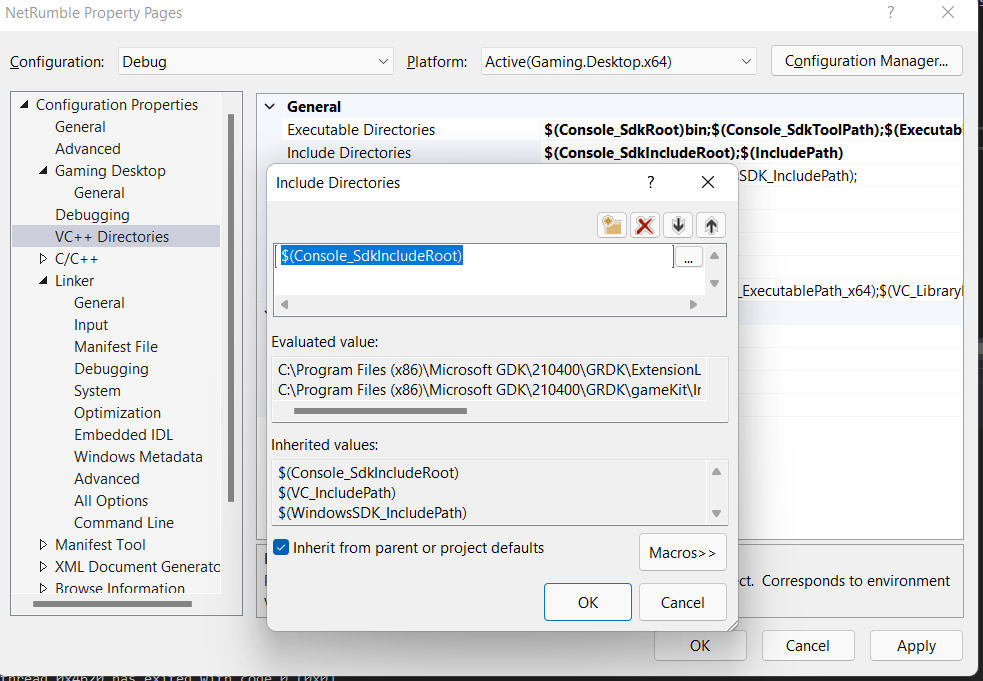


* **Configure environment variables**

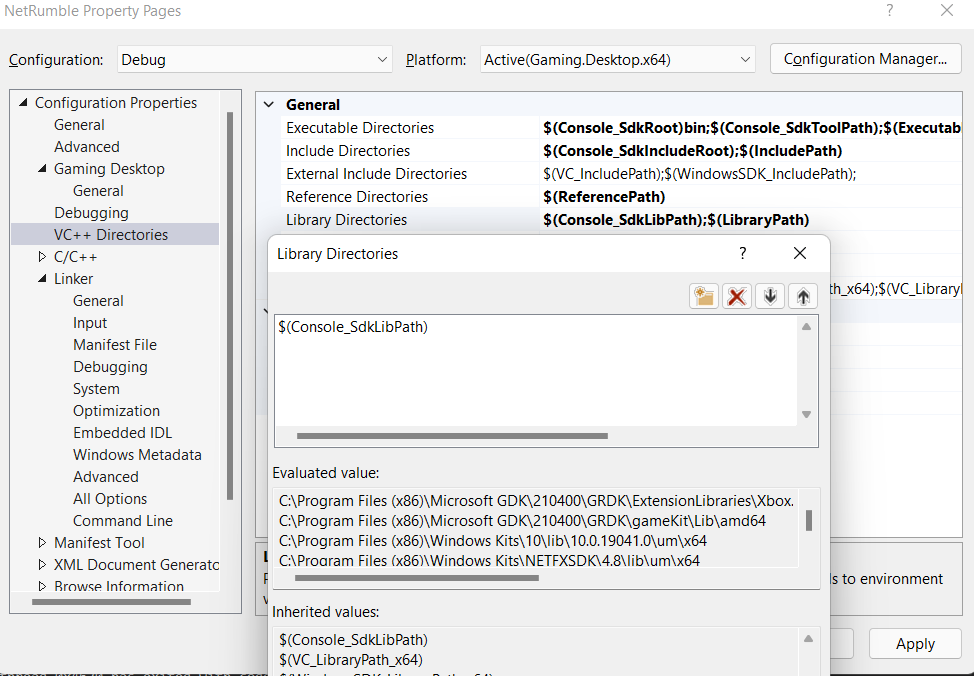
1. Properties à VC++ Directories à General à Executable Directories à Edit à New Line, add $(Console\_SdkRoot)bin and *$(Console\_SdkToolPath)*



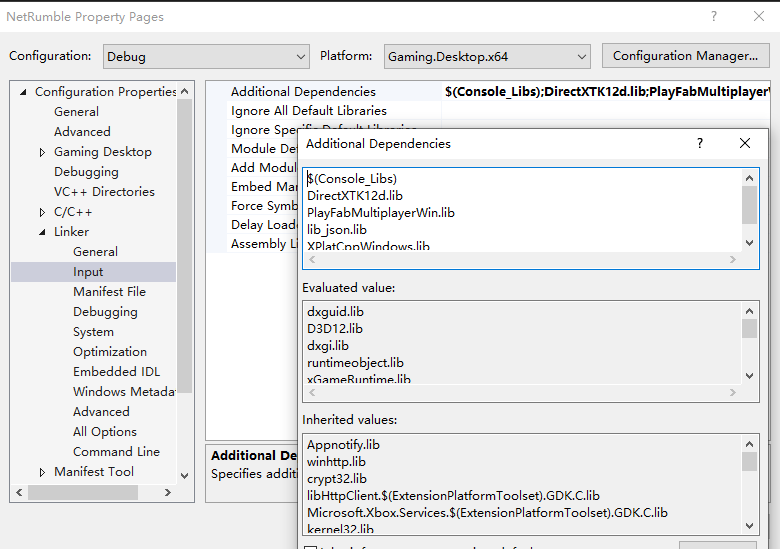
1. Properties à VC++ Directories à General à Include Directories à Edit à New Line add *$(Console\_SdkIncludeRoot)*



1. Properties à VC++ Directories à General à Library Directories à Edit à New Line add *$(Console\_SdkLibPath)*

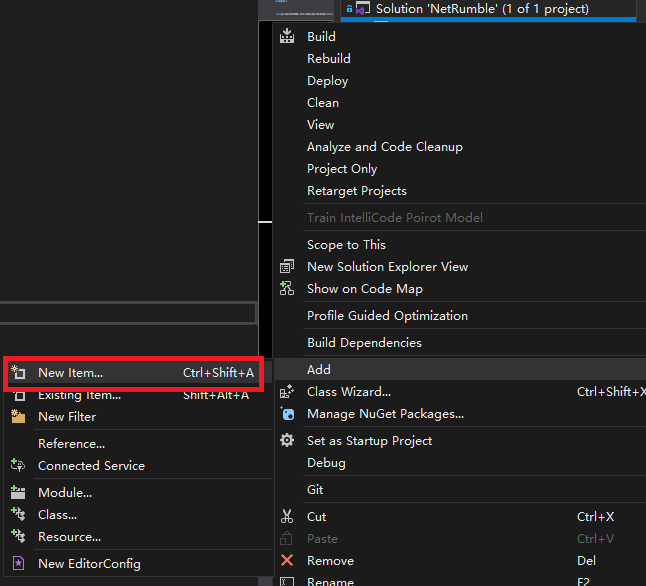


1. Properties à Linker à Input à Additional Dependencies à Edit à New Line add *$(Console\_Libs)*

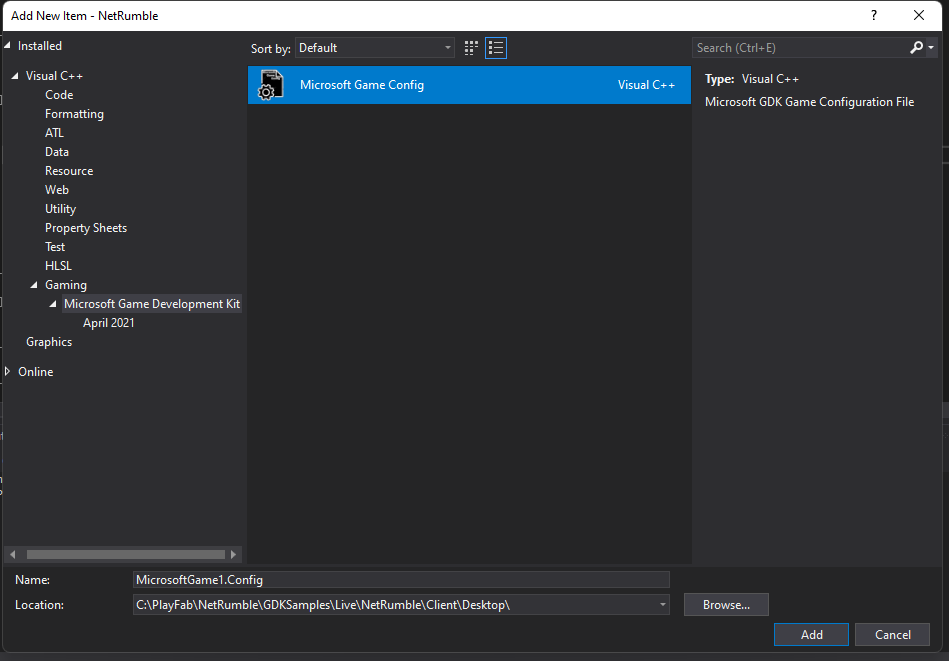


* **Add MicrosoftGame.Config**

1. Right-click the Solution à Add à New Item…

****

1. Select Gaming à Microsoft Game Development Kit à Microsoft Game Config.



1. For *Identity Name, Publisher* and *TitleId*, please refer to configuration of your Xbox Live game, these parameters can be found in your **Partner Center**.

Configure MicrosoftGame.config

The file GDKSamples/Live/NetRumble/Client/Desktop/MicrosoftGame.Config needs to be updated with configuration information specific to your Xbox Live game title from Xbox Partner Dev Center before building, installing, and running the app.

It is expected that a developer should be familiar with the process of building and configuring applications for Microsoft Game Core (GDK). Please refer to corresponding documentation or sample code as necessary.

On the screenshot below blue stickers indicate the data that needs to be provided. The picture below may be a little different than the content of provided MicrosoftGame.Config file. You may specify only the values used in the provided file; the other parameters referenced in the picture are not necessary.

A picture containing graphical user interface

Description automatically generated

Identity Name: Package Identity Name from Xbox Live configuration.

Identity Publisher: Package Identity Publisher from Xbox Live configuration.

StoreId, TitleId and MSAAppId are parameters from Xbox Live configuration.

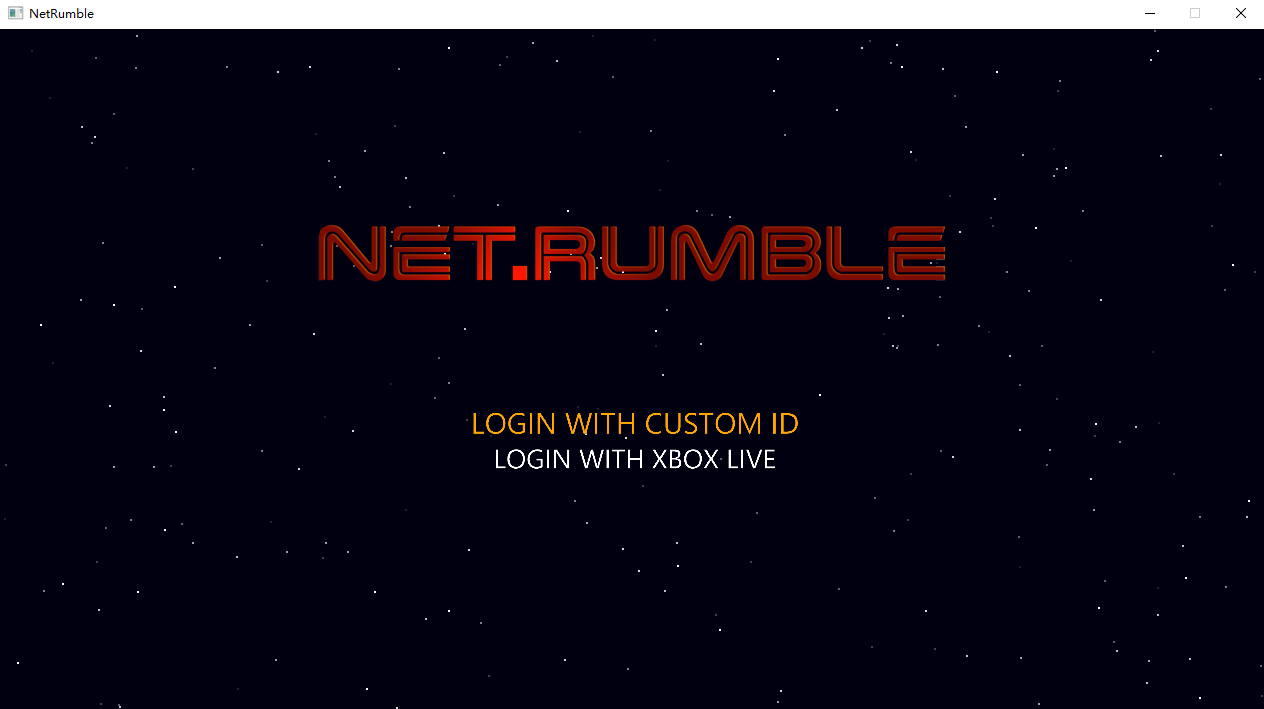
Executable Name and DefaultDisplayName should typically be the same, this is the application name and the name of game’s executable file (without the extension).

PublisherDisplayName: Publisher Display Name from Xbox Live configuration.

Please, also remember to specify a required value in SandboxIds, it should match the Sandbox(es) in Xbox Live configuration.

# Using the sample

**Start Menu Screen**

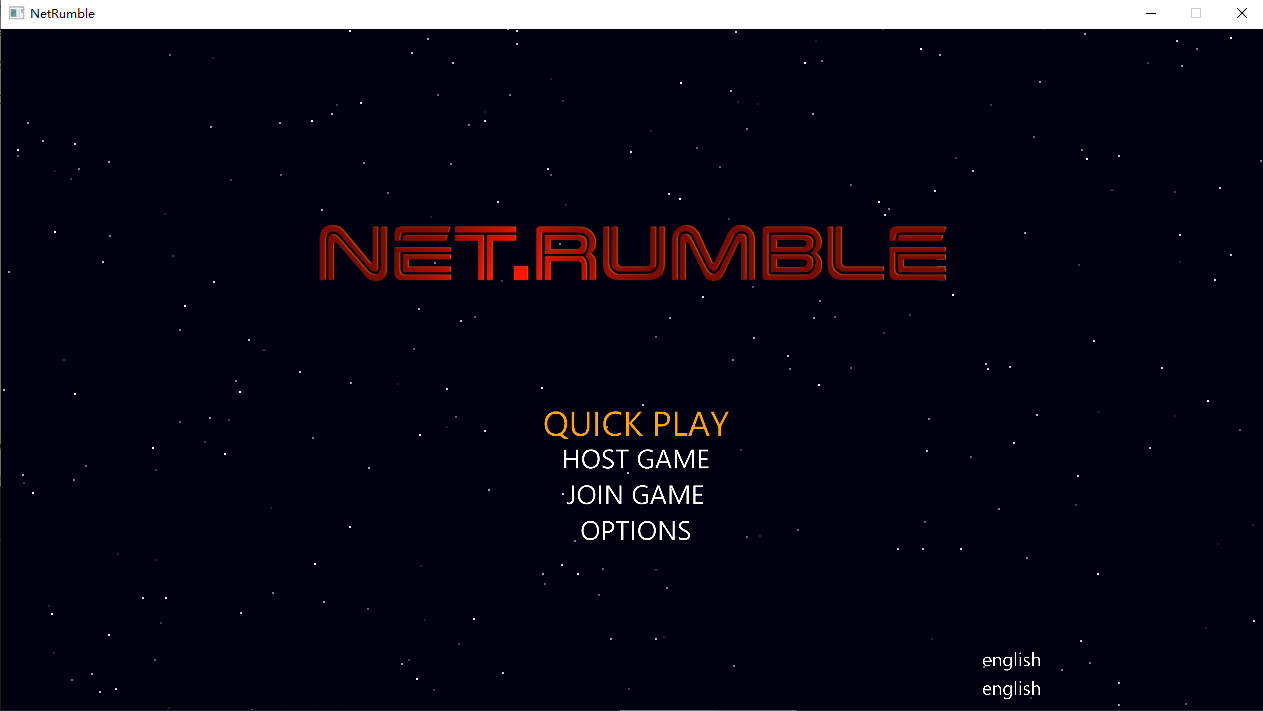
****

There are two login methods: login with custom ID and login with Xbox Live. Players can login using either of these methods and enter the Main Menu Screen after a successful login.

Action:

Choose between Login with Custom ID and Login with Xbox Live: keyboard arrow up/down

**Main Menu Screen**

****

The Main Menu Screen shows the main features provided by the game.

**QUICK PLAY:** Automatically find and join the lobby that meets the criteria. If no matching lobby is found, a matchmaking lobby is created.

**HOST GAME:** Create a lobby and join the lobby as owner.

**JOIN GAME:** Find and join lobby

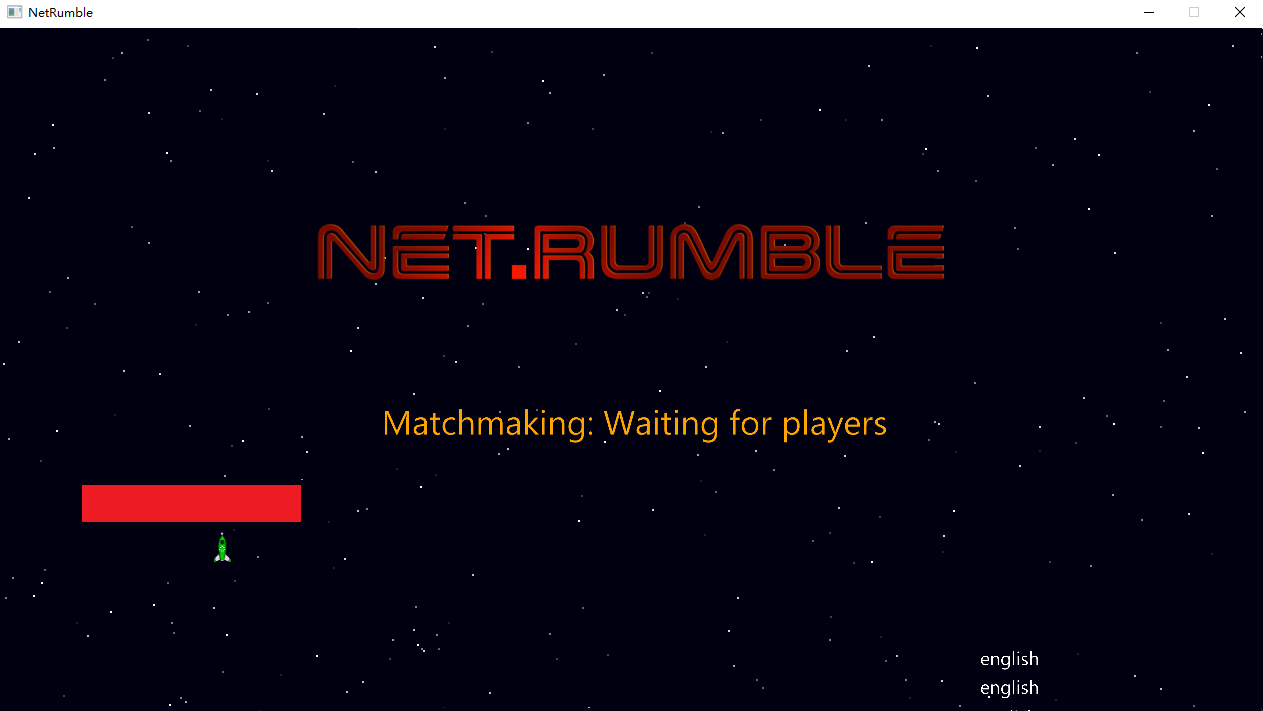
**OPTIONS:** Various game options

Action:

Choose between QUICK PLAY, HOST GAME, JOIN GAME and OPTION: Arrow Up/Down

Select menu item: Enter

**QUICK PLAY Screen**



If a match is successful, the user will directly enter the lobby. If no match is found, a new matchmaking lobby will be created automatically, and the user will be added as the owner.

Action：

Cancel matchmaking: Esc/Backspace

Go back to Main menu: Esc/Backspace

**HOST GAME Screen**



In the lobby screen the player can choose the color of the plane and the style of the plane.

Up to 4 players can play together. When the number of players reaches the lobby limit, the lobby can no longer be discovered and joined by other players.

The lobby is normally owned by the first creator of the lobby, and only when all players in the lobby are ready can the lobby begin the game's start countdown.

Action：

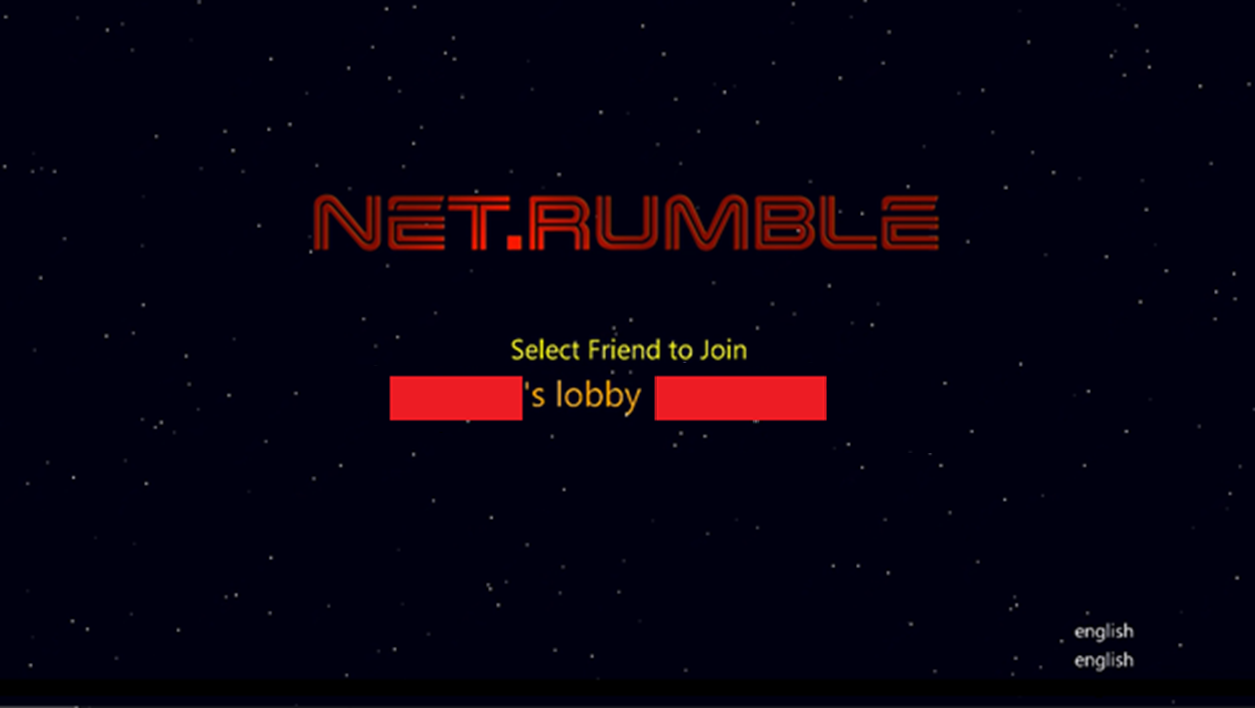
Mark your player state as ready/Cancel: Enter/Keyboard A

Change ship color: Left Arrow

Change ship appearance: Right Arrow

Go back to Main menu: Esc/Backspace

**JOIN GAME Screen**



Join Game can display up to 5 lobbies that meet the criteria, and players can select a lobby and join it.

Action:

Select between Lobbies: Up Arrow/Down Arrow

Join the lobby: Enter

Go back to Main menu: Esc/Backspace

Game Screen 

Player scores are displayed at the top of the screen in game. Players lose points for hitting meteorites and dying, and score points for killing other players.

The game is over when the player wins after scoring 5 points.

In game players can communicate through voice chat. When the game ends, the voice chat automatically stops.

Action:

Move the ship: Keyboard W, A, S, D

Fire primary weapon: Arrow Keys

Drop mines: Space

Go back to Main menu: Esc/Backspace

Notes

**Login with Custom ID**

Custom ID is a simple authentication option supported by PlayFab servers. It can often be used during the game development or testing phase and provides a “lightweight” way of authentication with PlayFab. However, it has its limitations. Since it is not supported by Xbox/GDK platform some GDK-dependent PlayFab SDK API (like initializing sound devices for voice chat) are not functional when this variant of the game (GDK UWP) is used with Custom ID login option. As a result, voice chat is not available in game when it is used with Custom ID login.

Another limitation is that only one of the two available login methods can be used (Custom ID or Xbox Live), any subsequent login attempts during the same game session can be performed only using the same login option. You cannot re-login with another option until the game is restarted.

**Voice chat**

In this game, we implemented in-game voice chat. After joining a lobby, voice chat is used to communicate with all players in the game. The voice chat stops when the game is won and players leave the party and lobby, or when the player exits the game.

# Privacy statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in App.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).