NetRumble Demo Game

with PlayFab Party and Multiplayer

(Steam Store with Steam login variant)

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

# 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 using Steamworks SDK, creating, configuring and developing games for Steam and uploading them to Steam Storefront with a Steam developer account, before building this sample game. 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 a Steam game for this project with their Steam developer account and know its AppId. They will need to configure Steam Add-on in PlayFab developer portal with their Steam AppId and Steam Web API Key. This sample app is intended to demonstrate how developers can use PlayFab Party + Multiplayer C++ API and Steamworks API together to perform the following functions:

* Login to the PlayFab server using a Custom ID
  + Call PlayFab API LoginWithCustomID()
* Login to Steam using a Steam user account (make sure to login to Steam client first)
  + Call Playfab API LoginWithSteam()
  + Login to the PlayFab server using Steam authentication token (Steam Ticket)
  + Get the player's name and player ID of the Steam user
* Create a game lobby
* Invite your Steam friends to join the 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
* Get and update player inventory using Steamworks API
* Get player achievements, scores, leaderboards already joined, and create new leaderboards using Steamworks API
* Use Steam Cloud
* Implement a multiplayer game

Dependencies

This example depends on the PlayFab C++ SDKs and Steamworks SDK. 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**

* Steamworks SDK download link: <https://partner.steamgames.com/downloads/steamworks_sdk.zip>

Download and unzip the Steamworks SDK to this directory: GDKSamples/Live/NetRumble/Dependencies/SteamSDK

* PlayFab Party C++ SDK is automatically installed by VS as a nuget from the public [nuget.org](https://www.nuget.org/) feed (release 1.7.11 for Windows (Microsoft.PlayFab.PlayFabParty.Cpp.Windows) was used at the moment of writing this document)
* PlayFab Multiplayer C++ SDK download link: [PlayFab/PlayFabMultiplayer: PlayFab Multiplayer C++ SDK (github.com)](https://github.com/PlayFab/PlayFabMultiplayer) (release v1.3.1 **for Windows** 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:   
  [Release XPlatCppSdk version 3.94 · PlayFab/XPlatCppSdk (github.com)](https://github.com/PlayFab/XPlatCppSdk/releases/tag/3.94.220926)

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 and install DirectX, Steam client app and Steamworks SDK

* **Download and install DirectX**

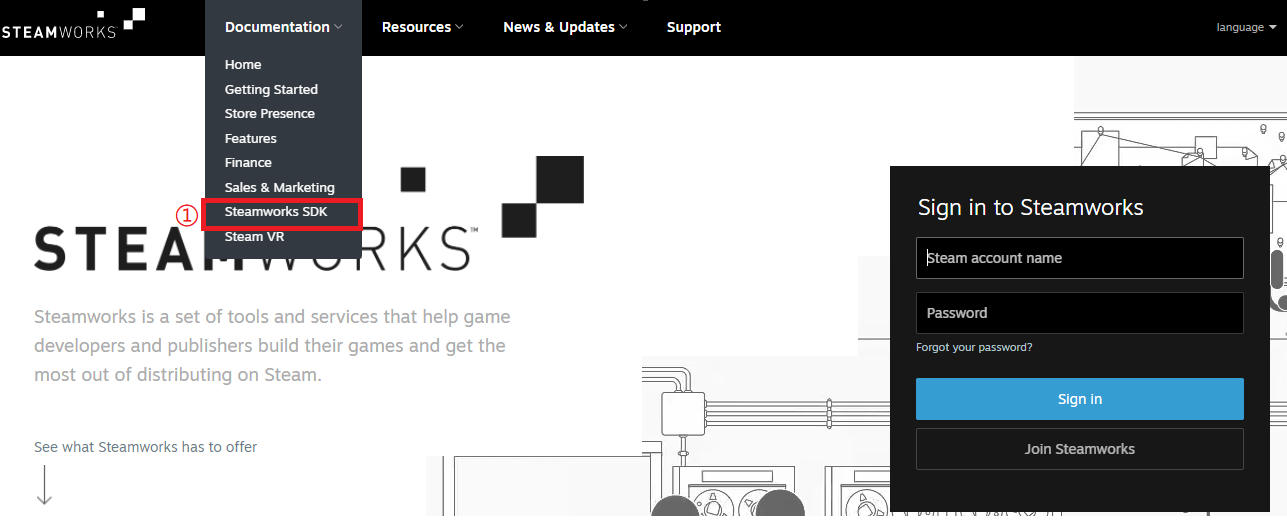
You can click here ([Download DirectX Software Development Kit from Official Microsoft Download Center](https://www.microsoft.com/en-us/download/details.aspx?id=6812)) to download DirectX. Next, follow the default prompts to install DirectX.

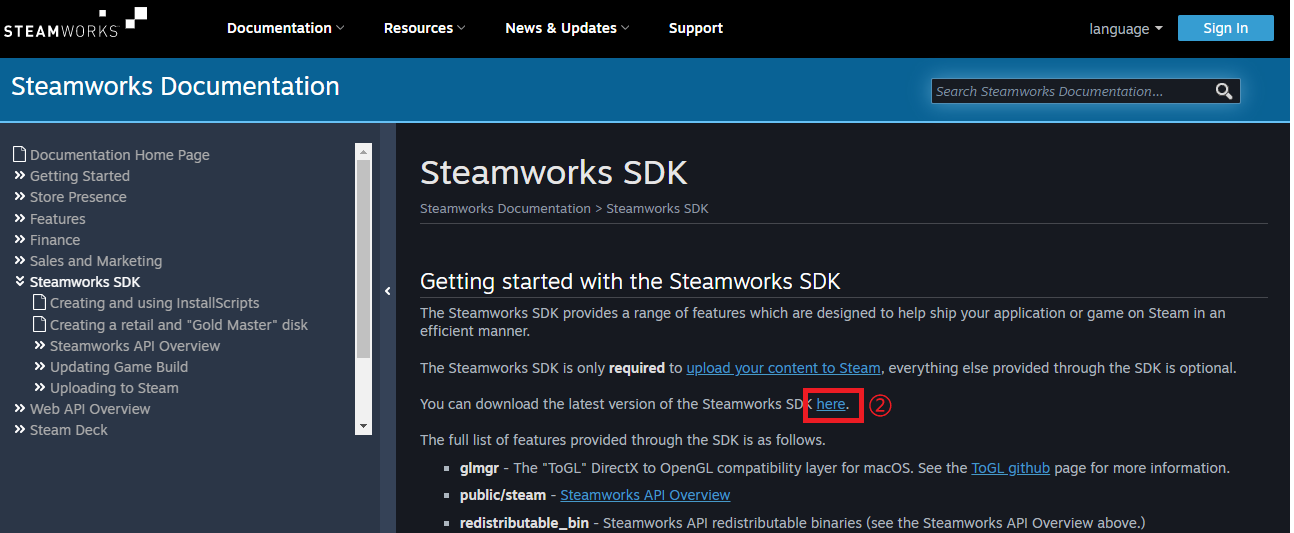
* **Download and install the Steam client app**

You can click here ([Steam (steampowered.com)](https://store.steampowered.com/)) to download and install the Steam client app. Also, please make sure you have a Steam developer account, created a Steam app, created a PlayFab title and configured your PlayFab title to use your Steam app and Steam Web API Key.

* **Download Steamworks SDK from Steamworks**

You can click here ([Steamworks (steamgames.com)](https://partner.steamgames.com/)) to download Steamworks SDK:





* **Configure Steamworks SDK and DirectX in the project (the NetRumble source code comes already pre-configured with corresponding paths, this step is for reference only)**

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

…\SteamSDK\public

…\DirectXTK12\Inc

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

…\DirectXTK12\x64

…\SteamSDK\redistributable\_bin\win64

…\SteamSDK\public\steam\lib\win64

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

DirectXTK12d.lib

steam\_api64.lib

* **Set your own Steam App ID in the project**
  + Replace the following ID with your own Steam App ID in GDKSamples/Live/NetRumble/Client/Desktop/Main.cpp file:  
      
    Text

    Description automatically generated
  + Set your Steam App ID value for “**appid**” in GDKSamples/Live/NetRumble/SteamItemDef.json file.

Configure Steam Inventory Service in Steamworks

The Steam Inventory Service is a set of features that allow a game to enable persistent player inventories without having to run special servers to manage users or items. For additional details please ([click here](https://partner.steamgames.com/doc/features/inventory))

You can find our project definition file here: GDKSamples/Live/NetRumble/SteamItemDef.json

Here will show the description of some common parameters:

|  |  |
| --- | --- |
| Name | Description |
| appid | The ID for your application. |
| item | An item type that can be found in a player inventory. |
| bundle | Represents a collection of ItemDefs, with an associated quantity of each type. When this item is granted, it automatically expands into the set of items configured in the bundle property. |
| itemdefid | The ID of this itemdef. This must be lower than 1,000,000 for non-workshop items. |
| type | Internal value. ('item' | 'bundle' | 'generator' | 'playtimegenerator' | 'tag\_generator') |
| display\_type | The item's "type". |
| name | The name of your item. |
| description | The description of your item. |
| background\_color | The color to display in the inventory background as a 6-digit hex number. |
| icon\_url | The URL to the item's small icon. The URL should be publicly accessible because the Steam servers will download and cache. Recommended size is 200x200. |
| icon\_url\_large | The URL to the item's large image. The URL should be publicly accessible because the Steam servers will download and cache. Recommended size is 2048x2048. |
| name\_color | The color to display the name in the inventory as a 6-digit hex number. |
| store\_images | Image URLs delimited by ";" character. These images will be proxied and used on the detail page of the Steam item store for your app. |
| tradable | true/false. Whether this item can be traded to other users using Steam Trading. |
| marketable | true/false. Whether this item can be sold to other users in the Steam Community Market. |

For more information on the parameters, please [click here](https://partner.steamgames.com/doc/features/inventory/schema).

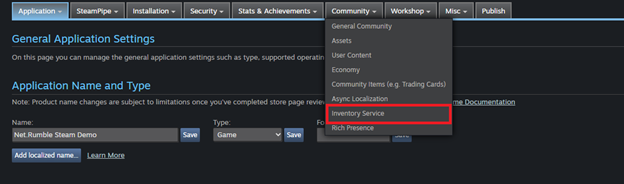
This file should be uploaded to [Steamworks](https://partner.steamgames.com/doc/features/inventory/schema) (you may make optional modifications).

Please follow these steps:

1. Click "Edit Steamworks Settings":

Graphical user interface, text, website

Description automatically generated

1. Select the "Inventory Service":  
     
   
2. At the bottom of the page, we can find "Item Definitions". You can click on "Choose File" to select your SteamItemDef.json and then click on "Upload" to upload it:  
     
   Graphical user interface

   Description automatically generated
3. We can modify the values of these items in two ways
   1. Click "Edit Row Definition", then click "Edit" in the window that pops up:

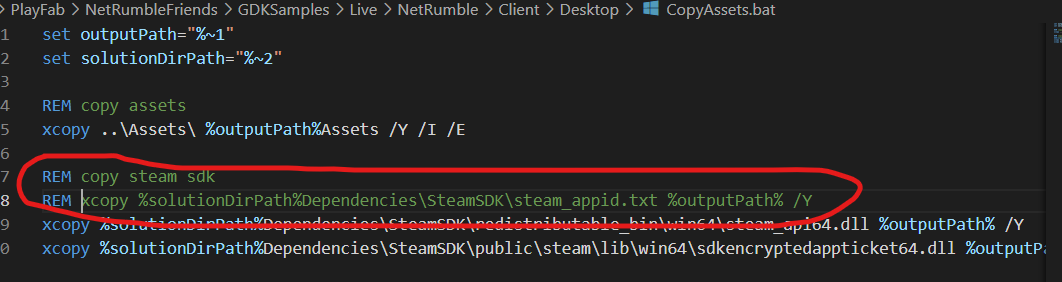
Graphical user interface

Description automatically generated

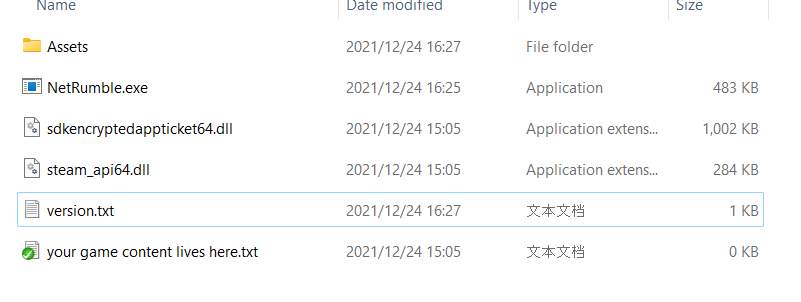
* 1. If you want to modify these values in batches, you can make changes in SteamItemDef.json and re-upload this file.

# Packaging Tutorial

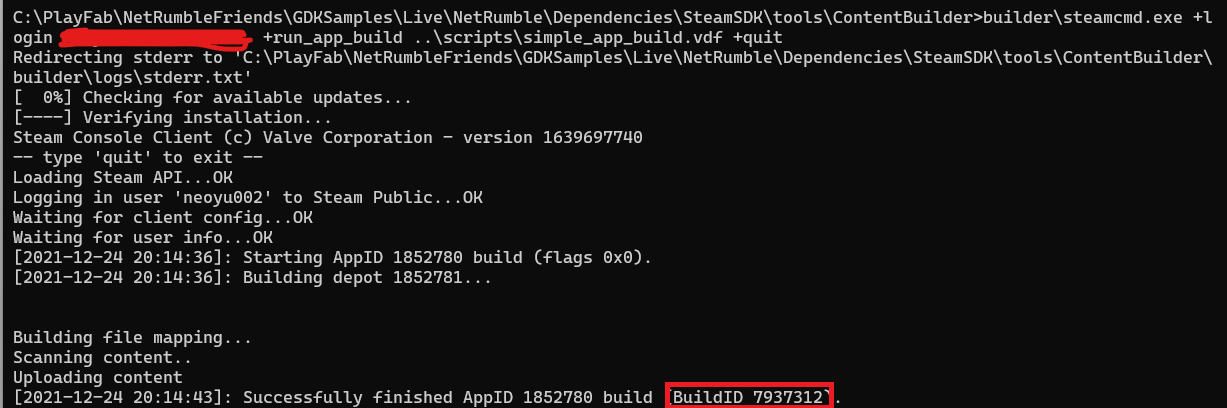
1. Manually delete all files in the current project output path *(.\GDKSamples\Live\NetRumble\x64\\*)*
2. Open the CopyAsset.bat*(.\GDKSamples\Live\NetRumble\Client\Desktop\CopyAssets.bat)* file, and comment out lines 7 and 8:

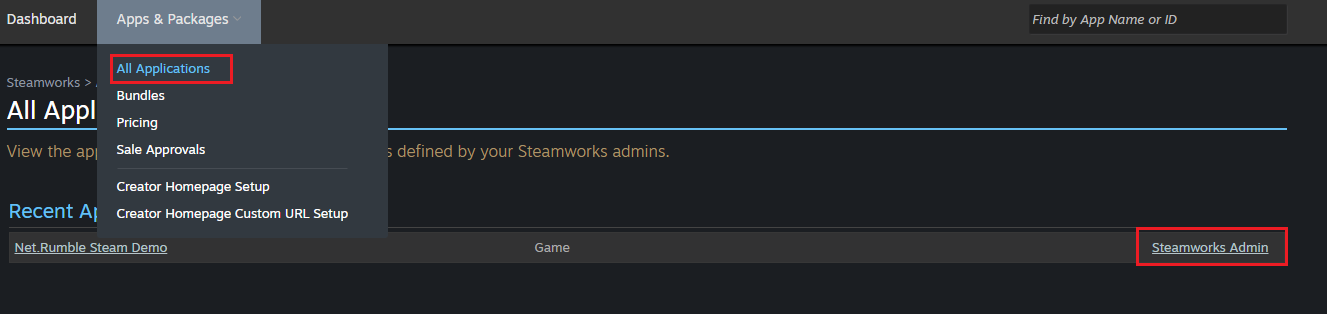


1. Build VS solution
2. Copy the build output to directory:  
   *.\GDKSamples\Live\NetRumble\Dependencies\SteamSDK\tools\ContentBuilder\content,* and delete the *.pdb* file, you can modify the number in *version.txt* if needed:

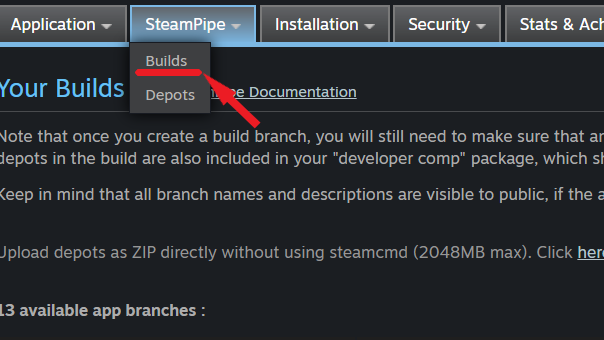


1. Modify *.\GDKSamples\Live\NetRumble\Dependencies\SteamSDK\tools\ContentBuilder\scripts\simple\_app\_build.vdf*. Please provide your own Steam AppID and Steam DepotID:  
     
   Text

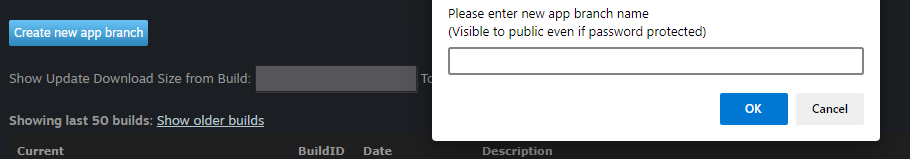
   Description automatically generated
2. Modify *.\GDKSamples\Live\NetRumble\Dependencies\SteamSDK\tools\ContentBuilder\run\_build.bat* by adding the required Steam account login and password information. Run *run\_build.bat*.
3. Please remember **BuildID**: 
4. Open the Steam Dashboard, log in, and click the All Applications button. Find your project, click the Steamworks Admin:



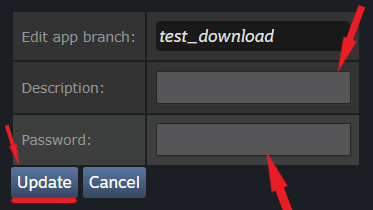
1. Click SteamPipe and click Builds:



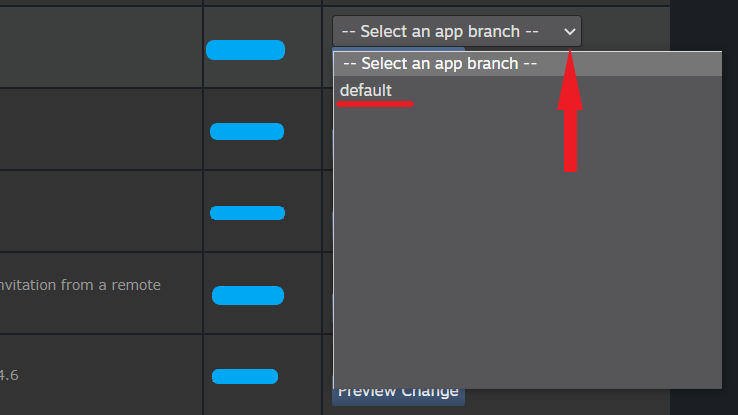
1. Click the "Create new app branch", enter the branch name to input box, and click OK.



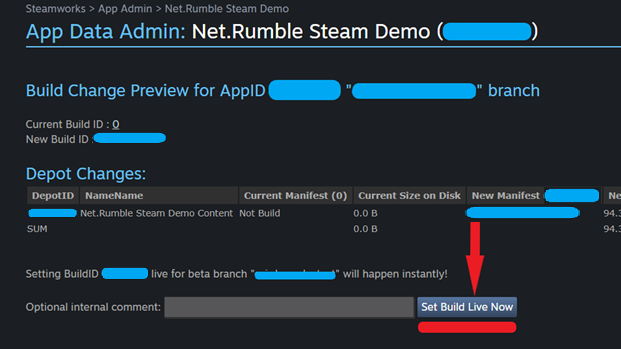
1. Enter "Description" and "Password", then click "Update" button:



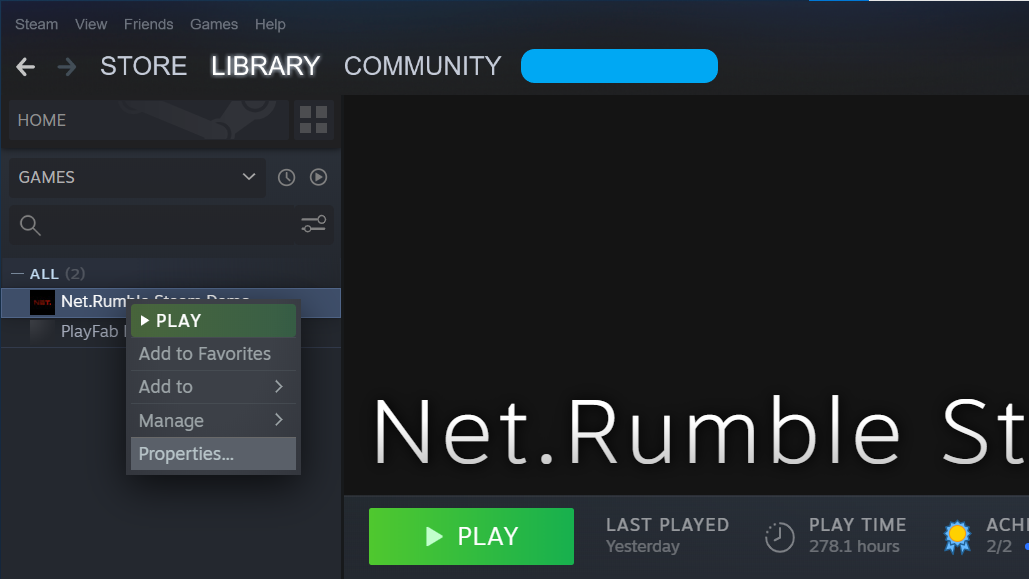
1. Update the newly generated build version (use your **BuildID**) to the created branch. Select the branch and click it:



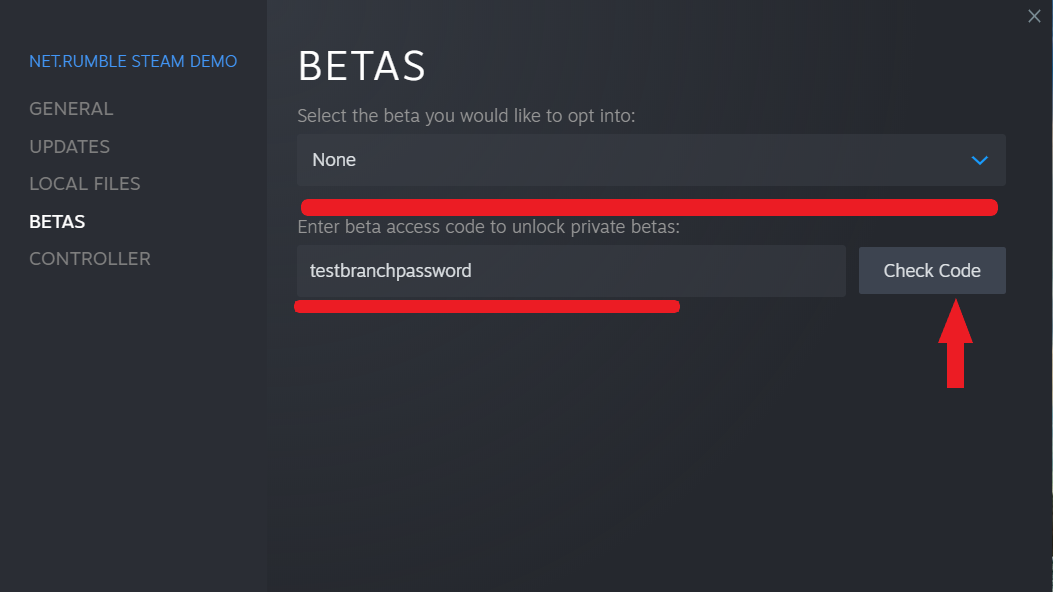
1. Click the “Set Build Live Now” button to update the build version:



1. Open Steam, click Library, right-click the test game. Click the Properties button to jump to the properties interface:



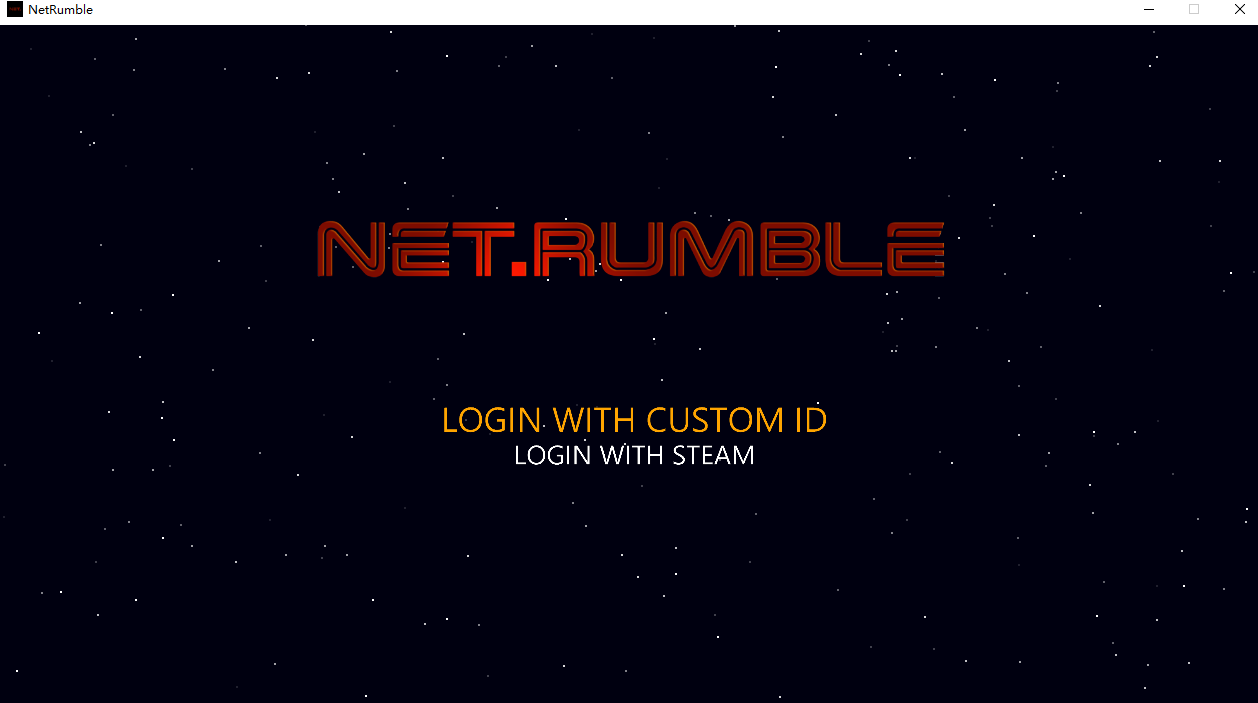
1. Click the BETAS button to open the BETAS screen. Enter beta access code to unlock a private beta. Click the Check Code button:



1. Steam will automatically update apps downloaded from the beta branch.
2. Now you can click the Play button to run Net.Rumble sample game.

# Using the sample

**Start Menu Screen**

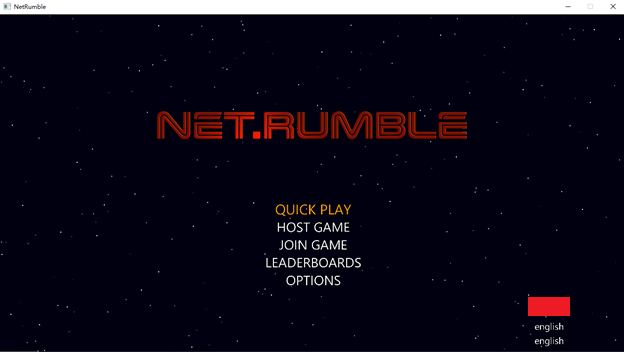


There are two login methods: login with custom ID and login with Steam. Players can login using either of these methods and enter the Main Menu Screen after a successful login.

Action:

Choose between Login with Custom ID and Login with Steam: 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 lobby is created and joined.

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

**JOIN GAME:** Find and join lobby

**LEADERBOARDS:** View leaderboard information.

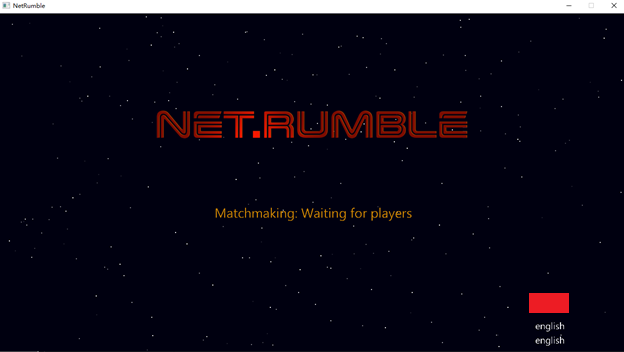
**OPTIONS:** Various game options.

Action:

Choose between QUICK PLAY, HOST GAME, JOIN GAME, LEADERBOARDS 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

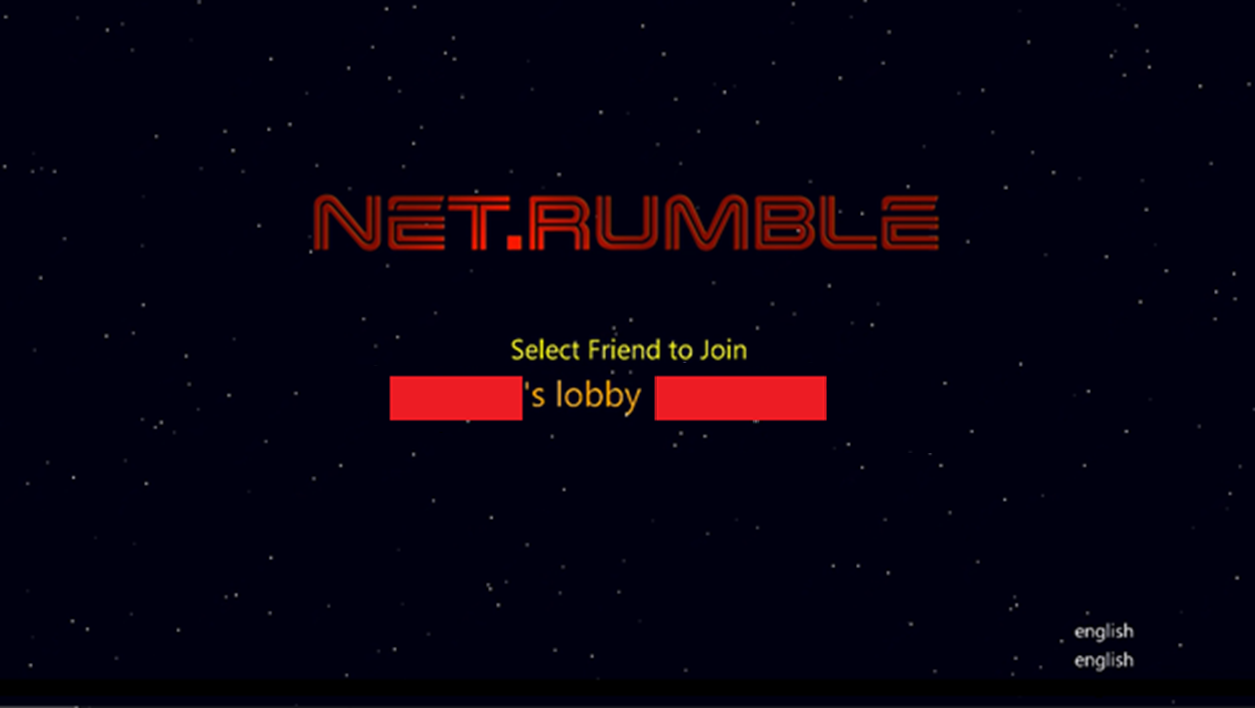
Change ship color: Left Arrow

Change ship appearance: Right Arrow

Invite a friend: Keyboard Y

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

**Leaderboards Screen**



Here you can see the top 5 and your own score.

Action:

Go back to Main menu: Esc/Backspace

Game Screen 

Player scores are displayed at the top of the screen in the 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.

Only one of the two available login methods can be used (Custom ID or Steam), 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.

**Inventory**

Allows an inventory of items to be set in the game. You can set a specific drop rate for inventory items. See inventory class SteamInventory where you can find and modify all inventory types and quantities.

**Friends**

In this variant of the game, we can add Steam friends and invite friends to play with us through the invite feature.

**Leaderboards**

In this variant of the game, we support leaderboards for all players in the game and allow to get the leaderboard information for up to top 100 players (in our version only show the top 5 players).

**Stats & Achievements**

In this variant of the game, we have achievements and statistics systems. There are two achievements and three statistics systems:

* Achievements:
  + First Game: start the game for the first time
  + First Death: First death in the game
* Stats:
  + Start Game Count: Count the number of times you start the game
  + Death Game Count: Count the number of times your death
  + Victory Count: Count the number of times you win 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/).