**ThunderRumble Multiplayer Sample**

*This sample is compatible with Windows 10 (17134).*

# Description

This sample demonstrates scalable cloud dedicated server functionality using the PlayFab Multiplayer Servers technology to build a complete end-to-end game experience.

An administration tool is also included to demonstrate how to integrate the management of PlayFab Multiplayer servers into your existing tooling and/or create new tooling.

# Known issues

* The network mesh provided with this sample is to be used as a baseline for your own implementation. The mesh should not be considered resilient enough for production use.
* The matchmaking service provided with this sample is for demonstration purposes only and should not be used in a production environment.

# Building the sample

**Build Requirements**

* Visual Studio 2017 (15.7 update) or later
* Windows 10 April 2018 Update SDK (17134)
* PlayFab XPlatCPPSDK
  + This library is included; however, you are required to provide your own dependencies including libcurl, openssl, zlib and jsoncpp.

# Local Development

This sample also demonstrates how to perform local development & debugging of your PlayFab Multiplayer Server with a flow representative of that which occurs when running in the cloud.

This is possible by registering the development server with the backend service. The backend service looks for local development servers first before attempting to allocate a server hosted in the cloud. An additional step which the service must perform is the passing of the session cookie for initial startup. Session cookies are typically a JSON blob which is used to inform the allocated server of relevant data (e.g. a lobby ID to query session information from the backend service).

You can see which section of the sample is specifically for local development vs cloud by following the g\_bIsRunningInComputeEnvironment variable throughout the Bootstrapper codebase.

If you are looking to mimic the containerization of PlayFab Multiplayer Servers locally, refer to the documentation available here: <https://api.playfab.com/docs/tutorials/landing-tournaments/multiplayer-servers-2.0/debugging-playfab-multiplayer-platform-integration-locally>

# Using the sample

**Requirements**

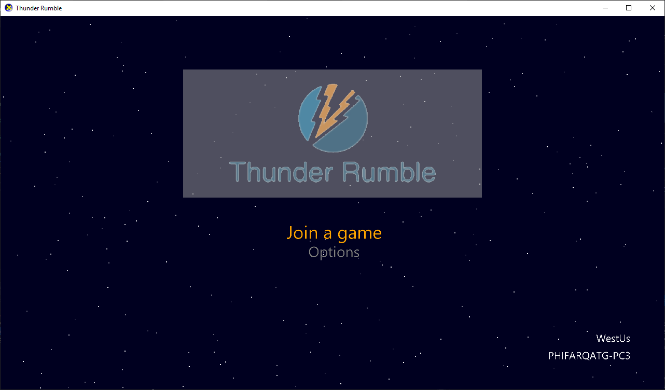
* PlayFab title configured for Multiplayer (see Service Configuration document)

**Initial Setup**

* Configure the following fields to match those of your PlayFab game title as specified in the Service Configuration document
  + PlayFabSettings::titleId in PlayFabManager.cpp
  + PlayFab.PlayFabSettings.TitleId in WebServer.cs
  + PlayFab.PlayFabSettings.DeveloperSecretKey
* Run the ‘mock lobby service’ (MockLobbyService.exe)
  + This service provides server allocation and matchmaking. You can change the required number of players to form a lobby or start a match in MatchmakingSettings within MatchmakingManager.cs
* Run the game client (ThunderRumbleWin32.exe)
  + By default, the client will communicate with the mock lobby service on localhost. If you wish to play over LAN or online, you will have to update the IP to the host machine address. This is done by modifying SERVICE\_ADDR in GameServiceHTTPRequest.h
* Compile and package a game server build (Build in Visual Studio, and then execute .\CreateServerPackage.ps1 <Debug/Release>
  + You can also run the game server locally by executing “ATGBootstrapper.exe” – note: the mock lobby service is still required and should be run first.
* Follow the instructions in the ‘Service Configuration’ document to upload and configure this server build correctly
* Proceed through the client flow as shown below.

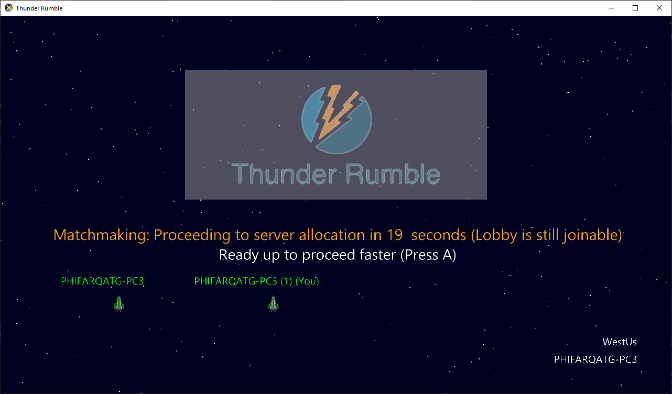
**Multiplayer Menu**

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Select menu item | A button | Enter |
| Go back | B button | Esc or Backspace |



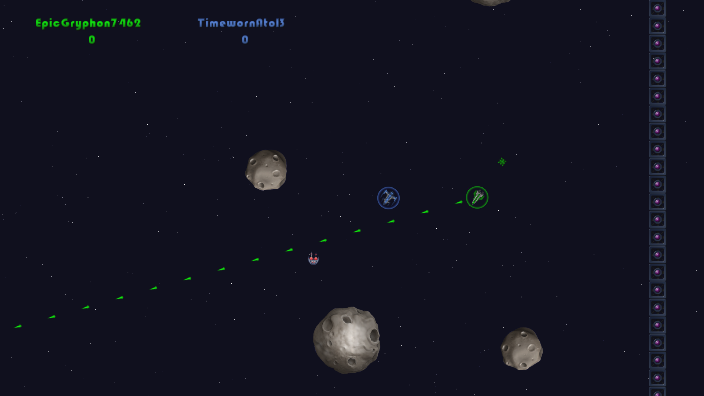
**Lobby Screen**

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Mark as Ready (with menu item selected) | A button | Enter |
| Go back | B button | Esc or Backspace |



**Gameplay Screen**

|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Keyboard |
| Move ship | Left thumbstick | W,A,S,D |
| Fire primary weapon | Right thumbstick | Arrow keys |
| Drop mines | Right trigger | Space |

****

# Implementation notes

This sample is a showcase game that demonstrates how to implement PlayFab Multiplayer Servers into your product, including the client, server and game service.

The sample consists of the following components:

**Client –** Game client which performs QoS using the PlayFab QoS API & communicates with the mock service to allocate servers

**Server –** Game server which can be run locally to demonstrate local development flow & debugging capabilities. The game server is also capable of being packaged and deployed to PlayFab Multiplayer Servers and demonstrates how to correctly port bind & how to perform logging to the host machine.

**Bootstrapper –** The bootstrapper is responsible for running the game server and enables additional debug functionality such as remote debugger. You can enable remote debugger for Visual Studio 2017 by enabling the flag in config.json before packaging the server build. The bootstrapper is also responsible for feeding the connected player list to PlayFab to be shown on the server overview page.

**AdminTool** – A sample demonstration on how to integrate PlayFab Multiplayer Servers administration functionality into your existing tooling and/or create new tools. This tool can perform a wide array of functions such as querying builds, VM’s, allocating a specific build with a session cookie and build deployment.

**PlayFabComputeHelper.cs** – A useful helper class which demonstrates all service & admin functionality provided by PlayFab Multiplayer Servers

# Update history

**Initial Release: *November 14, 2018***

Released with the PlayFab Multiplayer Servers launch

# 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/).