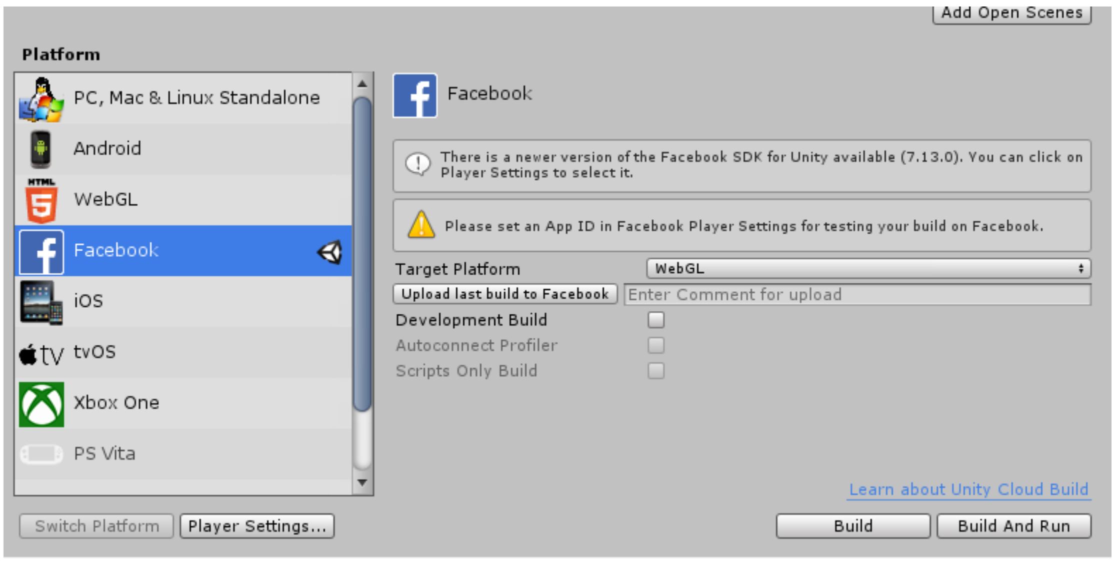
README

This document shows how to port game to Facebook Instant Game platform with Unity.

This asset only supports Unity version 2018/2019.
This WebGL project use Facebook Instant SDK 6.2 in 2018 version, SDK 6.3 in 2019 version.

How to run demo after imported package.

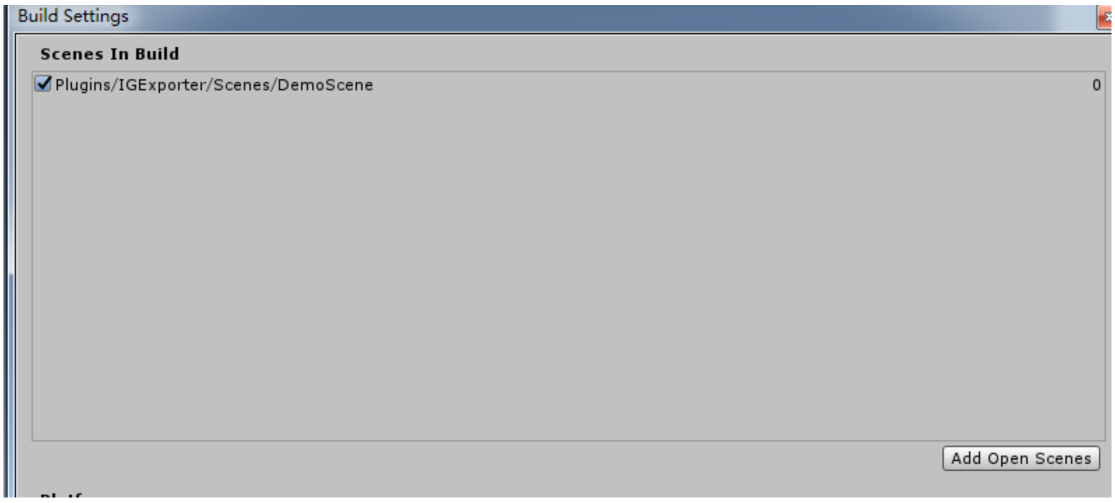
1. **Create App on Facebook**
2. **Add Instant Game product on Facebook app dashboard.**
3. **Require install WebGL platform in Unity.**
4. **Require install Facebook platform in Unity.**
5. **Switch platform to Facebook, and selected Facebook Target Platform as WebGL.**


There is a newer version of the Facebook SDK for Unity available (7.13.0). You can click on Player Settings to select it.

Please set an App ID in Facebook Player Settings for testing your build on Facebook.

Target Platform: WebGL
Development Build
Autoconnect Profiler
Scripts Only Build

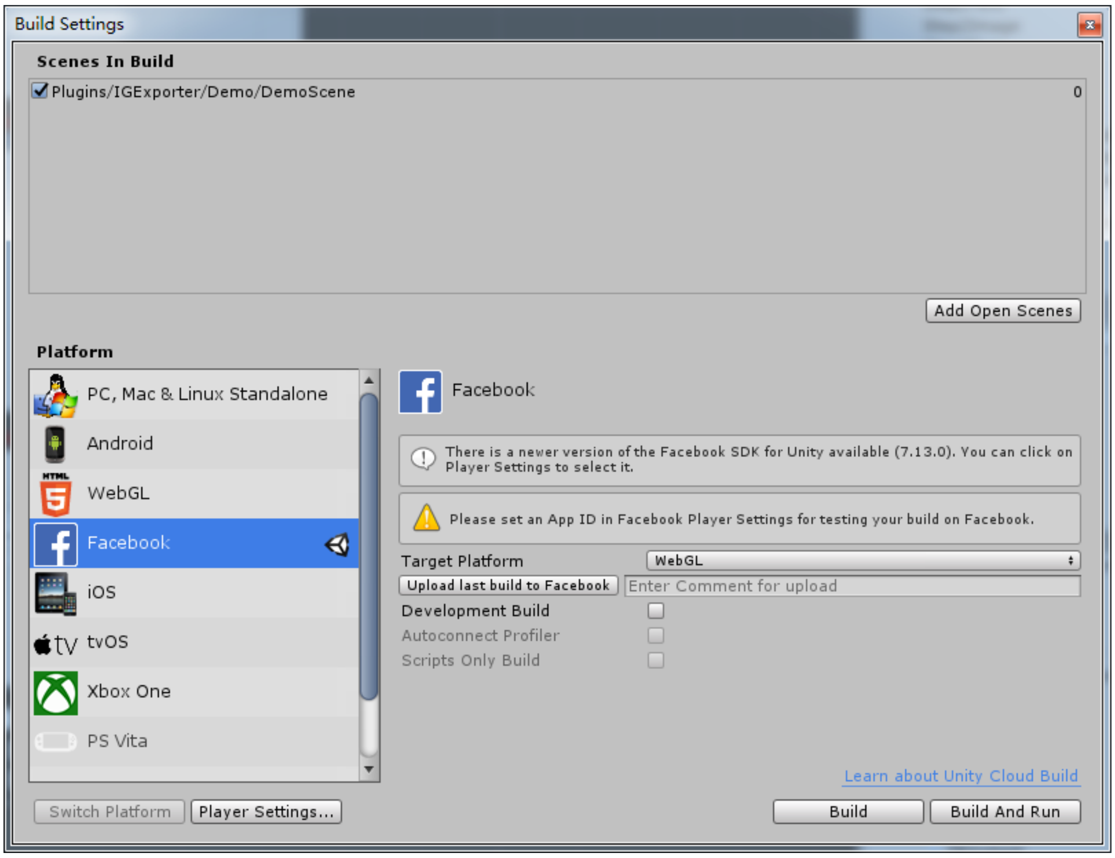
6. Search DemoScene in “Project Tab” and add the scene to Build Settings. And set it as first scene in Scenes In Build as below.


Build Settings

Scenes In Build

☑ Plugins/IGExporter/Scenes/DemoScene 0

7. Click “Build” butto to build WebGL project, the output directory **MUST** named “web”.


Build Settings

Scenes In Build

☑ Plugins/IGExporter/Demo/DemoScene 0

Platform

Facebook

There is a newer version of the Facebook SDK for Unity available (7.13.0). You can click on Player Settings to select it.

Please set an App ID in Facebook Player Settings for testing your build on Facebook.

Target Platform: WebGL

Upload last build to Facebook | Enter Comment for upload

Development Build ☐

Autoconnect Profiler ☐

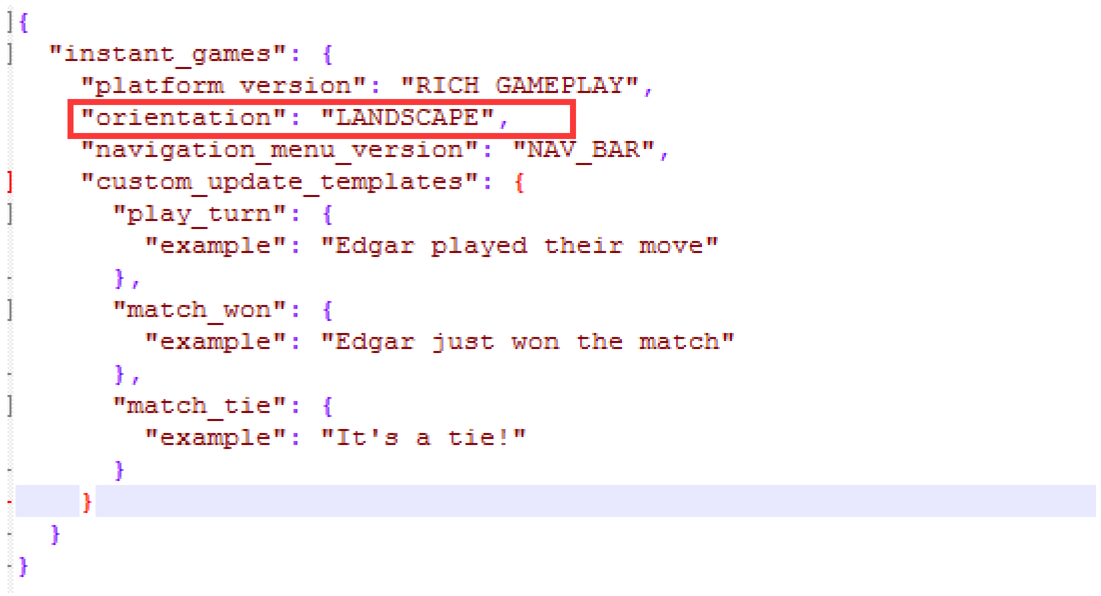
Scripts Only Build ☐

8. Test the game on localhost, Facebook document here: https://developers.facebook.com/docs/games/instant-games/test-publish-share
9. Zip the “web” directory and upload the .zip to Facebook via “Web Hosting” tab.
10. Test the game on mobile device both Android & iOS and web platform.

How to start your game logic

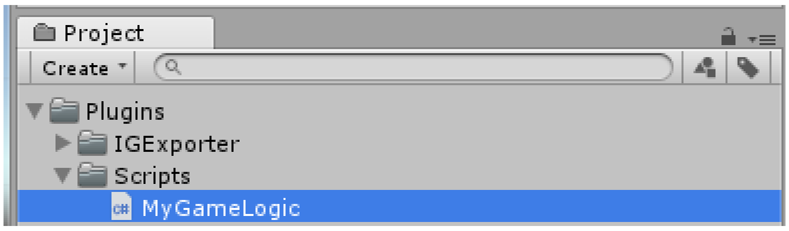
After the demo was run well, it's time to start your game logic now!!!
Please follow steps here:

1. **Custom your fbapp-config.json, if you never "Build Project" then you should change it in Plugins\IGExporter\Editor\TemplateFile\ver_2018\fbapp-config.json, else if you run demo success than your should change it in your "web" directory. Because IGExporter only copy fbapp-config.json file to "web" directory if it does not exists. You can change screen orientation or other bundle settings here. Be careful if you reimport this package, it maybe overwrite your settings. You should backup this file first before reimport.**



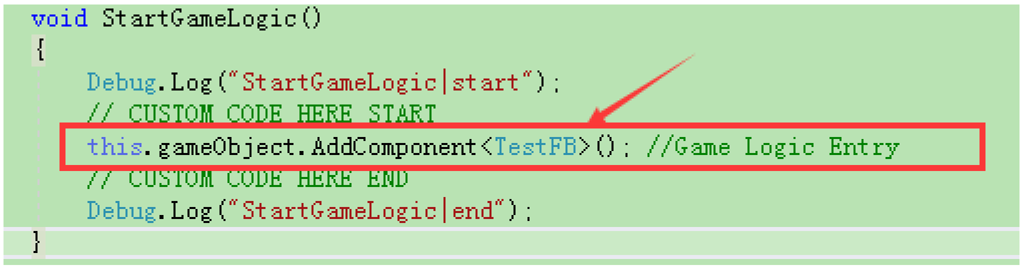
```
{
  "instant_games": {
    "platform_version": "RICH_GAMEPLAY",
    "orientation": "LANDSCAPE",
    "navigation_menu_version": "NAV_BAR",
    "custom_update_templates": {
      "play_turn": {
        "example": "Edgar played their move"
      },
      "match_won": {
        "example": "Edgar just won the match"
      },
      "match_tie": {
        "example": "It's a tie!"
      }
    }
  }
}
```

2. **Create MonoBehaviour Component, example named MyGameLogic.cs, this file should be under *Plugins* directory. Example:**

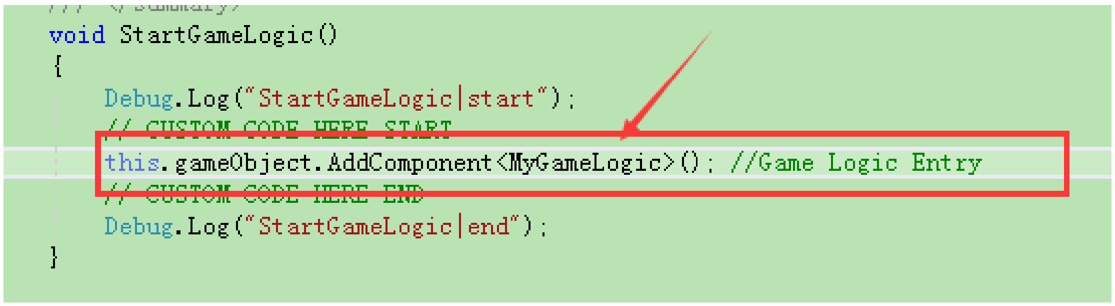


3. **Change the code in IGExporter.cs, find StartGameLogic method, and change original "TestFB" to "MyGameLogic".**

Before:


```
void StartGameLogic()
{
    Debug.Log("StartGameLogic|start");
    // CUSTOM CODE HERE START
    this.gameObject.AddComponent<TestFB>(); //Game Logic Entry
    // CUSTOM CODE HERE END
    Debug.Log("StartGameLogic|end");
}
```

After:


```
void StartGameLogic()
{
    Debug.Log("StartGameLogic|start");
    // CUSTOM CODE HERE START
    this.gameObject.AddComponent<MyGameLogic>(); //Game Logic Entry
    // CUSTOM CODE HERE END
    Debug.Log("StartGameLogic|end");
}
```

4. **Test the game on localhost again. If you feel ok than you should zip & upload to the Facebook "web hosting".**
5. **After the version staged to testing or published, test it on mobile device.**

API Documentation

1. List<**string**> FBInstant.getSupportedAPIs()
2. **void** FBInstant.switchGameAsync(string appId, Dictionary<string, object> entryPointData)
3. **string** FBInstant.player.getID()
4. **string** FBInstant.player.getName()
5. **string** FBInstant.player.getPhoto()
6. **void** FBInstant.player.getDataAsync (List<string> keys, System.Action<Dictionary<string, object>> cb)
7. **void** FBInstant.player.setDataAsync(Dictionary<string, object> gameData, System.Action cb)
8. **void** FBInstant.leaderboard.setScoreAsync (string keyName, long score, Dictionary<string, object> extraData, System.Action cb)
9. **void** FBInstant.leaderboard.getPlayerEntryAsync (string keyName, System.Action<FBLeaderboardEntry> cb)
10. **void** FBInstant.leaderboard.getEntriesAsync (string keyName, int limit,

System.Action<FBLeaderboardEntry[]> cb)

11. **void** FBInstant.leaderboard.getConnectedPlayerEntriesAsync (string keyName, int limit, System.Action<FBLeaderboardEntry[]> cb)
12. **void** FBInstant.PreloadInterstitialAd (string adId)
13. **void** FBInstant.ShowInterstitialAd (System.Action preloadMethod)
14. **void** FBInstant.PreloadRewaredVideoAd (string adId, System.Action<bool> onReadCallback = null)
15. **void** FBInstant.ShowRewaredVideoAd (System.Action completeCallback, System.Action<FBError> errorCallback, System.Action preloadMethod)
16. **void** FBInstant.shareAsync (Dictionary<string, object> p, System.Action cb)
17. **void** FBInstant.context.chooseAsync (Dictionary<string, object> p, System.Action cb)
18. **string** FBInstant.context. getType()
19. **string** FBInstant.context. getID()
20. **void** FBInstant.updateAsync (Dictionary<string, object> p)
21. **string** FBInstant. getPlatform()
22. **string** FBInstant. getSDKVersion()
23. **void** FBInstant. quit()
24. **void** FBInstant. logEvent(string eventName, long valueToSum, string jsonStr)
25. **string** FBInstant. getEntryPointData()
26. **string** FBInstant. getLocale()
27. **string** HTMLCanvas.GetScreenShotBase64()
28. **void** FBInstant.player.canSubscribeBotAsync(System.Action<bool> cb)
29. **void** FBInstant.player.subscribeBotAsync(System.Action successCallback, System.Action<FBError> errorCallback)
30. void FBInstant.context.getPlayersAsync(System.Action<ContextPlayerEntry[]> cb)
31. void FBInstant.payments.onReady(System.Action cb)
32. void FBInstant.payments.getCatalogAsync(System.Action<FBError, Product[]> cb)
33. void FBInstant.payments.purchaseAsync(PurchaseConfig purchaseConfig, System.Action<FBError, Purchase> cb)
34. void FBInstant.payments.getPurchasesAsync(System.Action<FBError, Purchase[]> cb)
35. void FBInstant.payments.consumePurchaseAsync(string purchaseToken, System.Action<FBError> cb)
36. bool FBInstant.payments.isSupportPayments()

Knowing Issue

1. Require include font files to display other languages expect English.
2. Build time is a little longer.