Prerequisite

Supported version

Unity 2019 to Unity 6000.0.X

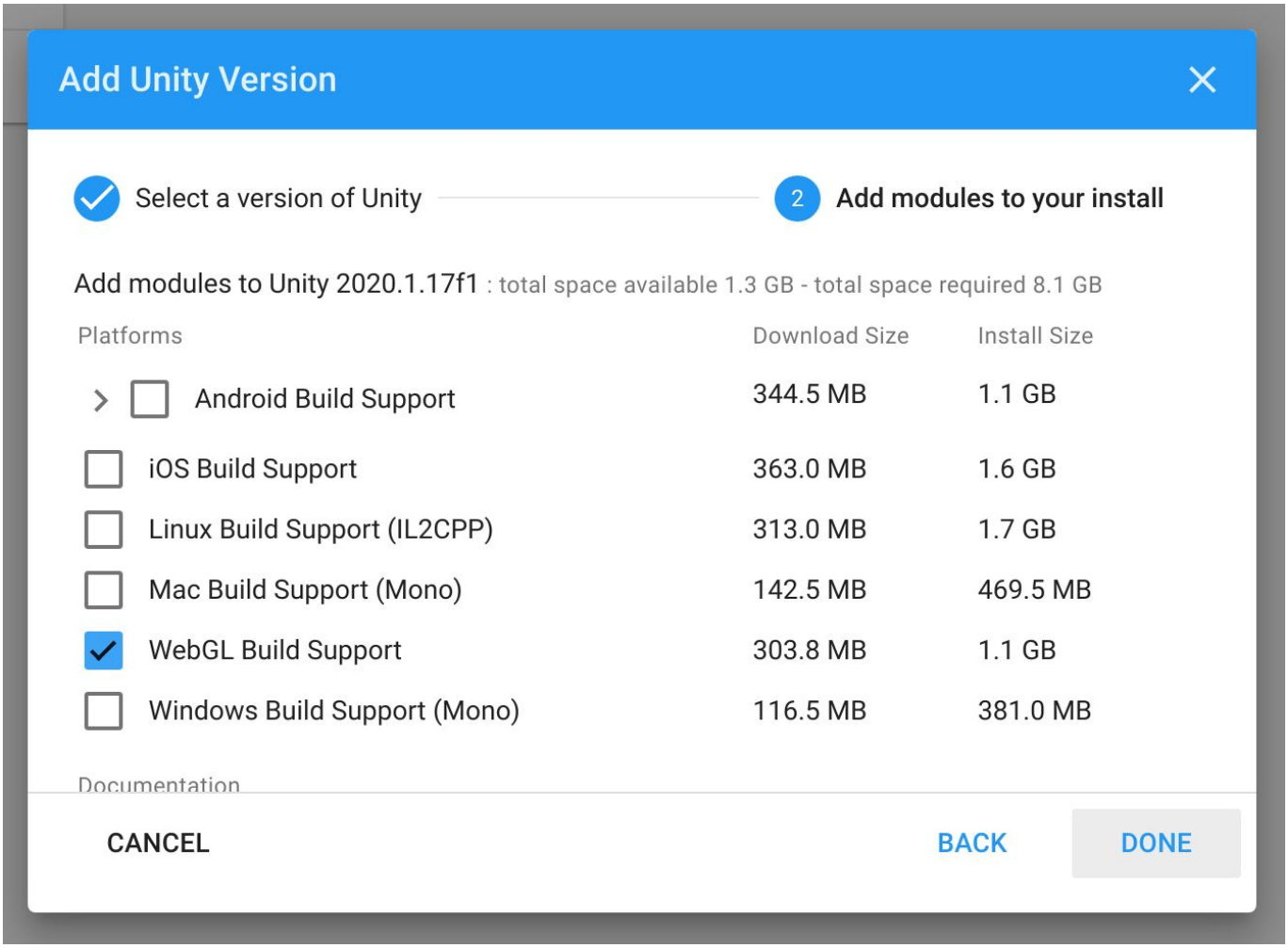
It is recommended to use the latest supported release

Not recommended Unity releases:

- 2021.1 and older (issue #1178725, issue #1325989)
- 2021.2 (issue #1411380)

Installing Unity

To publish game on GamePix portal you need to install Unity with **WebGL Build support**:



In this guide we will use **Unity 2020.1.17f**

Obtaining project

Get our sample project

This is a fully functional game that are designed to help you get started with Unity and GamePix plugin.

Installing GamePix plugin

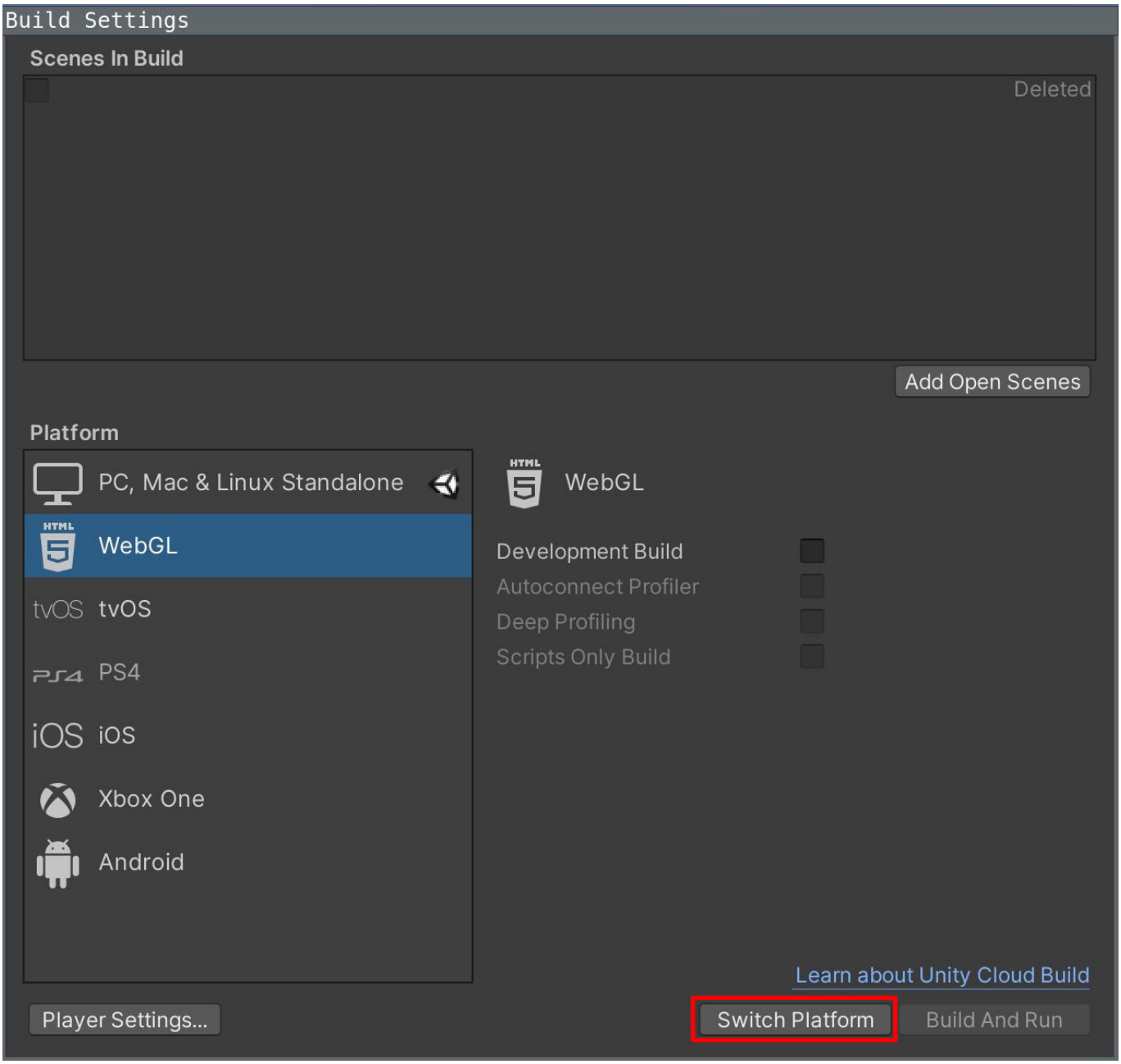
- Download latest version of GamePix plugin for:
  - Unity 2019 - Unity 2021.1
  - Unity 2021.2 - Unity 2022.1
  - Unity 2022.2 - Unity 2023.1
  - Unity 2023.2
- Extract contents of this archive into **Assets** folder of sample project

At the end plugin should be in the **Assets/Plugins/GamePix** directory.

Using plugin

Building project

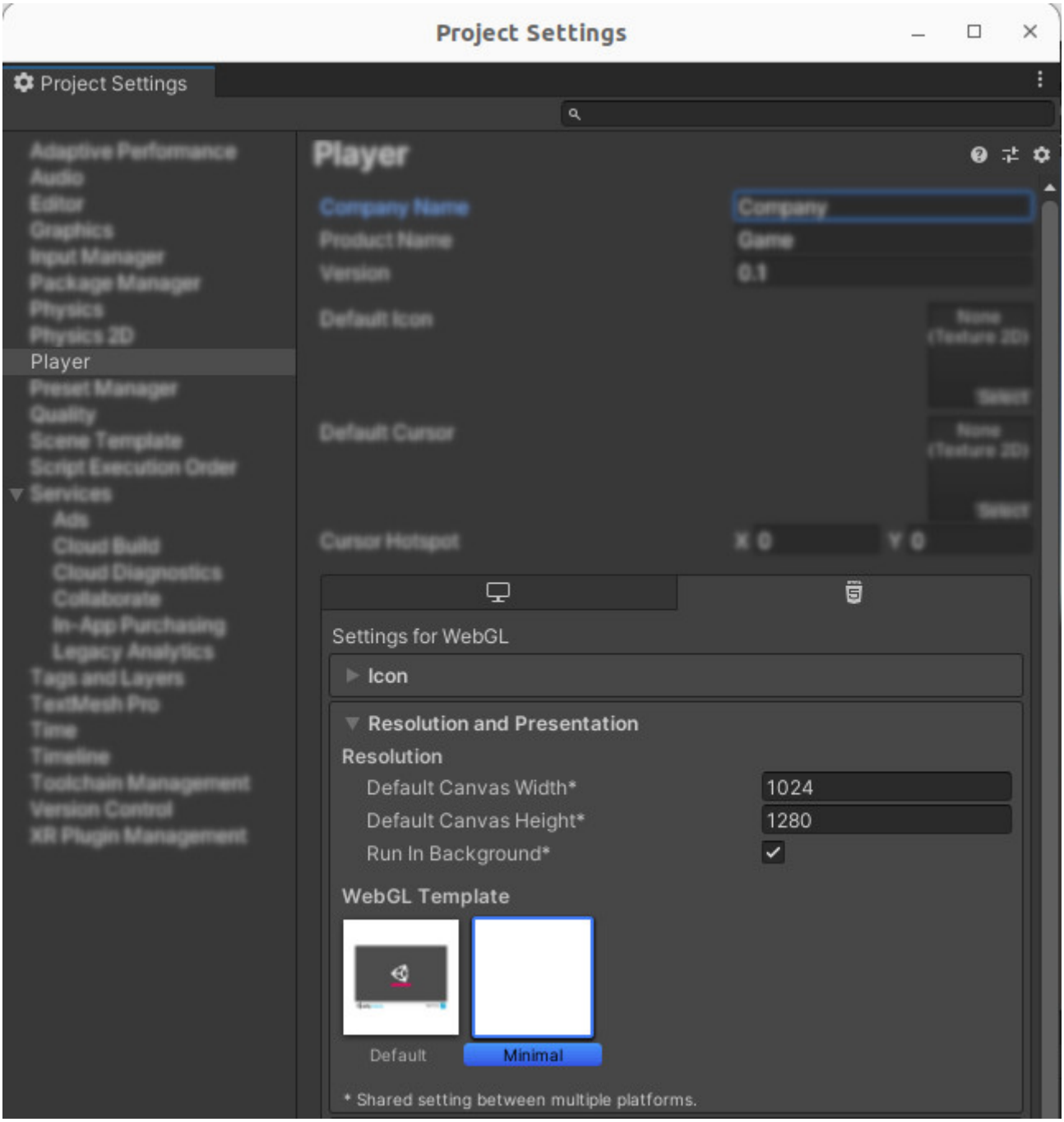
Open the project in Unity and then change platform to **WebGL** in Build Settings.



IMPORTANT: Ensure that there are no errors in Unity console. If you have some, please fix them before going forward.

It's also recommended to update all your packages to recent versions.

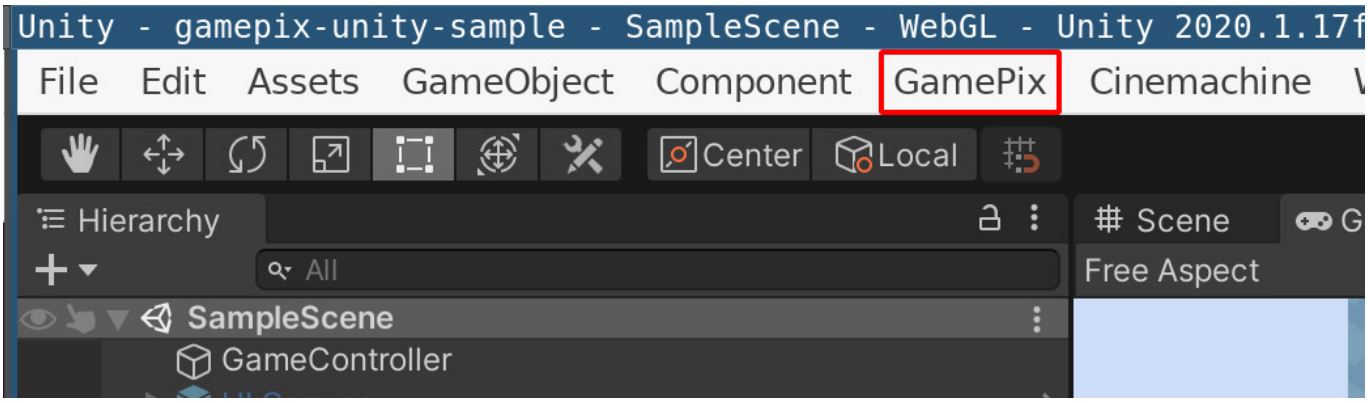
Set **default canvas width and height** for the game (Menu "Edit" -> Project Settings -> Player -> Resolution and Presentation).



Default canvas width and height will be used to calculate the game aspect ratio.

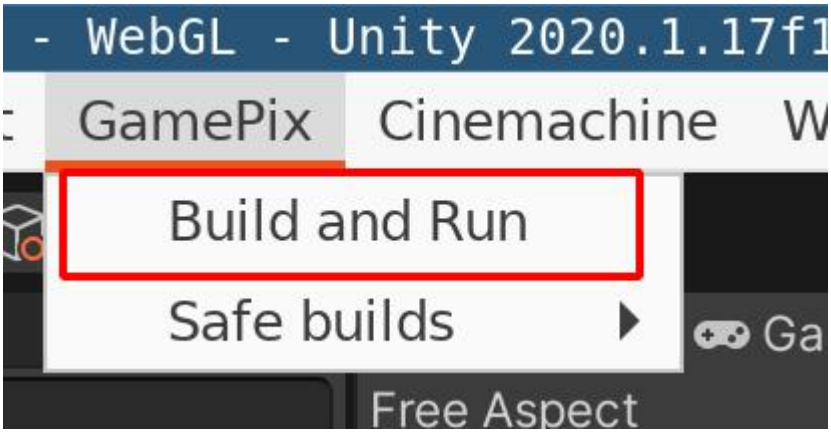
IMPORTANT: In browser the size of the game canvas will change automatically depending on the screen size, but the aspect ratio value will remain constant.

If GamePix plugin installed correctly then Unity Editor will have **GamePix** menu item.

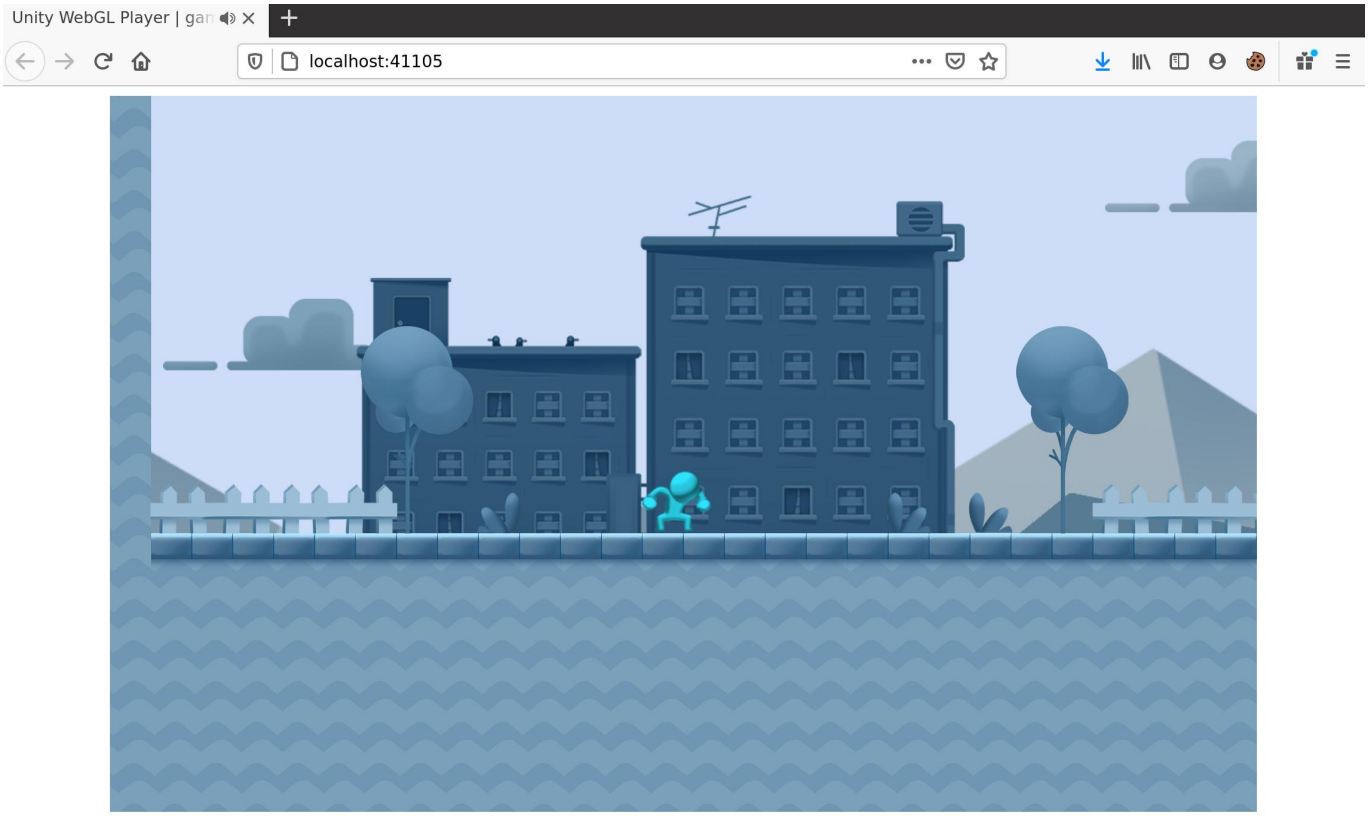


Press **Build and Run** from the menu.

IMPORTANT: Before build make sure that there are no important data in the folder **html5**. It will be cleared.



You should wait until build process is ended. It will take long time. However, at the end of build process Unity will automatically start browser with your game. Please check that game works correctly in the browser.



Publishing project

After a successful build, the archive with the game will be placed in the folder **html5** and will have the name like **CompanyName_GameName_GameVersion.gpx**.

Open the GamePix dashboard and follow instructions to publish **gpx** archive. Don't worry that file is very big, the game will be optimized and minified by GamePix publishing system.

Integrating GamePix SDK

The GamePix game SDK provides a series of utility methods needed for running your game in the GamePix ecosystem.They enhance the user experience of your game and complete the integration with our ads network and revenues system. We assume that you already read SDK Documentation.

Automatic integration

GamePix Unity plugin trying to automate everything as much as possible. Features below is automatically integrated, **do not try to integrate them manually**:

- Build already have latest SDK linked to it
- `GamePix.loading`, `GamePix.loaded`, `GamePix.pause`, `GamePix.resume` is managed by plugin itself
- `GamePix.localStorage` is used by Unity under the hood. You can use default unity storage system, it will be mapped to `GamePix.localStorage` automatically

Manual integration (Optional)

Inside the plugin SDK features are logically structured and putted inside **Gpx** global object. You may need to manually integrate:

- `Gpx.CurrentLanguage` - gets the current preferred language of player
- `Gpx.Ads.InterstitialAd` - request to show interstitial advertising
- `Gpx.Ads.RewardAd` - request to show reward advertising
- `Gpx.Events.UpdateScore` - should be called immediately every time the current score is updated

The sample project already uses some of this features in
Assets/Scripts/Gameplay/PlayerEnteredVictoryZone.cs

Pause

Interstitial and reward ads require the game to be paused, but browsers can't completely block the main thread, like on other platforms.

The plugin pauses Unity time-dependent event (animations, time-dependent moves, corotines, etc) and sounds internally.

If needs to pause time-independent functions, you can use bool value:

```
Gpx.IsGamePaused
```

which shows that an ad is being displayed.

For example, the following code pauses the rotation while an ad is running:

```
public class SampleScript : MonoBehaviour
{
    [SerializeField] private float tilt;

    private void Update()
    {
        if (Gpx.IsGamePaused)
        {
            return;
        }

        transform.Rotate(Vector3.up * tilt);
    }
}
```

Interstitial Advertising

The Intersitial ad should be shown periodically, for example: you can show this ad when the player finishes the level.

The `InterstialAd` method has only one optional argument - a callback that will be called when the ad is finished. However, you can omit it and use it like this:

```
Gpx.Ads.InterstitialAd();
```

Another example with callback:

```
using GamePix;
using Platformer.Core;
using Platformer.Mechanics;
using Platformer.Model;

namespace Platformer.Gameplay
{

    /// <summary>
    /// This event is triggered when the player character enters a trigger
with a VictoryZone component.
    /// </summary>
    /// <typeparam name="PlayerEnteredVictoryZone"></typeparam>
    public class PlayerEnteredVictoryZone :
Simulation.Event<PlayerEnteredVictoryZone>
    {
        public VictoryZone victoryZone;
```

```
        PlatformerModel model = Simulation.GetModel<PlatformerModel>();

        public override void Execute()
        {
            // show interstital ads
            Gpx.Ads.InterstitialAd(OnInterstitalAdSuccess);

            model.player.animator.SetTrigger("victory");
            model.player.controlEnabled = false;
        }

        [AOT.MonoPInvokeCallback(typeof(Gpx.gpxCallback))]
        public static void OnInterstitalAdSuccess()
        {
            Gpx.Log("SUCCESS");
        }

    }
}
```

IMPORTANT: To work with GamePix events, all callback functions must be static and use attribute **AOT.MonoPInvokeCallback** like in example above.

Reward Advertising

The Reward ad should be used when you want to reward the player after watching an ad. The `RewardAd` method takes two arguments: `success` and `fail` callback. `Success` is called when the ad is watched to the end and `fail` is called in all other cases.

Example of using `RewardAd`:

```
    private static Action onSuccess;
    public void ShowRewardAd(Action success)
    {
        onSuccess = success;
        Gpx.Ads.RewardAd(OnRewardAdSuccess, OnRewardAdFail);
    }

    [AOT.MonoPInvokeCallback(typeof(Gpx.gpxCallback))]
    public static void OnRewardAdSuccess()
    {
        onSuccess?.Invoke();
        onSuccess = null;
    }

    [AOT.MonoPInvokeCallback(typeof(Gpx.gpxCallback))]
    public static void OnRewardAdFail()
    {
        onSuccess = null;
    }
```

IMPORTANT: To work with GamePix events, all callback functions must be static and use attribute **AOT.MonoPInvokeCallback** like in example above.

Troubleshooting

If the **Build and Run** output is not working in browser, you can try a chance with:

- *GamePix>Safe builds->Build and Run (Default resources)*

However this is not recommended, because output is less optimized and have some perfomance limitations. Instead we recommend to fix game's source code until default build works.

If nothing helps, you can ask GamePix support to help build your game.

Freeze

If game freezes in browser without any error in browser console, it likely run into infinite loop. You should check the code and avoid any infinite loops. Like this:

```
    if (UseOnlineTime)
    {
        WWW www = new WWW(OnlineTimeUrl);
        while (!www.isDone)  {  }
    }
```

Best Practice

Take in account that our target environment is a default mobile browser (Safari, Android Webview, etc).

Do not expect that environment will support modern technologies.

Most common environment will be **WebGL 1 without any extension**. Please try to use this basic stack only.

Another important thing - target environment is **Single Threaded**, do not run any long-running background task.

However, all this tips are **optional** you should follow them to create most optimized and mified version of game.

Assets and packages

Delete all unused assets and packages.

It is highly desirable that everything not used in the game be deleted from both **Assets** and **Packages** folders.

Packages

Do not use any additional packages

Fonts

Do not use fonts and fonts assets or fonts packages.

Exceptions

Do not use. Using "try-catch" block slows down the game a lot. This is especially noticeable in JavaScript.