



PLUGIN YOUR GAMES 2.0

ПОДКЛЮЧАЙТЕ СВОИ ИГРЫ

[Online documentation \(beautiful, convenient and up-to-date\)](#)

[Telegram channel](#) — news about updates, useful and important information.

[Telegram chat](#) — ask a question, find information.

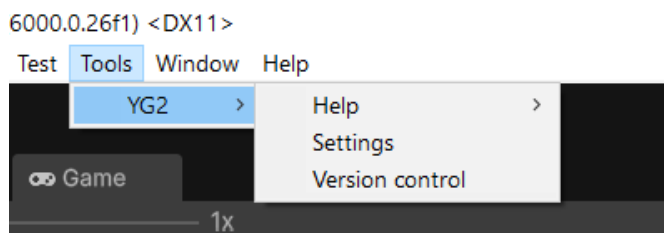
[PluginYG2 website](#) — benefits and other information.

[Video](#) - useful for visual understanding. But it is still recommended to focus on the documentation!

Getting started

Unity versions supported: 2021.3.18+, 2022.3+, 2023.2+ including Unity 6

After importing PluginYG2, you will be greeted with a welcome window. It duplicates the buttons from the context menu:



Open **settings** plugin is the fundamental window. Please note that many fields have tooltips!

It is recommended that you familiarize yourself with all the parameters in the section **Basic Settings**.

If, when importing the plugin, you refused to set the optimal settings for the standard Yandex Games platform, then for the YAI you need to select a template **YandexGame** or enable the checkbox **Auto Apply Settings**. You can see what settings will be applied by following the link to the platform settings in the field **Platform**.

Local host

Testing the game on localhost using Build And Run works. In this case, SDK initialization (receiving platform data) will be ignored.

When hosting a game on your domain on Yandex Games, enable the Own Host option, it is located in the platform settings (Platform field in the Basic Settings section). This option will add the appropriate connection string.

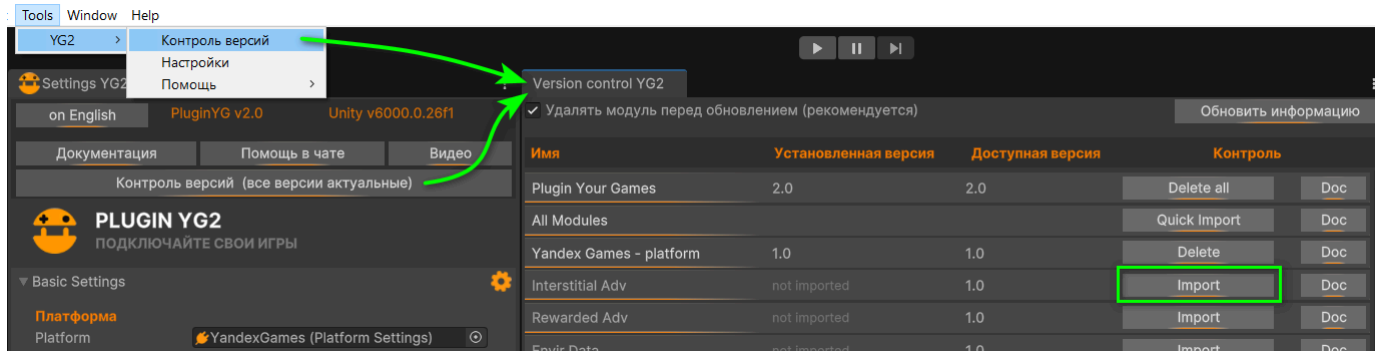
Please note that your host may not support some project settings. For example, Brotli compression. If, when importing the plugin, you agreed to install optimal settings, then PluginYG2 itself changes some project parameters. To control this process, go to the platform settings and make changes according to your preferences.

To test the game locally with connection to the SDK, in Index.html, change the IsLocalHost function to suit your needs. This file can be edited in the WebGLTemplates/YandexGames folder.

Modular system

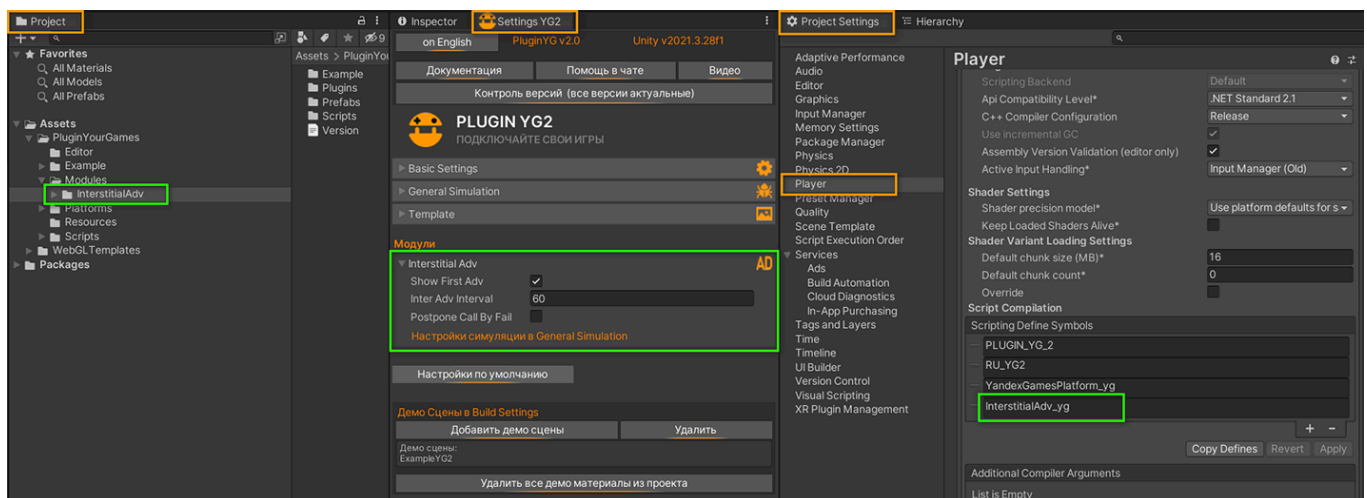
All plugin functions are divided into separate modules. You can read more about the advantages of this approach and about the plugin in general at

Open your version control tool. Let's download and import the example module **InterstitialAdv**:



When the module is imported:

1. It goes into the folder **Modules**. Any folder in the Modules directory will be defined as a plugin module.
2. If the module has global settings, they will be displayed in the plugin settings.
3. All modules will have their own definition ([preprocessor directive](#)). You can use it in your code.



Module definitions have a yg signature at the end. Modules - platforms are also designated by the inscription Platform. If problems arise, you can check whether all the definitions are installed correctly and correct them. You can also try updating the data

about the modules in the project by deleting the file
PluginYourGames/Editor/**ModulesListYG2.txt**

Having discovered **obvious** problems with the plugin, please report it in chat [Plugin Your Games](#). There you can find discussions with ready-made solutions, communicate on game development topics, keep track of plugin updates and other important information.

Import all modules

There is "quick import". Click on **Quick Import** and select modules to import. Some modules have dependencies (other modules). When importing several modules at the same time, you need to take this into account. If there are critical errors in the project, when it is impossible to open the version control window, you can manually download and use [package](#), which contains all the modules.

Working with code

PluginYG is in the YG namespace, import it at the beginning of your script: `using YG;`
Almost all functions are in the YG2 class. So, for example, we can find a method for calling advertising in this class: `YG2.InterstitialAdvShow();`
How to work with advertising and other modules, see the relevant sections.

Other objects that are contained in PluginYG2 without separate modules:

Basic

`platform` — example: `YandexGames`, `CrazyGames`.

`onGetSDKData` — an event signaling that the platform SDK has been initialized (can be useful when using the synchronous loading option).

`isSDKEnabled` — the field is true — when the SDK is initialized.

`isFirstGameSession` — a field informing whether this is the first game session. If the first one, it will be true for the entire current gaming session.

Pause the game

`onPauseGame` — game pause event. Returns bool. More details in the Interstitial Adv section.

`PauseGame` - method of pausing the game. Accepts a bool parameter: true - set a pause, false - continue the game.

The PauseGame method has other parameters so that when you turn on pause, you can choose whether to control sound, time, mouse cursor, Event System.

`PauseGameNoEditEventSystem` — pause method without stopping EventSystem. It is convenient to perform it through EventsYG2 (more about it below). Control over the EventSystem can be completely disabled using the Edit Event System option (plugin settings → Basic Settings → Pause Game).

`isPauseGame` — the field is true — when the game is stopped.

Game trick

`onShowWindowGame` - the event is triggered when the game comes into focus.

`onHideWindowGame` - going out of focus.

`onFocusWindowGame` — returns a bool value.

`isFocusWindowGame` — using this field you can find out whether the game is currently in focus.

Gameplay markup

`GameReadyAPI` — a method that needs to be called when the game has loaded and the

user can begin gameplay. Must be used if the Auto GRA checkbox is disabled in Basic Settings.

By default, this method is performed automatically after loading the game, because the AutoGRA option is enabled in the plugin settings. If your game still has additional delays after loading it, turn off AutoGRA and call the method manually.

GameplayStart - use this method to signal the platform to start gameplay.

GameplayStop - use this method to signal the platform to stop gameplay.

When opening and closing advertisements or other pauses, methods are executed

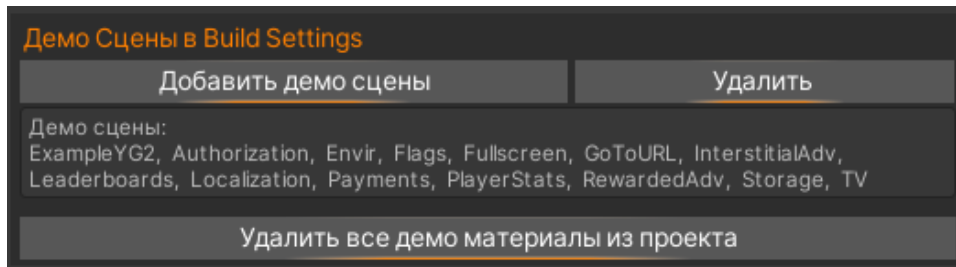
GameplayStart And **GameplayStop**. But only if the method was called earlier in the game **GameplayStart**.

That is, if you use markup methods in the code, they will be automatically executed when viewing advertisements and other similar pauses. And after the pause, the gameplay state will be set to the same as it was before the pause

isGameplaying — a field informing whether gameplay is currently in progress.

YG2.optionalPlatform.HappyTime - happy time.

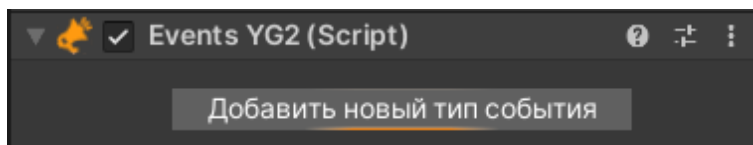
Demo materials



All demo materials are located in folders **Example**. Open the main demo scene: PluginYourGames/Example/Scenes/**ExampleYG2**. The main scene displays demo objects of all modules. If the module has a scene with a demo object, it will be displayed. But first you need to click on the "**Add demo scenes**". Then the module scenes will be added to [Build Settings](#). Before releasing the game or after completing the study of the plugin, you can delete demo scenes from Build Settings by clicking on the "button **Delete**" or delete all demo materials from the project.

When you delete a module, the module's scene will remain in Build Settings and will be undefined. Before deleting the module, it is advisable to click on the "Delete" button in the demo scenes section.

EventsYG2 component



- The EventsYG2 component stores most of the plugin's available methods. With its help, without code, you can, for example, attach a method for calling an advertisement to a button.
- The second purpose of EventsYG2 is to execute methods on an event. To do this, click on the button "**Add a new event type**" and select the event at which you want to execute a method. This works the same way as you bind methods to a button click in a component **Button**.

Assembling the build

Now you know the basics of working with Plugin Your Games 2.0 and with this knowledge you can already assemble a working build. But there are a couple more interesting points. Next, I will talk about the build number and a little about the correct configuration of the project for WebGL.

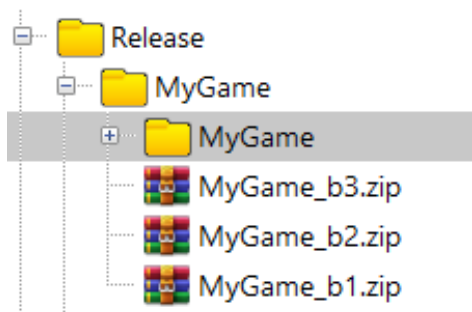
Build number

PluginYG2 stores build information in the file PluginYourGames/Editor/**BuildLogYG2.txt**. The build number is written there. For example, if you are building a project with PluginYG imported for the third time, then the build number will be = 3. This number is displayed in:

- Unity Editor log after building the project
- In the name of the build zip archive (b3)
- In the browser console (in the first logs)
- In the corner of the game screen - when turned on **Developer Build** mode (more details below)

(To reset the build information, you can delete the file **BuildLogYG2.txt** and even the entire PluginYourGames/ folder**Editor**)

After building the project, if the checkbox is enabled **Archiving Build** in the plugin settings in the section **Basic Settings**, then the game build will be automatically archived in the format **zip**, so as not to archive manually every time. PluginYG does not delete the original build and does not place an archive inside it, because The plugin does not break any core Unity logic when building a build. Therefore, the location of the created archive may seem strange. For convenience, inside the folder in which you are going to build the build, create another folder with the same name of the game. And assemble the game build inside it:

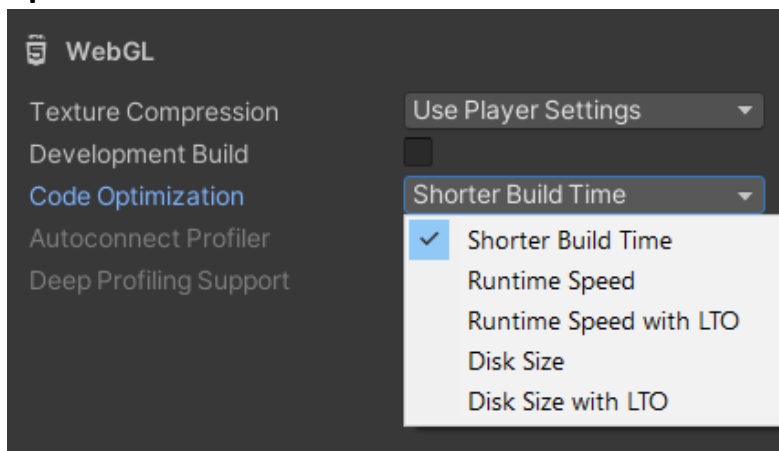


Project settings

In the plugin settings in the section **Template** you can enable the checkbox **Developer Build** while testing the game. Then the assembly will include scripts that display the build number in the corner of the game screen. This makes it possible to instantly verify that there is a current build on the platform server and signals that this build is not final. You can click on the plate with the build number to make it disappear.



Before the final build you need to turn off **Developer Build** plugin mode or the mode of the same name from Unity. Also in **Build Settings** there is an option for WebGL **Code Optimization**:



To save time when assembling a test build, set the parameter **Shorter Build Time**. For the release build **Runtime Speed with LTO**. See more details in [official documentation](#).

The plugin sets other basic project settings for WebGL automatically if you allow it, [more details above](#).

The option is not applied by default [Color Space](#). For WebGL it is recommended to use the value **Gamma**. IOS games will only work in Gamma mode. Other devices may also not support a different Linear mode. The same applies to the option **Auto Graphic API** (recommended to be enabled).

Among the main recommendations, you can also note the option **DSP Buffer Size** for sounds - sound optimization settings, which can also help eliminate the problem with

extraneous noise, if any. And in the settings of the sounds themselves for WebGL, the parameter [Load Type](#) recommended to be displayed on **Decompress On Load**.

Another problem for beginners is that the font is not displayed in the WebGL build. You cannot use the standard Unity font, replace it.