PluginYG Documentation

✔ Table of contents

📄 Introduction

This is the documentation for the plugin that enables your to integrate the Yandex Games SDK into your projects. Almost all SDK functions are supported at the moment. The plugin is automated and designed to make SDK integration as easy as possible. Please don't get scared by the documentation size. There are so many features in the plugin now, and I tried to give more detail to make everything clear. Working with the plugin is actually simple!

Many of the plugin's parameters are described in tooltips. So, in this documentation, I will ignore such parameters, as well as ones whose names are self-explanatory.

Don't miss to the demo scene! It has examples of all the functions and offers many sample scripts for the plugin.

In the Unity development environment, the plugin isn't really establishing a connection to the Yandex SDK, but simulating it. You shouldn't get many errors in the console when running the plugin.

The plugin initializes and checks everything automatically when starting the game. Cloud saves and leaderboards are supported both for authorized and non-authorized users. At startup, the plugin loads all the data, including the game saves and language (which is updated automatically when the data is retrieved).

The plugin builds upon [this template](#) for the full-screen mode. It fully solves every screen issue for you. Moreover, it lets you add an image for the startup screen.

When testing your game, please disable AdBlock and similar extensions. Also, be sure to enable ads and wait for it to get running. Apart from this, I won't describe working with Yandex Games or their developer console anymore. Nor will I describe working with the project settings for WebGL. I have articles on these topics, you can find them in my [Trello](#). Perhaps, the biggest collection of information on Yandex Games [is their official Telegram chat.](#) If you have questions about Yandex Games, first try to find an answer by searching the official chat. For issues with the plugin, you can also try searching the [plugin's chat](#).

🔗 Links

To see all discussions regarding the plugin and update alerts, welcome to the Telegram chat.

[PluginYG](#)

All information about the plugin and other useful Trello links. Here you can view the plugin versions, a description of fixes and revisions for each version, download links, implemented features, and todo lists.

[Trello](#)

📤 Importing a plugin

The plugin is packaged as a unitypackage. Importing a .unitypackage package is similar to importing an asset from the Unity Asset Store. It means that when you import a package, you can uncheck a demo scene that you don't need to add to your project. You can easily update the plugin. All the files will get updated and changed as needed. But occasionally, you may face issues during updates. In this case, try deleting the plugin's files and re-importing them.

If this is not the first time you are importing the plugin into the project, you can clear the SavesYG checkbox for the script at the YandexGame/WorkingData path to preserve your saved data (if any).

After the data is imported, you may see errors mentioning Newtonsoft, Json.net, LanguageYG. If If there are such errors, then you do not have the Newtonsoft package. Json-for-Unity and you need to download it.

To find out the plugin version, see the Readme file in the YandexGame/Documentation folder.

⚙️ Getting started

Mandatory steps to make the plugin function:

1. Add the YandexGame prefab **to every scene.**
2. Select a WebGL template.

At this point, you can already upload the build of your game to your Yandex Games draft: everything will work just fine, even Fullscreen ads will run!

💡 *Ads need to be enabled in the developer console for each game! This applies both to full-screen games and rewarded games. When enabled, the ad will start running within the hour or the day, depending on how lucky you are…*

Adding YandexGame to a scene

Here's the best way to add the **YandexGame** prefab to a scene:
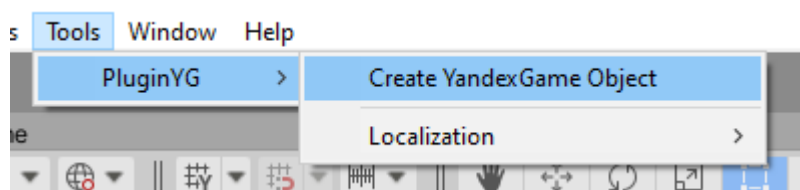
The **YandexGame prefab** includes the YandexGame object and the YandexGame script (this is your most important script). It handles all the data exchange between the SDKs of Yandex Games. It includes all the methods and fields you might need to run the plugin.

Selecting a WebGL template

Select a template under **Project Settings → Player → Resolution and Presentation → WebGL Template**
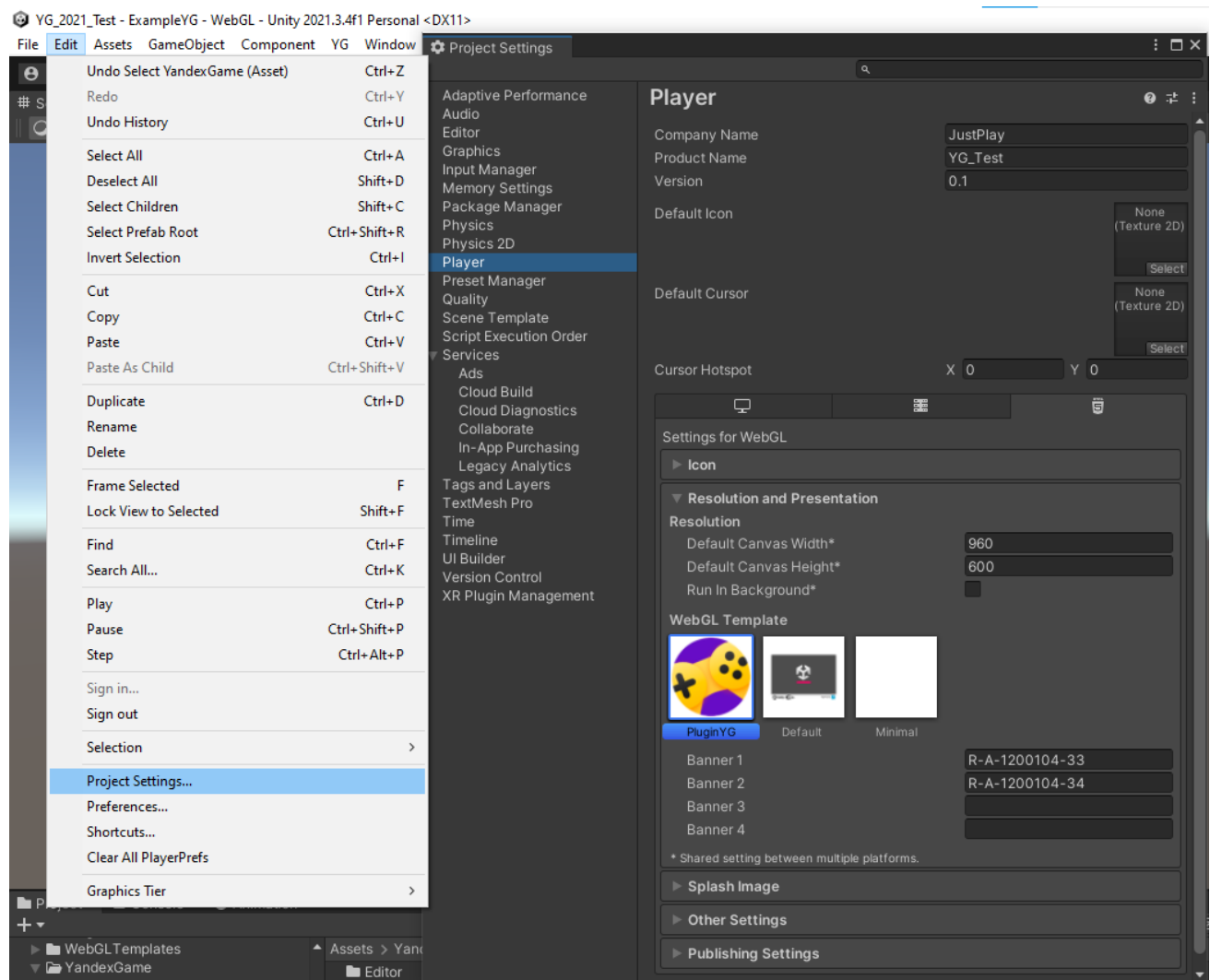
You can replace the image shown on the game's startup screen. To do this, replace the image at the path: **WebGLTemplates → PluginYG → logo.png.** But don't change the file name! Another way is to replace the logo.png file in the final build of the game.

Plugin settings (InfoYG)

The plugin settings are in the **WorkingData** folder's **InfoYG** file. You can also find it in the YandexGame component.

We'll revisit settings a couple of times below.

From now on, I'll always use InfoYG when referring to the plugin's settings.

Plugin graphics settings

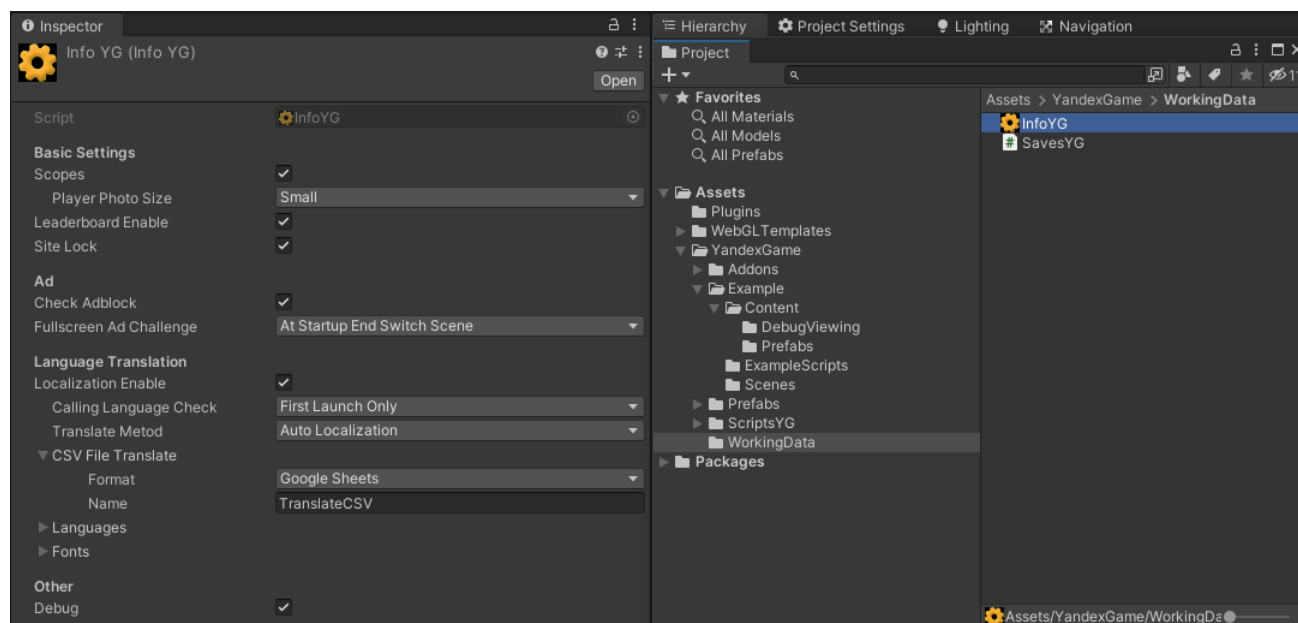Ready-to-use graphics settings in **GraphicSettingsYG** with plugin-enabled translation and possible errors when running the script

When you start the demo scene, you may see some errors. The errors are due to the GraphicSettingsYG script. If you don't need it, you can simply delete it or disable the "plugin settings" object. Otherwise, you need to configure it. The configuring includes specifying numbers of fields in relevant arrays. Those are the arrays storing languages you want to translate your game to. The number of fields in the arrays must match the number of graphic presets in **Project Settings** → **Quality**. For each relevant array, add names of graphics presets for a given language to the array's fields.

YG namespace

All the plugin scripts you need have an icon and a YG suffix at the end of the name. The other scripts are demo scripts.

All the plugin scripts belong to the YG namespace. This means that you can find components and add them to an object using the Add Component button like this:
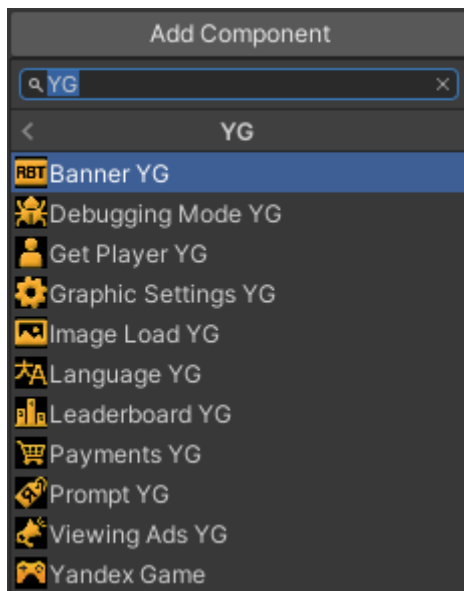
Moreover, to access the plugin's scripts from your scripts, you need to add the YG library as follows:
```
using YG;
```

Working with the YandexGame component

All the fields and methods you might need are in the YandexGame script. The script includes both static methods accessed via a script (*for example,* `YandexGame.SaveProgress();`) and non-static methods accessed via Unity Events. The screenshot below shows an example of how to trigger a Fullscreen ad block when clicking a button in the scene:
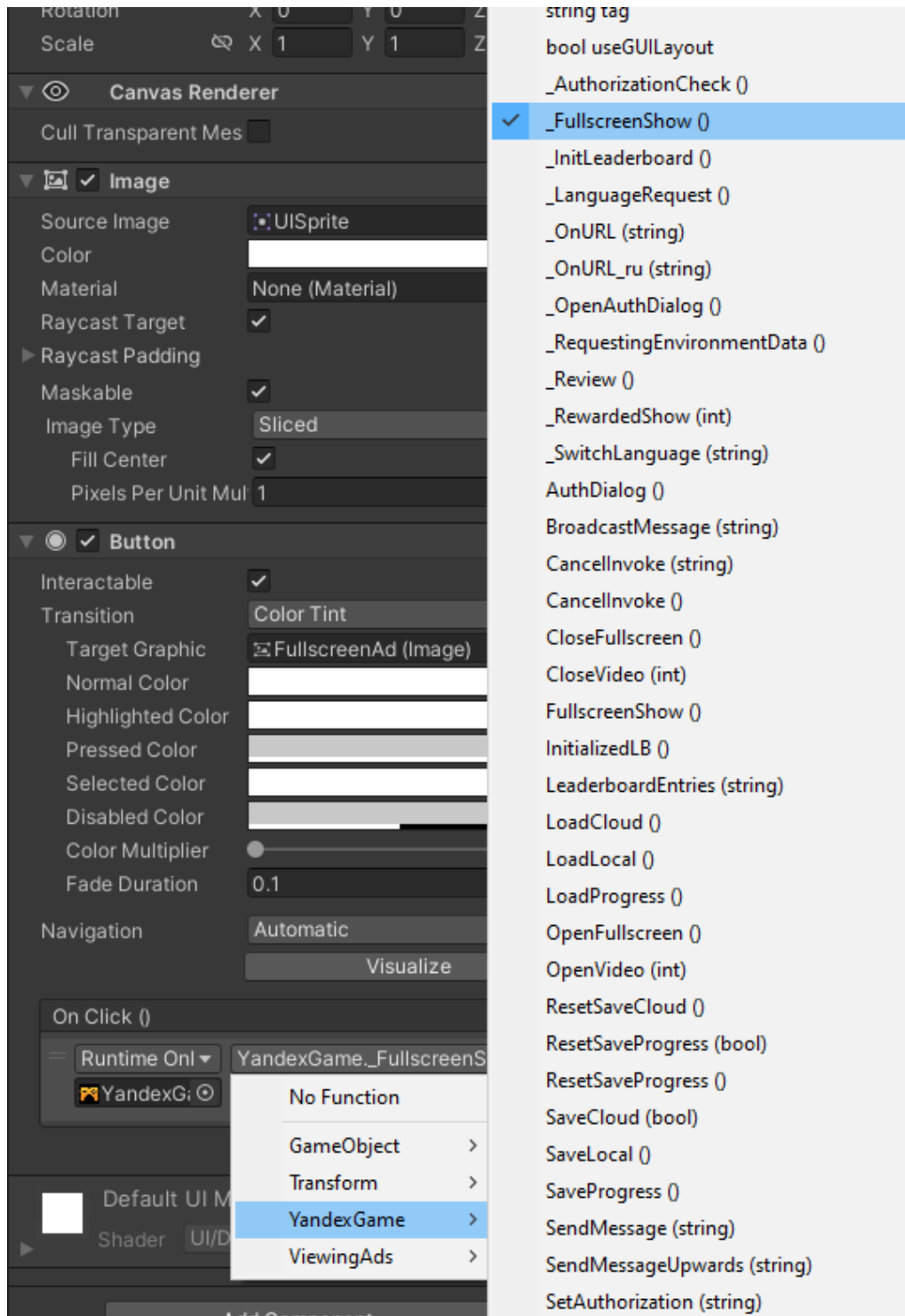
💡 *You can only use methods with underscores _ this way.*

Method definitions: **Authorization Check**: An authorization check (performed at game startup, no more checks are needed).

Init Leaderboard: Initializing the leaderboards (initialization is performed at game startup, no repeat initialization is needed).

Language Request: Requesting a language (the request is made at game startup, no repeat request is needed).

URL Yandex Define Domain

Open a link to a game or developer account. (domain is detected automatically).

Link of this format: https://yandex.defineDomain.yourPromotion

- Example for recording a link to a game: app/189792

Any URL

Open any link (your direct link).

Open Auth Dialog: Opening the authorization dialog window.

Requesting Envelopment Data: Getting the data from the Yandex Games SDK. (The request is made at game startup, no repeat requests are needed).

Review Show: Open a window to evaluate the game. Accepts an overload of the boolean type. Set to true to open the authorization window if the user is not authorized. Rewarded Show.

Rewarded Show: Showing video ads for a reward.

Sticky Ad Activity: Activate/deactivate stick banners.

Switch Language: Changing the language. Specify the language as the string value. For example, "ru", "en", or "tr". After you change the language, the game is saved.

Events in the YandexGame component

In the screenshot below, you can see the same Unity Events as the standard buttons from the Unity UI have. You can assign objects to events and select methods triggered by the events. For example, you can add a pause when opening an ad or grant a reward after the user watches a video ad.
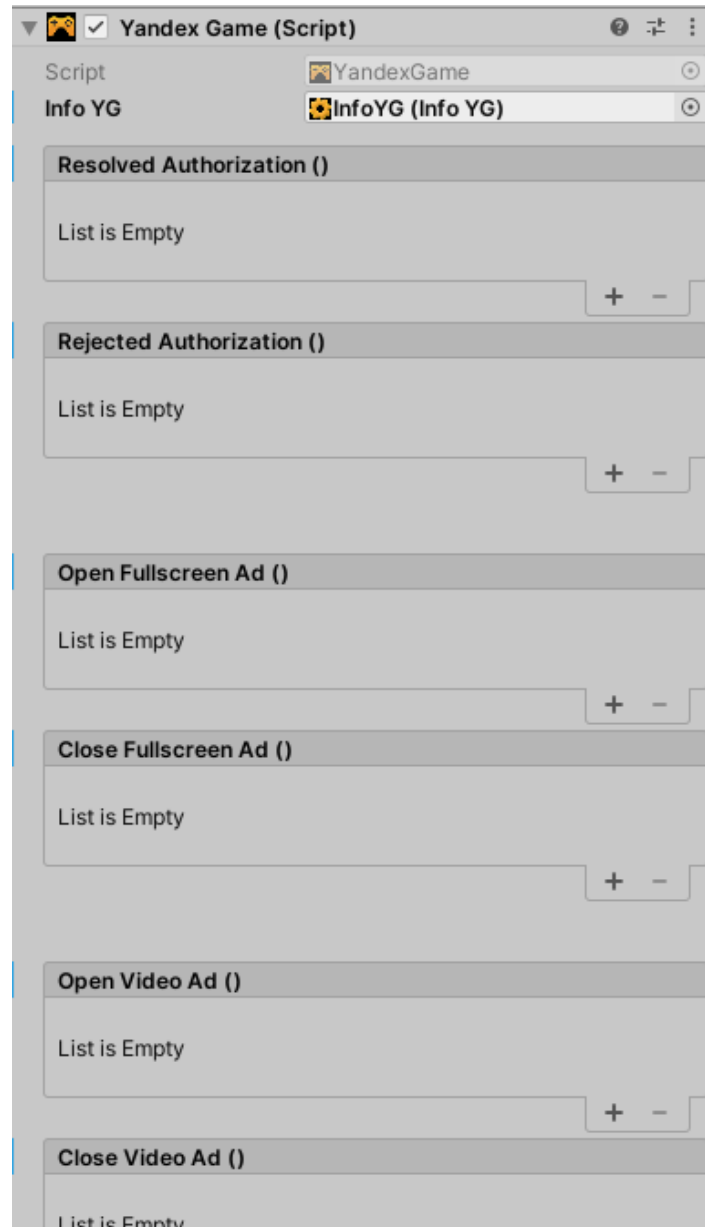
For example, the Open Fullscreen Ad event is raised when you open fullscreen ads. If you hang your script with your method on it, it will mean that you have signed your method to the Open Fullscreen Ad event. And now each time you open fullscreen ads, the method that you signed will be called. Accordingly, if there is a signed method(s) on Close Fullscreen Ad, then the signed method will work when you close fullscreen ads.

Description of events:

- Resolved Authorization

  Called the first time the game is launched after the SDK is initialized, if the player is authorized.

- Rejected Authorization

  Called the first time the game is launched after the SDK is initialized, if the player is **not** authorized.

- Open Fullscreen Ad

  Called when you open a full-screen ad.

- Close Fullscreen Ad

  Called when you close a full-screen ad.

- Open Video Ad

  Called when you open a video ad for a fee.

- Close Video Ad

  Called when closing video ads for a fee.

- Reward Video Ad

  The user watched and closed the advertisement. You have to give him a reward.

- Error Video Ad

  An error occurred while opening a rewarded ad.

- Purchase Success

  Called when a successful purchase payment is made.

- Purchase Failed

  It is called in case the payment is not made.

  The event may fire if:

  - The developer console does not add a product with such an id.
  - The user did not log in, changed his mind and closed the payment window.
  - The time allotted for the purchase has expired, there was not enough money, etc.

- Prompt Do

  Invoked after the user has successfully installed a shortcut to the desktop.

- Review Do

  The player left a review.

📞 Initialization

The plugin initializes itself at game startup. You only need to use the plugin data and methods after initialization. Follow the steps described below when working with the data or methods, like when saving a game.

The plugin doesn't run immediately when the game starts. If you need to access the plugin data or call some of its methods when starting the game, use the method subscribed for the `GetDataEvent` event rather than the start method. This event is triggered when all the data is received from the Yandex SDK.

To check that the plugin is enabled, use the boolean `SDKEnabled` field.

Basically, if you only need to do something once at game startup, do it in the method subscribed to GetDataEvent.

If you need to do something every time you load a new scene, do it in the start method. However, if you use the start method to get data from the plugin, then when you start the game, the data you got may become invalid. Or the called method may cause an error or even totally crash the game. In such cases, use SDKEnabled to check that the plugin is enabled, then use the GetDataEvent event described above. I suggest using the following structure:

```
// Subscribe to the GetDataEvent event in OnEnable
private void OnEnable() => YandexGame.GetDataEvent += GetData;

// Unsubscribe from the GetDataEvent event in OnDisable
private void OnDisable() => YandexGame.GetDataEvent -= GetData;

private void Start()
{
        // Check if the plugin has started
        if (YandexGame.SDKEnabled == true)
        {
            // If it has started, then run your method
            GetData();

            // If the plugin hasn't fully loaded yet, then the method
             // won't start in the Start method, however, it will
             // start when the GetDataEvent event is called, once the
             // plugin is fully loaded
    }
}

// Your method that will run at startup
public void GetData()
{
        // Get data from the plugin and do whatever you want with it
}
```

🅰 Fullscreen Ads

Enable ads in the developer console. To be able to enable ads in the Advertising section, fill out a draft.

💡 *The ads won't get running immediately! It usually takes about three hours, or sometimes even a full day.*

💡 *Test ads are shown at game startup! Don't confuse them with real ads. Test ads are shown even if real ads aren't running yet.*

InfoYG has the setting **Fullscreen Ad Challenge**.

- By default, the **At Startup End Switch Scene option is enabled.** This means that the fullscreen ad will start even when the game is just loading and will run when switching scenes.
- The option **Only At Startup** means that the ad will only run at game startup.

💡 *Don't worry if ads are called too often or an ad block is opened twice. The plugin knows how to handle these situations.*

To use the script for calling fullscreen ads, use the `FullscreenShow` method.

You can subscribe to the fullscreen ad opening event `OpenFullAdEvent` and its closing event `CloseFullAdEvent`.

🎞️Reward Ad

Activate Reward Video Ads in the developer console.

> 💡 *The ads won't get running immediately! It usually takes about three hours, or sometimes even a full day.*

To call video ads through a script, use the `RewVideoShow(int id)` method.

You can subscribe to the `OpenVideoEvent` video opening event and the `CloseVideoEvent` closing event.

There is also an `ErrorVideoEvent` event. Subscribe to it if you want to notify players about the failure to play ad for a fee.

Use the `RewardVideoEvent (int id)` event to reward the player for viewing ads.

The video ad method (`RewVideoShow`) takes a single value of type integer. It's an advertising ID. It's for several types of rewards.

Let's say you have a reward of "+100 coins" and a reward of "+weapons".

When calling a video ad with a "+100 coins" reward, write the ID of 1: `RewVideoShow(1)`.

For the "+weapon" reward, write the ID of 2: `RewVideoShow(2)`.

In your script, subscribe your reward method to the `RewardVideoEvent` event. The subscribed method must accept a single integer value. This value will be the ID that will return the number we wrote when calling the ad. Add an if-else statement to the subscribed method. If ID = 1, reward "+100 coins". If ID = 2, reward "+weapons".

There is a simple RewardedAd demo script in the demo scene. You can use it as an example, or use the following approach:

```
using YG;

// Subscribe to the ad opening event in OnEnable
private void OnEnable() => YandexGame.CloseVideoEvent += Rewarded;

// Unsubscribe from the ad opening event in OnDisable
private void OnDisable() => YandexGame.CloseVideoEvent -= Rewarded;

// Subscribed reward granting method
void Rewarded(int id)
{
        // If ID = 1, grant "+100 coins"
        if (id == 1)
                AddMoney();

        // If ID = 2, grant "+weapons".
        else if (id == 2)
                AddWeapon();
}
```

Continuation of the code:

```
// Method for calling video ads
void ExampleOpenRewardAd(int id)
{
        // Call the method to open the video ad
        YandexGame.RewVideoShow(id)
}
```

If you have only one reward, you can simply write the ID as 0 and omit if-else checks inside your reward method.

📢 Pausing the game when viewing ads

By default, YandexGame prefab is assigned the ViewingAdsYG script. When viewing a full-screen or video ad, this script pauses the audio or time scale in the game. It can also pause both of them (if you enable the Pause Type option).

Pause Type

Audio Pause — Pause audio when viewing ads.

Time Scale Pause — Set the time scale to 0 (stop the time) when viewing an ad.

All — Pause both the audio and time scale when viewing ads.

Pause Method

Remember Previous State — Put the game on pause when opening an ad. After the ad closes, the audio and/or the time scale return to their original value (before the ad has opened).

Custom State — Specify custom values to be set when opening and closing the ad.

⚡ Sticky-banner

To enable sticky banner display:

1. Open the Developer Console and click the **Draft** tab.
2. In the **Sticky banners** block, configure the display of banners:
   1. For mobile devices, in the **Sticky banner field in portrait orientation**, select the location **Bottom** or **Top**.
   2. For tablets, in the **Sticky banner field in landscape orientation**, select the location **Bottom** or **Top**.
   3. For computers — enable the **Sticky-banner** option **on the desktop**. The banner will be displayed on the right.

   By default, the sticky banner appears when you start the game and displays the entire session. To customize when a banner is displayed:

   1. In the **Sticky banners** block, enable the **Do not show sticky banner at the start option.**
   2. Use the Sticky Banner Display Management API.Have a Banner Sticky Display Management

Manage sticky banner display

Activate or deactivate sticky banner using the method: `YandexGame.StickyAdActivity(bool activity)`

Overload activity sets the activity of the Sticky banner:

`StickyAdActivity(true)` — enable Sticky banner;

`StickyAdActivity(false)` — turn off the Sticky banner.

AB Custom banner

You can make a banner (div block) displayed by means of the "html element".

This can be useful for creating additional content that does not depend on the Unity interface, but is controlled from Unity. This can be a block with the developer's games when downloading the game or, say, a block with music ... Previously, banners were used to implement adaptive banner advertising.

The size and position of the banner can be statically specified initially in the style properties, or the banner can be dynamically changing. A dynamic banner retrieves information from Unity. It copies position, shape, and activity from an element's UI from Unity.

Types of banners

Dynamic banners are the banners that you add to the Unity interface and that stretch dynamically to fit the Unity interface. They can hide and reappear during the game.

Static banners are the banners that will always be fixed in the same position (their sizes adapt to different screens, in a percentage ratio). You mostly need static banners on the startup screen. You can use them during the game, but can't control them like dynamic banners. You won't be able to change their size, position, or hide them: they will be displayed constantly during the game.

Settings of WebGL template for banners

In order for the banner to work, you need to enter something in the appropriate line:
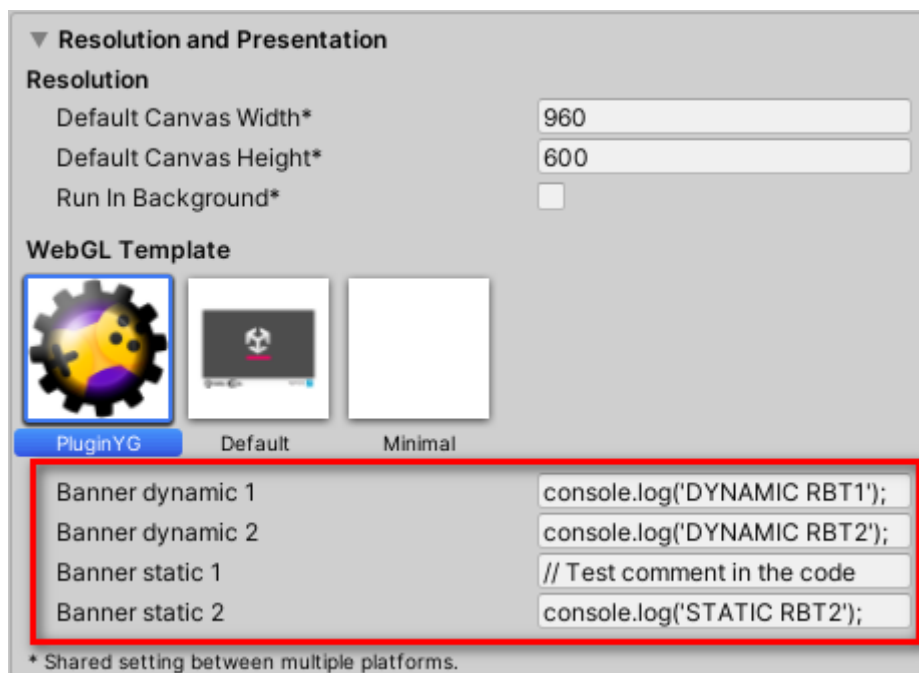
If there is any text in the field, the banner code will compile and fall into the build, and the game will automatically generate div blocks. What you write in these fields is the code that will be executed at certain points, in which it will be described below. This code is compiled in the index file in place of the following line: {{{ BANNER_DYNAMIC_1 }}}. You can, of course, write down your logic there.

For static banners, the code will execute only at startup (when initializing the Yandex SDK).

To execute code for dynamic banners, there is the ExecuteCodeRBT method in the BannerYG script or you can activate the "Code Execution Timer 31Sec" property so that the code of dynamic banners is executed every 31 seconds. The BannerYG script will be described below.

Possible issues with the template

If you can't see fields for block IDs, click on some other templates, then go back to the PluginYG template. Note that values in block ID fields disappear when you switch templates, and you need to re-enter them. After you update the project to a new Unity version or update the plugin, also double check that the fields aren't missing.

Creating dynamic banners

To create a dynamic banner, you need to specify the banner ID in the template and drag a ready-to-use prefab with the relevant number to the scene. Banner prefabs are located at: **YandexGame → Prefabs→ Banners.** If, for example, you have only one banner, use prefab number one. If you have two banners, then you must fill out the field for the second banner in the template and only use banners with the numbers 1 and 2 in the scene. If you use a banner with the number 4 in the scene, but the field for banner 4 is empty in the template, this will cause an error.

To create a banner, drag-and-drop the prefab to the scene. A prefab is a canvas with a BannerYG script and one child **Render Block object.** Set the dimensions of the Render Block object, its position, and anchors. Generally speaking, lay out your blocks similarly to regular elements in the Unity interface.

💡 *Canvas doesn't need to be configured!*

You don't need to drag the Render Block object to another Canvas!

Don't use singleton for your banners!

In the final build, the block won't be visible in the game's UI interface.

Don't edit Pivot. If you change the Pivot object of the Render Block object, the Pivot will automatically return to the position in the upper-left corner, shifting the block.

BannerYG component parameters

RTB_Number — The number of the banner whose ID you specified in the template.

Device — The device where the banner will be displayed.

Desktop And Mobile - The banner will be displayed on all devices.

Only Desktop - The banner will only be displayed on computers.

Only Mobile - The banner will only be displayed on mobile devices (phones and tablets).

💡 *You can add two banners with the same number to the scene, for different devices. That way, the same banner can have different block sizes or its anchor on different devices.*

Min Size — A minimum block size. If set, the RBT block won't be smaller than the set value. X is the minimum width. Y is the minimum height.

Code Execution Timer 31Sec — Execute code every 31sec.

UI Scale Mode — The block's scaling mode. Use the BannerYG component to adjust the scale, don't change the parameters in the Canvas components! For more information about scale modes and their parameters, see the Unity documentation.

Creating static banners

To create static banners, you only need to add the ID to the template. The banner will then be running, other operations are optional.

The first static banner is fixed at the bottom of the screen by default. The second banner is fixed at the top of the screen. By default, both banners have a width of 80% and a height of 12% of the screen. If desired, you can resize and move banners as you like. Below is an example of editing a banner block:

How to display a static banner both when loading the game and inside the game?

For this, use the **Static RBT In Game** parameter in InfoYG. Enable this option, and your static banners won't disappear after the game has loaded.

🗄 Player details

You can get the player's name, avatar, user device, and so on.

Most data is taken directly from the YandexGame class. Other SDK objects and parameters are stored in a dedicated YandexGame.EnvironmentData class. There is also a class for saving the game.

- `YandexGame.SDKEnabled;`
  Boolean type
    - `true` — If the SDK has loaded.
    - `false` — If the SDK has not loaded.

- `YandexGame.auth;`
  Boolean type
    - `true` — If the user has logged in.
    - `false` — If the user hasn't logged in.

- `YandexGame.playerName;`
  Stringify type
  If the player is logged in = **player name**
    - `"anonymous"` — If the player hasn't logged in.

- `YandexGame.playerId;`
  Stringify type
    - **Player ID**

- `YandexGame.adBlock;`
  Boolean type
    - **Check AdBlock** enabled

- `YandexGame.initializedLB;`
  Boolean type
    - `true` — If leaderboards have been initialized.
    - `false` — If leaderboards haven't been initialized.

- `YandexGame.playerPhoto;`
  Stringify type
    - A link to the **player's avatar image**

- `YandexGame.photoSize;`
  Stringify type
    - The **size** of the uploaded user **image**. It returns the value of the **Player Photo Size** parameter that you select in InfoYG.

- `YandexGame.EnvironmentData.domain;`
  Stringify type
    - **The domain.** For a list of supported domains, see the [Yandex Games official documentation](#)

21

- `YandexGame.EnvironmentData.deviceType;`
  Stringify type
  User device. It returns one of the following values:
    o `"desktop"` (a computer)
    o `"mobile"` (a mobile device)
    o `"tablet"` (a tablet)
    o `"tv"` (a TV set)

- `YandexGame.EnvironmentData.isMobile;`
  Boolean type
    o `true`: A mobile device.
    o `false`: Another device type.

- `YandexGame.EnvironmentData.isDesktop;`
  Boolean type
    o `true`: A computer.
    o `false`: Another device type.

- `YandexGame.EnvironmentData.isTablet;`
  Boolean type
    o `true`: A tablet.
    o `false`: Another device type.

- `YandexGame.EnvironmentData.isTV;`
  Boolean type
    o `true`: A TV set.
    o `false`: Another device type.

- `YandexGame.EnvironmentData.appID;`
  Stringify type
    o **Game ID**

- `YandexGame.EnvironmentData.browserLang;`
  Stringify type
    o **Browser language**
  💡 *I don't recommend using it*

- `YandexGame.EnvironmentData.payload;`
  Stringify type
    o Read about the payload parameter in the Deep Linking section.

- `YandexGame.savesData.isFirstSession;`
  Boolean type
    o An auxiliary field for the plugin. It changes to `true` after the first initialization of the game.

- `YandexGame.savesData.language;`
  Stringify type
    o **The language.** For a list of supported languages, see the [Yandex Games official documentation](#)

22

💾 Cloud saves

- Yandex Games supports both cloud and local saves. Yandex will save the data itself, depending on the authorization status.
- The saves work within the Unity environment. Outside of Yandex, the data is saved to a separate file. This enables you to test your game safely during development.
- Your game saves are preserved when you update your game in Yandex, which is not true of other saving methods that aren't linked to Yandex.
- Using saves in the plugin is simple. Create a custom list of data to be saved in a separate class. The data is stored in JSON format, so even arrays are saved. You can add new saved fields in new game versions: everything will be preserved when you upload the game to Yandex.
-

Creating custom saved data

1. Open the **SavesYG** script at the path: **YandexGame → Banners → WorkingData.**
2. You can already see some fields under the `// Your saves` comment. These fields are for a demo scene, feel free to delete them.
3. Add your fields. You can assign some values to them. In this case, these values will be used as defaults when the game is loaded for the first time.

```
namespace YG
{
    [System.Serializable]
    Ссылок: 6
    public class SavesYG
    {
        public bool isFirstSession = true;
        public string language = "ru";

        // Ваши сохранения
        public int money = 1;
        public string newPlayerName = "Hello!";
        public bool[] openLevels = new bool[3];
    }
}
```

Loading the saves

Data saves are loaded automatically after SDK initialization. The class used for the saves is static. This means that even if a scene changes in the game, the data won't change, so you will never need to add the `LoadProgress` loading method.

To get the game saves, use the same approach as with regular plugin data — the YandexGame script and its static savesData class: `YandexGame.savesData.yourField;`

To get the game saves at startup, do this after initialization in the `GetDataEvent` event mentioned above.

Saving game progress

To save game progress, use the method `YandexGame.SaveProgress();`

However, before saving the data, write it to the `savesData` class.

In the demo scene, you can find a good script for the "SaverTest" example. Here is another example:

```csharp
// Subscribe to the GetDataEvent event in OnEnable
private void OnEnable() => YandexGame.GetDataEvent += GetLoad;

// Unsubscribe from the GetDataEvent event in OnDisable
private void OnDisable() => YandexGame.GetDataEvent -= GetLoad;

private void Start()
{
        // Check if the plugin has started
        if (YandexGame.SDKEnabled == true)
        {
                // If it has started, then run your loading method
                GetLoad();

                // If the plugin hasn't fully loaded yet, then the method
                 // won't start in the Start method, however, it will
                 // start when the GetDataEvent event is called, once the
                 // plugin is fully loaded
        }
}

// Your loading method that will run at startup
public void GetLoad()
{
        // Get data from the plugin and do whatever you want with it for
         // example, we want to write to the UI.Text component how many coins
         // the player has:
        textMoney.text = YandexGame.savesData.money.ToString();
}

// Let's say this is your saving method
public void SaveProgress()
{
        // Write data to the plugin example, we want to save the number
         // of coins the player has:
        YandexGame.savesData.money = money;

        // Finally, we save the data
        YandexGame.SaveProgress();
}
```

Resetting Saves

The file of saves for game testing is saved in the WorkingData folder. To reset the saves, if needed, delete the saveyg.yg file from the WorkingData folder.

🏅 Leaderboards

The documentation in English is outdated, there is no translation yet. Watch the demo scene. Sorry!

Creating a leaderboard in the Developer Console

Go to the **Leaderboards** section, then add an **Auxiliary name** for the leaderboard. We don't need to **localize the name**, and the other options have explanations.

💡 Your leaderboards will not be enabled immediately upon creation!

Adding a score to the leaderboard

For this, use the method `NewLeaderboardScores(string, int);`

```
YandexGame.NewLeaderboardScores("Auxiliary name of your leaderboard", int new score);
```

This creates a learderboard to be shown on your game page. But you can also display leaderboards within your game.

💡 *Don't save scores too often!* *This overloads Yandex servers, and your game may be rejected. By too often I mean, for example, on every frame. Once every 10 seconds is probably fine. Though why would you…*

Creating a leaderboard in Unity

Leaderboards are displayed by the **LeaderboardYG** script. All its options have tooltips.

For **simple** description of scores, add a link to the UI.Text component to the LeaderboardYG component's **Entries Text** option. The script will write the leaderboard data to it. You can also take a prefab with a preconfigured leaderboard at the path **YandexGame → Prefabs → Leaderboards** and edit the leaderboard's interface.

The **Advanced** mode is also supported. You don't need to fill out the Entries Text option for it. It searches for certain child objects to write data to them. It will create individual elements for every entry. The player's position in the rating, their avatar, nickname, and record will be in separate UI elements. That's why it makes sense to just take the preconfigured leaderboard prefab.

The LeaderboardYG script has the following public methods:

- `UpdateLB()`

  Updating the leaderboard.

- `NewScore(int score)`

Writing a new score.

有 Localization

To get started, go to InfoYG and select your localization languages in the **Languages option.**

A total of 27 languages is available. The first 19 languages correspond to countries that have dedicated domains. These 19 and the other languages include those recommended to me by Yandex.

Here I share a list of languages that Yandex recommended for translating the game into:

> English, Spanish, Turkish, Portuguese, Arabic, Indonesian, French, Japanese, Italian, German, Hindi.

The **Calling Language Check** parameter.

- **First Launch Only** — The language will be checked only at the first game launch and saved at that point. The next game launches will use the language selected at the first launch.
- **Every Game Launch** — The language will change every time you start the game.
- **Do Not Change Language Startup** — The language won't change when the game starts.

The **Translate Method** parameter is a development-only localization method. Method choice will not affect localization in the final game. You only need it for your work in Unity.

- The **Auto Localization** parameter enables you to automatically translate text using the Google Translate API.
- Issues with Auto Localization

  Google Translate has limitations. You won't be able to translate lots of text in a short period of time. If you want to use automatic localization, then I recommend translating the text in small chunks. As you loop through your game interface development cycles, you'll have natural intervals between translations. It also seems that Google won't translate the text after the dot. It's easy to get around this… My translation script can be revised to circumvent this restriction. I'm planning to work on this.

  Nevertheless, I think that auto localization can be quite helpful in the boring task of game translation.

- Tool for translating an entire scene

  Auto Localization Masse. You will find it in the top tab **YG → Localization.** It's quite easy to figure out

  💡 *This and all other tools that search for objects on the scene ignore disabled objects!*

- With the **Manual** method, redundant component options are removed. This means that you will have to translate everything manually.

26

- Using the **CSVFile** method, you can store all keys with translations in a separate CSV file that you can open in Office Excel or Google Sheets. This method is often used to send a CSV file with application keys to a translation provider.
- Working with the CSV file saving method.

When you select the CSVFile method, the translation import and export buttons appear in the LanguageYG component.

When you export translations, if the CSV file doesn't exist, it is created in the **Resources** root folder. If the file exists, then when you click the export button in the LanguageYG component, old translations of the key written in the ID field is overwritten by translations from the LanguageYG component.

There is also a tool for importing and exporting translations for an entire scene. It is in the top tab **YG → Localization → Import\Export Language Translations.** It won't overwrite old keys with new ones if they exist. It will write only the keys absent from the CSV file. To save all translations once again, delete the CSV file or rename it. Or edit the file name in InfoYG.

💡 *Excel sometimes fails to read CSV files properly. If you have any such issues, I recommend opening the CSV file in Google Sheets. You can do it online. Then, you can save the file from Google Sheets, and this file will open properly in Excel.*

💡 *After the import, in your table processor, you might see " * " instead of a comma. This has been done intentionally for technical reasons. You can quickly replace the asterisks with commas using a quick replace tool. Before exporting translations from Unity, also double check that all your commas are followed by spaces. Example: Bad list — one, two, three. Good list — one, two, three.*

Fonts

IngoYG provides arrays of fonts for each language. If you drag an individual font to the zero element of the English language array for example, then if English localization is selected in the game, the fonts will be replaced with the specified font. As another example, you can also drag the font to the first element of the array. In this case, if you specify the **Font Index** field equal to one in the LanguagesYG component, then the font from the first element of the array will be used for this text.

The LanguagesYG component also has the **Unique Font** field. If you drag some font to this **Unique Font** field, then this text will always be loaded with the specified font.

For the **"Replace font with standard" button** to work, you need to specify a default font in InfoYG.

There is also a tool that enables you to replace fonts throughout a scene with the default font. You can find it in the top tab **YG → Localization → Font Default Masse.**

🛒 In-game purchases

The documentation in English is outdated, there is no translation yet. Watch the demo scene. Sorry!

📜 Deep Linking

You can pass any value to the game by a link using a hyper operator.

This way, you can, for example, use links to open a specific level in the game, grant an in-game bonus, or pass a value for game testing.

How to pass a value to the game

For example, the URL of your game: https://yandex.ru/games/app/012345

At the end of the URL, add: ?payload=

After the equal sign, add the value you want to pass to the game. Let's say the value is: debug123

The resulting link is: https://yandex.ru/games/app/012345?payload=debug123

Now, the game can get the "debug123" value and process it.

How to get and process a value

After initialization of the SDK, the passed value is written to the payload field.

Get the value as follows:

```
YandexGame.EnvironmentData.payload;
```

A regular link without Deep Linking won't pass any value to payload. In this case, the payload field will be empty.

You can see a clear example of Deep Links in the next section.

🐞 Debugging tool

You can easily build a debugging tool into your game where you can start calling the plugin's main methods. Roughly speaking, you will have a panel in the game with a "mini demo scene plugin". You can also see the drawing of blocks for advertising banners (RTB-Blocks).

What you need to do to run the game in debug mode

1. In Unity, find the **DebuggingModeYG** prefab. It is located in the **YandexGame folder → Prefabs**.
2. Add a prefab to the main scene in your project (which opens first when you start the game). The DebuggingModeYG prefab is a singleton object. A singleton is an object that will not be deleted when you move to another scene, and will not be recreated.
3. Configure DebuggingModeYG. Specify a value for **the Payload Password** parameter. This is the value that you will pass using Deep Linking. You can write a word, for example, debug and add your own password, for example, 123. The result is debug123.
4. Can you now run the game with a note**? payload=debug123** at the end of the game link, and it will start in debug mode.
5. When you run the game in debug mode, you should see a button to deploy the control panel in the lower right corner.

It should look like this:
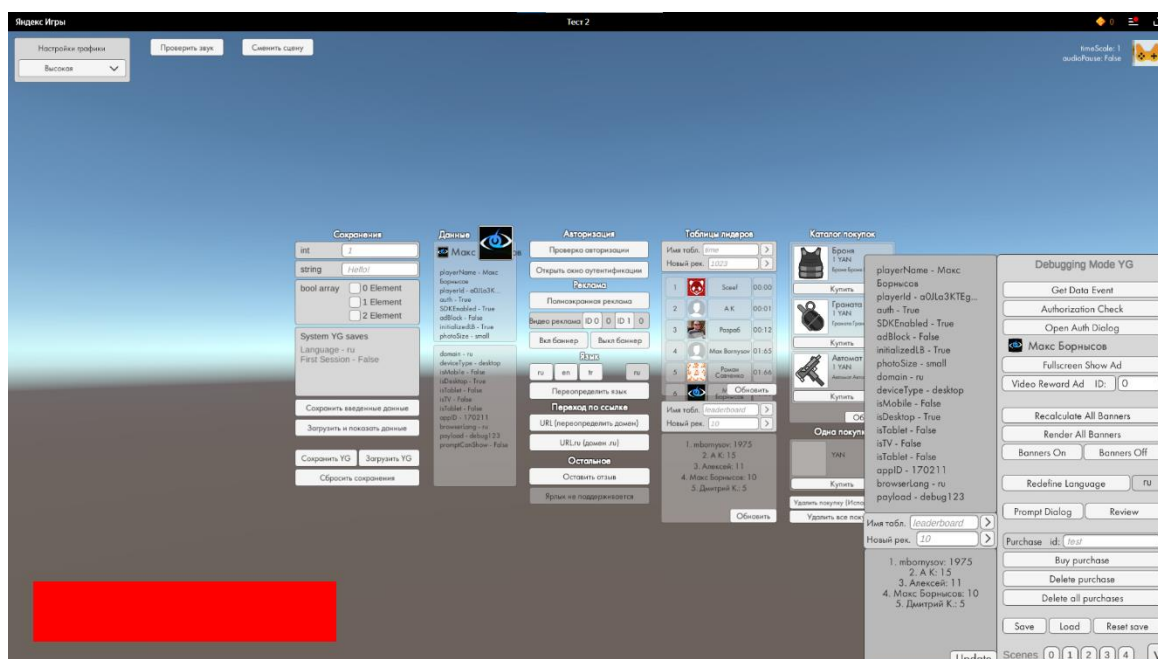
On the right side of the screenshot, you can see the control panel. On the left is a block in which the banner should be drawn.

💡 *Such red blocks are drawn only for dynamic banners!*

◆ Desktop shortcut

Using the native dialog box, you can offer the user to add a shortcut to the desktop — a link to the game.

Opening a dialog box

Use the Yandex Game method `YandexGame.PromptShow()`, to open a dialog box in which you will be prompted to install a shortcut.

The availability of the option depends on the platform, the internal rules of the browser and the limitations of the Yandex Games platform. In order to make sure that a shortcut can be added, the plugin uses the parameter `YandexGame.EnvironmentData.promptCanShow`. After initializing the SDK, this parameter will be true if the shortcut can be set. You can also use the parameter `promptCanShow` to write your own scripts for the shortcut, such as, for example, the PromptYG script. Or you can just use this script.

PromptYG script

In the folder with prefabs there is a ready-made, configured Desktop shortcut prefab with the PromptYG script. In the prefab, you only need to change the buttons to match the style of your game. The PromptYG script will show the desired button itself after initializing the SDK.

To use the PromptYG script, you need three buttons (three states):

Show Dialog — You can set a shortcut. In the ShowDialog state, a button will be shown that calls the method described above `PromptShow().`

Not Supported — The shortcut is not supported. If the status is Not Supported, a disabled button will be shown, inside which an explanation that the shortcut is not supported.

Done — The shortcut is already installed. When the status is Done, a disabled button will be shown, inside which an explanation that the shortcut is already installed.

The shortcut is installed

After the user installs the shortcut, two things will happen:

1. Parameter `YandexGame.savesData.promptDone` will be equal to `true`. With this parameter, when you try to install a shortcut again, you can check whether the user has already done this operation before. A reward can be given to a player for establishing a label. Therefore, in order not to reward the player twice, you can do a check using `promptDone`.
2. An event will be triggered `YandexGame.PromptSuccessEvent`. You can subscribe to it to reward the user for setting a shortcut. Or display a message about a successful operation.

☆ Game rating

You can ask the user to rate the game and write a comment in a pop-up window (will appear at the time of the rating request, closing the background of the application). A pop-up window will not be displayed if the user is not authorized or has previously evaluated the game.

To call the window to evaluate the game, use the YandexGame.ReviewShow(bool authDialog) method. This method will open a window to evaluate the game if such an option is available. Accepts a boolean type overload. Set it to true to open the authorization window if the user is not authorized.

This method can also be executed through the ReviewYG script, in which case the Auth Dialog value is set in the ReviewYG component.

In order to make sure that the pop-up window for evaluating the game can be displayed, the plugin uses the YandexGame parameter. EnvironmentData. reviewCanShow. After the SDK is initialized, this parameter will be true if you can open a window to evaluate the game. You can also use this reviewCanShow parameter to write your own scripts, such as the ReviewYG script. Or use this script.

In the folder with **YandexGame → Prefabs** there is a Review prefab with the **ReviewYG** script.

The ReviewYG script has four events:

Review Available - Perform the action that should occur if a review is available (if reviewCanShow = true). The action can be, for example, a show button.

Review Not Available - Perform the action that should occur if the revocation is **NOT** available (if reviewCanShow = false). The action can be, for example, hide the button.

Left Review - perform the action that should occur if the user left a review. For this, you can give a reward. You can also subscribe to this event through the YandexGame script. In the YandexGame component, this event is called ReviewDo.

Not Left Review - perform the action that should occur if the user did **NOT** leave a review, but closed the window to rate the game.

To write your scripts, use the YandexGame event. ReviewSentEvent.

When the user closes the window or leaves feedback, the ReviewSentEvent(bool sent) event will be called from the YandexGame script. The sent overload will be equal to:

true — if the user left a review;

false — if the user has closed the window. *In this case, the YandexGame* parameter. *reviewCanShow will become false for this game session.*