



# PluginYG Документация

## ✓ Содержание

[Содержание](#)

[Предисловие](#)

[Ссылки](#)

[Импорт плагина](#)

[Часто задаваемые вопросы](#)

[Начало работы](#)

[Добавление Yandex Game на сцену](#)

[Выбор WebGL шаблона](#)

[Настройки плагина \(\[Info YG\]\(#\)\)](#)

[Game Ready API](#)

[Настройки графики от плагина](#)

[Пространство имен YG](#)

[Работа с компонентом Yandex Game](#)

[Ивенты в компоненте Yandex Game](#)

[Инициализация](#)

[Полноэкранная реклама \(Fullscreen Ad\)](#)

[Реклама за вознаграждение \(Reward Ad\)](#)

[Пауза игры при просмотре рекламы](#)

[Простой способ настройки Viewing Ads YG](#)

[Sticky-баннер](#)

[Данные](#)

[Сохранения](#)

[Создание своих данных для сохранения](#)

[Загрузка сохранений](#)

[Сохранение](#)

[Сброс сохранений](#)

[Сохранение массива массивов](#)

[Лидерборды](#)

[Создание таблицы в консоли разработчика](#)

[Запись рекорда в соревновательную таблицу](#)

[Создание таблицы в Unity](#)

[Симуляция отображения таблицы в Unity Editor](#)

[Time тип](#)

[Локализация](#)

[Шрифты](#)

[Внутриигровые покупки](#)

[Симуляция отображения покупок в Unity Editor](#)

[Готовое решение](#)

[Обработка покупок](#)

[Компоненты покупок](#)

[Работа с кодом](#)

## Deep Linking

 Инструмент для отладки

 Ярлык на рабочий стол

[Вызов диалогового окна](#)

[Скрипт PromptYG](#)

[Ярлык установлен](#)

## Оценка игры

 Яндекс Метрика

[Настройка метрики](#)

[Отправка метрики](#)

## Пользовательский баннер

[Типы баннеров](#)

[Настройки WebGL шаблона для баннеров](#)

[Создание динамического баннера](#)

[Создание статических баннеров](#)

## Релизы

[Версия 1.5.2](#)

[Версия 1.5.1](#)

[Версия 1.5](#)

[Версия 1.4](#)

[Версия 1.3.6](#)

[Версия 1.3.5](#)

[Версия 1.3.4](#)

[Версия 1.3.3](#)

[Версия 1.3.2](#)

[Версия 1.3.1](#)

[Версия 1.3](#)

[Версия 1.2](#)

[Версия 1.1](#)

[Версия 1.0](#)



## Предисловие

Это документация по плагину для интеграции SDK Яндекс Игры в Ваши проекты. Плагин автоматизирован и сделан так, чтобы максимально облегчить интеграцию SDK.

Многие параметры в плагине описаны во всплывающих подсказках. Поэтому в данной документации будут игнорироваться такие параметры, и параметры названия которых говорят сами за себя.

Обратите внимания на демо сцену! В ней есть примеры всех функций и много скриптов для образца работы с плагином.

В среде разработки Unity плагин не имеет реального подключения к SDK Яндекса, но симулирует его. Ошибки в консоли при работе с плагином - это ненормальное явление!

Все инициализации и проверки плагин производит автоматически при запуске игры. Облачные сохранения и лидерборды работают как для авторизованных пользователей, так и для неавторизованных. После инициализации SDK Яндекс Игр плагин загружает все данные, включая загрузку сохранений игры и языка.

Плагин использует за основу [этот шаблон](#) для полного экрана. Он полностью избавляет от всех проблем с экраном. К тому же с ним вы можете сделать своё изображение для загрузочного экрана.

При тестировании игры не забывайте, пожалуйста, отключать AdBlock и другие аналогичные расширения. Также не забывайте активировать рекламу и подождать, пока она заработает.

В данной документации не будет описываться работа с настройками проекта под WebGL или даваться подробные описания работы с Яндекс Играми или РСЯ. Уже существуют статьи на подобные темы, и их вы сможете найти у меня на [Trello](#). Больше всего информации по Яндекс Играм Вы найдете в [официальном Telegram чате](#). Если у Вас возникают вопросы по Яндекс Играм, сначала попробуйте найти ответ с помощью поиска по официальному чату. Если проблема касается плагина, Вы так же можете попробовать найти её решение с помощью поиска в [чате по данному плагину](#).

Изначально был сделан видео-урок. Но на данный момент он не объясняет новые функции. Это не проблема, новые функции было бы логичнее объяснить в новых отдельных видео. Проблема в том, что некоторые моменты поменялись. Всё же, основная суть в видео остаётся неизменной! Поэтому советую прочитать документацию и посмотреть видео. В видео есть много полезной информации, которая хорошо усваивается визуально.

[https://www.youtube.com/watch?v=iS5a\\_LD0hv&t=92s](https://www.youtube.com/watch?v=iS5a_LD0hv&t=92s)

## Ссылки

Все обсуждения по плагину и оповещение об обновлениях в телеграм чате:

## PluginYG

Обсуждения по плагину

 [https://t.me/yandexgame\\_plugin](https://t.me/yandexgame_plugin)



*В закрепах найдете обновления и всю информацию в том числе ссылки на скачивание.*

## Asset Store

### PluginYG - Yandex Game integration

Get the PluginYG - Yandex Game integration package from Max Bornysov and speed up your game development process. Find this & other Add-Ons options on the Unity Asset Store.

 <https://assetstore.unity.com/packages/add-ons/pluginyg-yandex-game-integration-235877>



## GitHub

<https://github.com/JustPlay-Max/PluginYG>

**Вся информация по плагину и другие полезные ссылки на Trello.** Здесь вы сможете удобно посмотреть версии плагина, описание исправлений и дополнений для каждой версии, ссылки на скачивание, список реализованных функций и планы на будущее:

### Trello

Your browser was unable to load all of Trello's resources. They may have been blocked by your firewall, proxy or browser configuration. Press Ctrl+F5 or Ctrl+Shift+R to have your browser try again and if that doesn't work, check out our troubleshooting guide .

 <https://trello.com/b/Wd4wWOp1/yandexgame-plugin>

## Видео урок на Дзен:

JustPlay | Yandex Game Плагин | Реклама, Таблицы лидеров, Облачные сохранения, Защита от AdBlock и т.д.

Скачать плагин в телеграм чате: [https://t.me/yandexgame\\_plugin](https://t.me/yandexgame_plugin) Все сведения на Trello:  
<https://trello.com/b/Wd4wWOp1/yandexgame-plugin> Раздача...

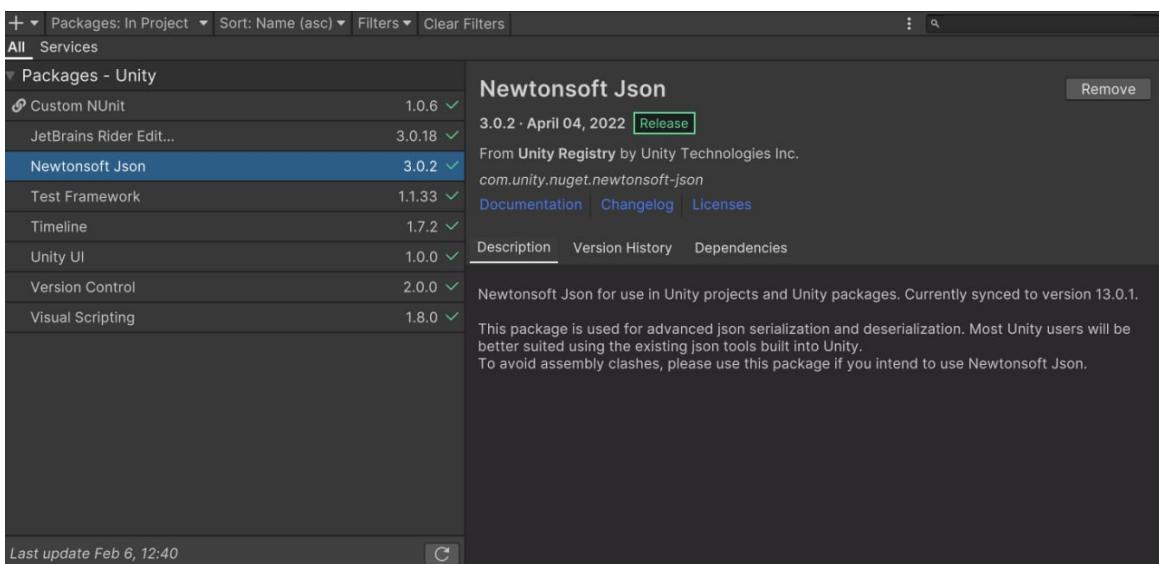
 <https://zen.yandex.ru/video/watch/6234f7331c98a06699258833>

## Библиотека Newtonsoft.Json .NET

С версии **1.5** для импорта данной библиотеки нужно лишь нажать на кнопку активации Newtonsoft в настройках плагина **Info YG**.

В ином случае, Вы можете импортировать библиотеку через Package Manager:

1. Нажмите на **+** в левом верхнем углу
2. Выберите **Add Package from git URL...**
3. Введите **com.unity.nuget.newtonsoft-json**
4. Нажмите **Add**



## ⬆️ Импорт плагина



Плагин поддерживается для Unity 2021 версии и выше!



Не перемещайте импортированные с плагином папки в проекте!

При импорте вы можете снять галочку для добавления в Ваш проект, допустим, демо сцены, если она Вам не нужна. Вы сможете легко обновлять плагин. Все файлы сами дополняются и изменяются как надо. Но иногда всё-таки бывают проблемы при обновлении плагина. В таком случае попробуйте удалить файлы плагина и импортировать по-новой.

Если Вы импортируете плагин в проект не первый раз, Вы можете убрать галочку напротив папки **YandexGame → WorkingData**, чтобы не слетели Ваши сохранения и настройки плагина, если таковые имеются.

После импорта у Вас могут возникнуть ошибки со словами **Newtonsoft**, **Json.net**, **LanguageYG**. Если такие ошибки есть, значит в вашем проекте отсутствует библиотека Newtonsoft.Json и вам необходимо её скачать (ссылки на скачивание выше, в разделе документации “[Ссылки](#)”).



Узнать версию плагина можно в файле **Readme** в папке **YandexGame** → **Documentation**.

## ❓ Часто задаваемые вопросы

### ▼ Нужно ли мне что то прописывать в **HTML**?

Нет. Для работы плагина Вам не нужно заходить в html и css скрипты. Но для собственного расширения функционала, конечно, вы можете что то изменять в любых скриптах плагина.

### ▼ Как узнать **устройство**, с которого запущена игра?

Нужно получить данные об устройстве (смотрите раздел “[Данные](#)”). Данные можно получать только после инициализации SDK (смотрите раздел “[Инициализация](#)”).

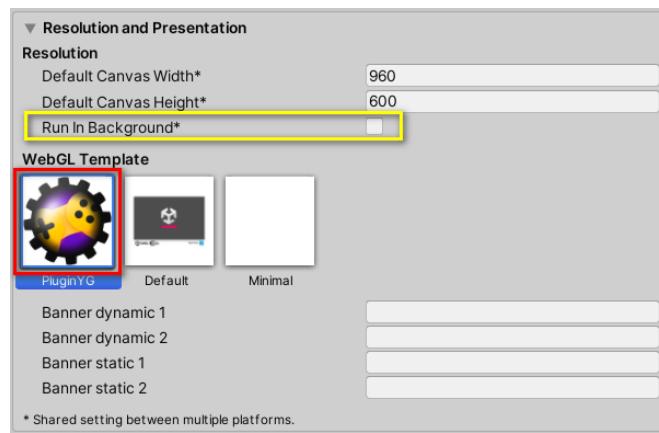
### ▼ Ошибки при **импорте** плагина?

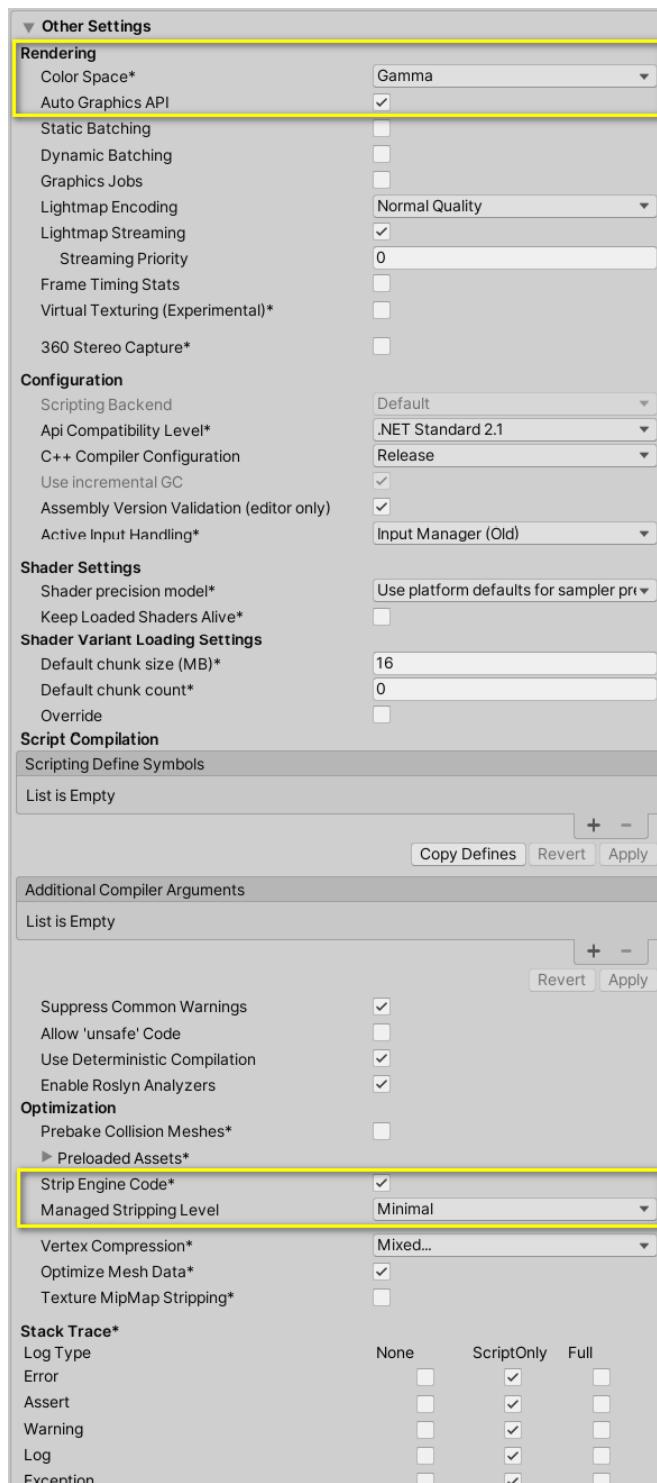
1. В проекте отсутствует библиотека **Newtonsoft.Json.net** (Смотрите раздел “[Импорт плагина](#)”)
2. Не корректно настроен компонент **Graphic Settings YG** (Смотрите раздел “[Настройки графики от плагина](#)”)
3. Иные ошибки могут быть из-за устаревшего ПО. Плагин работает на Unity 2021 версиях и выше.
4. Редактор кода не поддерживает синтаксис плагина. В таком случае может помочь обновление программы редактора кода или её замена.
5. Может помочь удаление папки Library.

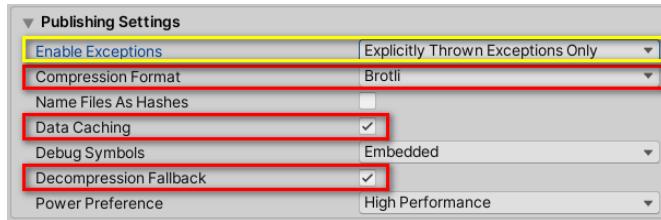
### ▼ Не билдится проект (**билд** происходит с ошибкой)

1. Не выбран или слетел шаблон плагина. (Смотрите раздел “[Выбор WebGL шаблона](#)”). Так же может помочь переоткрытие меню Project Settings, перезагрузка Unity, удаление папки WebGLTemplates и её реимпорт. И всё это вместе взятое в разной последовательности... К сожалению, в Unity не очень стабильно работают WebGL шаблоны.

2. Не правильно настроен проект. Вы можете найти рекомендации по настройке и оптимизации проекта в видео плагина и на Trello. (Смотрите раздел “[Ссылки](#)”). Настройки проекта находятся в **Project Settings → Player**. Ниже представлены скриншоты с примером настроек проекта. Красным цветом отмечены - необходимые параметры, жёлтым - рекомендуемые:







▼ Крашится игра при тестировании проекта в черновике Яндекса? (Вечная загрузка)

1. Не выбран или слетел шаблон плагина. (Смотрите раздел “[Выбор WebGL шаблона](#)”)
2. Использование методов и данных плагина до инициализации SDK. Необходимо инициализировать данные в соответствии с документацией. (Смотрите раздел “[Инициализация](#)”)
3. Не правильно настроен проект. ([Смотрите рекомендации по настройке проекта](#))
4. Возможно, по какой то причине, вам нужно установить задержку перед инициализацией SDK. За это отвечает параметр **SDK Start Delay** в настройках плагина **InfoYG**. Читайте описание данного параметра во всплывающей подсказке при наведении.
5. Убедитесь, что билд упакован в zip архив и целостность файлов внутри не нарушена.
6. Убедитесь, что Вы делаете билд в правильную папку, что Вы заливаете именно тот билд, что планировали.
7. Грузится Ваш старый, не рабочий билд (если таковой имелся). Требуется очистка кеша браузера, что бы запускался свежезалитый билд. Для Яндекс Браузера достаточно просто полностью закрыть браузер и открыть снова.

▼ Не отображается **шрифт** в WebGL билде?

Нельзя использовать дефолтный Unity шрифт, который по умолчанию назначен в компоненте текста. Нужно использовать какой-либо посторонний шрифт, который находится в проекте как файл, и назначить шрифт в компоненты текста.

Или, если Вы используете локализацию от плагина, в настройках InfoYG назначьте default шрифт в массиве шрифтов, тогда плагин сам будет применять данный шрифт в игре.

▼ Нет **звучка** на мобильном устройстве?

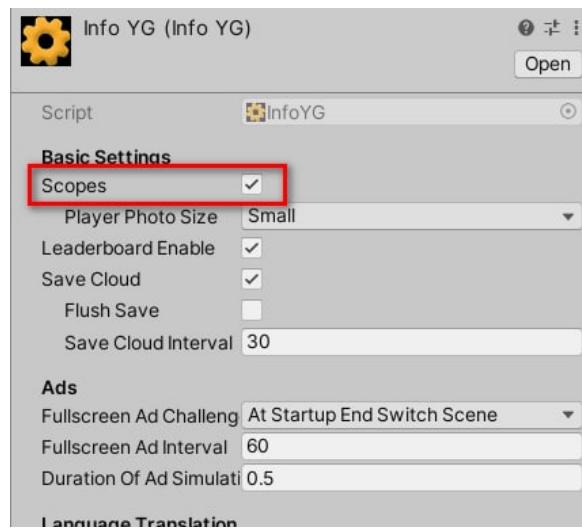
В игре на Unity звук может не работать в Яндекс браузере, Opera и других. Смотрите в сторону плагина [AudioYB](#). Так же, возможно, звуки будут исправно работать, если использовать [FMOD](#) (Требуется не только ассет, но и программа FMOD)

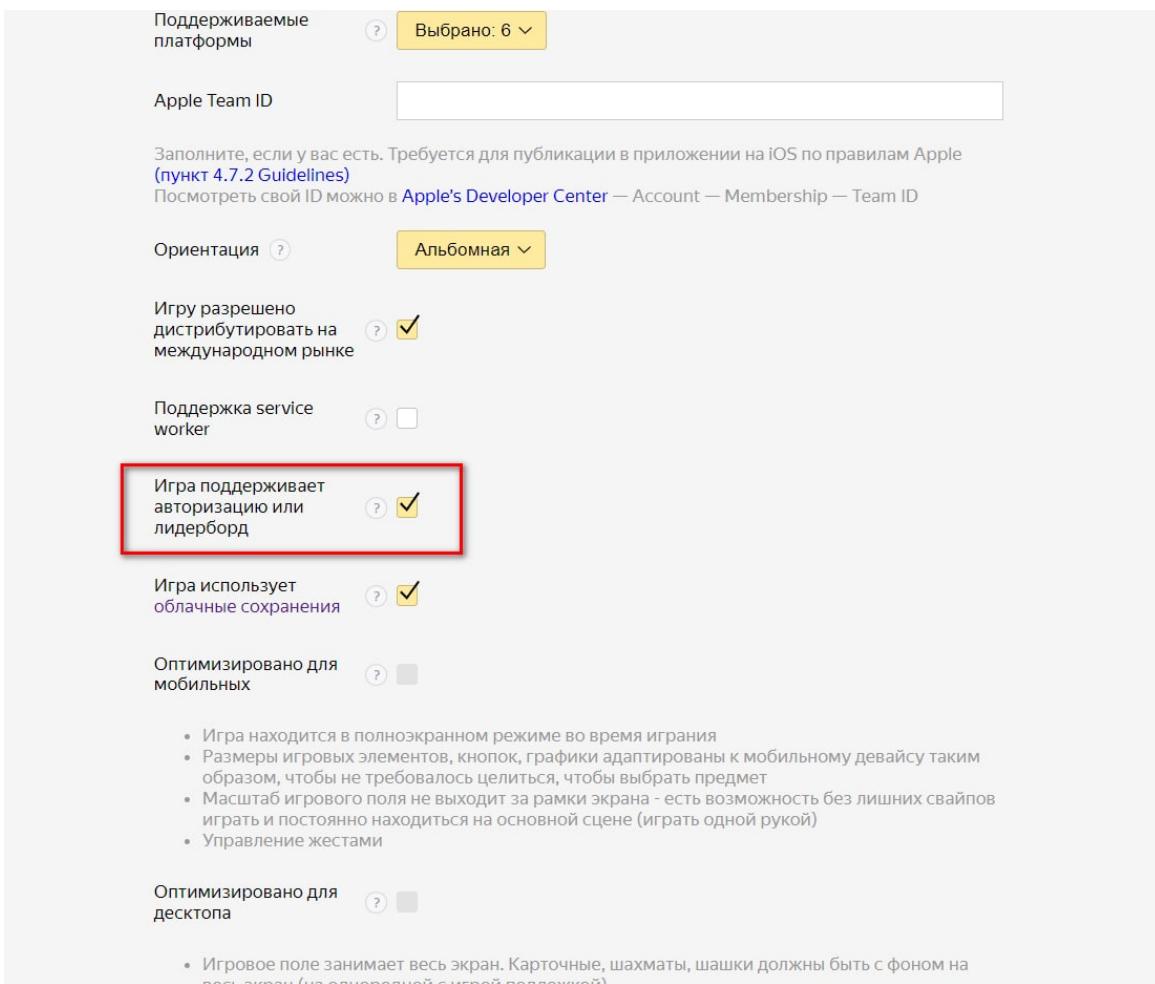
▼ Плеер (звуковой) из игры от браузера отображается в **нотификациях** (в шторке телефона)?

У клипов (звуков) в Unity параметр Load Type должен быть переключён на Decompress On Load.

▼ Не отображается ник или аватар игрока?

Должны быть включены данные параметры:





Если при включённых параметрах всё же не отображаются ник и аватар, значит:

1. При первом запуске игры в открывшемся окне не было дано разрешение на использование личных данных аккаунта.
2. Вышеописанное окно по каким то причинам не было показано и разрешение не было получено.

При любом из вариантов нужно попробовать протестировать черновик через другой аккаунт. Может быть поможет чистка кеша браузера или запуск игры в другом браузере.

▼ Нужно использовать Unity ниже **2021** версии?

Для этого Вы можете использовать версию плагина **0.4.4**. В ней меньше функций, но она рабочая.

▼ Как сделать вертикальную **ориентацию экрана**, изменить прогресс бар?

Найдите файл **style.css**, в нём прописаны такие данные. В css файле можно сменить соотношение экрана, размеры прогресс бара, его цвет. Более подробную информацию можно найти в поиске по чату [PluginYG](#) и в интернете.

[Также есть сторонние шаблоны.](#)

▼ Не работает **авто-локализация?**

Включите разрешение загрузки по протоколу HTTP для Unity Editor в **Project Settings**  
→ **Player** → **Allow downloads over HTTP**.

Попробуйте сменить в настройках плагина (**Domain Auto Localization**).

▼ Ошибка: компонент **Language YG** не найден?

У некоторых появляется неизвестная ошибка при попытке создания компонента **Language YG**. Должно помочь переименование скрипта в **Language**. Возможно, ошибка возникает только в 2022+ версии Unity.

## Начало работы

Обязательные пункты для функционирования плагина:

1. Добавить префаб **YandexGame** на каждую сцену (или воспользоваться Singleton).
2. Выбрать WebGL шаблон.
3. Заменить лого при загрузке игры на своё (обязательно).

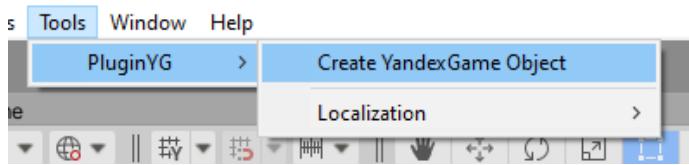
При этом Вы уже сможете загрузить билд Вашей игры в черновик Яндекс Игр, и все будет работать, и даже будет показываться Fullscreen реклама!



Рекламу нужно активировать в консоли разработчика для каждой игры! Как полноэкранную, так и за вознаграждение. После активации реклама заработает не сразу!

## Добавление Yandex Game на сцену

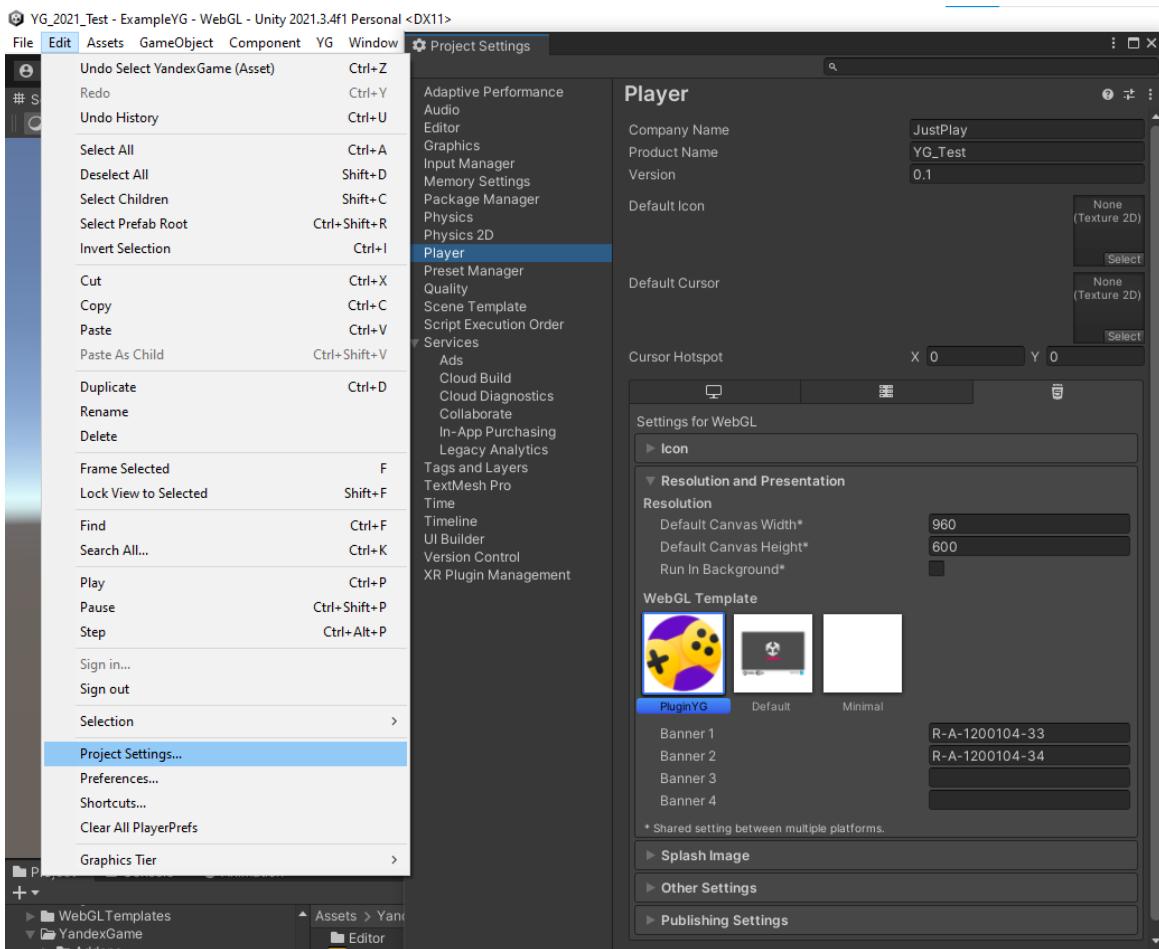
| Добавить префаб **YandexGame** на сцену удобнее всего таким образом:



Предфаб **YandexGame** – это объект с одноименными названием и скриптом, который является самым важным скриптом. Через него проходит весь обмен данными между SDK Яндекс Игр. В нём содержатся все методы и поля, которые Вам понадобятся для работы с плагином.

## Выбор WebGL шаблона

Выберите шаблон в **Project Settings** → **Player** → **Resolution and Presentation** → **WebGL Template**



Вы можете заменить изображения (логотип, задний фон), которые показываются на загрузочном экране игры. Для этого замените изображения на своё по пути **WebGLTemplates** →

**PluginYG** → **logo.png**, **background.png**. Имя файла должно оставаться неизменным. Так же Вы можете заменить эти файлы в готовом билде игры.

**logo.png** – логотип

**background.png** – задний фон (*поддерживается начиная с версии плагина 1.1*)

▼ Возможные проблемы с шаблоном

Если у Вас есть какие либо проблемы с шаблоном, попреключайте разные шаблоны и вернитесь к PluginYG. Важно отметить, что после переключений шаблонов заполненные поля пропадают, и их приходится записывать заново. После обновления проекта до новой версии Unity или обновления плагина, убедитесь в том, что поля не пропали.

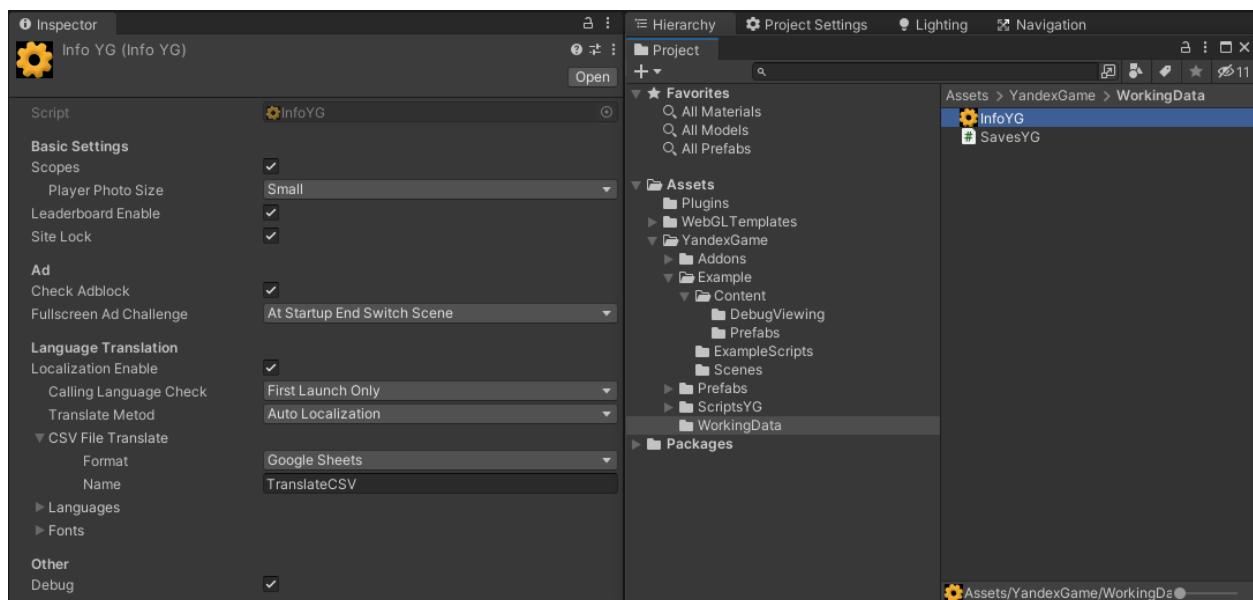


Изображение logo.png не нужно делать большого разрешения, иначе картинка растягивается за пределы нормы, и появится полоса прокрутки, что повлечёт за собой отклонение игры модераторами.

Изображение заднего фона background.png наоборот должен быть нормального размера, 1920 на 1080 нормально. Именно logo.png не нужно делать большим.

## Настройки плагина (Info YG)

Настройки плагина находятся в папке **WorkingData**, файл **InfoYG**. Также Вы сможете найти его на компоненте **YandexGame**.



В ходе документации я еще буду возвращаться к настройкам.

В дальнейшем настройки плагина я буду называть только **Info YG**.

## Game Ready API

Это необязательный метод из SDK Яндекс игр, он отображает момент, когда игра загрузила все ресурсы и готова к взаимодействию с пользователем.

В Info YG есть галка **Auto Game Ready API**. Если она включена, то плагин сам выполнит метод Game Ready API (по умолчанию она включена).

Если в Вашей игре имеются свои реализации загрузки игры, например, загрузка первой сцены, то Вам необходимо снять галку **Auto Game Ready API** и самостоятельно выполнять этот метод, когда игра будет полностью загружена. Выполнение метода: `YandexGame.GameReadyAPI();`

## Настройки графики от плагина

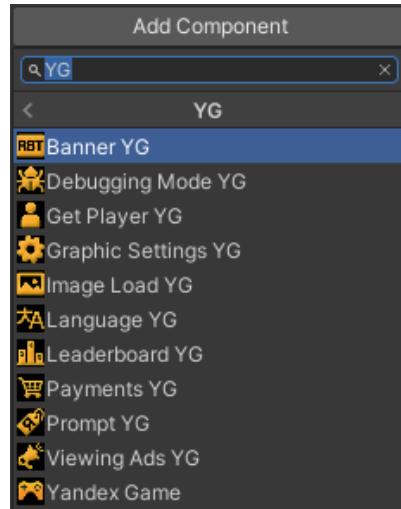
Готовые настройки графики **Graphic Settings YG** с переводом с помощью плагина и возможные ошибки при использовании данного скрипта

При запуске демо сцены, возможно, у Вас появятся ошибки. Это связано со скриптом **Graphic Settings YG**. Если он Вам не нужен, можете просто удалить его или деактивировать объект "Настройки плагина" в сцене. В противном случае, Вы должны настроить его. Для этого укажите количество полей у нужных массивов. Нужные массивы - это массивы с языками, на которые вы будете переводить игру. Количество полей в массивах должно соответствовать количеству графических пресетов в **Project Settings → Quality**. Для каждого нужного массива заполните поля названиями пресетов графики на определенном языке.

## Пространство имен YG

Все скрипты плагина, которые вам нужны, имеют иконку и приписку YG в конце имени.  
Остальные скрипты - это демонстрационные скрипты.

Все скрипты плагина находятся в пространстве имен YG. Значит Вы можете находить и добавлять компоненты на объект с помощью кнопки Add Component таким образом:



Также это значит: чтобы обратиться к скриптам плагина через Ваши скрипты, нужно подключать библиотеку YG таким образом: `using YG;`

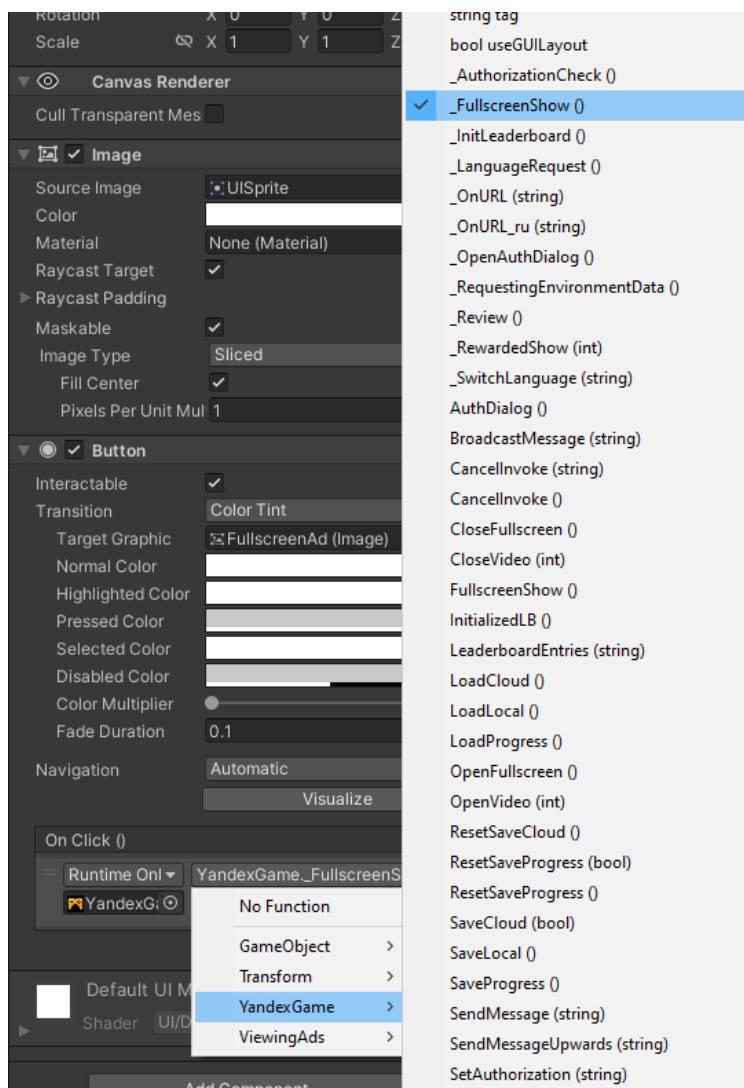
## Работа с компонентом Yandex Game

Все нужные Вам поля и методы есть в скрипте YandexGame.

Через скрипт методы вызываются таким образом *пример:* `YandexGame.SaveProgress()`

Для Unity Events есть дублирующие методы с нижнем подчеркиванием перед названием.

Скриншот ниже - пример того, как вызвать открытие блока Fullscreen рекламы при нажатии на кнопку в сцене:



Можно использовать таким образом только методы с нижним подчеркиванием \_.

## Определения методов:

### ▼ Authorization Check

Проверка на авторизацию (производится при старте игры, больше производить проверку не требуется).

### ▼ Buy Payments

Открыть окно для совершения покупки.

▼ Consume Purchase

Использовать покупку по её id, если будет найдена такая неиспользованная покупка.

▼ Consume Purchases

Использовать все оплаченные, но по какой то причине не использованные покупки.

▼ Full Screen Show

Вызов полноэкранной рекламы.

▼ Get Payments

Обновить информацию о покупках.

▼ Init Leaderboard

Инициализация соревновательных таблиц (инициализация производится при старте игры, больше производить инициализацию не требуется).

▼ Language Request

Запросить язык (запрос производится при старте игры, больше производить запрос не требуется).

▼ URL Yandex Define Domain

Открыть ссылку на игру или аккаунт разработчика. (домен определяется автоматически).

Ссылка получается такого формата: <https://yandex.определитьДомен.вашеПрожлдение>

- Пример для записи ссылки на игру: app/189792
- Пример для записи ссылки на аккаунт разработчика: developer?name=JustPlay

▼ Any URL

Открыть любую ссылку (Ваша прямая ссылка).

▼ Open Auth Dialog

Открыть диалоговое окно авторизации.

▼ **Prompt Show**

Показать диалоговое окно для установления ярлыка на рабочий стол.

▼ **Requesting Environment Data**

Получение дополнительных данных SDK Яндекс Игры (запрос производится при старте игры, больше производить запрос не требуется).

▼ **Reset Save Progress**

Сброс сохранений (работает в Unity Editor и в готовом билде).

▼ **Review Show**

Открыть окно для оценки игры. Принимает перегрузку boolean типа. Поставьте true, чтобы открывалось окно авторизации в случае если пользователь не авторизован.

▼ **Rewarded Show**

Показать видео рекламу за вознаграждение.

▼ **Sticky Ad Activity**

Активировать/деактивировать стики-баннеры.

▼ **Switch Language**

Сменить язык. Укажите в перегрузку string значение языка. Например, "ru", "en", или "tr".

После смены языка произойдет сохранение игры.

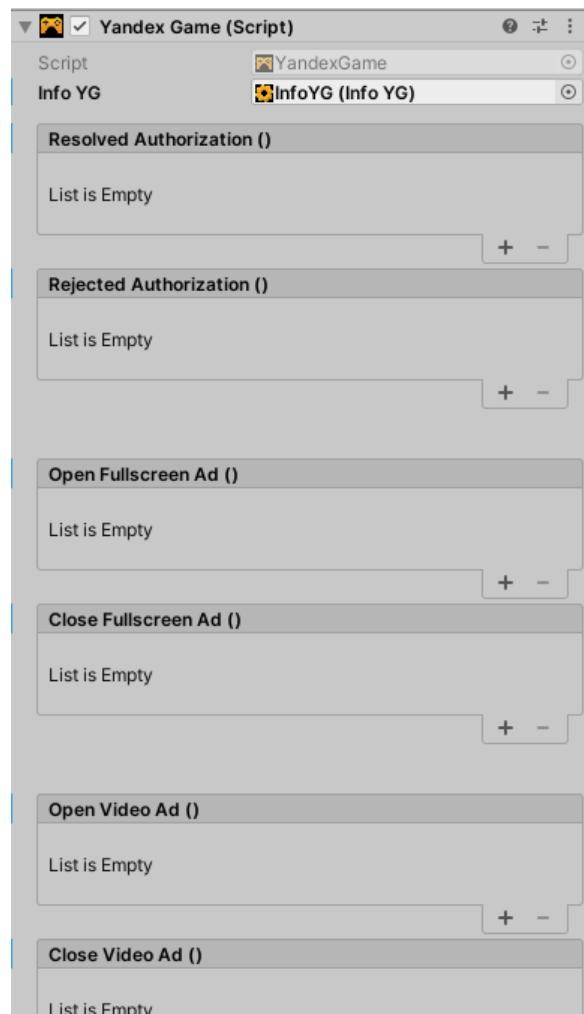


Работа с этими и другими методами подробно описана в соответствующих разделах документации.

## Ивенты в компоненте Yandex Game

На скриншоте ниже показаны такие же Unity Events, как и у стандартных кнопок из Unity UI. Вы можете повесить какие-то объекты и выбрать метод, который будет запускаться при

срабатывания какого-то из событий. Допустим, Вы можете выдать вознаграждение после просмотра видео рекламы.



Например, событие Open Fullscreen Ad вызывается при открытии Fullscreen рекламы. Если Вы повешаете на него свой скрипт со своим методом, то это будет означать, что Вы подписали свой метод на событие Open Fullscreen Ad. И теперь при каждом открытии Fullscreen рекламы будет вызываться метод, который Вы подписали. Соответственно, если будут подписанный метод/методы на Close Fullscreen Ad, то подписанный метод сработает при закрытии Fullscreen рекламы.

## Описание ивентов:

### ▼ Resolved Authorization

Вызывается при первом запуске игры после инициализации SDK, если игрок авторизован.

▼ Rejected Authorization

Вызывается при первом запуске игры после инициализации SDK, если игрок **не** авторизован.

▼ Open Fullscreen Ad

Вызывается при открытии полноэкранной рекламы.

▼ Close Fullscreen Ad

Вызывается при закрытии полноэкранной рекламы.

▼ Open Video Ad

Вызывается при открытии видео рекламы за вознаграждение.

▼ Close Video Ad

Вызывается при закрытии видео рекламы за вознаграждение.

▼ Reward Video Ad

Пользователь посмотрел и закрыл рекламу. Надо дать ему вознаграждение.

▼ Error Video Ad

Ошибка при открытии рекламы за вознаграждение.

▼ Purchase Success

Вызывается при совершении успешной оплаты покупки.

▼ Purchase Failed

Вызывается в случае, если оплата не будет произведена.

Событие может сработать если:

- В консоли разработчика не добавлен товар с таким id.
- Пользователь не авторизовался, передумал и закрыл окно оплаты.
- Истекло отведенное на покупку время, не хватило денег и т. д.

#### ▼ Prompt Do

Вызывается после успешной установки пользователем ярлыка на рабочий стол.

#### ▼ Review Do

Игрок оставил отзыв.

## Инициализация

Инициализацию плагин производит сам при запуске игры. Вам нужно только пользоваться данными и методами плагина после инициализации. Выполните нижеописанные действия при работе с данными или методами, такими как, например, сохранение игры.

После запуска игры плагин “запускается” не сразу. Если вам нужно обратиться к данным плагина при запуске игры или вызвать какой-то метод, сделайте это не в методе start, а в подписанным методе на событие `GetDataEvent`. Это событие вызывается после получения всех данных от SDK Яндекса.

Для проверки активности плагина есть поле `SDKEnabled` boolean типа.

Таким образом, если Вам нужно сделать что-то один раз только при запуске игры, просто сделайте это в методе, подписанным на событие `GetDataEvent`.

Если Вам нужно делать что-то каждый раз при загрузке новой сцены, Вы обычно просто делаете это в методе `Start` или `Awake`. Но если вы будете брать данные из плагина просто в методе `Start` или `Awake`, то при запуске игры данные, которые Вы берете, могут оказаться некорректными. Или метод, который вызываете, может вызвать ошибку и даже полный краш игры. Поэтому в таких случаях делайте проверку на активность плагина с помощью `SDKEnabled` и пользуйтесь описанным выше событием `GetDataEvent`. Рекомендуется следующая конструкция:

```
// Подписываемся на событие GetDataEvent в OnEnable
private void OnEnable() => YandexGame.GetDataEvent += GetData;

// Отписываемся от события GetDataEvent в OnDisable
private void OnDisable() => YandexGame.GetDataEvent -= GetData;

private void Awake()
{
    // Проверяем запустился ли плагин
    if (YandexGame.SDKEnabled == true)
    {
        // Если запустился, то запускаем Ваш метод
        GetData();

        // Если плагин еще не прогрузился, то метод не запуститься в методе Start,
        // но он запустится при вызове события GetDataEvent, после загрузки плагина
    }
}

// Ваш метод, который будет запускаться в старте
```

```
public void GetData()
{
    // Получаем данные из плагина и делаем с ними что хотим
}
```



## Полноэкранная реклама (Fullscreen Ad)

В Info YG есть настройка **Ad When Loading Scene**.

По умолчанию **Ad When Loading Scene = true** — это значит, что показ рекламы будет вызываться при загрузке любой сцены в игре. Значение **false** — реклама не будет показываться при загрузке сцен.

Независимо от выбора параметра **Ad When Loading Scene**, по умолчанию первая реклама показывается еще до загрузки игры, это можно настроить с помощью параметра **Show First Ad** в Info YG.



В Unity Editor симуляция рекламы не будет происходить при запуске игры, это отключено, что бы не мозолило глаза. Так же, что бы симуляция рекламы Вас не отвлекала, Вы можете уменьшить параметр **Fullscreen Ad Interval** в Info YG.

Не переживайте на счет того, что вызов рекламы будет происходить слишком часто или на счёт двойного открытия рекламного блока. Плагин обрабатывает такие ситуации.

Для вызова полноэкранной рекламы через скрипт есть метод `FullscreenShow()`.

Вы можете подписаться на событие открытия полноэкранной рекламы `OpenFullAdEvent()` и на событие закрытия `CloseFullAdEvent()`.



## Реклама за вознаграждение (Reward Ad)

Для вызова видео-рекламы через скрипт есть метод `RewVideoShow( int id )`.

Вы можете подписаться на событие открытия видео-рекламы `OpenVideoEvent` и на событие закрытия `CloseVideoEvent`.

Так же есть событие `ErrorVideoEvent`. Подпишитесь на него, если хотите уведомить игроков о неудачном воспроизведении рекламы за вознаграждение.

Используйте событие `RewardVideoEvent( int id )` для вознаграждения игрока за просмотр рекламы.

Метод вызова видео рекламы (`RewVideoShow`) принимает одно значение типа `integer`. Это ID рекламы. Он нужен для нескольких видов вознаграждения.

Допустим, у Вас есть вознаграждение “+100 монет” и вознаграждение “+оружие”.

При вызове видео-рекламы за “+100 монет” запишите ID как 1 `RewVideoShow( 1 )`.

А для вознаграждения “+оружие” запишите ID как 2 `RewVideoShow( 2 )`.

В своём скрипте подпишите свой метод вознаграждения на событие `RewardVideoEvent`.

Подписанный метод должен принимать одно значение типа `integer`. Это значение и будет ID, которое вернётся тем числом, которое мы записывали при вызове рекламы. И в подписанным методе сделайте проверку. Если ID = 1, то выдаём “+100 монет”. Если ID = 2, то выдаём “+оружие”.

В демо сцене есть простой демо скрипт `RewardedAd`. Вы можете делать по его примеру, или по этому:

```
using YG;

// Подписываемся на событие открытия рекламы в OnEnable
private void OnEnable() => YandexGame.RewardVideoEvent += Rewarded;

// Отписываемся от события открытия рекламы в OnDisable
private void OnDisable() => YandexGame.RewardVideoEvent -= Rewarded;

// Подписанный метод получения награды
void Rewarded(int id)
{
    // Если ID = 1, то выдаём "+100 монет"
    if (id == 1)
        AddMoney();

    // Если ID = 2, то выдаём "+оружие".
    else if (id == 2)
        AddWeapon();
}

// Метод для вызова видео рекламы
void ExampleOpenRewardAd(int id)
{
    // Вызываем метод открытия видео рекламы
    YandexGame.RewVideoShow(id)
}
```



Если у Вас всего одно вознаграждение, Вы можете просто записывать ID как 0 и не делать проверок внутри своего метода для вознаграждения.



## Пауза игры при просмотре рекламы

По умолчанию на префабе **Yandex Game** висит скрипт **Viewing Ads YG**. При просмотре полноэкранной или видео-рекламы данный скрипт ставит на паузу звук, временной масштаб,

скрывает курсор в игре или всё вышеперечисленное, на Ваш выбор (за это отвечает опция Pause Type).

---

## Pause Type

**Audio Pause** — Ставить звук на паузу при просмотре рекламы.

**Time Scale Pause** — Выставить временную шкалу на 0 (остановить время) при просмотре рекламы.

**Cursor Activity** — Скрывать курсор.

**All** — Ставить на паузу и звук и время, скрывать курсор при просмотре рекламы.

**Nothing To Control** - Не контролировать никакие параметры (подпишите свои методы в опции Custom Events).

---

## Pause Method

▼ **Remember Previous State** — Ставить паузу при открытии рекламы. После закрытия рекламы звук и/или временная шкала придут в изначальное значение (до открытия рекламы).



При использовании **Remember Previous State** на каждой сцене компонент ViewingAdsYG должен выглядеть одинаково, с одними и теми же настройками.



Метод **Remember Previous State** может вызывать конфликты из-за чего может возникать не корректная работа скрипта ViewingAdsYG. Более простой и устойчивый метод — это **Custom State**. Используйте его, если испытываете трудности.

▼ **Custom State** — Укажите свои значения, которые будут выставляться при открытии и закрытии рекламы.

▼ **Opening AD Values** — Установить значения при открытии рекламы.

**Time Scale** — Значение временной шкалы при открытии рекламы.

▼ **Closing AD Values** — Установить значения при закрытии рекламы.

**Time Scale** — Значение временной шкалы при закрытии рекламы.

**Audio Pause** — Значение аудио паузы при закрытии рекламы.

**Cursor Visible** — Показать или скрыть курсор при закрытии рекламы?

**Cursor Lock Mode** — Выберите мод блокировки курсора при закрытии рекламы.



При выборе опции **Nothing To Control** в **Pause Type** — без разницы что будет выбрано в **Pause Method**.

▼ **Custom Events** — Ивенты для кастомных методов.

**Open Ad** — Подпишите свой метод, который будет выполняться при **открытии** полноэкранной рекламы или рекламы за вознаграждение.

**Close Ad** — Подпишите свой метод, который будет выполняться при **закрытии** полноэкранной рекламы или рекламы за вознаграждение.



Можно иметь несколько компонентов ViewingAdsYG на сцене/на объекте. Чтобы, например, один компонент отвечал за контроль звука, второй за контроль курсором. Но не делайте несколько компонентов, если используете Pause Type — All.

**Do Close Void On Awake** — Выполнить метод закрытия рекламы (**Closing AD Values** в **Viewing Ads YG**) в методе **Awake** (то есть при старте сцены).

Это позволит не прописывать события вроде аудио пауза = false или timeScale = 1 в ваших скриптах в методах Start.

## Простой способ настройки Viewing Ads YG

1. Не прописывайте в своих скриптах в методах в **Start**, **Awake** или **On Enable** параметры такие как: `AudioListener.pause;` `Time.timeScale;` `Cursor.visible;` `Cursor.lockState;`
2. В компоненте **Viewing Ads YG** установите параметр **Pause Method = Custom State**.
3. Поставьте галку **Do Close Void On Awake**.
4. Настройте параметры из раздела **Closing AD Values** как вам нужно для конкретной сцены в момент закрытия рекламы и при старте сцены.



## Sticky-баннер

Чтобы включить показ sticky-баннера:

1. Откройте консоль разработчика и перейдите на вкладку **Черновик**.
2. В блоке **Sticky баннеры** настройте отображение баннеров:

- Для мобильных устройств — в поле **Sticky-баннер в портретной ориентации** выберите расположение **Внизу** или **Вверху**.
- Для планшетов — в поле **Sticky-баннер в альбомной ориентации** выберите расположение **Внизу** или **Вверху**.
- Для компьютеров — включите опцию **Sticky-баннер на десктопе**. Баннер будет показываться справа.

По умолчанию sticky-баннер появляется при запуске игры и отображается всю сессию. Чтобы настроить момент показа баннера:

1. В блоке **Sticky баннеры** включите опцию **Не показывать sticky-баннер на старте**.
2. Используйте API управления показом Sticky баннера.

## Управление показом Sticky баннера

Активируйте или деактивируйте Sticky баннер с помощью метода:

```
YandexGame .StickyAdActivity( bool activity)
```

Перегрузка `activity` задаёт активность Sticky баннера:

`StickyAdActivity( true )` — включить Sticky баннер;

`StickyAdActivity( false )` — выключить Sticky баннер.



## Данные

Вы можете получить имя игрока, его аватар, девайс пользователя и т.д.

Большинство данных берутся напрямую из класса `YandexGame`. Другие объекты и параметры SDK в отдельном классе `YandexGame .EnvironmentData`. Также есть класс для сохранений игры и класс для внутриигровых покупок.

### ▼ `YandexGame .SDKEnabled`

типа `boolean`

- `true` — Если SDK загрузился;
- `false` — Если SDK не загрузился.

### ▼ `YandexGame .auth`

типа `boolean`

- `true` — Если пользователь авторизован;
- `false` — Если пользователь не авторизован.

▼ `YandexGame .playerName`

типа `stringify`

- “имя игрока” — Если игрок авторизован;
- “`anonymous`” — Если игрок не авторизован.

▼ `YandexGame .playerId`

типа `stringify`

- `ID игрока`

▼ `YandexGame .adBlock`

типа `boolean`

- Активность функции `Check AdBlock`

▼ `YandexGame .initializedLB`

типа `boolean`

- `true` — Если лидерборды инициализированы;
- `false` — Если лидерборды не инициализированы.

▼ `YandexGame .playerPhoto`

типа `stringify`

- Ссылка **изображения аватарки игрока**

▼ `YandexGame .photoSize`

типа `stringify`

- **Размер** подкаченного **изображения** пользователя. Возвращает значение параметра **Player Photo Size**, которое вы выбираете в InfoYG.

▼ `YandexGame .nowAdsShow`

типа `boolean`

- `true` — Полнояркная или видео реклама **открыта** в данный момент.

- `false` — Полнэкранная или видео реклама **закрыта** в данный момент.

▼ `YandexGame .nowFullAd`

типа `boolean`

- `true` — Полнэкранная реклама **открыта** в данный момент.
- `false` — Полнэкранная реклама **закрыта** в данный момент.

▼ `YandexGame .nowVideoAd`

типа `boolean`

- `true` — Видео реклама за вознаграждение **открыта** в данный момент.
- `false` — Видео реклама за вознаграждение **закрыта** в данный момент.

---

▼ `YandexGame . EnvironmentData .language`

типа `stringify`

- **Язык.** Поддерживаемые языки вы можете посмотреть в [официальной документации Яндекс Игр.](#)

▼ `YandexGame . EnvironmentData .domain`

типа `stringify`

- **Домен.** Поддерживаемые домены вы можете посмотреть в [официальной документации Яндекс Игр.](#)

▼ `YandexGame . EnvironmentData .deviceType`

типа `stringify`

**Устройство пользователя.** Возвращает одно из значений:

- `"desktop"` (компьютер)
- `"mobile"` (мобильное устройство)
- `"tablet"` (планшет)
- `"tv"` (телевизор)

▼ `YandexGame . EnvironmentData .isMobile`

типа `boolean`

- `true` — мобильное устройство;
- `false` — иное устройство.

▼ `YandexGame . EnvironmentData .isDesktop`

типа `boolean`

- `true` — компьютер;
- `false` — иное устройство.

▼ `YandexGame . EnvironmentData .isTablet`

типа `boolean`

- `true` — планшет;
- `false` — иное устройство.

▼ `YandexGame . EnvironmentData .isTV`

типа `boolean`

- `true` — телевизор;
- `false` — иное устройство.

▼ `YandexGame . EnvironmentData .appID`

типа `stringify`

- **ID игры**

▼ `YandexGame . EnvironmentData .browserLang`

типа `stringify`

- **Язык браузера**



Рекомендуется использовать `YandexGame . EnvironmentData .language` (Структура i18n).

▼ YandexGame . EnvironmentData .payload

тип `stringify`

- О параметре payload читайте в разделе **Deep Linking**

▼ YandexGame . EnvironmentData .promptCanShow

тип `boolean`

- Используется для того, чтобы убедиться, что ярлык можно добавить.

▼ YandexGame . EnvironmentData .reviewCanShow

тип `boolean`

- Используется для того, чтобы убедиться, что отзыв можно оставить.

---

▼ YandexGame . savesData .isFirstSession

тип `boolean`

- Техническое поле для плагина. Становится `true` после первой инициализации игры.

▼ YandexGame . savesData .language

тип `stringify`

- Язык возвращаемый SDK Яндекс Игр. Поддерживаемые языки вы можете посмотреть в [официальной документации Яндекс Игр](#)

▼ YandexGame . savesData .promptDone

тип `boolean`

- Становится `true`, когда ярлык игры установлен.

---

▼ YandexGame . purchases

тип `Purchase[]`

- Все товары (информация о товарах).

▼ YandexGame . Purchase .id

типа `stringify`

- **Идентификатор товара**, который Вы записывали при создании товара в консоли разработчика.

▼ `YandexGame . Purchase .title`

типа `boolean`

- **Название** товара.

▼ `YandexGame . Purchase .description`

типа `stringify`

- **Описание** товара.

▼ `YandexGame . Purchase .imageURIimageURI`

типа `stringify`

- **URL изображения** товара.

▼ `YandexGame . Purchase .priceValue`

типа `stringify`

- **Стоимость** товара.

▼ `YandexGame . Purchase .consumed`

типа `boolean`

- **Использована ли покупка**
- `true` — использована;
- `false` — не использована.

## Сохранения

В плагине предусмотрено 3 вида сохранений:

1. Для тестирования в Unity Editor — при разработке игры в Unity, данные сохраняются в файл `saveyg.yg` ([подробнее ниже](#)).

- Облачные сохранения — сохранения на облако Яндекса. Для работы облачных сохранений необходимо в черновике поставить галочку напротив “Игра использует облачные сохранения”.
- Локальные сохранения — если данные не могут сохраниться или загрузиться из облака Яндекс Игр по каким то причинам, то используются локальные сохранения (Local Storage). Если Вы хотите использовать только локальные сохранения, отключите облачные сохранения в настройках плагина InfoYG → параметр **Save Cloud**.

Все виды сохранений работают автоматически, Вам нужно только сделать следующие действия:

- Создать свои данные для сохранений в отдельном скрипте “savesData”.
- В Вашем скрипте присвоить созданным Вами сохранениям в классе “savesData” - значения, которые требуется сохранить.
- Выполнить метод сохранения.
- Загрузка сохранений происходит следующим образом: берёте из класса “savesData” значения нужных данных и делаете с ними что требуется. Если нужно получать данные сразу при старте игры, то это нужно делать после инициализации SDK, т.к. сохранённый ранее прогресс игрока загружается в класс “savesData” - после инициализации SDK.



Данные хранятся в Json формате, так что сохраняются даже массивы. Вы можете добавлять новые поля сохранений в новой версии игры, и после загрузки игры в Яндекс - ничего не слетит.

## Создание своих данных для сохранения

- Откройте скрипт **SavesYG**, расположенный по пути **YandexGame** → **WorkingData**.
- Под комментарием `// Ваши сохранения` уже есть некоторые поля. Это поля для демо-сцены, вы можете их удалить.
- Запишите свои поля. Вы можете задать им какие-то значения, тогда эти значения будут дефолтными, и при первой загрузке игры данные будут равные дефолтным значениям.

```
1  namespace YG
2  {
3      [System.Serializable]
4      public class SavesYG
5      {
6          public bool isFirstSession = true;
7          public string language = "ru";
8
9          // Ваши сохранения
10         public int money = 1;
11         public string newPlayerName = "Hello!";
12         public bool[] openLevels = new bool[3];
13     }
14 }
15
16
```

## Загрузка сохранений

Загрузка сохранений происходит автоматически после инициализации SDK. Класс с сохранениями статический. Это значит, что даже если в игре переключится сцена, данные не изменятся. Поэтому Вам никогда не нужно будет делать метод загрузки (`LoadProgress`).

Чтобы получить сохранения игры, просто делайте это как с обычными данными плагина через скрипт YandexGame и его статический класс savesData: `YandexGame . savesData . вашеПоле;`

Чтобы получить сохранения игры в старте, делайте это после инициализации в уже рассмотренном нами событии `GetDataEvent` (В разделе “Инициализация”).

## Сохранение

Для сохранение игрового прогресса есть метод `YandexGame . SaveProgress();`

Но прежде чем сохранить данные, нужно записать их в класс `savesData`.

В демо-сцене вы найдете хороший скрипт для примера “SaverTest”. Вот еще пример:

```
// Подписываемся на событие GetDataEvent в OnEnable
private void OnEnable() => YandexGame.GetDataEvent += GetLoad;

// Отписываемся от события GetDataEvent в OnDisable
private void OnDisable() => YandexGame.GetDataEvent -= GetLoad;

private void Start()
{
    // Проверяем запустился ли плагин
    if (YandexGame.SDKEnabled == true)
    {
        // Если запустился, то выполняем Ваш метод для загрузки
        GetLoad();

        // Если плагин еще не прогрузился, то метод не выполнится в методе Start,
        // но он запустится при вызове события GetDataEvent, после прогрузки плагина
    }
}
```

```

}

// Ваш метод для загрузки, который будет запускаться в старте
public void GetLoad()
{
    // Получаем данные из плагина и делаем с ними что хотим
    // Например, мы хотим записать в компонент UI.Text сколько у игрока монет:
    textMoney.text = YandexGame.savesData.money.ToString();
}

// Допустим, это Ваш метод для сохранения
public void MySave()
{
    // Записываем данные в плагин
    // Например, мы хотим сохранить количество монет игрока:
    YandexGame.savesData.money = money;

    // Теперь остаётся сохранить данные
    YandexGame.SaveProgress();
}

```

## Сброс сохранений

Файл сохранений для тестирования игры в Unity сохраняется в папке **YandexGame**

→ **WorkingData**. Вы можете сбросить сохранения, для этого удалите файл **saveyg.yg** в папке **WorkingData**.

Есть метод для сброса сохранений `YandexGame .ResetSaveProgress()`. Он не удаляет файл **saveyg.yg**, он сбрасывает все сохранения до дефолтных значений. Также данный метод сбрасывает сохранения и в готовом билде игры.

После сброса, сохранение не происходит. Если Вы хотите, чтобы после сброса данные сразу же сохранились в дефолтном состоянии - выполните метод сохранения `YandexGame .SaveProgress();`

## Сохранение массива массивов

Для сохранения массива массивов необходимо использовать класс **JsonConvert** из библиотеки **JsonNet**. Для этого есть возможность переключаться между **JsonUtility** и **JsonConvert**.

Недостаток класса **JsonConvert** в том, что при его использовании в билд проекта добавляется библиотека **JsonNet** и конечный вес игры увеличивается на ~2мб.

Преимущество класса **JsonConvert** — сохранение массива в массиве. Используйте **JsonUtility**, если Вам не нужно сохранять массив в массиве, для этого ничего не требуется делать, **JsonUtility** используется по умолчанию.

Чтобы использовать **JsonConvert**: нажмите соответствующую кнопку активации в настройках плаcтина **Info YG**.



## Лидерборды



Для версий ниже 1.5 смотрите документацию внутри плагина.

## Создание таблицы в консоли разработчика

Перейдите в раздел **Лидерборды** и запишите **Техническое название** соревновательной таблицы. **Локализация наименования** нам не нужна, а остальные опции имеют пояснения.

## Запись рекорда в соревновательную таблицу

Это делается с помощью метода:

```
YandexGame . NewLeaderboardScores ( string "техническое название таблицы", int новый рекорд);
```

### ▼ Дополнительные опции при работе с кодом

После инициализации SDK и инициализации лидербордов поле boolean типа

`YandexGame .initializedLB` будет равно true.

После инициализации можно выполнять метод `YandexGame .GetLeaderboard`. Метод имеет параметры необходимые для заполнения.

После выполнения метода `GetLeaderboard` вызовется событие `YandexGame .onGetLeaderboard`. Оно передаёт класс **LB Data** содержащий следующую информацию о лидерборде:

#### ▼ `technoName`

типа `stringify`

- Техническое название таблицы

#### ▼ `entries`

типа `stringify`

- Описание таблицы в тексте.

Это тот текст, который записывается в таблицу при выборе простого режима (**Advanced = false**).

▼ `isDefault`

типа `boolean`

- Является ли основным лидербордом

▼ `isInvertSortOrder`

типа `stringify`

- Сортировка

`false` = сортировка по убыванию

`true` = сортировка по возрастанию

▼ `decimalOffset`

типа `integer`

- Размер десятичной части счёта

Это число определяет, сколько знаков из целого числа счета должны быть отображены после запятой.

Например, значение 5712 в лидерборде при размере десятичной части равном 2 будет отображено как 57. 12.

▼ `type`

типа `stringify`

- Тип таблицы: `numeric` или `time`

▼ `thisPlayer`

класс `LBThisPlayerData`

Это информация о пользователе аккаунта (игрока играющего в игру)

Имеет два параметра:

- `rank`

- **score**

▼ `players`

класс `LBPlayerData[]`

- **Список игроков.**

Массив игроков в таблице лидеров. Элементы массива содержат информацию о пользователе.

Класс **LB Player Data** содержит следующую информацию о игроке:

▼ `rank`

типа `integer`

- **Ранк игрока (позиция в таблице).**

▼ `name`

типа `stringify`

- **Ник пользователя.**

▼ `score`

типа `integer`

- **Рекорд пользователя**

▼ `photo`

типа `stringify`

- **Ссылка на аватар пользователя.**

▼ `uniqueID`

типа `stringify`

- **Уникальный идентификатор пользователя.**



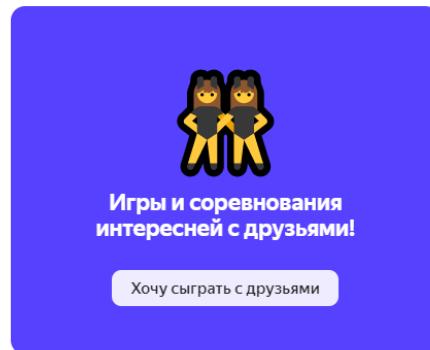
В таблицу запишется новое число, даже если оно меньше предыдущего. Необходимо самостоятельно выполнять сравнение рекорда перед новой записью.

При этом у Вас уже будет существовать соревновательная таблица, которая будет отображаться на странице Вашей игры таким образом:

### Лучшие игроки

|      |  |                    |         |
|------|--|--------------------|---------|
| 1    |  | Матвей Долганин    | 1387645 |
| 2    |  | Рейм Кьювир        | 1116519 |
| 3    |  | Zzzz Zzzz          | 1040476 |
| 1387 |  | Анастасия Левченко | 7647    |
| 1388 |  | Пользователь скрыт | 7646    |
| 1389 |  | mbornysov          | 7645    |
| 1390 |  | Владимир           | 7637    |
| 1391 |  | Михаил К.          | 7626    |
| 1392 |  | Ann                | 7617    |
| 1393 |  | Ангелина Харченко  | 7615    |

### Вызов друзьям



Запрос можно отправлять не чаще, чем раз в секунду. В противном случае он будет отклонен.

## Создание таблицы в Unity

Отображением таблицы занимается скрипт **Leaderboard YG**. Все его опции имеют всплывающие подсказки. Обязательно заполните параметр **Name LB** — это техническое название таблицы, которое мы записывали в консоли разработчика.

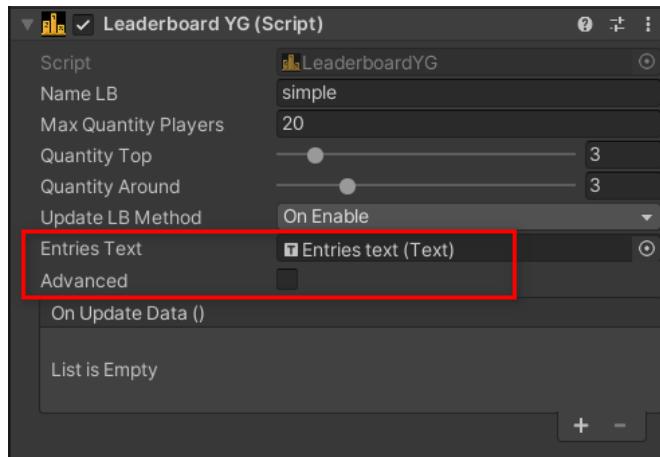
В папке **YandexGame → Prefabs → Leaderboards** расположены примеры таблиц, возьмите их за основу.



Префаб таблицы нужно поместить в **Canvas** объект.

### ▼ Простой вариант отображения таблицы

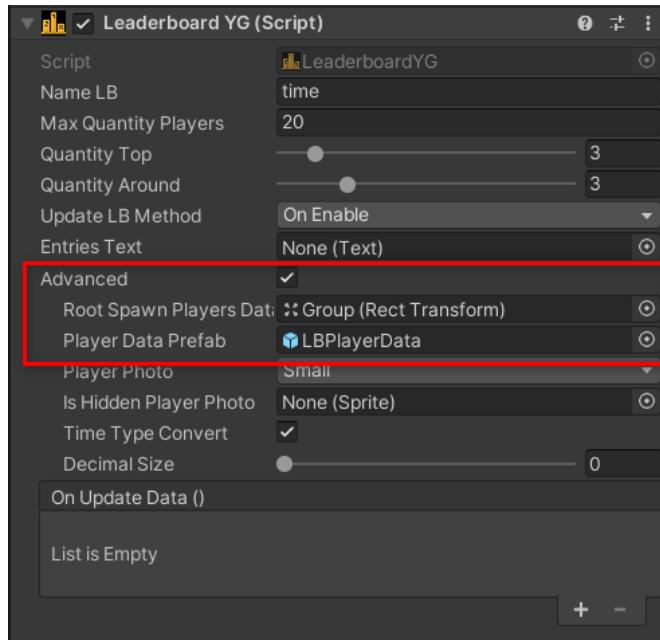
Для режима **простого** отображения рекордов достаточно указать в компоненте **Leaderboard YG** ссылку на компонент **Text** в опции **Entries Text**, галочка **Advanced** должна быть отключена для простой таблицы. И скрипт сам будет записывать в него данные таблицы.



#### ▼ Продвинутый вариант отображения таблицы

Отдельными блоками отображает всех игроков, их рейтинг, аватар, ник, рекорд. П

Позволяет выделить топ игроков и пользователя аккаунта.



1. Включите галочку **Advanced**.

2. **Entries Text** заполнять необязательно.
3. Укажите **Root Spawn Players Data** — это объект в иерархии которого будут создаваться объекты с компонентом **LB Player Data YG**. Каждый такой объект с компонентом LB Player Data YG — это игрок в таблице (информация о пользователе).
4. Укажите **Player Data Prefab** — это должен быть префаб с компонентом **LB Player Data YG**. Создайте такой префаб по аналогии с готовыми примерами. Поля в **LB Player Data** заполняются по желанию.

У скрипта **Leaderboard YG** есть следующие публичные методы:

▼ `UpdateLB()`

Обновление таблицы.

▼ `NewScore( int score)`

Запись нового рекорда.

▼ `NewScoreTimeConvert( float score)`

Запись нового рекорда и конвертация в Time тип.



Префаб таблицы Advanced типа по умолчанию загружает аватарки с помощью компонента **Raw Image**.

Чтобы картинка загружалась в обычный Image компонент, найдите объект **Player Photo**, у компонента **Image Load YG** укажите объект с компонентом **Image**.

## Симуляция отображения таблицы в Unity Editor



Доступно с версии 1.5 и выше.

Для отображения данных в таблице в Unity Editor, необходимо создать эти данные:

**Info YG** → **Leaderboards** → **Leaderboard Simulation** — это массив таблиц для симуляции. Элемент этого массива — это лидерборд со всеми параметрами, и все их настраивать необязательно. Для компонента **Leaderboard YG** используется только **Techno Name** (техническое название) и массив **Players**.

1. Создайте таблицу в массиве или измените уже имеющуюся.
2. Запишите **Techno Name**.
3. Для отображения игроков в таблице создайте их, каждый элемент массива это игрок отображающийся в таблице. Заполните данные игрока по своему усмотрению.  
**Photo** — это ссылка на скачивание изображения для аватарки.  
**Unique ID** может пригодиться для выделения пользователя в таблице:  
Для этого **Unique ID** должен совпадать с одноимённым параметром в **Info YG** → **Basic settings** → **Scopes** → **Player Info Simulation** → **Unique ID**.



Если не настроить симуляцию, лидерборд внутри Unity Editor будет отображать только надпись “Нет данных”.

## Time тип

1. В консоли разработчика выберите **Time** тип лидерборда.
2. Скорее всего, на первых местах таблицы Вы захотите видеть рекорд игрока, который прошёл уровень за наименьшее время (быстрее всех). Для этого установите параметр **Направление сортировки** → **Сортировка по возрастанию**.
3. **Размер десятичной части счета** оставьте на 0.
4. В компоненте **Leaderboard YG** поставьте галочку напротив параметра **Time Type Converter**.
5. При записи нового рекорда вместо метода `NewLeaderboardScores` используйте метод  
`NewLBScoreTimeConvert( string "техническое название таблицы" , float новый рекорд );`

Рекорд записывается в формате «секунды». Например, рекорд «3 минуты» должен записываться как число «180». Также может использоваться сотая часть: «180.135». Таким образом, код таймера в вашей игре может выглядеть следующим образом:

```
using YG;

class Class: MonoBehaviour
{
    float timer;

    void Update()
    {
        timer += Time.unscaledDeltaTime;
    }

    // Пример записи нового рекорда Time типа
    void NewRecordExample()
    {
        YandexGame.NewLBScoreTimeConvert("tableName", timer);
    }
}
```

```
}
```

## 有 **Локализация**

Плагин поставляется с инструментами локализации. Ниже будет описание работы с ними. Если же Вам нужно только брать язык из SDK Яндекс Игр, Вы можете сделать это после инициализации SDK таким образом:

▼ `YandexGame . EnvironmentData .language`

тип `stringify`

- **Язык.** Поддерживаемые языки вы можете посмотреть в [официальной документации Яндекс Игр.](#)

Зайдите в **Info YG** и выберите языки, на которые будете переводить игру в опции **Languages**.



Обратите внимание на кнопки внизу в Info YG. Для работы авто-локализации и Text Mesh Pro требуется их активация.

### Настройки локализации в Info YG:

▼ Параметр **Calling Language Check**.

- **First Launch Only** — язык будет проверяться только при первом запуске игры, и сохраняется. Последующие запуски игры будут с языком, который выбрался при первом запуске.
- **Every Game Launch** — язык будет меняться при каждом запуске игры.
- **Do Not Change Language Startup** — Язык не будет меняться при запуске игры.



При выборе **First Launch Only** рекомендуется при отправке игры на модерацию сделать пометку разработчика о том, что язык в игре определяется только при первом запуске.

▼ Параметр **Translate Method** — это метод работы с локализацией. Выбор метода не будет влиять на локализацию в готовой игре. Он нужен для работы в Unity Editor.

▼ **Auto Localization**

Auto Localization Даст возможность автоматически переводить текст через API Google Translate.

▼ Проблемы авто-локализации

У Google Translate есть ограничения количество символов. Может помочь смена домена в настройках плагина.

Перевод может не выполняться дальше первой точки в тексте.

▼ Инструмент для перевода всей сцены

**Auto Localization Masse.** Вы найдете его в верхней вкладке **YG → Localization.**

▼ **Manual**

Перевод вручную. С Manual методом интерфейс компонента для перевода упрощается.

▼ **CSVFile**

С CSVFile методом Вы сможете хранить все переводы по ключам в отдельном csv файле, который открывается через такие программы, как Office Excel или Google Sheets. Так обычно делают, чтобы отправить csv файл в компанию для перевода приложений.



Требует доработки, возможны баги.

▼ Работа с методом сохранения в csv файл.

При выборе метода CSVFile, в компоненте LanguageYG появятся кнопки импорта и экспорта перевода.

При экспорте, если csv файла не существует, то он создастся в корневой папке **Resources**. Если файл существует, то при нажатии на кнопку экспорт в LanguageYG компоненте, старые переводы ключа, который записан в поле ID перезапишется на переводы из компонента LanguageYG.

Есть еще инструмент для импорта и экспорта переводов всей сцены. Он находится в верхней вкладке **YG → Localization → Import\Export Language Translations.** Он не будет перезаписывать старые ключи новыми, если такие уже существуют. Он запишет только те ключи, которых еще нет в csv файле. Для того, чтобы по новой сохранить весь перевод, удалите csv файл или измените его название. Или запишите новое название для файла в InfoYG.

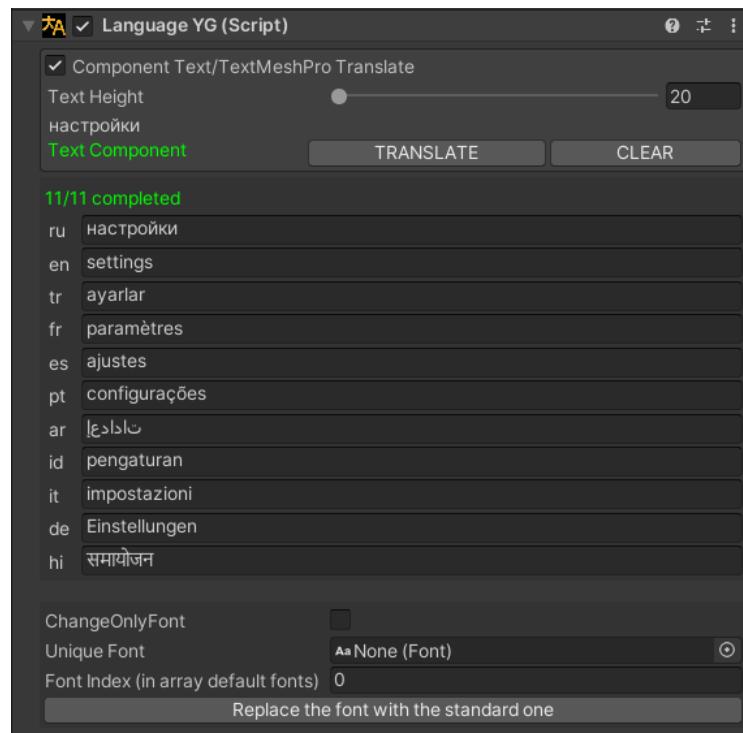


Excel не всегда правильно читает csv файл. Если у Вас возникнут с этим проблемы, откройте csv файл в Google Sheets. Это можно сделать онлайн. И через Google Sheets уже можно и пересохранить файл, тогда он и в Excel откроется.



После импорта, в программе для редактирования таблиц Вы можете увидеть знак « \* » вместо запятой. Это сделано специально по техническим причинам. Вы можете быстро заменить их на запятые с помощью инструмента быстрой замены. Так же, следите, чтобы перед экспортом переводов в Unity не было знаков запятой без пробела после запятой. Пример: **Плохо,наверное,получилось.** Хорошо, наверное, **получилось.**

За перевод текста отвечает компонент **Language YG**. Он должен висеть на объекте с компонентом Text или с компонентами Text Mesh Pro. Если компонент текста на объекте не будет найден **Language YG** предложит их создать.



## Шрифты

В **Info YG** есть массивы шрифтов для каждого языка. Если вы перетянете отдельный шрифт, допустим, в нулевой элемент массива английского языка, то в игре при английской локализации шрифты заменятся на тот, что вы указали. Вы также можете закинуть шрифт, допустим, в первый элемент массива. Тогда, если в компоненте **Languages YG** вы укажите поле **Font Index** равное одному, то для этого текста будет ставиться шрифт из первого элемента массива.

Также, в компоненте **Languages YG** есть поле **Unique Font** (уникальный шрифт). Если вы перетащите туда какой-либо шрифт, то данный текст всегда будет грузиться с указанным шрифтом.

Также есть инструмент по замене шрифтов на всей сцене на дефолтный. Он находится в верхней вкладке **YG** → **Localization** → **Font Default Masse**.

В **Info YG** массив **Font Size Correct** поможет задать корректировку для каждого шрифта.

Обратите внимание на компонент **Lang YG Additional Text** — Добавляет текст к переведенному тексту. Через код можно менять поле **Additional Text** и всё корректно применится.



## Внутриигровые покупки

Прочтайте раздел документации Я.Игр “[Внутриигровые покупки](#)”.

После добавления товаров в каталог покупок можно приступать к настройке покупок в Unity.



Для версий ниже 1.5 смотрите документацию внутри плагина.

## Симуляция отображения покупок в Unity Editor

Чтобы плагин симулировал получение каталога покупок в Unity Editor, настройте эту симуляцию в **Info YG** → **Purchases** → **Purchases Simulation**.

**Purchases Simulation** — это массив покупок для симуляции. Создайте или измените элементы массива и их параметры.

## Готовое решение

Самый простой способ — использовать готовый префаб **Payments Catalog** из папки **YandexGame** → **Prefabs** → **Payments**. Он автоматически отображает каталог всех покупок с выводом информации и кнопкой для совершения покупки.

Вам понадобится обрабатывать покупки (выдать вознаграждение). В этом может помочь скрипт **Receiving Purchase Example**. Он имеет Unity Events (иVENTЫ что в инспекторе): закрытия окна покупки, и иVENT успешной покупки. Это годится, например, для вывода уведомления о успешности покупки. Но кроме этого необходимо произвести саму выдачу покупки, в

большинстве случаев это будет происходить внутри кода. В скрипте **Receiving Purchase Example** есть такой пример:

```
// Подписываемся на ивенты успешной/неуспешной покупки
private void OnEnable()
{
    YandexGame.PurchaseSuccessEvent += SuccessPurchased;
    YandexGame.PurchaseFailedEvent += FailedPurchased;
}

private void OnDisable()
{
    YandexGame.PurchaseSuccessEvent -= SuccessPurchased;
    YandexGame.PurchaseFailedEvent -= FailedPurchased;
}

// Покупка успешно совершена, выдаём товар
void SuccessPurchased(string id)
{
    // Ваш код для обработки покупки. Например:
    if (id == "50")
        YandexGame.savesData.money += 50;
    else if (id == "250")
        YandexGame.savesData.money += 250;
    else if (id == "1500")
        YandexGame.savesData.money += 1500;

    YandexGame.SaveProgress();
}

// Покупка не была произведена
void FailedPurchased(string id)
{
    failedPurchased?.Invoke();
}
```

Покупка может быть неудачной, если:

- В консоли разработчика не добавлен товар с таким id.
- Пользователь не авторизовался, передумал и закрыл окно оплаты.
- Истекло отведенное на покупку время, не хватило денег и т. д.

## Обработка покупок

Во время оплаты, если были проблемы вроде потери соединения с интернетом, может возникнуть ситуация, что информация о произведении покупки “не дойдёт до игры” и оплаченный товар не применится в игре.

В нормальной ситуации, после “выдачи покупки в игре” выполняется метод подтверждения покупки (использования/consume). Но если этого не происходит неиспользованная, но

оплаченная покупка остаётся в массиве неиспользованных покупок в SDK Я.Игр. По этому, при запуске игры необходимо проверять этот массив и применить неиспользованные покупки.

Для применения всех неиспользованных покупок есть скрипт **Consume Purchases YG**. По умолчанию данный компонент есть на префабе **Payments Catalog**.

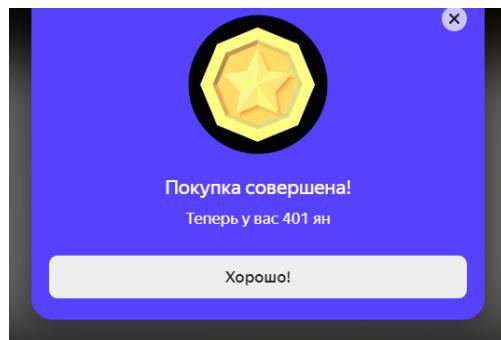
**Consume Purchases YG** - простой скрипт, который после инициализации SDK один раз вызывает метод `YandexGame .ConsumePurchases();`.

Метод `ConsumePurchases()` применяет все необработанные покупки, но как? - вызывает вышеуказанный иvent `YandexGame .PurchaseSuccessEvent` с передачей ID необработанной покупки. Это значит, что в момент выполнения `ConsumePurchases()` в игре скрипт получения товара должен подхватить обработку покупки.

Если по простому - при запуске игры на сцене должны располагаться скрипт **Consume Purchases YG** и скрипт выдачи товара, например **Receiving Purchase** на не деактивированных объектах желательно. Если в проекте что то по сложнее, сделайте свою реализацию применения необработанных покупок с помощью метода `ConsumePurchases()`.

#### ▼ Как протестировать обработку неиспользованных покупок?

После совершения покупки не нажмите на кнопку хорошо:



Перезагрузите страницу. После загрузки игры должен произойти ивент получения товара.

## Компоненты покупок

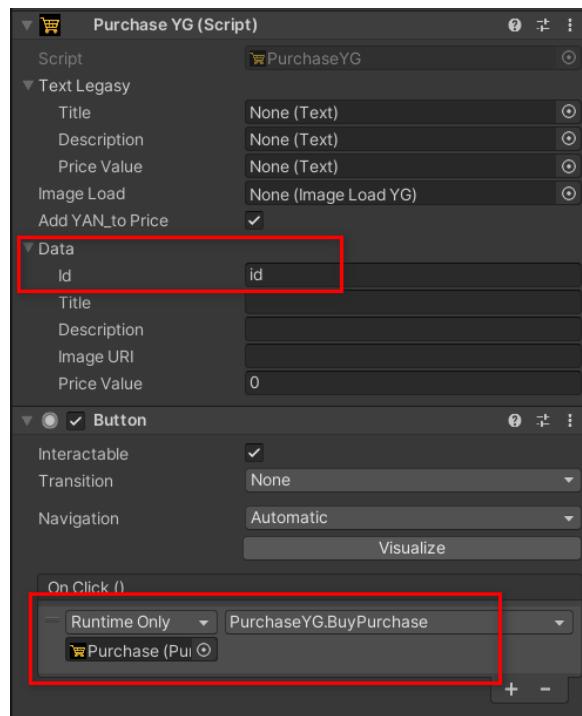
#### ▼ Подробнее о компонентах покупок

Для отображения покупок и контроля над ними используется два компонента: **Payments Catalog YG** и **Purchase YG**, или отдельно только **Purchase YG**:

#### ▼ Purchase YG

**Purchase YG** — это контейнер для информации о покупке. В нем содержатся ссылки на текст названия покупки, её описания, цену и иконку. Поля необязательны для заполнения. Purchase YG призван отображать информацию о покупке. Но сама по себе

в нём информация не обновляется. Если вам не нужно получение информации о покупке из SDK Я.Игр, вы можете оформить покупку по своему усмотрению и всё же воспользоваться компонентом Purchase YG для вызова окна совершения покупки. Для этого достаточно указать Id в классе Data и выполнять метод `BuyPurchase()` (можно через Unity Button).



## ▼ Catalog YG

**Catalog YG** — обновляет информацию в компонентах **Purchase YG**.

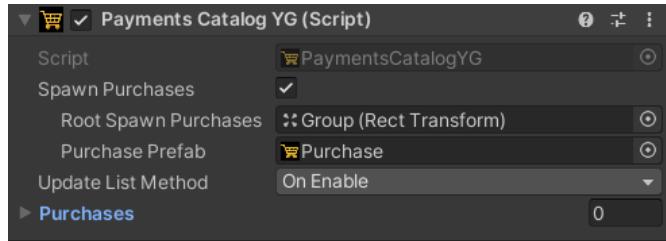
▼ **1 вариант** — автоматическое отображение всего каталога покупок.

Поставьте галочку **Spawn Purchases**.

Заполните поле **Root Spawn Purchases** объектом, внутри иерархии которого будут создаваться префабы из поля **Purchase Prefab**.

Требуется указать ссылку на префаб с компонентом **Purchase YG**. Теперь каталог товаров будет автоматически создаваться при обновлении каталога.

Когда будет обновляться каталог — можно настроить в опции **Update List Method**.



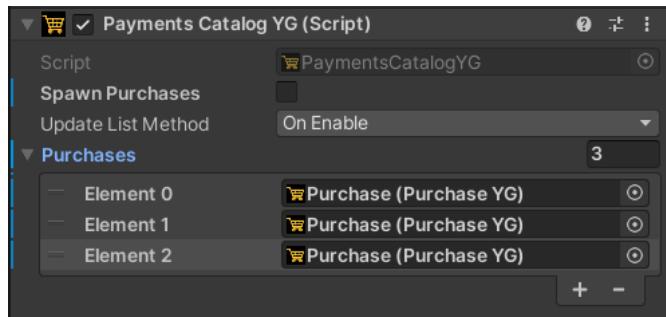
▼ **2 вариант** — создать список покупок.

Полезно, если требуется отображение не полного каталога товаров или ручное заполнение информации о покупках.

галочку **Spawn Purchases**.

Заполните список **Purchases** объектами с компонентом **Purchase YG**. В компонентах покупок требуется указать **Id**. По Id будет производиться загрузка информации.

Когда будет производиться загрузка информации — можно настроить в опции **Update List Method**.



▼ Как исключить показ определённых покупок, например, если покупка одноразовая и нужно её скрыть, если она уже приобретена.

Готовой реализации нет, т.к. реализация может быть совсем разной и может зависеть от определённых сохранений.

Но здесь будет дана подсказка:

*После обновления каталога удалим нежелательную покупку из списка.*

В скрипте **Catalog YG** есть Unity Event **On Update Purchases List**. Подпишите на него метод, в котором возьмите список **Purchases** из скрипта **Catalog YG** и удалите из списка нежелательные покупки с помощью поиска по Id. Например:

```
PaymentsCatalogYG catalog;
for (int i = 0; i < catalog.purchases.Length; i++)
{
    if (catalog.purchases[i].data.id == deletePurchaseID)
    {
        Destroy(catalog.purchases[i].gameObject);
    }
}
```



Язык информации о покупке может быть (*на момент написания док.*) только русским и английском языках. Отображение языка зависит от домена. Отображение Yan зависит от языка из сохранений плагина. Yan будет на русском как Ян, на остальных языках как Yan. Текст Yan меняется только при обновлении каталога в компоненте **Catalog YG**.

## Работа с кодом

### ▼ Подробнее о работе с кодом

После инициализации SDK выполняется метод `YandexGame .GetPayments` — метод получает информацию о покупках и “передаёт в игру”.

После “передачи информации” срабатывает ивент `YandexGame .GetPaymentsEvent`.

Метод `YandexGame .GetPayments` выполняется плагином автоматически после инициализации SDK и больше не требуется выполнение данного метода.

В `YandexGame` есть поле **purchases** — это массив покупок. Элемент массива — это класс **Purchase**. Он содержит следующую информацию о товаре:

#### ▼ `id`

типа `stringify`

- **Идентификатор товара**, который Вы записывали при создании товара в консоли разработчика.

#### ▼ `title`

типа `boolean`

- **Название** товара.

▼ `description`

типа `stringify`

- **Описание** товара.

▼ `imageURI`

типа `stringify`

- **URL изображения** товара.

▼ `priceValue`

типа `stringify`

- **Стоимость** товара.

▼ `consumed`

типа `boolean`

- **Использована ли покупка**
- `true` — использована;
- `false` — не использована.

Есть метод `YandexGame .PurchaseByID( string id )` возвращающий информацию о товаре (класс **Purchase**). Он ищет товар в списке **purchases** по **id**, который требуется указать в аргументе.

Метод `YandexGame .ConsumePurchaseByID( string id )` — обработка (использование) определённой покупки по **id**.

Метод `YandexGame .ConsumePurchases` — обработка (использование) всех необработанных покупок.

Метод `YandexGame .BuyPayments( string id )` — открыть окно, в котором можно совершить покупку.

Ивент `YandexGame .PurchaseSuccessEvent< string id >` — покупка успешно совершена.

Ивент `YandexGame .PurchaseFailedEvent< string id >` — покупка не была совершена.



Вы можете передавать какое-либо значение в игру через ссылку с помощью гипер-оператора. Таким образом можно, например, открывать конкретный уровень в игре, переходя по такой ссылке, давать бонус в игре при переходе по ссылке, передавать значение для тестирования игры.

## Как передать значение в игру

Например, адрес Вашей игры: <https://yandex.ru/games/app/012345>

Допишите к адресу приписку: ?payload=

После равно напишите значение, которое хотите передать в игру. Допустим, значение будет: debug123

Мы получили ссылку: <https://yandex.ru/games/app/012345?payload=debug123>

Теперь мы можем получать значение "debug123" в игре и делать с ним что захотим.

## Как получить и обработать значение

Передаваемое значение после инициализации SDK записывается в поле payload.

Получайте значение таким образом:

```
YandexGame . EnvironmentData .payload
```

Обычная ссылка без Deep Linking не передаёт никаких значений в payload. В таком случае, поле payload будет пустым.

Наглядный пример работы с Deep Linkings Вы можете наблюдать в следующем разделе.



## Инструмент для отладки

Вы можете легко встроить в игру инструмент для отладки, в котором можно запустить вызов основных методов плагина. Грубо говоря, у Вас в игре будет панель с "мини демо-сценой плагина". Также Вы сможете увидеть отрисовку блоков для рекламных баннеров (RTB-Блоки).

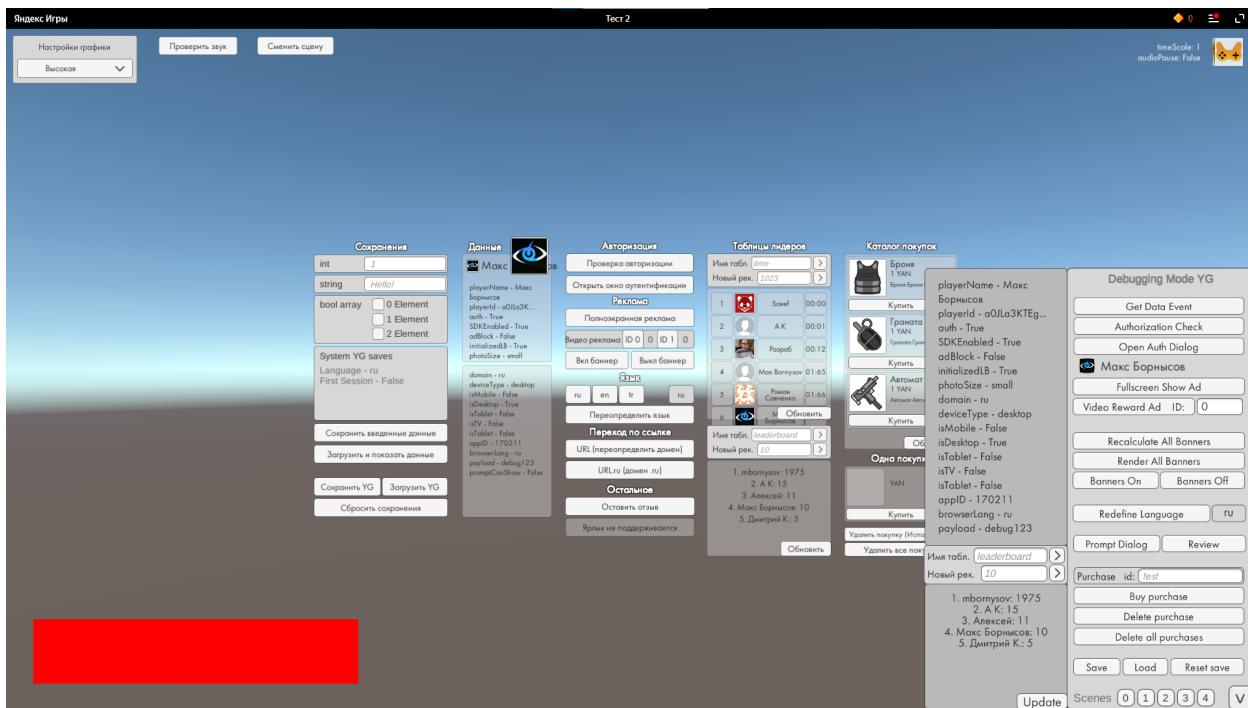
## Что нужно сделать, чтобы запустить игру в отладочном режиме

1. В Unity найдите префаб DebuggingModeYG. Он находится в папке YandexGame → **Prefabs**.
2. Добавьте префаб на главную сцену в проекте (которая открывается первой при запуске игры). Префаб DebuggingModeYG - это синглтон объект. Синглтон - это объект, который не будет удаляться при переходе на другую сцену, и не будет создаваться заново.
3. Настройте DebuggingModeYG. Укажите значение для параметра **Payload Password**. Это значение, которое Вы будете передавать с помощью Deep Linking. Можете написать слово,

например, debug и добавить свой пароль, например, 123. Получится debug123.

4. Теперь вы можете запускать игру с припиской **?payload=debug123** в конце ссылки игры, и она запустится в отладочном режиме.
5. При запуске игры в отладочном режиме Вы должны увидеть кнопку для развертывания панели управления в правом нижнем углу.

Это должно выглядеть следующим образом:



Справа на скриншоте Вы можете видеть панель управления. Слева блок, в котором должен рисоваться баннер.



Такие красные блоки рисуются только для динамических баннеров!



## Ярлык на рабочий стол

С помощью нативного диалогового окна Вы можете предложить пользователю добавить на рабочий стол ярлык — ссылку на игру.

## Вызов диалогового окна

Используйте метод `YandexGame .PromptShow()`, чтобы вызвать диалоговое окно, в котором будет предложено установить ярлык.

Доступность опции зависит от платформы, внутренних правил браузера и ограничений платформы Яндекс Игры. Для того, чтобы убедиться, что ярлык можно добавить, плагин использует параметр `YandexGame . EnvironmentData .promptCanShow`. После инициализации SDK данный параметр будет равен `true`, если ярлык можно установить. Вы также можете использовать параметр `promptCanShow` для написания своих скриптов для ярлыка, таких как, например, скрипт PromptYG. Или можете просто использовать данный скрипт.

## Скрипт PromptYG

В папке с префабами есть готовый, настроенный префаб DesktopShortcut со скриптом PromptYG. В префабе нужно только изменить кнопки под стилистику Вашей игры. Скрипт PromptYG после инициализации SDK сам покажет нужную кнопку.

Для использования скрипта PromptYG нужны три кнопки (три состояния):

**Show Dialog** — Можно установить ярлык. При состоянии Show Dialog будет показана кнопка, которая вызывает описанный выше метод `PromptShow()`.

**Not Supported** — Ярлык не поддерживается. При состоянии Not Supported, будет показана отключённая кнопка, внутри которой пояснения о том, что ярлык не поддерживается.

**Done** — Ярлык уже установлен. При состоянии Done будет показана отключённая кнопка, внутри которой пояснения о том, что ярлык уже установлен.

## Ярлык установлен

После того, как пользователь установит ярлык, произойдут две вещи:

1. Параметр `YandexGame . savesData .promptDone` будет равен `true`. С помощью данного параметра при повторной попытке установить ярлык, можно проверять, делал ли уже эту операцию пользователь раньше. За установление ярлыка игроку можно давать награду. Поэтому, чтобы не награждать игрока дважды, можно делать проверку с помощью `promptDone`.
2. Будет вызвано событие `YandexGame .PromptSuccessEvent`. Вы можете подписаться на него, чтобы наградить пользователя за установление ярлыка. Или вывести сообщение об успешно выполненной операции.



## Оценка игры

Вы можете попросить пользователя оценить игру и написать комментарий во всплывающем окне (появится в момент запроса оценки, закрывая фон приложения). Всплывающее окно не будет показано, если пользователь не авторизован или оценивал игру ранее.

Чтобы вызвать окно для оценки игры используйте метод `YandexGame .ReviewShow( bool authDialog )`. Данный метод откроет окно для оценки игры, если такая опция доступна. Принимает перегрузку

`boolean` типа. Поставьте `true`, чтобы открывалось окно авторизации в случае, если пользователь не авторизован.

Данный метод так же можно выполнит через скрипт ReviewYG. В этом случае значение `Auth Dialog` устанавливается в компоненте ReviewYG.

Для того, чтобы убедиться, что всплывающее окно для оценки игры можно отобразить, плагин использует параметр `YandexGame . EnvironmentData .reviewCanShow`. После инициализации SDK данный параметр будет равен `true`, если можно открыть окно для оценки игры. Вы также можете использовать данный параметр `reviewCanShow` для написания своих скриптов, таких как, например, скрипт ReviewYG. Или используйте данный скрипт.

В папке с **YandexGame → Prefabs** есть префаб **Review** со скриптом **ReviewYG**.

Скрипт ReviewYG имеет четыре ивента:

**Review Available** — выполните действие, которое должно произойти, если отзыв доступен (если `reviewCanShow = true`). Действие может быть, например, показать кнопку.

**Review Not Available** — выполните действие, которое должно произойти, если отзыв **НЕ** доступен (если `reviewCanShow = false`). Действие может быть, например, скрыть кнопку.

**Left Review** — выполните действие, которое должно происходить в случае, если пользователь оставил отзыв. За это можно выдавать награду. На данный ивент так же можно подписаться через YandexGame скрипт. В компоненте YandexGame данный ивент называется `ReviewDo`.

**Not Left Review** — выполните действие, которое должно происходить в случае, если пользователь **НЕ** оставил отзыв, а закрыл окно для оценки игры.

Для написания своих скриптов используйте ивент `YandexGame .ReviewSentEvent`.

Когда пользователь закроет окно или оставит отзыв, из скрипта `YandexGame` вызовется ивент `ReviewSentEvent( bool sent )`. Перегрузка `sent` будет равна:

`true` — если пользователь оставил отзыв;

`false` — если пользователь закрыл окно. При этом параметр

`YandexGame . EnvironmentData . reviewCanShow` станет равен `false` на данную игровую сессию.



## Яндекс Метрика

### Настройка метрики

1. Создайте счётчик в “[Яндекс.Метрика](#)”
2. Создайте цель в “Яндекс.Метрика” по [данной документации](#)
3. Скопируйте номер счётчика ([Как найти номер счетчика](#))

4. В настройках плагина **InfoYG** включите опцию **Metrica Enable** и вставьте номер счётчика в параметр **Metrica Counter**.

## Отправка метрики

Для отправки метрики используется класс **YandexMetrica** — он представляет собой статический класс, который содержит методы для отправки метрик в сервис "Яндекс.Метрика".

### Описание методов:

Метод `YandexMetrica. Send( string eventName )` с одним параметром типа "string" используется для отправки метрики без параметров.

Метод `YandexMetrica. Send( string eventName, IDictionary<string, string> eventParams )` с двумя параметрами — строковым и словарем типа "`IDictionary<string, string>`" — используется для отправки метрики с дополнительными параметрами. Если словарь параметров пустой или равен null, вызывается вышеописанный метод Send с одним параметром. Иначе, метод сериализует словарь параметров в JSON-строку с помощью вспомогательного класса "JsonUtils" и отправляет событие с параметрами пользователя.

Если приложение запущено в режиме редактора Unity, информация о метриках будет выводиться в консоль. В противном случае, метрики будут отправлены на сервер "Яндекс.Метрика".



## Пользовательский баннер

Вы можете сделать баннер (блок div), отображаемый по средствам "html элемента".

Это может быть полезно для создания дополнительного контента, не зависящего от интерфейса Unity, но контролируемого из Unity. Это может быть блок с играми разработчика при загрузке игры или, допустим, блок с музыкой... Раньше баннеры служили для внедрения адаптивной баннерной рекламы.

Размеры и положение баннера могут быть статично заданные изначально в свойствах стиля, или баннер может быть динамично изменяемый. Динамичный баннер получает информацию из Unity. Он копирует положение, форму и активность с UI элемента из Unity.

## Типы баннеров

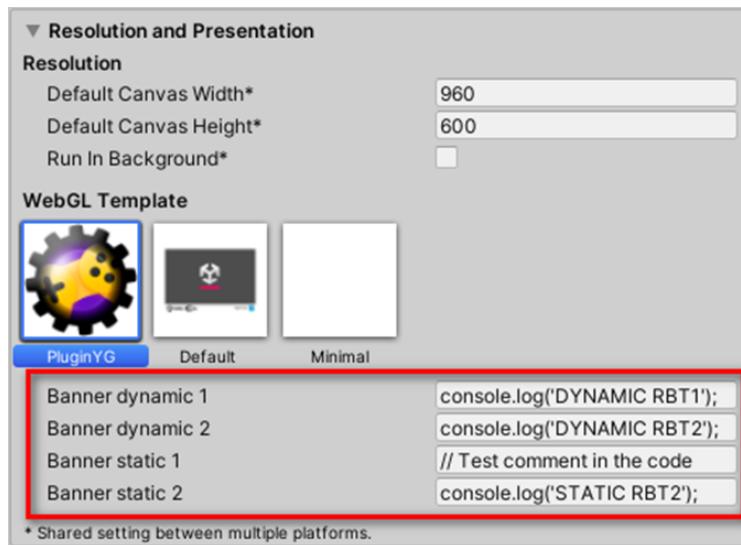
**Динамические баннеры** — это баннеры, которые Вы размещаете в интерфейсе Unity, которые будут динамически растягиваться, как интерфейс Unity. Их можно скрывать во время игры и показывать снова.

**Статические баннеры** — это баннеры, которые будут всегда зафиксированы в одном положении. Их размеры адаптируются под разные экраны. Статические баннеры нужны, по большей части, чтобы отобразиться только при загрузке игры. Их можно использовать и в

процессе игры, но тогда их нельзя будет контролировать как динамические. Не получится поменять размер, положение, и нельзя будет их скрыть, они будут присутствовать в игре постоянно.

## Настройки WebGL шаблона для баннеров

Чтобы баннер заработал, нужно что-то вписать в соответствующую строку:



Если в поле будет любой текст, код баннера скомпилируется и попадёт в билд, и в игре автоматически будут создаваться div блоки. То, что вы напишите в эти поля – это код, который будет выполняться в определённый моментах, в каких будет описано ниже. Данный код компилируется в index файле на месте такой строки: {{ BANNER\_DYNAMIC\_1 }}. Вы, конечно, можете записать туда свою логику.

Для статических баннеров код будет выполнять только при старте (при инициализации SDK Яндекса).

Что бы выполнить код для динамических баннеров есть метод ExecuteCodeRBT в скрипте BannerYG или вы можете активировать свойство «Code Execution Timer 31Sec», чтобы код динамических баннеров выполнялся каждые 31 сек. О скрипте BannerYG еще будет описано ниже.

## Создание динамического баннера

Чтобы создать динамический баннер, нужно указать ID баннера в шаблоне и перетащить готовый префаб с соответствующим номером на сцену. Префабы баннеров находятся по пути **YandexGame → Prefabs → Banners**. Если у Вас всего, допустим, один баннер, используйте префаб под номером один. Если, допустим, два баннера, то вы должны в шаблоне заполнить поле для второго баннера и использовать на сцене баннеры только под номерами 1 и 2. Если

на сцене будет баннер под номером 2, а в шаблоне поле для второго баннера будет пустым, то это вызовет ошибку.

Для создания баннера Вам нужно перетащить префаб на сцену. Префаб — это канвас со скриптом BannerYG и одним дочерним объектом **Render Block**. Настройте размеры объекта Render Block, его положение, якоря. В общем, поработайте с блоками так же, как с обычным элементом интерфейса Unity.



*Canvas настраивать не нужно!*

*Перетаскивать объект Render Block в другой Canvas не нужно!*

*Не делайте баннеры синглтоном!*

*В готовом билде блока в UI интерфейсе игры видно не будет.*

*Не меняйте Pivot. Смена Pivot объекта Render Block приведет к автоматическому возвращению Pivot'a на позицию левого верхнего угла, что приведет к смещению блока.*

#### Параметры компонента **BannerYG**

**RTB\_Number** — Номер баннера, соответствующий номеру, который вы указывали в шаблоне.

**Device** — Девайс, на котором будет отображаться баннер.

Desktop And Mobile - Отображение баннера на всех устройствах.

Only Desktop - Отображение баннера только на компьютере.

Only Mobile - Отображение баннера только на мобильных устройствах (телефонах и планшетах).

**Min Size** — Минимальный размер блока. RBT-блок не будет меньше установленного значения. X - минимальная ширина. Y - минимальная высота.

**Code Execution Timer 31Sec** — Выполнять код каждые 31сек.

**UI Scale Mode** — Режим масштабирования блоков. Настраивайте масштаб в компоненте BannerYG, не изменяйте параметры компонентов Canvas'a! Подробнее о режимах масштабирования и о их параметрах Вы можете почитать в документации Unity.

#### Активация и деактивация динамических баннеров

Вы можете де/активировать баннеры. Для этого нужно де/активировать сам префаб (объект с Cava'sом). Методы де/активирования для блоков срабатывают в OnEnable и OnDisable, поэтому при переключении сцены блоки будут включаться и выключаться в зависимости от того, есть ли префаб баннера на сцене.

Изменение положения и размера блока пересчитывается при де/активировании и при смене размера игрового экрана.

## Создание статических баннеров

Для создания статических баннеров нужно только лишь заполнит поле в шаблоне.

Первый статический баннер по умолчанию закреплен к нижней части экрана. Второй к верхней. Оба баннера по умолчанию имеют ширину 80% от экрана и высоту 15% для десктоп устройств. Для мобильных устройств зафиксирован размер в 320px на 50px.

При желании, Вы можете поменять размер и положение баннеров как Вам угодно в index файле.

### Отображение статических баннеров не только при загрузке игры, но и в самой игре

Для этого есть параметр **Static RBT In Game** в InfoYG. Включите данную опцию, и тогда статические баннеры не будут отключаться после загрузки игры.

**Custom Events** — Ивенты для кастомных методов.

## Релизы

### Версия 1.5.2

#### ▼ Game Ready API

Теперь есть контроль над Game Ready API. Об этом можете почитать в [новом разделе](#).

#### ▼ Лидерборды

- При симуляции лидерборда в Unity Editor, если при обновлении лидерборда не нашлось по его названию в Info YG соответствующего эмуляционного лидерборда, то раньше возникала невяная ошибка. Сейчас лидерборд корректно загружается, но показывает "нет данных".
- В лидербордах, исправлено иногда неточное отображение десятичной части счета в рекордах внутри игры.

#### ▼ Новые разделы документации

- [Симуляция отображения таблицы в Unity Editor](#)
- [Симуляция отображения покупок в Unity Editor](#)
- [Game Ready API](#)

## Версия 1.5.1

### ▼ Реклама

- Выключение первого показа рекламы в версии **1.5** сломалось. В версии **1.5.1** - работает.
- Устранена возможность фокуса игры при показе первой рекламы в момент когда игра загрузилась.

- Добавлен параметр **Rewarded After Closing** в Info YG.

Из моего [поста](#) в игре “Эпик Шутер” [второй пункт отклонения](#) был из за того, что после выдачи вознаграждения по какой то причине воспроизвёлся звук, но реклама еще не была закрыта. Чтобы исключить такую ситуацию, включённый параметр **Rewarded After Closing** будет означать, что вознаграждение будет получено только после [просмотра и закрытия](#) рекламы.

- Поле `YandexGame.timerShowAd` стало публичным. Его можно использовать для отслеживания сколько осталось времени до следующего показа рекламы. Это может пригодиться для таймера перед показом рекламы.

### ▼ Усовершенствована обработка инициализации

Добавлены дополнительные проверки на инициализацию.

Теперь даже если инициализация не будет произведена, в игру будут переданы данные как для неавторизированного пользователя и Get Data Event сработает. Но параметры окружения загружены не будут.

### ▼ В скрипте **Viewing Ads YG** новый параметр **Do Close Void On Awake**

Выполнить метод закрытия рекламы (**Closing AD Values** в **Viewing Ads YG**) в методе **Awake** (то есть при старте сцены).

Это позволит не прописывать события вроде аудио пауза = false или timeScale = 1 в ваших скриптах в методах Start.

**Обратите внимание** на новый раздел документации - “[Простой способ настройки Viewing Ads YG](#)”.

### ▼ Доп проверки на вкл/выкл кастомных баннеров

(Для тех, у кого ошибочно кастомные баннеры компилировались и попадали в проект, что могло вызвать ошибки).

# Версия 1.5

! Необходимо удалить старую версию плагина перед импортом v1.5 (папку WorkingData можно не удалять).

## ▼ Локализация

- Поддержка **Text Mesh Pro**
- Новый компонент **Lang YG Additional Text**. [Подробнее в разделе локализации.](#)
- Автолокализация теперь работает и на деактивированных объектах.

## ▼ Define system

В **Info YG** появились кнопки для отключения и включения:

- Библиотеки **Newtonsoft** для использования **автоматической локализации**.
- Библиотеки **Newtonsoft** для использования **в системе сохранений**. Newtonsoft json net позволяет сохранять массивы в массивах, классы. Это продвинутая работа с json. Но в сборке игра будет весить на 2мб больше с использованием Newtonsoft.
- Библиотеки **TMPro** (Text Mesh Pro).

Также, после импорта плагина в списке scripting define symbols будет создан define **YG\_PLUGIN\_YANDEX\_GAME**. Активность библиотек также контролируется с помощью списка scripting define symbols.

## ▼ Реклама

- Добавлен параметр:

▼ `YandexGame .nowAdsShow`

типа **boolean**

- `true` — Полноэкранная или видео реклама **открыта** в данный момент.
- `false` — Полноэкранная или видео реклама **закрыта** в данный момент.

Ранее были только параметры:

▼ `YandexGame .nowFullAd`

типа **boolean**

- `true` — Полноэкранная реклама **открыта** в данный момент.

- `false` — Полнозадранная реклама **закрыта** в данный момент.

▼ `YandexGame .nowVideoAd`

типа `boolean`

- `true` — Видео реклама за вознаграждение **открыта** в данный момент.
- `false` — Видео реклама за вознаграждение **закрыта** в данный момент.

- 
- Если время просмотра рекламы за вознаграждение составляет менее двух секунд, то вознаграждение не будет получено. Вместо этого вызовется ивент ошибки.
  - Теперь и при включённом параметре **Singleton** будет показываться реклама при открытии новой сцены. Если, конечно, включен параметр **Ad When Loading Scene** в **Info YG**.
  - Если при загрузке игры была открыта реклама, и после загрузки она также остаётся открытой, то после инициализации SDK вызывается ивент открытия рекламы.

▼ Дополнительный контроль над симуляцией

В **Info YG** найдёте настройки симуляции для:

- Параметры игрока. Например, имя, аватар, устройство пользователя.
- Создание эмулированных лидербордов вплоть до заполнения информации о игроах.
- То же самое для внутриигровых покупок.

▼ Доработка сохранений загрузки игрового прогресса

- Добавлена опция **Save On Quit Game** в **Info YG**. Выполняет метод `SaveProgress` при выходе из игры (закрытии страницы).
- Файл сохранений для тестирования в Unity Editor теперь сохраняется в папку **YandexGame/WorkingData/Editor** и имеет расширение **json**. Теперь можно удобно читать и менять сохранения в файле.
- Устранена проблема, когда компилятор ссылался не на ту ошибку при сохранении или загрузке.

▼ Настройка иконки загрузочного экрана

В **Info YG** параметр **Logo Image Format** задаёт расширение изображения иконки, и возможность её отключить.

▼ Новый компонент Activity On Authorization YG

Позволяет выполнить действие в зависимости от авторизации пользователя.

Например, легко отключить какую-либо кнопку, если пользователь неавторизован.

▼ Обновление задело практически весь код плагина

**Особо можно выделить:**

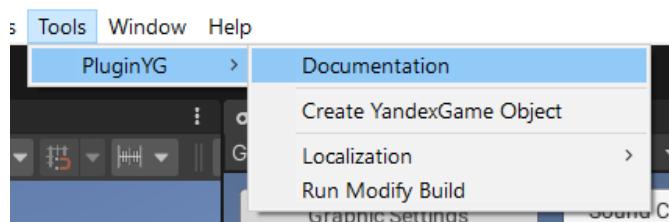
- Таблицы лидеров
- Внутриигровые покупки

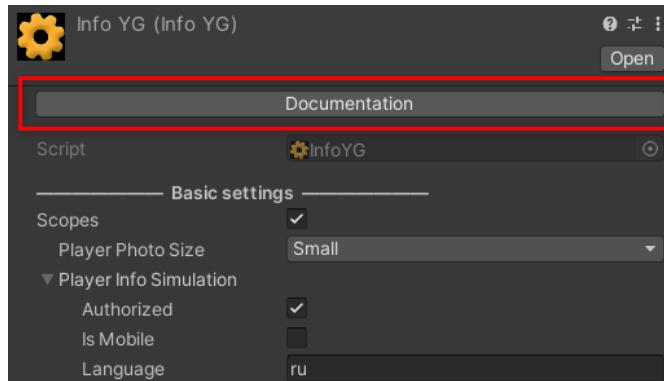
Для этих пунктов смотрите обновлённую документацию.

- Post process build система
- Скрипт оценки игры
- Debugging инструмент
- Удаление лишнего
- Реструктуризация кода
- Подготовка к модульной системе

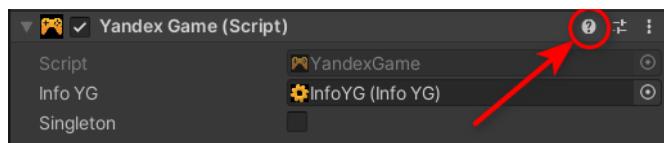
▼ Ссылки на документацию

Добавлены быстрые ссылки на документацию:





Также добавлены ссылки на соответствующий раздел документации в разных компонентах:



## Версия 1.4



Желательно удалить старую версию плагина перед импортом **v1.4** (папку *WorkingData* можно не удалять). Перераспределены файлы в папке **Examples** и перемещён скрипт **ArchivingBuild**.

### ▼ Яндекс Метрика с полным функционалом

Добавлена возможность отправлять метрики с параметрами и без. Документация Яндекс Метрики

### ▼ Перенесены параметры из шаблона PluginYG

Были перенесены параметры из полей шаблона в настройки плагина InfoYG. Кроме параметров баннеров (это не рекламные баннеры), они еще требуют доработки. Теперь включить и настроить метрику, опцию “Pixel ratio”, первый показ рекламы - можно не заходя в Player настройки.

### ▼ Более тонкая настройка сохранений

В **InfoYG** в настройках сохранений теперь есть параметр **Save Local Sync**. Описание: синхронизировать облачные сохранения с локальными? Если `localSaveSync = false` при

включенных облачных сохранениях, то локальные просто не будут использоваться.

▼ Доработка перехода по ссылке

Методы перехода по ссылке (OnURL, \_OnURL\_Yandex\_DefineDomain) адаптированы под браузер Safari и другие браузеры в которых раньше могли не открываться ссылки.

▼ Статичный метод для конвертации рекорда в time type

Объяснение работы с методами конвертации в разделе «[Лидерборды](#)».

▼ Переименованы методы В YandexGame

Переименованы некоторые, методы связанные с инициализацией. Когда-то они имели функционал авторизации. Имена, содержащие слово **Authorization** теперь называются со словом **Initialization**.

## Версия 1.3.6

▼ Фикс в методе Load Local

При **отключении** облачных сохранений – локальные не работали. В методе LoadLocal() была опечатка. Исправлено!

▼ Фикс в компоненте ViewingAdsYG

Исправлен баг некорректной работы **Remember Previous State** в параметре **Pause Method**. После показа рекламы иvent открытия и закрытия рекламы проходил два раза если скрипт **ViewingAdsYG** висел на объекте **YandexGame**, из-за идентичных по названию методов, которые принимали **Send Message** из index.html.

## Версия 1.3.5

! Удалите старую версию плагина перед импортом v1.3.5 (папку WorkingData можно не удалять).

▼ Устранено зависание игры

На мобильных устройствах в некоторых браузерах при открытии некоторых окон и нажатии на шторку Я.игры игра зависала. Сейчас не должно такого быть.

#### ▼ Сохранение массива массивов

Вернулся JsonUtility и появилась возможность выбрать между JsonUtility и JsonConvert.

Недостаток класса JsonConvert в том, что при его использовании в билд проекта добавляется библиотека JsonNet и конечный вес игры увеличивается на ~2мб.

Преимущество класса JsonConvert — сохранение массива в массиве. Используйте JsonUtility, если Вам не нужно сохранять массив в массиве, для этого ничего не требуется делать, JsonUtility используется по умолчанию.

Чтобы использовать JsonConvert: раскомментируйте первую срочку “`//#define JSON_NET_ENABLED`” в скрипте **YandexGame**.

#### ▼ Внедрён перевод слов “**unauthorized**” и “**anonymous**”

Перевод на все языки внедрён в скрипты “**LeaderboardYG**” и “**GetPlayerYG**”.

В Ваших собственных скриптах при получении данных, например, имя пользователя, если имя будет “unauthorized” или “anonymous”, то оно не будет переведено. То есть, в Ваших скриптах перевод нужно будет делать самостоятельно. Для этого можете воспользоваться методами:

```
YandexGame.Instance.infoYG. UnauthorizedTextTranslate() ;  
YandexGame.Instance.infoYG. IsHiddenTextTranslate() ;
```

Слово anonymous заменено на **Is Hidden** — означает “**скрытый**”.

Методы возвращают переведённое слово. Язык определяется в зависимости от настроек плагина. Так же методы имеют перегрузку **language** — Вы можете записать в перегрузку язык, на который требуется перевести слово.

#### ▼ Доработан ивент **CloseFullAdEvent**

Теперь вызывается только после успешного показа и закрытия рекламы.

#### ▼ Добавлен ивент **ErrorFullAdEvent**

ErrorFullAdEvent при ошибке отображения полноэкранной рекламы.

#### ▼ Доработан таймер интервала показа рекламы

Если при вызове полноэкранной рекламы она не была показана по причине ошибки или слишком частого вызова - таймер интервала показа полноэкранной рекламы сбрасывается.

#### ▼ Скрипт ReviewYG

Кнопка оценки игры не будет показана, если пользователь не авторизован и параметр **authDialog = false**. (Если пользователь не авторизован и **authDialog = true**, то кнопка будет показана и при нажатии на кнопку откроется диалоговое окно авторизации).

#### ▼ Метод OnURL\_Yandex\_DefineDomain

Если домен по какой то причине не определён, ссылка не откроется.

#### ▼ Компонент LeaderboardYG обзавёлся параметром "**Is Hidden Player Photo**"

Укажите ссылку на свой спрайт для отображения скрытых пользователей.

#### ▼ Ивент обновления таблицы в компоненте LeaderboardYG

"**On Update Data**" вызывается при получении данных таблицы лидеров.

#### ▼ Получение данных пользователей из таблицы лидеров

Получение данных пользователей из таблицы лидеров

У скрипта LeaderboardYG появился класс **PlayersData** - в этом классе содержатся данные пользователя: `name` , `rank` , `score` , `photo` . Скрипт LeaderboardYG содержит поле `playerData[]` - это массив, элементы которого являются вышеописанным классом PlayersData.

Соответственно, в скрипте LeaderboardYG Вы можете взять массив игроков и получать их данные. Можете брать данные по ивенту "**On Update Data**". Также был создан учебный скрипт "**GetLederboardData**" для примера использования данных из таблицы, и в демо-сцене этот скрипт показано как работает. В сцене есть объект "**Leaderboard Debug Data**". Обратите внимания на лидерборд "**Leaderboard Advanced & Time**" в сцене (в объекте Canvas) и его ивент "On Update Data", он связан с вышеописанным скриптом.

#### ▼ Метод для конвертации рекорда в time тип

В скрипте LeaderboardYG Вы сможете найти метод для конвертации рекорда в time тип:

`TimeTypeConvert(int);` Принимает параметр типа int. Как должен выглядеть параметр:

например, мы хотим конвертировать число 123.456f (123 секунды, 456 миллисекунд. Или 2 минуты, 3 секунды и 456 миллисекунд). В таком случае отправляемое число типа int должно выглядеть следующим образом: 123456. Метод TimeTypeConvert вернёт строку в

зависимости от параметра decimalSize. Если decimalSize будет равен, например, 1, то мы получим строку “02:03.4”

Это не самый удобный способ конвертации, потому что метод TimeTypeConvert был создан для получения и конвертации рекорда от Яндекса.

## Версия 1.3.4



*Игры с использованием плагина версий ниже **v1.3.4** будут отклонены модерацией!*



*Для перехода с предыдущих версий плагина на **v1.3.4** необходимо **перевыбрать шаблон!** (При этом слетят поля шаблона, в том числе метрика).*

### ▼ Изменена логика инициализации SDK

При тестировании черновика в режиме “черепашки” (ограничение таймаутов запросов) - не будут загружаться данные игрока (будет как не авторизированный пользователь), кроме сохранений, они будут браться из локальных сохранений.

### ▼ Внедрён “[Старт игр](#)”

Сигнализирование Яндексу, что игра загрузилась.

### ▼ Настройка Pixel Ratio

В шаблоне **PluginYG** поле **Pixel ratio mobile** отвечает за значение Device Pixel Ratio в index файле.

Заполните поле выбранным Вами значением, например: **1.3**

Оставьте данное поле пустым, если не хотите ничего менять.

### ▼ Сохранение массива массивов

В скрипте YandexGame - JsonUtility был заменён на JsonConvert.

### ▼ Вырезан сайтлок

В связи с новым пунктом **1.18** в [правилах требований к игре](#) был вырезан сайтлок, т.к. он блокировал страницу игры расположенную на ином домене.

- ▼ Скорректирован показ рекламы при запуске игры для разных устройств

В обновлении **v1.3.2** был такой же пункт. Теперь стало так же, как было до обновления **v1.3.2** в связи с изменением логики инициализации SDK.

## Версия 1.3.3

- ▼ Возможность выключить автоматическую архивацию билда

В настройках плагина **InfoYG** опция **Auto Build Archiving** отвечает за активность функции авто-архивации билда.

- ▼ Фикс бага, в котором событие закрытия полноэкранной рекламы могло не выполняться при симуляции в Unity

Если в момент симуляции показа полноэкранной рекламы производилась загрузка новой сцены, то событие закрытия рекламы могло не выполнятся, из-за этого далее реклама не показывалась. Данный баг присутствовал только в Unity Editor, но не в готовой сборке.

## Версия 1.3.2



Для перехода с предыдущих версий плагина на **v1.3.2**, может потребоваться перенастроить компонент **ViewingAdsYG** и опцию показа **рекламы при загрузке сцен!** Так же, обязательно **перевыберите шаблон!** (При этом слетят поля шаблона, в том числе метрика).

- ▼ Автоматическая архивация билда

После успешного создания билда игры, папка с содержанием билда пакуется в zip архив. При повторной сборке игры, архив не перезапишется, но создастся новый пакет с приписанным номером в названии файла.

- ▼ Обновлён скрипт Viewing Ads YG

Документация обновлена. При переходе с предыдущих версий плагина необходимо убедиться, что настройки компонента Viewing Ads YG удовлетворительны, т.к. были изменены некоторые опции.

- ▼ Скорректирована функция показа рекламы при загрузке сцен

В настройках плагина **InfoYG** опция показа рекламы при загрузке сцен изменилась. Теперь данная опция называется **Ad When Loading Scene**. По умолчанию = **true** — это значит, что показ рекламы будет вызываться при загрузке любой сцены в игре. Значение **false** — реклама не будет показываться при загрузке сцен.

▼ Отключение показа полноэкранной рекламы при запуске игры

По умолчанию первая реклама показывается еще до загрузки игры, это можно отключить:

1. Перейдите в настройки проекта **Project Settings** → **Player** → **Resolution and Presentation** → **WebGL Template**
2. Выберите шаблон **PluginYG**
3. Найдите поле **Off ads before loading game**
4. Заполните данное поле любым словом, например, запишите туда **true**
5. Теперь полноэкранная реклама не будет показываться сразу при запуске игры.

▼ Скорректирован показ рекламы при запуске игры для разных устройств

1. На Desktop реклама показывается сразу при открытии страницы.
2. На мобильных устройствах реклама показывается только после загрузки игры.  
Мгновенный показ рекламы после открытия страницы с игрой может раздражать, особенно, если реклама будет тормозить, и из-за этого плюсом еще её будет сложно закрыть, что может происходить как раз при загрузке игры. По этому на мобильных устройствах реклама будет показываться после загрузки игры.

▼ Кнопка вызова окна оценки игры

Кнопка для вызова окна оценки игры теперь отключается, если игра запущена на мобильном устройстве, т.к. на данный момент вызов такого окна по какой то причине может вывести игру из строя в каких то из браузеров.

Вы можете обратно включить кнопку, за это отвечает опция **Show On Mobile Device** в компоненте **Review YG**.

▼ Фикс для ESC

При работе с ECS-фреймворками (DOTS, LeoECS, Morpeh, etc...) часто используют выключение перезагрузки домена. Для данной опции была внедрена перезагрузка статических полей плагина, что должно обеспечить корректную работу с ECS.

#### ▼ Фикс для инкогнито режима

В предыдущих версиях плагина, в режиме инкогнито в определённых браузерах загрузка локальных сохранений происходила с ошибкой из-за ограничения доступа браузера к данным, и далее из-за ошибки плагин работал не корректно. Начиная с версии плагина 1.3.2 данная ошибка успешно обрабатывается.

#### ▼ Фикс для компонента Language YG

В предыдущих версиях плагина, если на сцене отсутствовал объект YandexGame, то компонент LanguageYG не мог обнаружить Scriptable Object “**InfoYG**”. Начиная с версии плагина 1.3.2, если на сцене нет объекта YandexGame, то InfoYG берётся из папки **YandexGame → WorkingData**.

## Версия 1.3.1

#### ▼ Быстрое внедрение Яндекс Метрики

Создайте счётчик на сайте Яндекс Метрика (<https://metrika.yandex.ru/list?>). Скопируйте код счётчика. Найдите поле "Metric" в шаблоне PluginYG. Вставьте код счётчика в данное поле.

#### ▼ Фикс зависания игры на мобильном устройстве при открытии диалогового окна

Теперь зависания нет, кроме окна оценки игры! (Зависание окна оценки игры проверено в Яндекс браузере и Google Chrome. В Chrome зависание отсутствует, в Яндексе игра крашится!).

#### ▼ Фикс ([https://t.me/yandexgame\\_plugin/13109](https://t.me/yandexgame_plugin/13109))

Фикс проблемы с загрузкой облачных сохранений при использовании json в сохраняемых данных.

#### ▼ Фикс ([https://t.me/yandexgame\\_plugin/12681](https://t.me/yandexgame_plugin/12681))

Фикс ([https://t.me/yandexgame\\_plugin/12681](https://t.me/yandexgame_plugin/12681)) бага с добавлением иконки на рабочий стол. Теперь при закрытии окна предлагающего создать иконку, кнопка вызова такого окна в игре скрывается, так как два раза такое окно открыть Яндекс не даёт.

## Версия 1.3

#### ▼ Time тип для лидербордов

Доработан! Документация обновлена.

▼ Обновлён Image Load YG

Компонент обзавёлся опцией Sprite Image. Перетащите в Sprite Image объект с компонентом Image, чтобы изображение загрузилось как спрайт.

▼ Обновлена система сохранений

Локальные сохранения переделаны из PlayerPrefs в localStorage.

Интервал облачных сохранений (Save Cloud Interval) теперь по умолчанию равен 5.

Параметр Flush по умолчанию равен false (выключен).

## Версия 1.2



Для перехода требуется удалить прошлую версию из проекта и импортировать новую. Заменить так же требуется скрипт SavesYG.

▼ Обновлена система сохранений

Решена проблема лимитов на облачные сохранения. Теперь данные сохраняются и локально и на облако. Всё происходит автоматически, от вас не требуется никаких действий. Появились новые параметры сохранений в InfoYG.

▼ Обновлен скрипт Viewing Ads YG

Добавлен контроль над курсором. Рекомендуется после обновления проверить настройки скрипта Viewing Ads YG в вашем проекте.

▼ Добавлен статический ивент onResetProgress

`YandexGame .onResetProgress` — это установка всех параметров и сохранений игры по умолчанию. Вызывается при первом запуске игры.

▼ Добавлен статический ивент onQualityChange

`GraphicSettingsYG .onQualityChange` — вызывается при смене графики по средствам компонента Graphic Settings YG.

- ▼ Заменен стандартный логотип при загрузке игры на лого плагина

Всё же, не используйте его. Используйте свой логотип!

- ▼ Синтаксис "new()"

Новый синтаксис `new()` возвращен на старый `new Тип()`. Больше не будет из-за этого ошибок на более старых версиях ПО.

## Версия 1.1



*Для перехода требуется небольшие правки вашего проекта. Изменены функции рекламы за вознаграждение, оценка игры, компонент LeaderboardYG и баннеры.*

- ▼ Создан функционал для Sticky-баннеров

Добавлен API для sticky баннеров. Документация обновлена.

- ▼ Вырезаны старые баннеры

Удален API для старых кастомных баннеров. Но возможность добавления адаптивных div блоков для иного функционала осталась. Актуальная документация по этому вопросу внутри пакета плагина.

- ▼ Оценка игры (отзыв)

Полный функционал оценки игры. Подробнее в документации.

- ▼ Изменен ивент вознаграждения за просмотр рекламы

Раньше за вознаграждение отвечал ивент Close Video. Теперь Close Video отвечает только за закрытие рекламы и не имеет перегрузок.

Для вознаграждения игрока используйте ивент Reward Video.

В компоненте YandexGame ивент Reward Video Ad

В скрипте: `RewardVideoEvent<int id>;`

- ▼ Удалена опция Check AdBlock

Функционал Check AdBlock и её ивенты заменены на один ивент Error Video.

В компоненте YandexGame ивент Error Video Ad

В скрипте: `ErrorVideoEvent;`

▼ Изображение на фон при загрузке игры

Уже давно, по какой то причине перестала отображаться фоновая картинка при загрузке игры. Вместо неё можно было наблюдать белый экран.

Чтобы отобразить фоновое изображение, положите свою картинку в папку **WebGLTemplates** → **PluginYG** под названием **background.png**. Точно так же, как с logo.png. Из настроек **Player** → **Splash Image** при этом нужно удалить **Background Image**.

▼ Доработка скрипта Leaderboard YG

Добавлен параметр Update LB Method.

Удалён параметр Load Avatars. Чтобы отключить загрузку аватарок выберите None Photo в параметре Player Photo.

▼ Фикс для автопереводчика

Добавлен параметр Domain Auto Localization в InfoYG. Поменяйте домен, если возникли проблемы с авто-переводом.

▼ Новые настройки рекламы в InfoYG

**Fullscreen Ad Interval** — Интервал запросов на вызов полноэкранной рекламы.

**Duration of Ad Simulation** — Длительность симуляции показа рекламы.

## Версия 1.0

▼ Корректировка масштаба шрифта отдельно для каждого языка и шрифта

В **InfoYG** найдете массив **Font Size Correct**. В нём содержатся еще массивы типа integer (для каждого языка). Создайте новый элемент массива, номер которого будет соответствовать номеру элемента нужного шрифта в массиве шрифтов (массив **Fonts** в **InfoYG**).

Корректировка осуществляется следующим образом:

Например, вы установили число 1 для Русского языка в первом элементе массива (для первого шрифта). В таком случае размер (Font Size) первого шрифта для Русского языка в

игре будет увеличен на 1. Соответственно, если вы установите размер, допустим, -2, то Font Size шрифта уменьшится на -2.

## ▼ Документация

Документация в PDF формате на Английском и Русском языках в папке **YandexGame**  
→ **Documentation**.



Документация в PDF формате может быть не всегда актуальной. Рекомендуется использовать онлайн документацию!