



PluginYG Документация

✓ Содержание

✓ Содержание

Предисловие

Ссылки

Импорт плагина

Часто задаваемые вопросы

Общее (читать всё)

Добавление Yandex Game на сцену

Выбор WebGL шаблона

Настройки плагина ([Info YG](#))

Пространство имен YG

Работа с компонентом Yandex Game

Ивенты в компоненте Yandex Game

Game Ready API

Разметка геймплея (Gameplay Start/Stop)

Настройки графики от плагина

Инициализация

События вкладки игры

Fullscreen методы (Полный экран)

Метод серверного времени

A Полноэкранная реклама (Fullscreen Ad)

Заглушка перед рекламой

Реклама в процессе игры (таймер 2с)

Реклама за вознаграждение (Reward Ad)

Пауза игры при просмотре рекламы

Простой способ настройки [Viewing Ads YG](#)

Sticky-баннер

Все данные

Сохранения

[Создание своих данных для сохранения](#)

[Загрузка сохранений](#)

Сохранение

Сброс сохранений

Сохранение массивов

Лидерборды

[Создание таблицы в консоли разработчика](#)

[Запись рекорда в соревновательную таблицу](#)

[Создание таблицы в Unity](#)

Симуляция отображения таблицы в Unity Editor

Time тип

Дополнительные опции при работе с кодом

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

Модули

Релизы

Версия 1.6.3 critical

Версия 1.6.2

Версия 1.6.1

Версия 1.6

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

Интеграция SDK Яндекс Игры для Unity. Плагин автоматизирован и сделан так, чтобы максимально облегчить интеграцию SDK.

Многие параметры в плагине описаны во всплывающих подсказках.

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.

В данной документации не будет описываться работа с настройками проекта под WebGL или даваться подробные описания работы с сервисами Яндекса. 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

ССЫЛКИ

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

PluginYG



Обсуждения по плагину

 https://t.me/yandexgame_plugin

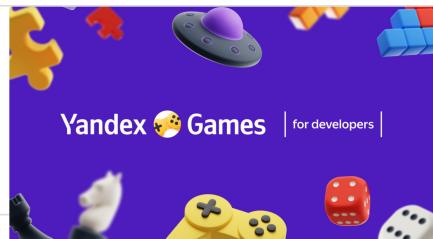
В закрепах найдете обновления и всю информацию в том числе ссылки на скачивание.

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 Все сведения на Trello:

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

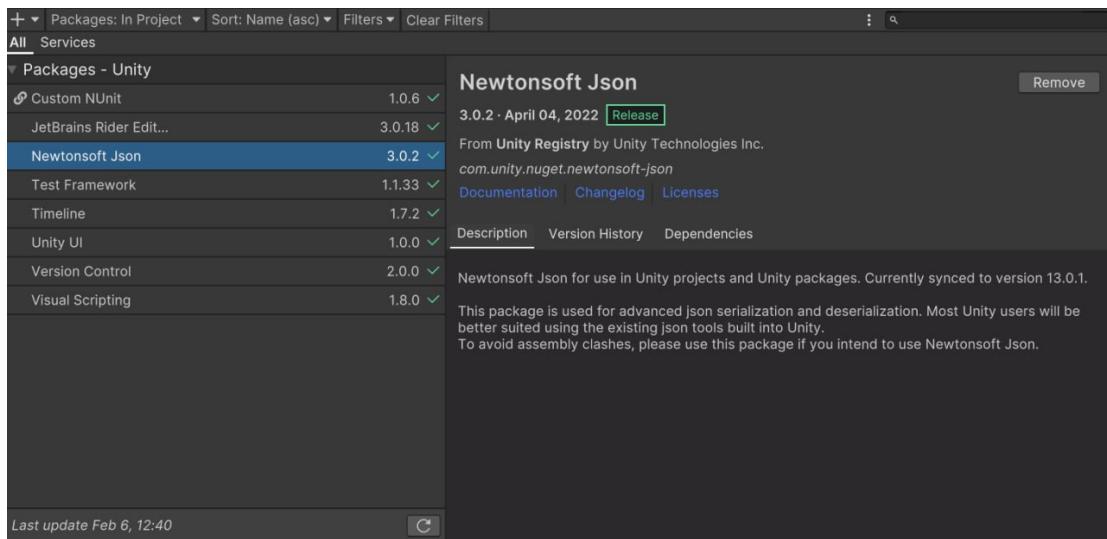
🎥 <https://zen.yandex.ru/video/watch/6234f7331c98a06699258833>

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

! Версии Unity, которые поддерживает плагин:

2021.3.18+

2022.3+

2023.2+



Не перемещайте импортированные с плагином папки в проекте!

Вы можете **скачать** плагин тремя путями:

▼ Unity Asset Store

Обычная установка как и все другие ассеты.

▼ GitHub

На GitHub нужно скачать архив и распаковать файлы плагина находящиеся в папке Assets в Ваш проект.

▼ Яндекс Диск | Телеграм чат (Закрепы)

Самые актуальные версии плагина можно найти в телеграм чате PluginYG в закрепе.

На диске хранятся версии плагина в формате unitypackage. Скачивав пакет, просто откроите его или перетащите в конкретный проект Unity в окно Projects.

Если обновляете плагин в проекте (устанавливаете не первый раз):

Перед импортом новой версии плагина рекомендуется удалить старую.

Удалять необходимо папки

YandexGame и **WebGLTemplates**.

Но! Оставьте папку **YandexGame → WorkingData**, в ней хранятся сохранения игры и настройки плагина.

Также не забудьте оставить Ваши изображения

background и **logo** в папке **WebGLTemplates → PluginYG**, если таковые имеются.



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

В папке

YandexGame также можно найти локальную документацию для импортированной версии плагина.

? 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?

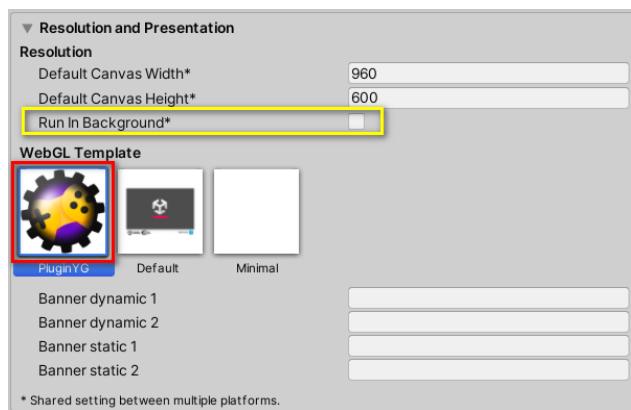
Нужно получить данные об устройстве (смотрите раздел "Данные"). Данные можно получать только после инициализации SDK (смотрите раздел "Инициализация").

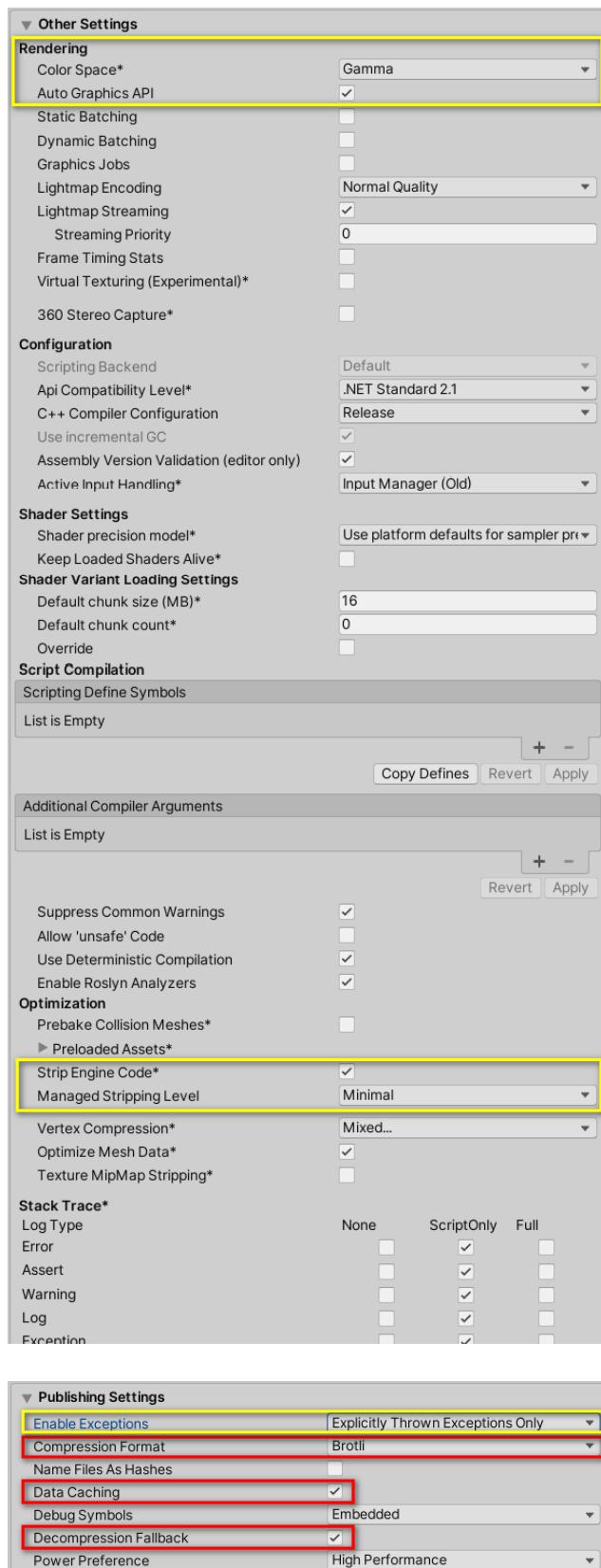
▼ Ошибки при **импорте** плагина?

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:

Basic Settings

Scopes

- Player Photo Size: Small
- Leaderboard Enable:
- Save Cloud:
- Flush Save:
- Save Cloud Interval: 30

Ads

- Fullscreen Ad Challenge: At Startup End Switch Scene
- Fullscreen Ad Interval: 60
- Duration Of Ad Simulation: 0.5

Language Translation

Поддерживаемые платформы: Выбрано: 6

Apple Team ID: [Input field]

Заполните, если у вас есть. Требуется для публикации в приложении на iOS по правилам Apple (пункт 4.7.2 Guidelines). Помогите себе в [Apple's Developer Center](#) — Account — Membership — Team ID.

Ориентация: Альбомная

Игру разрешено дистрибутировать на международном рынке

Поддержка service worker

Игра поддерживает авторизацию или лидерборд

Игра использует облачные сохранения

Оптимизировано для мобильных

- Игра находится в полноэкранном режиме во время игры
- Размеры игровых элементов, кнопок, графики адаптированы к мобильному девайсу таким образом, чтобы не требовалось целиться, чтобы выбрать предмет
- Масштаб игрового поля не выходит за рамки экрана - есть возможность без лишних свайпов играть и постоянно находиться на основной сцене (играть одной рукой)
- Управление жестами

Оптимизировано для десктопа

- Игровое поле занимает весь экран. Картинки, шахматы, шашки должны быть с фоном на весь экран (из опицентной с игрой полноткой)

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

- 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 расширенные шаблоны](#)

▼ Не работает **авто-локализация**?

1. Необходимо активировать автолокализацию в настройках плагина **InfoYG!** Для этого имеется соответствующая кнопка.
2. Включите разрешение загрузки по протоколу HTTP для Unity Editor в **Project Settings** → **Player** → **Allow downloads over HTTP**.
3. Попробуйте сменить в настройках плагина (**Domain Auto Localization**).

▼ Во время игры частично или полностью не переводится текст?

По какой то причине слетели ссылки внутри скрипта **Language YG**. Чтобы восстановить ссылки, есть метод **Reserialize** в контекстном меню. Нажмите на три точки на компоненте или на правую клн. мыши. В открывшемся списке снизу будет **Reserialize**.



При выполнении **Reserialize** сцена автоматически не сохраняется и не помечается как * "грязная (требуется сохранение)".

Для исправления всех компонентов на сцене такой метод также имеется в инструменте **Auto Localization Masse**. Так же, можно выделить все компоненты на сцене с помощью поиска в иерархии по полному имени компонента **LanguageYG**, и сделать **Reserialize** для всех выделенных объектов.

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

У некоторых появляется ошибка при попытке создания компонента Language YG. Должно помочь перекомпилирование скрипта **LanguageYG**. Как ни странно, помогает переименовывание данного скрипта в "**Language**".

Общее (читать всё)

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

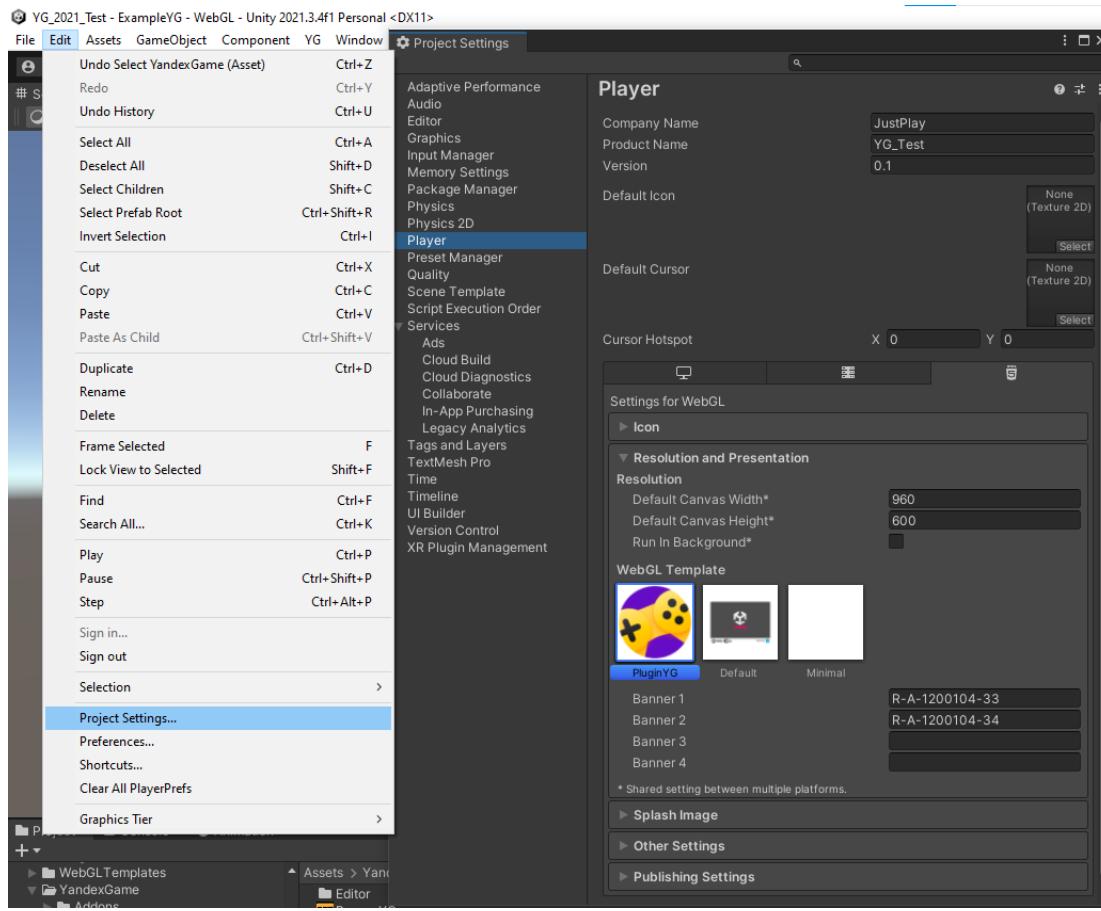
| Добавить префаб **YandexGame** на сцену удобнее всего таким образом:



Префаб **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 плафигна 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 плафигна, убедитесь в том, что поля не пропали.

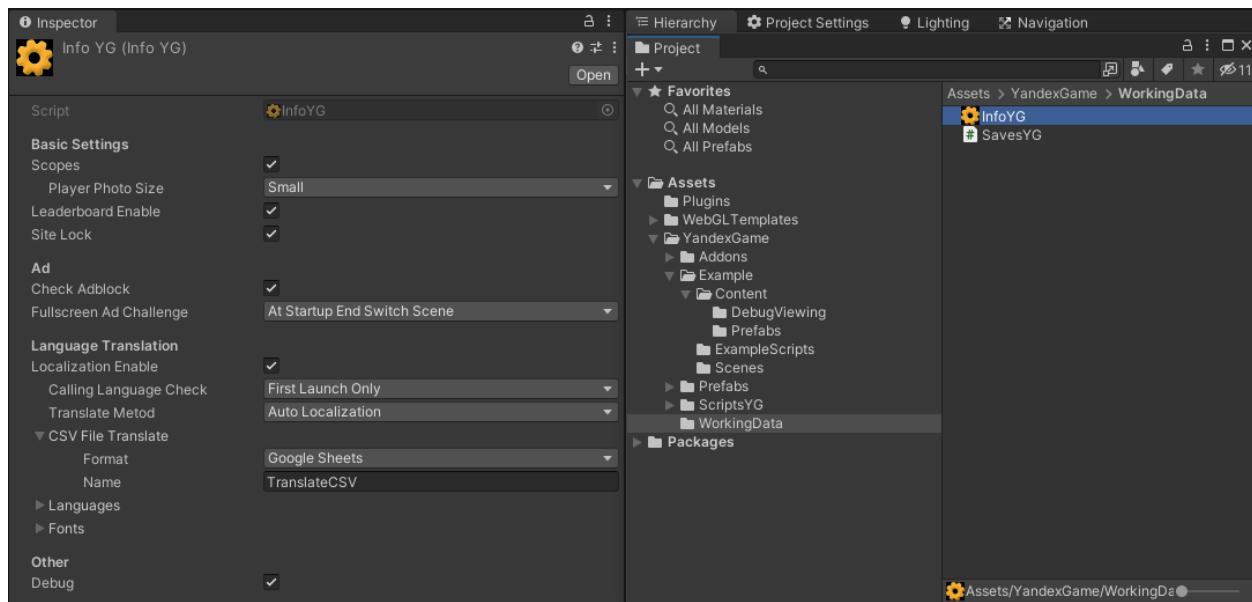


Изображение logo.png не нужно делать большого разрешения, иначе картинка растягивается за пределы нормы, и появится полоса прокрутки, что повлечёт за собой отклонение игры модераторами.

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.

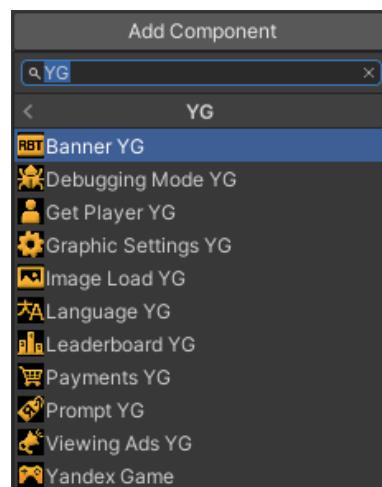
В дальнейшем настройки плагина я буду называть только **Info YG**.

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:



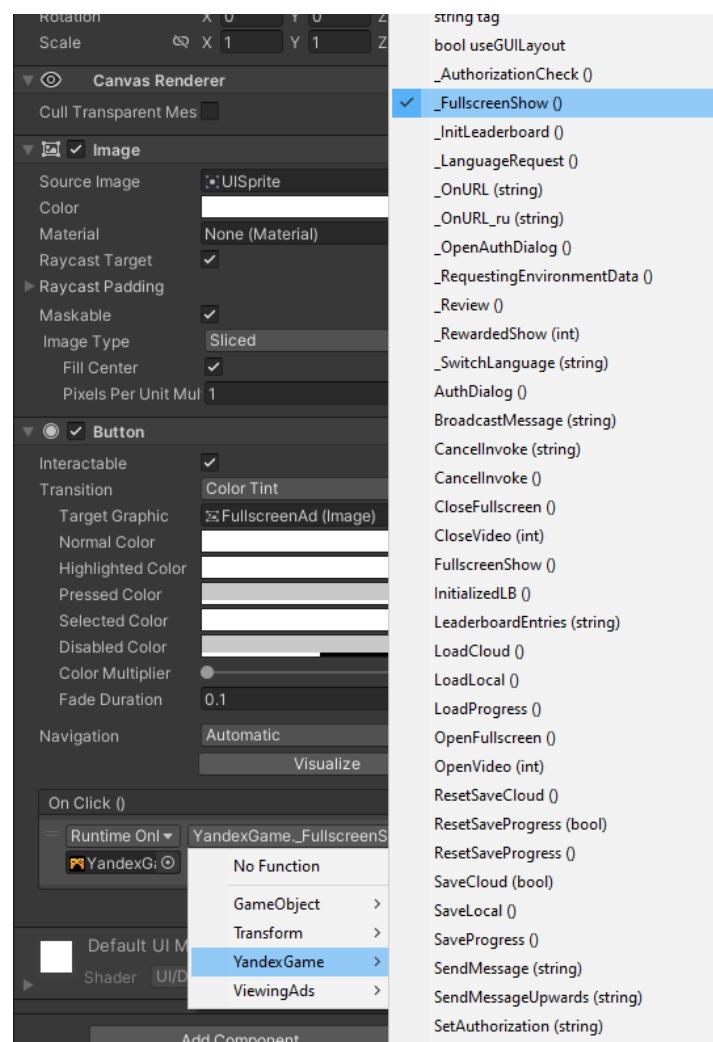
Также это значит: чтобы обратиться к скриптам плагина через Ваши скрипты, нужно подключать библиотеку YG таким образом: `using YG;`

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.вашаЗапись>

- 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.

▼ Gameplay Start

Разметка геймплея. Старт игры.

▼ Gameplay Stop

Разметка геймплея. Остановка игры.

▼ Set Fullscreen

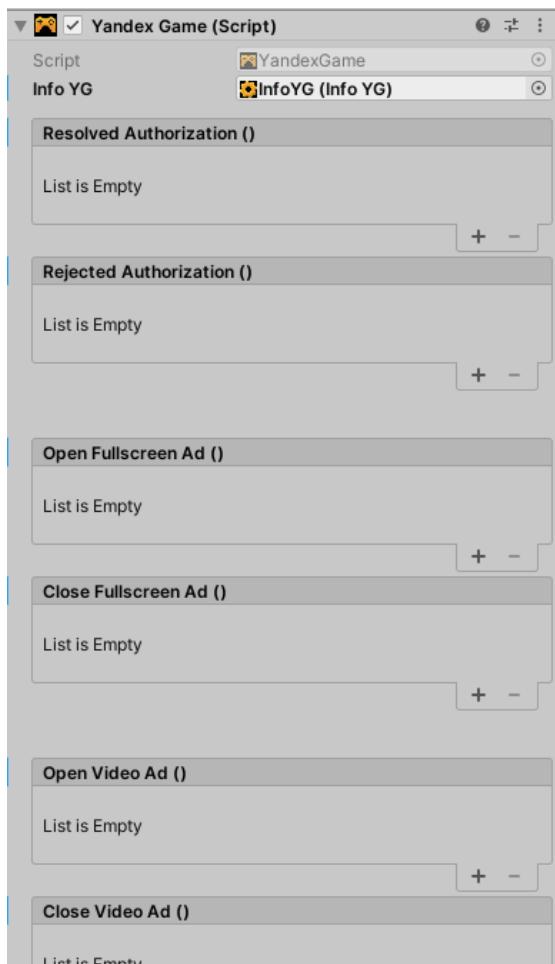
Установить полноэкранный режим в true или false.



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 подписаны 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.

Game Ready API



Game Ready API обязательное требование для размещения игры на Яндекс Играх!

Метод из SDK Яндекс при выполнении которого отражается 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 сразу после загрузки игры.

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();`

Разметка геймплея (Gameplay Start/Stop)



Рекомендуется к использованию! Разметка геймплея может стать обязательным требованием.

Прочтите [документацию разметки геймплея Яндекс Игр](#), чтобы понять где в игре необходимо вставлять методы старта и остановки игры.

Есть требование выполнять методы при уходе и возвращению на вкладку игры. Также необходимо выполнять методы при просмотре рекламы. Этого делать не нужно! Плагин сделает за Вас.

Выполните метод `YandexGame .GameplayStart();` для обозначения старта игры.

Метод `YandexGame .GameplayStop();` для остановки игры.

Плагин также запоминает состояние геймплея при уходе с вкладки игры, и по возвращению выполнит метод `GameplayStart()` только если при выходе состояние игры было активное.

Плагин не сделает лишних вызовов методов SDK Яндекс Игр старта и остановки. Можете выполнять методы плагина старта и остановки сколько угодно.

Вы можете узнать состояние геймплея с помощью поля `YandexGame .isGamePlaying`. Поле будет равно `true`, когда игра активна.

Для большего контроля, Вы можете воспользоваться ивентами закрытия и открытия рекламы и ивентами, которые срабатывают при закрытии/открытии вкладки с игрой.

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.

📞 Инициализация

SDK Яндекс Игр и плагин инициализируется до полного запуска игры. Вы можете пользоваться функциями плагина не дожидаясь пока что-то инициализируется, но Вам всё же может пригодится ивент, который вызывается после инициализации и проверка на инициализацию.

Ивент `YandexGame .GetDataEvent` срабатывает после инициализации. Также ивент сработает, если вы вручную вызовете загрузку сохранений или сброс сохранений.

Проверку на инициализацию Вы сможете сделать с помощью boolean поля `YandexGame .SDKEnabled`. Равно `true` когда плагин инициализирован.

События вкладки игры

При закрытии/открытии вкладки с игрой вызовется ивент:

`YandexGame . onVisibilityWindowGame <bool visible >` — требует подписать метод с параметром boolean типа.

visible = `true`, если вкладка была открыта, и `false`, если вкладка была закрыта.

Также есть ивент открытия вкладки `onShowWindowGame` и ивент закрытия вкладки `onHideWindowGame`.

И поле `YandexGame.isVisibilityWindowGame` — если равно `true`, значит вкладка в фокусе.

Пример использования:

```
using YG;

// Подписываемся на событие открытия/закрытия вкладки игры
private void OnEnable()
{
    YandexGame.onVisibilityWindowGame += OnVisibilityWindowGame;
}

// Отписываемся от события открытия/закрытия вкладки игры
private void OnDisable()
{
    YandexGame.onVisibilityWindowGame -= OnVisibilityWindowGame;
}

// Метод, который выполнится при открытии/закрытии вкладки игры
void OnVisibilityWindowGame(bool visible)
{
    // Ваша логика
}
```

Второй пример с событиями без параметра:

```
using YG;

private void OnEnable()
{
    YandexGame.onShowWindowGame += OnShowWindowGame; // Подписываемся на открытие
    YandexGame.onHideWindowGame+= OnHideWindowGame; // Подписываемся на закрытие
}

private void OnDisable()
{
    YandexGame.onShowWindowGame -= OnShowWindowGame; // Отписываемся от открытия
    YandexGame.onHideWindowGame-= OnHideWindowGame; // Отписываемся от закрытия
}

void OnShowWindowGame()
```

```
{  
    // Ваша логика при открытии вкладки игры  
}  
void OnHideWindowGame()  
{  
    // Ваша логика при закрытии вкладки игры  
}
```

Fullscreen методы (Полный экран)

Используйте метод `YandexGame. SetFullscreen (bool)` для установки экрана в полноэкранный режим или сброса полного экрана.

Метод

`SetFullscreen` принимает параметр boolean типа. Если параметр равен `true`, то будет установлен полноэкранный режим, `false` — сброс полного экрана.

Также есть поле `YandexGame. isFullscreen`, с помощью которого вы сможете узнать полный экран сейчас включён или нет.

Метод серверного времени

Используйте метод `YandexGame .ServerTime()` для того, чтобы узнать текущее время. Это может пригодиться, например, для выдачи ежедневного вознаграждения.

Метод `ServerTime()` возвращает переменную типа `long`. Полученное число — это время в миллисекундах, оно будет выглядеть примерно так: 1721201231000.

В Unity Editor будет отображаться фиксированное число для имитирования серверного времени, которое Вы можете установив в настройках плагина **InfoYG → Player Info Simulation → Server Time**.

A 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()`.

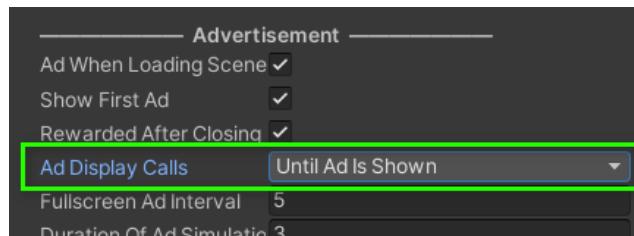
Заглушка перед рекламой

Реклама может загружаться с задержкой. Из-за этого может возникнуть ситуация, что реклама между уровнями показывается после загрузки уровня - получается в момент игры. Это не соответствует [правилам Яндекс Игр](#).

Если в игре между уровнями происходит загрузка в течении пары секунд, то наверняка реклама успеет загрузиться до открытия уровня. В ином случае есть смысл остановить игру до показа рекламы. В этом поможет компонент

Ad Notification YG.

При использовании заглушки, в настройках плагина **InfoYG** параметр **Ad Display Calls** выставьте на **Resetting Timer After Any Ad Display**. Иначе, с другим значением "Until Ad Is Shown" заглушка будет показываться при каждом запросе рекламы, т.к. таймер сбрасывается, если реклама не была показана.



Возмите готовый префаб **Ad Notification** в папке **YandexGame → Prefabs → Ads** и поместите на первую сцену, которая будет загружаться в проекте. Объект с компонентом **Ad Notification**

будет синглтоном. Скрипт имеет пояснения во всплывающих подсказках.

В

InfoYG можете поставить симуляцию задержки открытия рекламы для Unity Editor с помощью параметра

Load Ad With Delay Simulation. Так вы сможете протестировать заглушку. Заглушка работает и для рекламы за вознаграждение.



Лучше не ставить чисто чёрный или чисто белый цвет для заглушки, т.к. у Я.И. есть правило не заполнять такими цветами фон неигровой области.



Вы можете написать на заглушки например "Загрузка", но лучше не писать "Загрузка рекламы", т.к. в таком случае заглушки может промелькнуть слишком быстро и могут возникнуть вопросы относительно уведомления о загрузке рекламы.

Реклама в процессе игры (таймер 2с)

Готовое решение для рекламы в процессе игры. О вариантах отображения рекламы можете [почитать здесь](#).

Найдите префаб

YandexGame → Prefabs → ads → Timer Before Ads. Компонент **Timer Before Ads YG** простой в понимании и имеет всплывающие подсказки.

Объект не синглтон. Скрипт сам открывает таймер, когда рекламу можно показывать, и сам вызывает показ рекламы после отчёта таймера. Последняя цифра или любой ваш объект будет активен пока загружается реклама в течении пары секунд.

Рекомендуется изменить внешний вид интерфейса.

Галочка **Pause To_Viewing Ads YG = true** означает, что при показе таймера игра будет остановлена с помощью компонента **Viewing Ads YG**. Вы можете сделать свою реализацию остановки геймплея с помощью ивента **Do Pause**.



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 подписанном 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 некорректная 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 = Remember Previous State**.
3. Поставьте галку **Awake Set Values**.
4. Настройте параметры при старте сцены в **Awake Values** как вам нужно.

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 **Использовать API для показа sticky-баннера**.
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.



Все 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 .lang`

типа `stringify`

- Возвращает язык сортируя его из выбранных языков в настройках плагина InfoYG.

▼ YandexGame .auth

тип boolean

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

▼ YandexGame .playerName

тип stringify

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

▼ YandexGame .playerId

тип stringify

- ID игрока

▼ YandexGame .adBlock

тип boolean

- Активность функции **Check AdBlock**

▼ YandexGame .playerPhoto

тип stringify

- Ссылка **изображения аватарки игрока**

▼ YandexGame .photoSize

тип stringify

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

▼ YandexGame .nowAdsShow

тип boolean

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

▼ `YandexGame .nowFullAd`

типа `boolean`

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

типа `boolean`

- `true` — Игра открыта в полноэкранном режиме.
- `false` — Игра не развернута на полный экран.

▼ `YandexGame .isVisibilityWindowGame`

типа `boolean`

- `true` — Вкладка игры в фокусе.
- `false` — Вкладка игры не активна.

▼ `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

типа [stringify](#)

- О параметре payload читайте в разделе [Deep Linking](#)

▼ YandexGame . EnvironmentData .promptCanShow

типа [boolean](#)

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

▼ YandexGame . EnvironmentData .reviewCanShow

типа [boolean](#)

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

▼ YandexGame . EnvironmentData .platform

типа [stringify](#)

- Платформа, на которой запущена игра.
- Какие платформы могут быть смотрите в [других источниках](#).

▼ YandexGame . EnvironmentData .browse

типа [stringify](#)

- Браузер, в котором запущена игра.

Список браузеров:

- [Yandex](#)
- [Opera](#)
- [Firefox](#)
- [IE](#)

- Edge
- Chrome
- Safari
- Other

▼ YandexGame .purchases

типа 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 .price

типа stringify

- Стоимость товара в формате <цена> <код валюты>.

▼ YandexGame . Purchase .priceValue

типа `stringify`

- Стоимость товара в формате `<цена>`.

▼ `YandexGame . Purchase .priceCurrencyCode`

типа `stringify`

- Код валюты.

▼ `YandexGame . Purchase .currencyImageURL`

типа `stringify`

- Адрес иконки валюты.

▼ `YandexGame . Purchase .consumed`

типа `boolean`

- **Использована ли покупка**

- `true` — использована;

- `false` — не использована.

▼ `YandexGame .PayingStatus`

Четыре возможных значения, зависящих от частоты и объема покупок пользователя:

▼ **paying**

Пользователь купил портальную валюту на сумму более 500 рублей за последний месяц.

▼ **partially_paying**

У пользователя была хотя бы одна покупка портальной валюты реальными деньгами за последний год.

▼ **not_paying**

Пользователь не делал покупок портальной валюты реальными деньгами за последний год.

▼ **unknown**

Пользователь не из РФ или он не разрешил передачу такой информации разработчику.
В плагине unknown ещё означает — неавторизованный.

Сохранения

В плагине предусмотрено 3 вида сохранений:

1. Для тестирования в Unity Editor — при разработке игры в Unity, данные сохраняются в файл SavesEditorYG.json ([подробнее ниже](#)).
2. Облачные сохранения — сохранения на облако Яндекса. Для работы облачных сохранений необходимо в черновике поставить галочку напротив "Игра использует облачные сохранения".
3. 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:

1. Создать свои данные для сохранений в отдельном скрипте "savesData".
2. В Вашем скрипте присвоить созданным Вами сохранениям в классе "savesData" - значения, которые требуется сохранить.
3. Выполнить метод сохранения.
4. 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

1. Откройте скрипт **SavesYG**, расположенный по пути **YandexGame** → **WorkingData**.
2. Под комментарием `// Ваши сохранения` уже есть некоторые поля. Это поля для демо-сцены, вы можете их удалить.

3. Запишите свои поля. Вы можете задать им какие-то значения, тогда эти значения будут дефолтными, и при первой загрузке игры данные буду 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;`



Для версий ниже 1.6, 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 момент
        // но он запустится при вызове события GetDataEvent, после про-
    }
}

// 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** → **Editor**. Вы можете сбросить сохранения удалив файл **SavesEditorYG.json**.

Есть метод для сброса сохранений `YandexGame .ResetSaveProgress()`. Он не удаляет файл **SavesEditorYG.json**, 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 1.5смотрите документацию внутри плаcтина.

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 лидербордов поле boolean типа `YandexGame .initializedLB` будет равно `true`.

Есть метод `YandexGame .GetLeaderboard`, он имеет параметры необходимые для заполнения.

После выполнения метода `GetLeaderboard` вызовется событие `YandexGame .onGetLeaderboard`. Оно передаёт класс **LBData** 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 `LBPlayerData` 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.**

Обратите внимание. Вы можете получить ранк и рекорд пользователя из `thisPlayer`. Но это не официальные данные. Они получены путём поиска пользователя из списка `players` по уникальному id игрока.



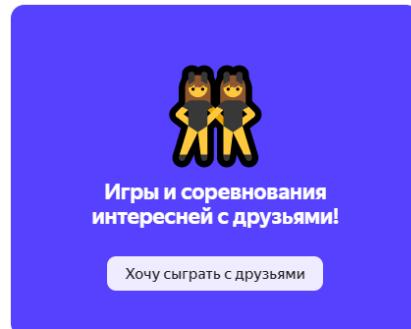
В таблицу запишется новое число, даже если оно меньше предыдущего. Необходимо самостоятельно выполнять сравнение рекорда перед новой записью.

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

Чтобы лидерборды работали необходимо включить параметры **Scopes** и **Leaderboard Enable** в **Info YG**.

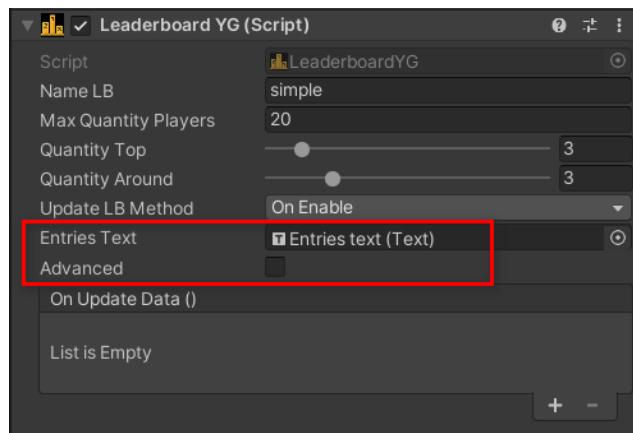
Отображением таблицы занимается скрипт **Leaderboard YG**. Все его опции имеют всплывающие подсказки. Обязательно заполните параметр **Name LB** — это техническое название таблицы, которое мы записывали в консоли разработчика.

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

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

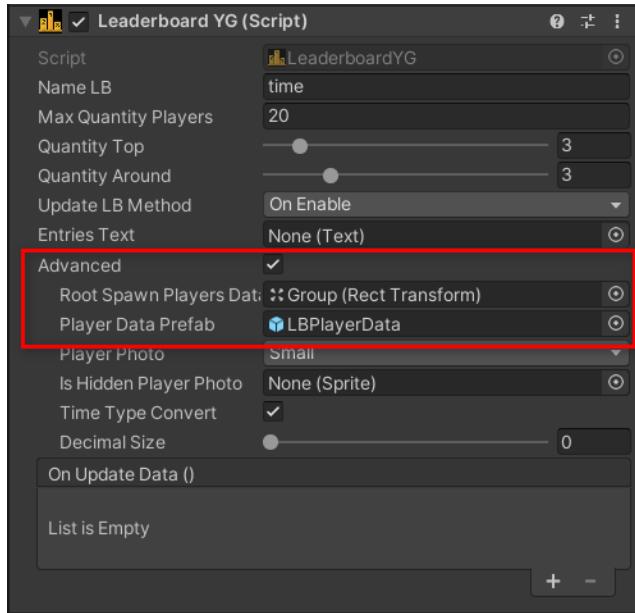
▼ Простой вариант отображения таблицы

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 (Advanced) вариант отображения таблицы

Отдельными блоками отображает всех игроков, их рейтинг, аватар, ник, рекорд. 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 следующим образом:

Info YG → Leaderboards → Leaderboard Simulation — это массив таблиц для симуляции. Элемент этого массива — это лидерборд со всеми параметрами, и все их настраивать необязательно. Не все данные влияют на симуляцию. Все данные отображаются, потому что используют тот же класс, в котором хранится информация о реальной таблице. Благодаря этому можно посмотреть какие реальные данные существуют.

1. Создайте таблицу в массиве или измените уже имеющуюся.
2. Запишите **Techno Name**.
3. **Entries** — будет работать для не продвинутой (no Advanced) таблицы.
4. **Is Invert Sort Order** — можете установить порядок расстановки по возрастанию или по убыванию. Также, как настраиваете в консоли ЯИ.
5. Для отображения игроков в таблице создайте их, каждый элемент массива это игрок отображающийся в таблице. Заполните данные игрока по своему усмотрению.

Rank — позиция в таблице (в симуляции выставляется автоматически).

Score — рекорд игрока. По рекорду будет выставляться расстановка игроков.

Name — имя игрока. Если назвать anonymous, то будет симуляция анонимного игрока.

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 );
```

Новый рекорд вы передаёте в типе `float`, число может выглядеть, например, так: 180,135.

Рекорд записывается в формате «секунды». Например, рекорд «3 минуты» должен записываться как число «180».

▼ Сейчас имеется баг с целым числом!

Когда вы тестируете лидерборд, вы можете передавать целое число, например 180. Целое число будет некорректным. Для конвертации, такое целое число должно выглядеть так: 180000.

В реальной игре вероятность выпадания целого числа ничтожно мала.

Просто, тестируйте с числами после плавающей точки (например 180,1). Или умножайте целые числа на 1000. В версиях плагина 2.0+ это будет исправлено.

Код таймера в вашей игре может выглядеть следующим образом:

```
using YG;

class Class: MonoBehaviour
{
    float timer;

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

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

Дополнительные опции при работе с кодом

После инициализации лидербордов поле boolean типа `YandexGame .initializedLB` будет равно true.

Есть метод получения данных лидерборда `YandexGame .GetLeaderboard`, он имеет параметры необходимые для заполнения.

После выполнения метода `GetLeaderboard` вызовется событие `YandexGame .onGetLeaderboard`.

Событие передаёт класс

LBData 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 `LBPlayerData` 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.**

Обратите внимание. Вы можете получить ранк и рекорд пользователя из `thisPlayer`. Но это не официальные данные. Они получены путём поиска пользователя из списка `players` по уникальному id игрока.

Пример:

```
string technoName = "myLeaderboard";
int rankPlayer;

// Получение данных лидерборда
public void GetLeaderboard()
{
    YandexGame.GetLeaderboard(technoName, 10, 3, 3, "small");
    // Последний параметр photoSizeLB может быть "nonePhoto", "small", "medium"
}

// Подписываемся и отписываемся от ивента получения данных
private void OnEnable()
{
    YandexGame.onGetLeaderboard += OnGetLeaderboard;
}
private void OnDisable()
```

```

{
    YandexGame.onGetLeaderboard -= OnGetLeaderboard;
}

// Метод получающий данные
private void OnGetLeaderboard(LBData lb)
{
    // Сверяем имя лидерборда
    if (lb.technoName == technoName)
    {
        // Получаем данные. В данном случае, получаем ранк пользователя.
        rankPlayer = lb.thisPlayer.rank;
    }
}

```

Локализация

Плагин поставляется с инструментами локализации. Ниже будет описание работы с ними. Если же Вам нужно только брать язык из SDK Яндекс Игр, используйте одно из двух полей:

▼ `YandexGame . EnvironmentData .language`

типа `stringify`

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

▼ `YandexGame .lang`

типа `stringify`

- Возвращает язык сортируя его из выбранных языков в настройках плагина InfoYG.

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 сохраняться. 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 количества 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.

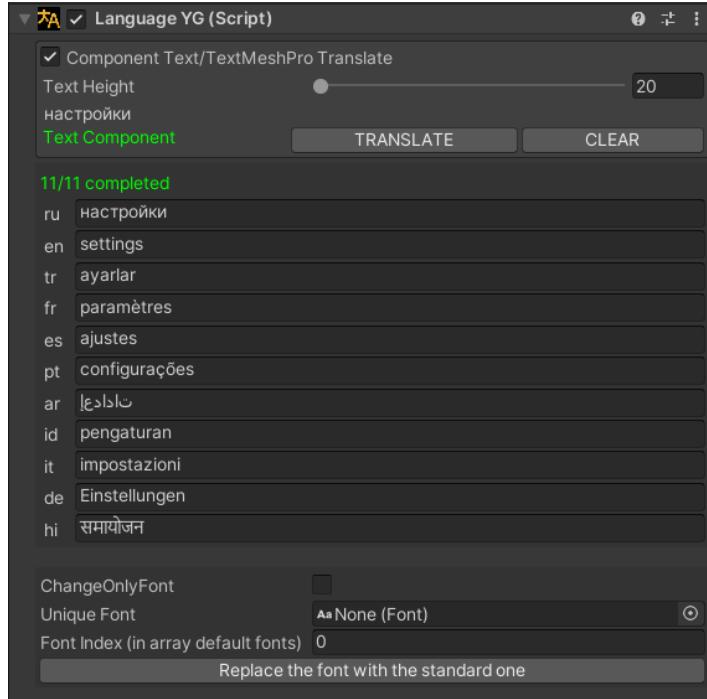


Excel не всегда правильно читает csv файл. Если у Вас возникнут с этим проблемы, откройте csv файл в Google Sheets. Это можно сделать онлайн. И через Google Sheets уже можно и пересохранить файл, тогда он и в Excel откроется.



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

Прочтите 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 иVENTы успешной/неуспешной покупки
private void OnEnable()
{
    YandexGame.PurchaseSuccessEvent += SuccessPurchased;
    YandexGame.PurchaseFailedEvent += FailedPurchased; // Необязательно
}

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

// 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)
{
    // Например, можно открыть уведомление о неуспешности покупки.
}

```

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**.

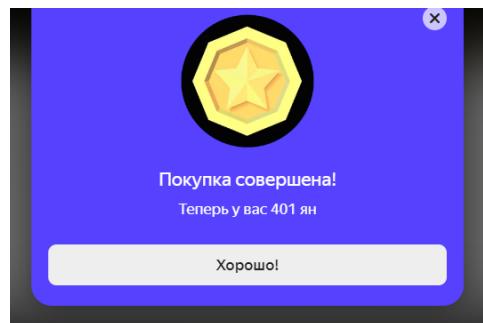
Consume Purchases YG - простой скрипт, который после инициализации SDK один раз вызывает метод `YandexGame .ConsumePurchases();`.

Метод `ConsumePurchases()` применяет все необработанные покупки, но как? - вызывает вышерассмотренный 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.

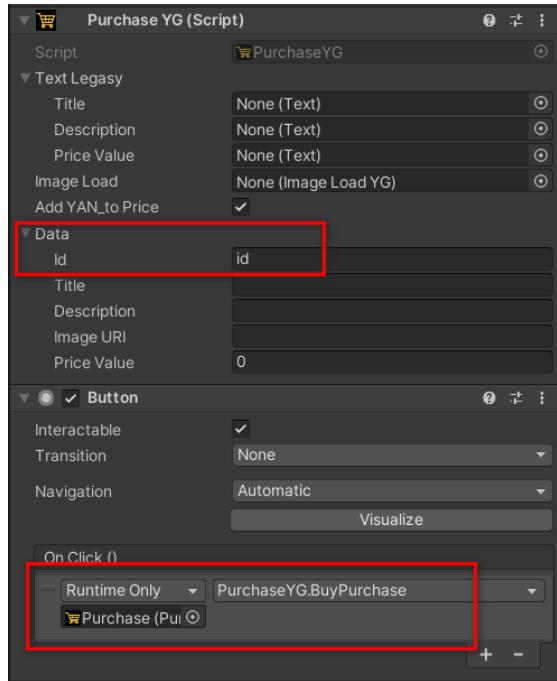
Компоненты покупок

▼ 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**.

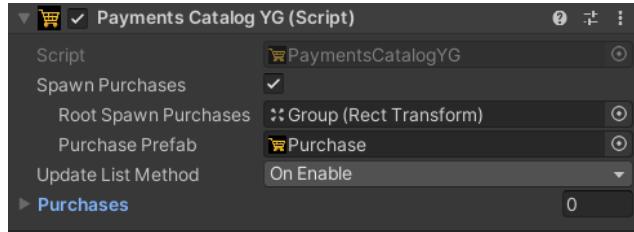
▼ **1 вариант** — автоматическое отображение всего каталога покупок.

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**.



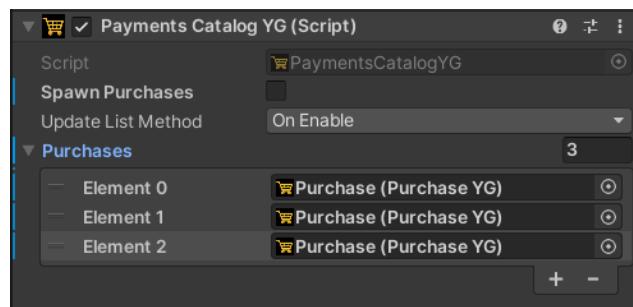
▼ **2 вариант** — создать список покупок.

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.

▼ `price`

типа `stringify`

- Стоимость товара в формате `<цена> <код валюты>`.

▼ `priceValue`

типа `stringify`

- Стоимость товара в формате `<цена>`.

▼ `priceCurrencyCode`

типа `stringify`

- Код валюты.

▼ `currencyImageURL`

типа `stringify`

- Адрес иконки валюты.

▼ `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.

Поле `YandexGame .PayingStatus` — Четыре возможных значения, зависящих от частоты и объема покупок пользователя:

▼ **paying**

Пользователь купил портальную валюту на сумму более 500 рублей за последний месяц.

▼ **partially_paying**

У пользователя была хотя бы одна покупка портальной валюты реальными деньгами за последний год.

▼ **not_paying**

Пользователь не делал покупок портальной валюты реальными деньгами за последний год.

▼ **unknown**

Пользователь не из РФ или он не разрешил передачу такой информации разработчику. В плагине unknown ещё означает — неавторизованный.



Deep Linking

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.



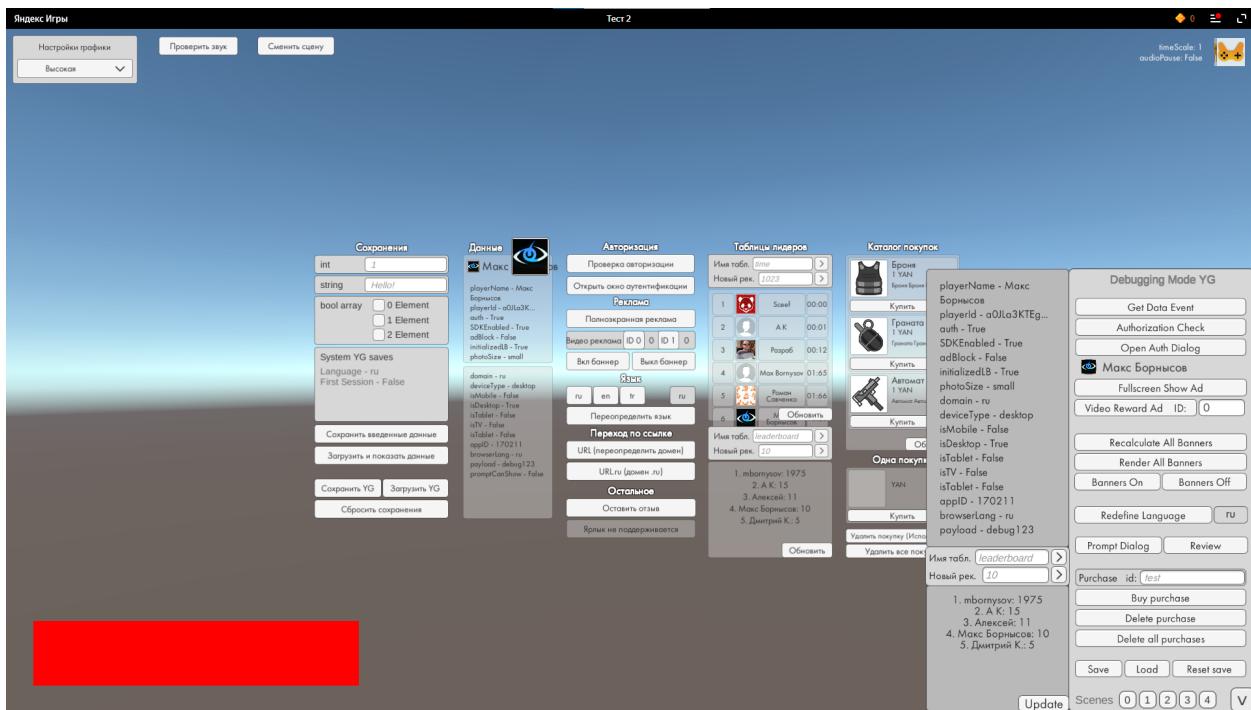
Инструмент для отладки

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

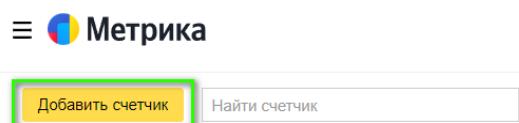
`YandexGame . EnvironmentData . reviewCanShow` станет равен `false` на данную игровую сессию.



Настройка метрики

▼ Описание процесса создания счётчика и метрик в сервисе "Яндекс.Метрика".

1. Создайте счётчик в "Яндекс.Метрика"



Новый счетчик

1

Задайте настройки

2

Установите код счетчика на сайт

Имя счетчика

Название игры, например

Адрес сайта ?

yandex.ru/games/app/257467

+ Дополнительные адреса ? Принимать данные только с указанных адресов ? Включая поддомены

По желанию

Часовой пояс

(GMT+03:00) Москва, Санкт-Петербург

Автоматические цели

Вкл

Метрика будет автоматически создавать цели на некоторые важные события на сайте — например, на клики по номеру телефона или email. [Подробнее](#)

Вебвизор, карта скроллинга, аналитика форм

Откл

Подробные записи действий посетителей на сайте: движения мышью, прокручивание страницы и клики.

 Я принимаю условия [Пользовательского соглашения](#)

2. Создайте цель

Счетчики Настройка аккаунта API Помощь

EpicShooter yandex.ru/games/app/191541 · 95086567

Цели

Цели

Сводка Отчеты Вебвизор Посетители Карты Цели Конверсии Эксперименты Сегменты Интеграции Привлечение клиентов Настройка

Добавить цель

Название цели*

Название метрики

Ретаргетинг

Тип условия*

Количество просмотров Посещение страниц JavaScript-событие Составная цель

Клик по номеру телефона Клик по email Отправка формы Переход в мессенджер

Скачивание файлов Поиск по сайту Клик по кнопке Переход в соц. сеть

Возвращение из платежной системы

Позволяет отследить нажатие на кнопку, заполнение формы и другие события на сайте. Такая же цель понадобится и для передачи онлайн-конверсий. Подробнее: Диагностика проблем.

Идентификатор цели: совпадает с id

Код цели для сайта
utm(95086567,'reachGoal','id')

Скопировать код

Доход

Добавить цель Отмена

3. Скопируйте номер счётчика ([Как найти номер счетчика](#))

Код счетчика

Скопировать код

При установке счетчика в HTML-код сайта, разместите код как можно ближе к началу страницы. Например, в пределах тегов <head></head> или <body></body>.

[Другие способы установки](#)

```
<!-- Yandex.Metrika counter -->
<script type="text/javascript" >
  (function(m,e,t,r,i,k,a){m[i]=m[i]||function(){(m[i].a=m[i].a||[])
    .push(arguments)};
    m[i].l=1*new Date();
    for (var j = 0; j < document.scripts.length; j++) {if
    (document.scripts[j].src === r) { return; }}
    k=e.createElement(t),a=e.getElementsByTagName(t)
    [0],k.async=1,k.src=r,a.parentNode.insertBefore(k,a)};
    (window, document, "script", "https://mc.yandex.ru/metrika/tag.js",
    "ym");
    ym[95271522].init, {
      clickmap:true,
      trackLinks:true,
      accurateTrackBounce:true
    });
</script>
<noscript><div></div></noscript>
<!-- /Yandex.Metrika counter -->
```

Устанавливая код счетчика на сайт, вы соглашаетесь со всеми условиями [Пользовательского соглашения](#)

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

После создания и настройки счётчика - автоматическая метрика уже будет отправляться. Но Вы так же можете отправлять свои метрики по событиям в игре.

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

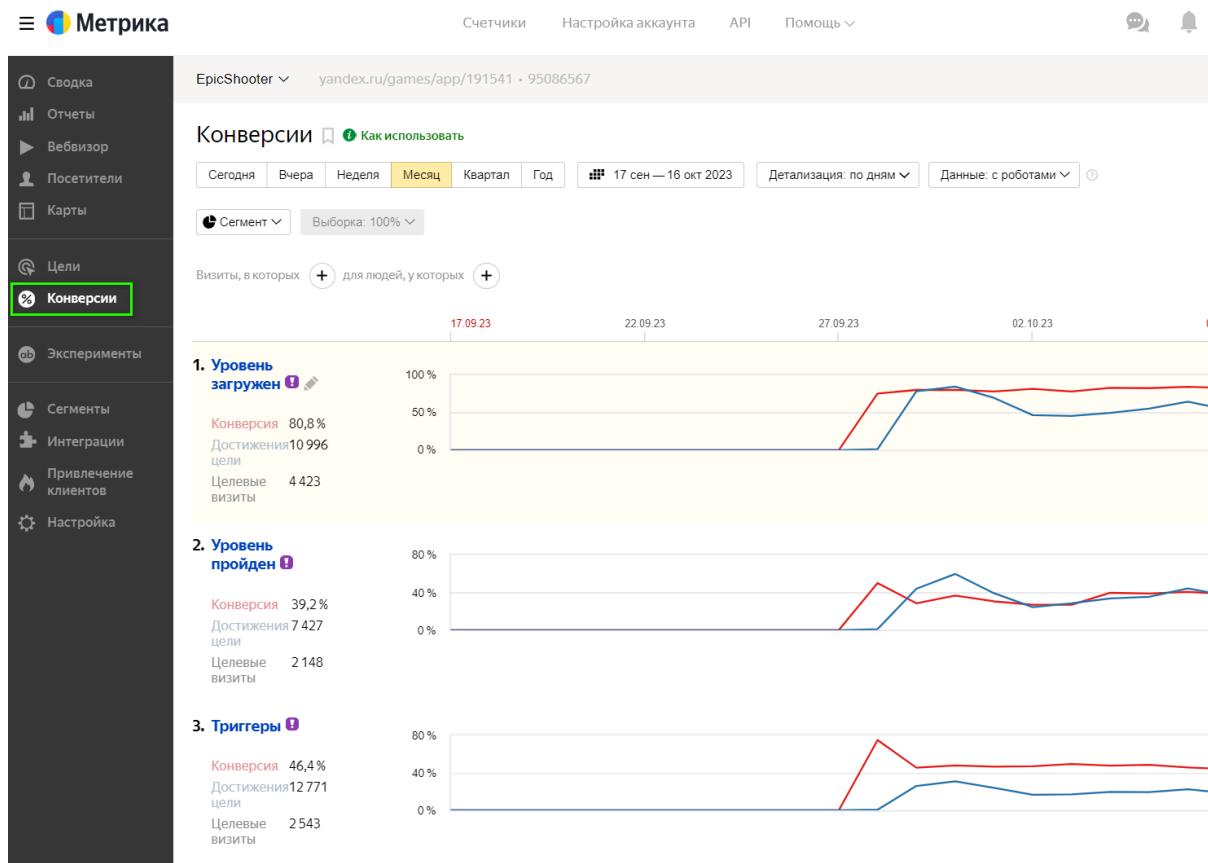
Метод `YandexMetrica. Send(string eventName)` с одним параметром типа "string" используется для отправки метрики без параметров.

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".

После отправки метрик, через некоторое время на сайте Вы сможете увидеть отправленные метрики в разделе "Конверсии":

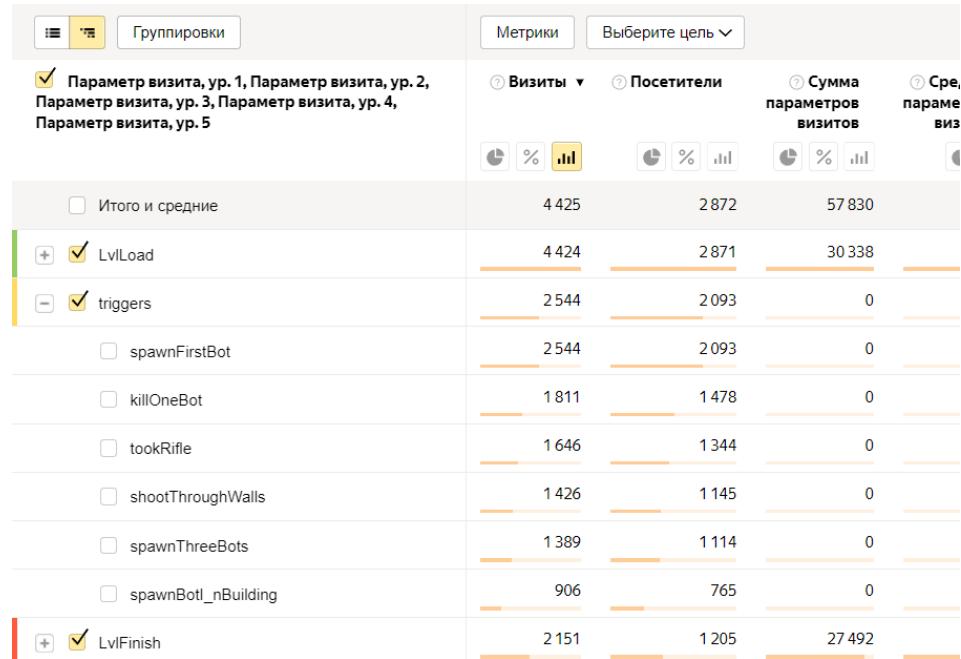


▼ Подробнее о дополнительных параметрах

Вложенные метрики могут быть более удобными и информативными.

Рассмотрим пример отправки событий в метрики по входению в триггеры в игре. В таком случае, у нас будет одна цель, которая в себе будет содержать сколько угодно событий.

Выглядит в метриках это так:



Необходимо создать лишь одну цель, и записывать в неё события с любым именем. И события будут в одном месте, будто в папке.

Пример класса для отправки метрик:

```
using System.Collections.Generic;
using UnityEngine;
using YG;

public class MetricaSender : MonoBehaviour
{
    public void Send(string id)
    {
        YandexMetrica.Send(id);
    }

    public void TrigerSend(string name)
    {
        var eventParams = new Dictionary<string, string>
        {
            { "triggers", name }
        };

        YandexMetrica.Send("triggers", eventParams);
    }
}
```

Метод `Send` в данном случае отправляет метрику без вложений.

Метод `TriggerSend` отправляет вложенные метрики в цель "triggers". В игре, при входе персонажем в триггер выполняется метод `TriggerSend`, в перегрузку записывается наименование триггера.

Посмотреть вложенные метрики на сайте:

The screenshot shows the Yandex Metrika interface. The left sidebar has a dark theme with various sections: Сводка, Отчеты (highlighted with a green box), Вебвизор, Посетители, Карты, Цели, Конверсии, Эксперименты, Сегменты, Интеграции, Привлечение клиентов, and Настройка. The main content area has tabs: Все отчеты (highlighted with a yellow box), Кто заходит, Откуда, Что смотрят, Как пользуются, Как часто, and Отдача от рекламы. Below these are three boxes: Конверсии, Посещаемость, and Источники, сводка. The main content area also includes: Популярное, Страницы входа, Страницы выхода, Заголовки страниц, По параметрам URL, Кнопка «Поделиться», Внешние переходы, Загрузки файлов, and Параметры визитов (highlighted with a green box). A search bar is at the top right. The bottom section lists: Параметры посетителей, Параметры визитов, and Параметры посетителей again.

🧩 Модули

📺 Яндекс TV

🆎 Div Adaptive Banner

💡 Яндекс Музыка Виджет

▶ Флаги (удаленная конфигурация)

📋 Надпись при загрузке игры

📌 Релизы

Версия 1.6.3 critical

! Критически важная версия! Рекомендуется обновиться, с сентября игры со старым API будут отклонять.
Скоро выходит PluginYG 2.0. Можно его подождать.



Изменён index.html

▼ Новый лоадер SDK Яндекс Игр (*критично*)

Изменён путь SDK `https://yandex.ru/games/sdk/v2` на `/sdk.js`. Новые требования платформы.

▼ Внутриигровые покупки (*критично*)

- Обновлён скрипт **PurchaseYG**, теперь он получает корректный код валюты. Добавлена опция подгрузки изображения валюты.
- В данные покупок (класс `Purchase`) добавлены поля `price`, `priceCurrencyCode` и `currencyImageURL`.

▼ Лидерборды

- Компонент LeaderboardYG теперь имеет улучшенную симуляцию для тестирования в Unity Editor. Будут также как в ЯИ отображаться первые игроки с лучшим рекордом и игроки вокруг текущего пользователя. Порядок расположения игроков в таблице будет зависеть от рекорда, который можно выставить у игроков в симуляции таблиц в InfoYG. Также, теперь на симуляцию повлияет галочка **Is Invert Sort Order**. Можно у какого-нибудь игрока из списка выставить **Unique ID**, например, 000. И в разделе **Player Info Simulation** выставить ID текущему игроку 000, тогда в таблице будет отмечаться текущий игрок и вокруг него будут выстроены ближайшие по рекорду игроки.
- При использовании события `onGetLeaderboard` отдельно от скрипта LeaderboardYG при старте игры информация передаваемая событием приходила с задержкой (после инициализации лидербордов, когда поле `initializedLB` было = `true`). Теперь данного поля нет и не нужно ждать инициализации, со старта данные приходят.
- Реорганизация и перемещение файлов и кода касающихся лидербордов.
- Теперь рекорд отправляется `long` типом (метод `NewLeaderboardScores`).

▼ Paying Status (статус платёжеспособности пользователя)

Добавлено поле **payingStatus**. Его можно найти в **InfoYG → Player Info Simulation** и установить значение для симуляции в эдиторе. В коде получить значение можно так:
`YandexGame .payingStatus`. Вернётся `string` значение.

Описание опции — четыре возможных значения, зависящих от частоты и объема покупок пользователя:

- **paying** — пользователь купил портальную валюту на сумму более 500 рублей за последний месяц.
 - **partially_paying** — у пользователя была хотя бы одна покупка портальной валюты реальными деньгами за последний год.
 - **not_paying** — пользователь не делал покупок портальной валюты реальными деньгами за последний год.
 - **unknown** — пользователь не из РФ или он не разрешил передачу такой информации разработчику. Плагин ещё устанавливает unknown, если пользователь не авторизирован.
- [Разметка геймплея \(Gameplay Start/Stop\)](#).
 - [Fullscreen методы \(Полный экран\)](#).
 - [События вкладки игры](#)
 - [Метод серверного времени](#)

▼ Timer Before Ads

Скрипт Timer Before Ads доработан (реклама в процессе игры). Яндекс Игры опубликовали статью с правилами показа рекламы. В связи с этим, по стандарту отчёт таймера будет начинаться с 2-ух секунд, а при активации таймера игра остановится с помощью компонента Viewing Ads YG. Можете сделать свою реализацию остановки, для этого теперь есть unity ивент Do Pause.

▼ Image Load YG

Усовершенствован компонент **Image Load YG**. Однаковые изображения не будут скачиваться несколько раз.

Версия 1.6.2



Немного изменён index.html

Исправления ошибок:

▼ Локализация

Была устранена путаница с получением параметра языка. Есть три поля: `YandexGame.lang`, `YandexGame.EnvironmentData.language`, `YandexGame.savesData.language`. При различных обстоятельствах поля могли выдавать ложную информацию. Сейчас всё стабильно. Советую использовать

`YandexGame.lang`, если используете локализацию от плагина. И `YandexGame.EnviromentData.language`, если нужно получить язык, который выдал SDK Яндекс Игр. `YandexGame.savesData.language` в будущем будет удалён.

▼ Исправлены ошибки компиляции

В InfoYG (настройки плагина) при нажатии на кнопки активации дополнительных библиотек для тех или иных задач, предположительно, в новых версиях Unity компиляция происходила по какой то причине не корректно. Сейчас производится дополнительная компиляция после всех операций. Но если всё же возникнут проблемы, они должны исчезнуть при следующей компиляции (измените и сохраните любой скрипт).

▼ Ивент onResetProgress

Вызывался раньше выполнения метода `OnEnable` стандартных скриптов, что затрудняло выполнить метод по событию при первом запуске игры. `onResetProgress` - вызывается при установке сохранений в дефолтные значения. Плагин вызывает ивент при первом запуске игры. Далее он будет вызываться только если вы выполните метод

`YandexGame.ResetSaveProgress()`.

▼ Background фоновое изображение при загрузке

Фоновое изображение загрузочного экрана не выгружалось после загрузки игры, что могло вызывать небольшую потерю оптимизации.

▼ Эмуляция ошибки при просмотре рекламы за вознаграждение

Снова работает. Параметр **Test Error Of Rewarded Ads In Editor** в Info YG - эмулирует вызов ошибки при просмотре рекламы за вознаграждение. (Только для Unity Editor).

- Другие правки...

Версия 1.6.1

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

! Версии плагина 1.6.1+ работают с версии Unity 2021.3.18+

▼ Данные о платформе и браузере

Информацию на какой платформе и в каком браузере запущена игра Вы сможете узнать из раздела "Данные".

▼ Реклама за вознаграждение

В плагине есть функция "анти-абуза" рекламы за вознаграждения. Это сделано для избежания накруток вознаграждения, когда реклама не открылась, но callback о вознаграждении приходил.

Также эта функция не давала вознаграждения, если во время просмотра рекламы перейти на другую вкладку или скрыть игру другим способом и таким образом пропустить просмотр рекламы. Это модерация не одобрила. Сейчас вознаграждение выдаётся если во время просмотра рекламы игра была в расфокусе. Но если реклама не откроется вовсе, то также "анти-абуз" сработает.

▼ Массив purchases от внутриигровых покупок

Для своей реализации покупок может понадобиться массив purchases. В версии плагина 1.6 массив инициализировался с задержкой после запуска игры. Сейчас данные массива приходят до выполнения вашего кода игры и можно использовать поле purchases со старта игры.

▼ Метод Reset Save Progress

Улучшена синхронизация двух устройств при сбросе прогресса.

▼ Вырезаны кастомные баннеры

Они доработаны и перемещены в отдельный модуль.

▼ Unity 2023

Заменены некоторые методы, которые являются устаревшими для Unity 2023+.

▼ Настройки дополнительных библиотек

В Info YG есть кнопки для активации и деактивации сторонних библиотек для тех или иных функций.

Теперь под кнопками надписи выделяются цветом в зависимости от активации библиотеки.

Теперь перед автоматическим импортом библиотек спрашивается разрешение.

После компиляции скриптов, выходит уведомление с просьбой перезагрузить Unity Editor. Перезагрузка Unity поможет избежать вопросов о возникших ошибках, т.к. были

обращения на то, что после активации библиотек были проблемы с инициализацией некоторых скриптов. После перезагрузки это должно прийти в норму.

- Другие исправления, доработки, перемещения файлов и кода

Версия 1.6



Необходимо удалить старую версию плагина перед импортом новой (папку WorkingData можно не удалять).

▼ Инициализация

- Загрузка игры будет происходить только после полной инициализация SDK Яндекс Игр.
- Обращение к данным плагина теперь необязательно. Раздел инициализации становится не актуальным.
- На локальном хосте игра запускается.

▼ Реклама

- Новый скрипт **Ad Notification YG** ([документация](#)).
- Новый скрипт **Timer Before Ads YG** ([документация](#)).
- В **Info YG** новый параметр **Load Ad With Delay Simulation**:

Задержка открытия полноэкранной рекламы. Может быть полезна для тестирования уведомления о том, что скоро откроется реклама, перед её показом (в момент ожидания рекламы).

- Для тестирования в Unity Editor ошибки при показе видео рекламы теперь есть флаг **Test Error Of Rewarded Ads In Editor** в **Info YG**. Если включить данный параметр, то при открытии рекламы за вознаграждение появится красный экран и вознаграждение не будет выдано.

▼ Улучшен скрипт Viewing Ads YG

- Увеличена надёжность. В том числе теперь скрипт деактивирует Event System во время показа рекламы.
- Параметр **Do Close Void On Awake** заменён на **Awake Set Values**.

▼ Лидерборды

- Рекорд запишется в таблицу только если пользователь авторизован.
- Возможность не записывать в таблицу скрытых пользователей.

Параметр

Save Score Anonymous Players в Info YG.

true

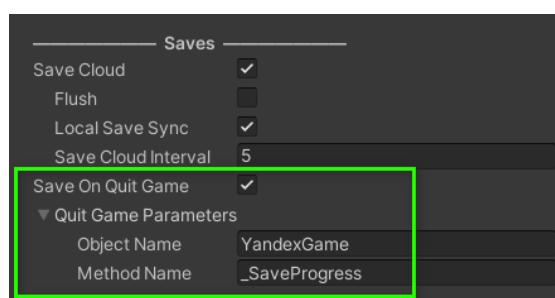
- анонимные пользователи будут записываться.

false - анонимные пользователи не будут записываться.



▼ Выполнение любого метода при закрытии игры.

Выполнение любого Вашего метода при закрытии или обновлении страницы с игрой.



Имеются всплывающие подсказки.

▼ Обновлена документация для метрик

Более подробная информация.

▼ Кодировка UTF-8

Не все скрипты имели необходимую кодировку UTF-8 для русского языка. Из-за этого русский язык у кого то мог не отображаться. Теперь все скрипты имеют кодировку UTF-8 поддерживающую кириллицу.

▼ Локализация

Исправлены некоторые малозначительные баги с компонентами **Language YG** и **Lang YG Additional Text**.

- Исправления мелких ошибок

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 неявная 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 **TMPPro** (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.

Раннее были только параметры:

▼ `YandexGame .nowFullAd`

типа boolean

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

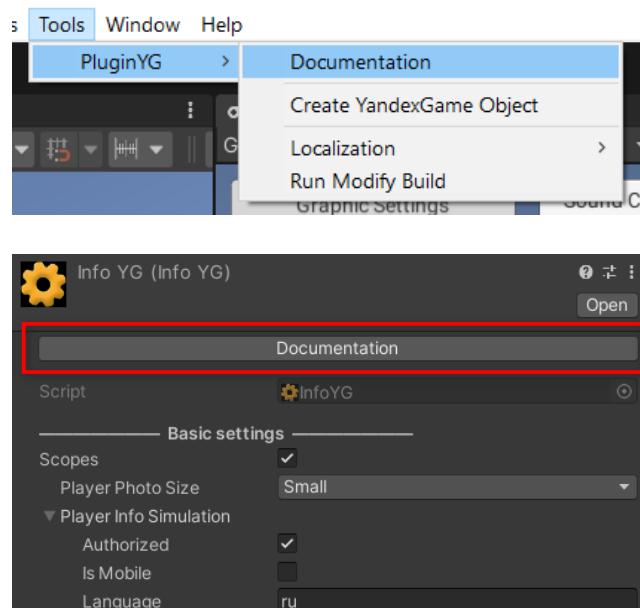
Для этих пунктов смотрите обновлённую документацию.

- Post process build система
- Скрипт оценки игры

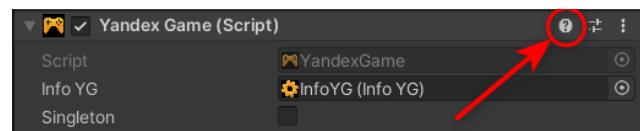
- Debugging инструмент
- Удаление лишнего
- Реструктуризация кода
- Подготовка к модульной системе

▼ 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 ивент 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 строку “`//#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 "**GetLeaderboardData**" 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 выполняться, 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 слепят 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)

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

▼ Фикс (https://t.me/yandexgame_plugin/12681)

Фикс (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!