

⚠ Внимание! [Более удобная и всегда актуальная документация на русском языке](#)

PluginYG Документация

✓ Содержание

✓ Содержание

Предисловие

Ссылки

Импорт плагина

Начало работы

Добавление Yandex Game на сцену

Выбор WebGL шаблона

Настройки плагина (Info YG)

Настройки графики от плагина

Настройки графики от плагина

Пространство имен YG

Работа с компонентом Yandex Game

Ивенты в компоненте Yandex Game

Инициализация

Полноэкранная реклама (Fullscreen Ad)

Видео реклама за вознаграждение (Reward Video Ad)

Античит (Check AdBlock)

Пауза игры при просмотре рекламы

Баннерная реклама

Настройки WebGL шаблона для баннеров

Типы баннеров

Создание динамического баннера

Создание статических баннеров

Данные игрока

Облачные сохранения

Создание своих данных для сохранения

Загрузка сохранений

Сохранение

Сброс сохранений

Лидерборды

Создание таблицы в консоли разработчика

Запись рекорда в соревновательную таблицу

Создание таблицы в Unity

Локализация

Шрифты

Внутриигровые покупки

Скрипт PaymentsYG

Активация процесса покупки

Обновление информации о товарах

Получение информации о товарах

Глубинное связывание (Deep Linking)

Инструмент для отладки

Ярлык на рабочий стол

Вызов диалогового окна

Скрипт PromptYG

Ярлык установлен

Предисловие

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

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

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

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

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

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

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

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

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

https://www.youtube.com/watch?v=iS5a_LD0hvk&t=92s

Ссылки

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

[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 функций, ожидающих реализации:

[Trello](#)

Видео-урок на Дзен:

[JustPlay | Yandex Game Плагин | Реклама, Таблицы лидеров, Облачные сохранения, Защита от AdBlock и т.д.](#)



Импорт плагина

⚠ *Плагин поддерживается только для Unity 2021.3 версий и выше!*

💡 *Не перемещайте импортированные с плагином папки в проекте!*

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

Если Вы импортируете плагин в проект не первый раз, Вы можете убрать галочку напротив SavesYG скрипта по пути YandexGame/WorkingData, чтобы не слетели Ваши сохранения, если таковые имеются.

После импорта у Вас могут возникнуть ошибки со словами Newtonsoft, [Json.net](https://www.json.net), LanguageYG. Если такие ошибки есть, распакуйте пакет JsonDotNet. Он лежит в папке YandexGame/Addons.

Узнать версию плагина можно в файле Readme в папке YandexGame.




Начало работы

Обязательные пункты для функционирования плагина:

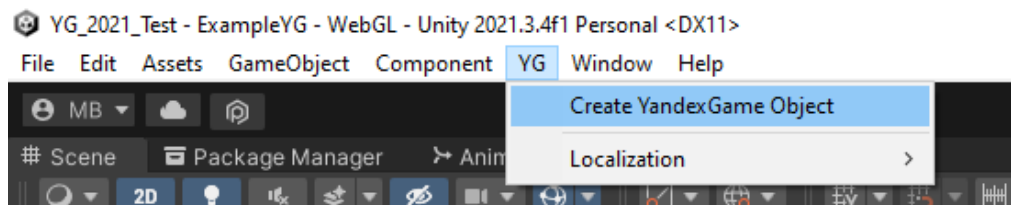
1. Добавить префаб YandexGame **на каждую сцену**.
2. Выбрать WebGL шаблон.

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

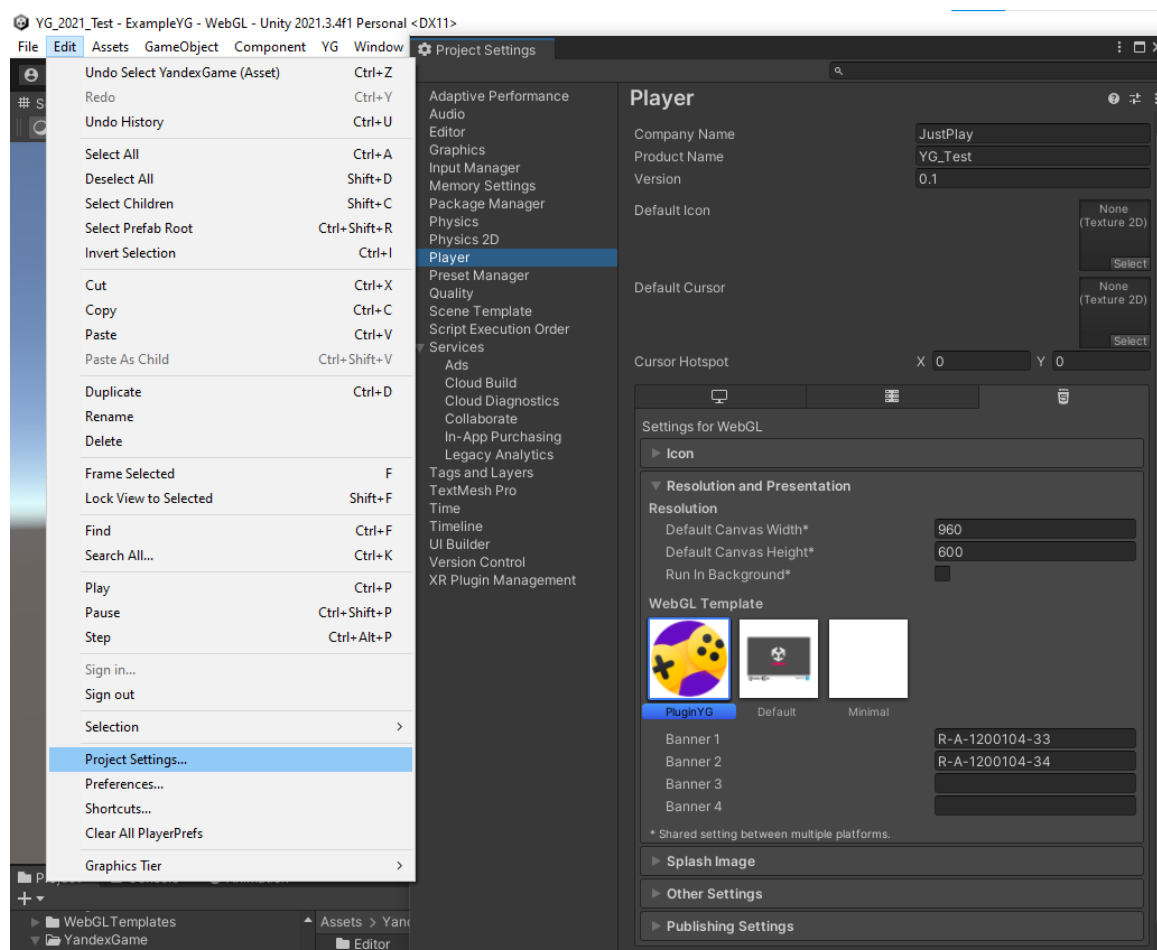
Добавить префаб **YandexGame** на сцену удобнее всего таким образом:



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

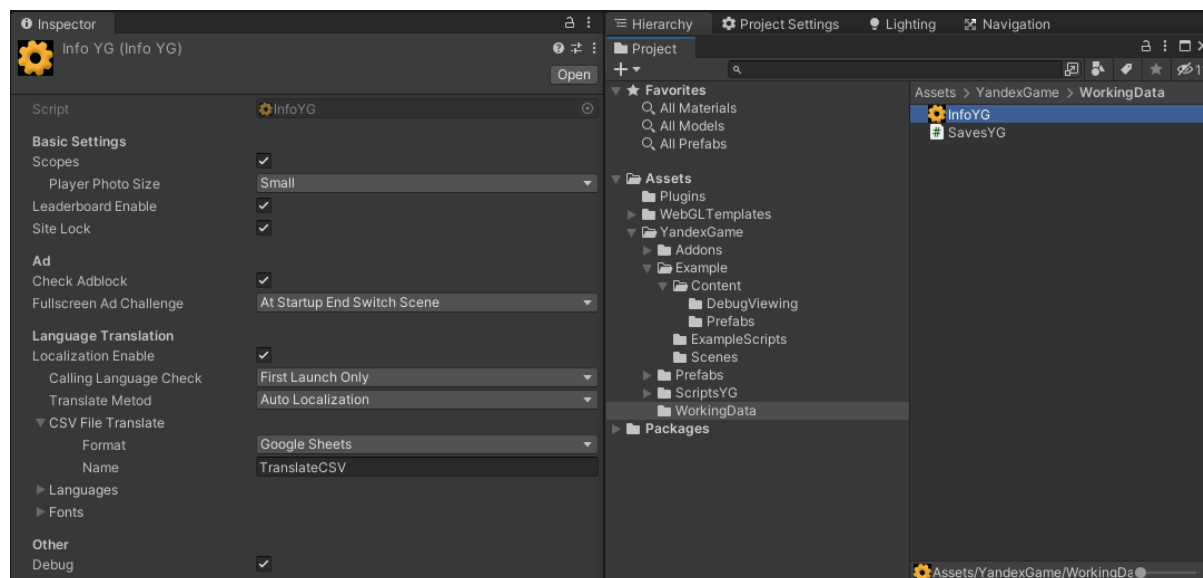
Выберите шаблон в **Project Settings** → **Player** → **Resolution and Presentation** → **WebGL Template**



Вы можете заменить картинку, которая показывается на загрузочном экране игры. Для этого замените изображение на своё по пути **WebGLTemplates** → **PluginYG** → **logo.png**. Но не меняйте имя файла! Или Вы можете заменить файл logo.png в готовом билде игры.

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 InfoYG.

Настройки графики от плагина

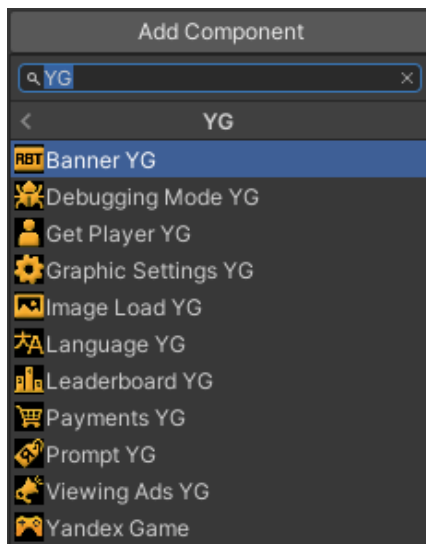
Готовые настройки графики **GraphicSettingsYG** с переводом с помощью плагина и возможные ошибки при использовании данного скрипта

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

Пространство имен YG

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

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



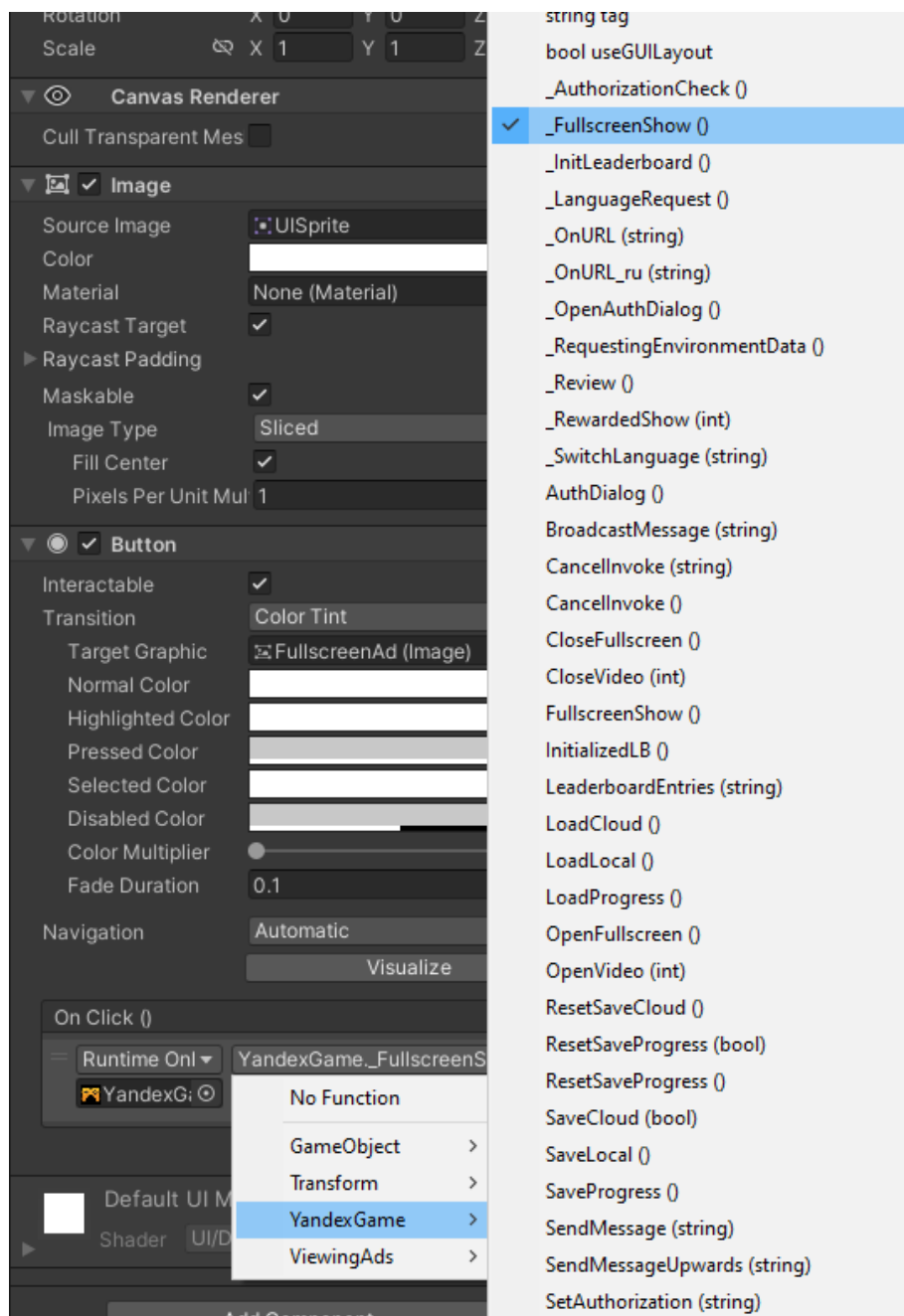
Также это значит: чтобы обратиться к скриптам плагина через Ваши скрипты, нужно подключать библиотеку YG таким образом: `using YG;`

Работа с компонентом Yandex Game

Все нужные Вам поля и методы есть в скрипте YandexGame.

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

Для Unity Events есть дублирующие методы с нижним 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.

- 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).

- On URL

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

- On URL_ru

Открыть ссылку только на русском домене.

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

- 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 Игр (запрос производится при старте игры, больше производить запрос не требуется).

- Reset Save Progress

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

- Review


Открыть окно для отзыва (окно будет открываться не всегда и только для авторизованных пользователей).

- Rewarded Show

Показать видео рекламу за вознаграждение.

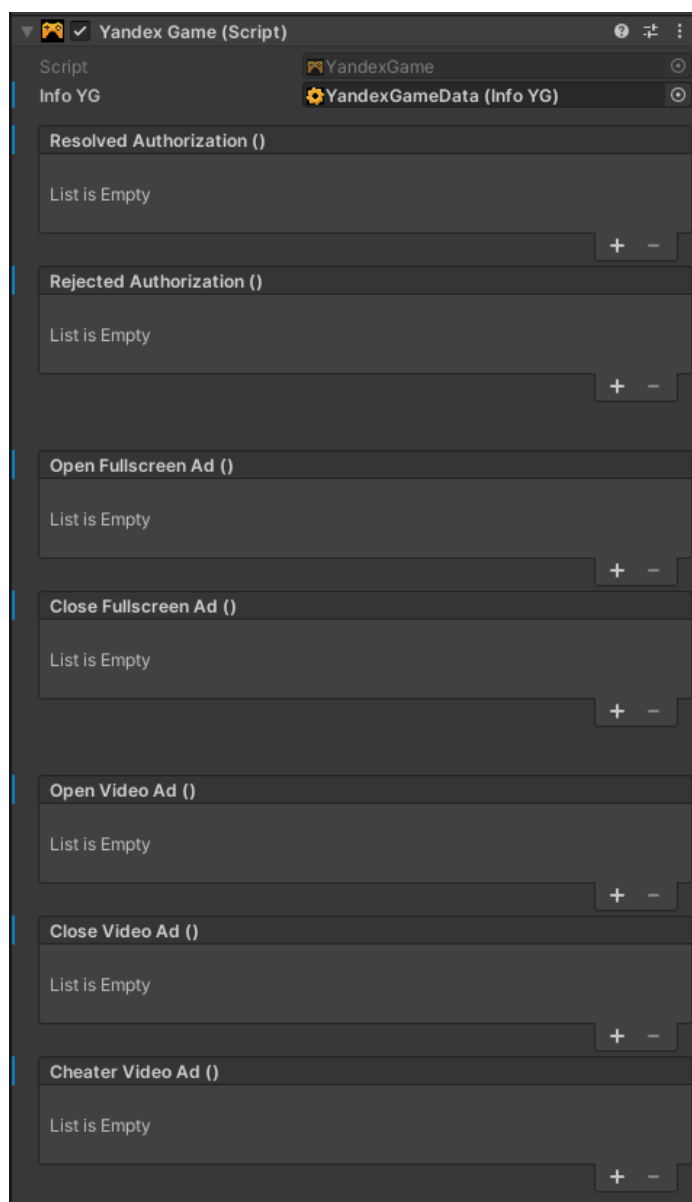
- Switch Language

Сменить язык. Укажите в перегрузку string значение языка. Например, “ru”, “en”, или “tr”. После смены языка произойдет сохранение игры.

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

Ивенты в компоненте Yandex Game

На скриншоте ниже показаны такие же Unity Events, как и у стандартных кнопок из Unity UI. Вы можете повешать какие-то объекты и выбрать метод, который будет запускаться при срабатывания какого-то из событий. Допустим, Вы можете выдать вознаграждение после просмотра видео рекламы.



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

Описание ивентов:

- Resolved Authorization
Вызывается при первом запуске игры после инициализации SDK, если игрок авторизован.
- Rejected Authorization
Вызывается при первом запуске игры после инициализации SDK, если игрок **не** авторизован.
- Open Fullscreen Ad
Вызывается при открытии полноэкранной рекламы.
- Close Fullscreen Ad
Вызывается при закрытии полноэкранной рекламы.
- Open Video Ad
Вызывается при открытии видео рекламы за вознаграждение.
- Close Video Ad
Вызывается при закрытии видео рекламы за вознаграждение.
- Cheater Video Ad
Вызывается если при открытии видео рекламы за вознаграждение обнаружится, что у пользователя установлено расширения блокировки рекламы или произошла какая-то ошибка. В таком случае, вознаграждение не будет выдано пользователю, а будет вызвано событие Cheater Video Ad. И при этом событии вы можете сделать показ уведомления пользователю о том, что произошла некая ошибка. Подробнее о Check AdBlock можно узнать в описании данной функции в разделе видео 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.



Инициализация

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

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

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

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

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

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

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

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

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

// Ваш метод, который будет запускаться в старте
public void GetData()
{
    // Получаем данные из плагина и делаем с ними что хотим
}
```



Полноэкранная реклама (Fullscreen Ad)

Активируйте рекламу в консоли разработчика. Для того, чтобы появилась возможность активации в разделе Реклама - нужно заполнить черновик.

💡 Реклама заработает не сразу! Через 3 - 24 часа.

💡 При старте игры показывается тестовая реклама! Не перепутайте её с настоящей. Тестовая реклама будет показана даже если настоящая еще не работает.

В InfoYG есть настройка **Fullscreen Ad Challenge**.

- По умолчанию выбрана опция **At Startup End Switch Scene**. Это означает, что полноэкранная реклама запустится еще когда игра только загружается, и будет запускаться при переключении сцен.
- Опция **Only At Startup** означает, что реклама запустится только при запуске игры.


💡 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()`.

Видео реклама за вознаграждение (Reward Video Ad)

Активируйте reward рекламу в консоли разработчика.

 Реклама заработает не сразу! Через 3 - 24 часа.

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`.

Метод вызова видео рекламы 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 `CloseVideoEvent`. 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 конструкцию из `if else`. 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 поэтому:

```
using YG;

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

// Отписываемся от события открытия рекламы в OnDisable
private void OnDisable() => YandexGame.CloseVideoEvent -= 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();
}
```

Продолжение кода:

```
// 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 if else внутри своего метода для вознаграждения.

Античит (Check AdBlock)

Это защита от накруток вознаграждения при использовании рекламы за вознаграждение. Она не даёт награду пользователям с AdBlock и другими аналогичными расширениями браузера., а также пользователям, которые закрывают рекламу раньше времени. Предотвращает открытие нескольких рекламных блоков и соответственно получения чрезмерной награды.

Если видео-реклама закроется раньше, чем требуется, то вместо события CloseVideoEvent вызовется событие CheaterVideoEvent. Вы можете подписаться на него, чтобы показать пользователю уведомление о том, что видео-реклама не была просмотрена, и произошла некая ошибка.

Для того, чтобы “Античит” работал в готовом билде игры, включите параметр **CheckAdBlock** в InfoYG.

Для тестирования функции Check AdBlock в среде разработки Unity, нужно ставить галочку наоборот:

При включённой галочке CheckAdBlock ****в InfoYG будет воспроизводиться симуляция просмотра реклама, и вызов события выдачи награды.

Чтобы протестировать срабатывание ошибки при открытии видео-рекламы в среде разработки Unity, нужно выключить CheckAdBlock в InfoYG.

Пауза игры при просмотре рекламы

По умолчанию на префабе YandexGame висит скрипт **ViewingAdsYG**. При просмотре полноэкранной или видео-рекламы данный скрипт ставит на паузу звук или временной масштаб в игре. Или и то и другое 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.

All — Ставить на паузу и звук и время при просмотре рекламы.

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).

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

AB Баннерная реклама

Создайте рекламный блок в личном [партнерском интерфейсе РСЯ](#):

1. Перейдите на вкладку **Реклама на сайтах** → **RTB-блоки**.
2. Нажмите кнопку **Добавить RTB-блок**.
3. Выберите **площадку** (игру).
4. Дайте запоминающееся **название**, ****т.к. Вам, возможно, понадобится в дальнейшем найти RTB-блок по названию.
5. **Уровень приватности** оставляем по умолчанию.
6. **Версия сайта нам не важна**, т.к. мы выберем адаптивный блок, в котором не важен выбор форматов. Можете выбрать десктоп или мобильная в зависимости от платформы, на которой будет Ваша игра.
7. В разделе **рекламные форматы** открываем **основной дизайн** (стандартный).
8. Слева выберите **Форматы** → **адаптивный**.
9. Справа **Основные** → **максимально количество объявлений**, ****желательно выбрать число 2 и более.
10. Можете настроить внешний вид баннера.
11. **Стратегия** - оставляем всё по умолчанию.
12. Нажимаем **создать** и переходим к **настройкам блока**.
13. **Копируем ID** блока следующим образом:

The screenshot shows the configuration page for an RTB block. At the top, there are buttons: "Найти в Яндексе", "Копировать", and "В заметки". The title of the block is "RTB-блок R-A-1200104-33 - Тест1". Below the title, there are tabs: "Общие", "Рекламные форматы", "Стратегия", "География", "Тематики", and "Бренды". The "Общие" tab is selected. The form contains the following fields:

- Площадка ***: game.yandex.ru, Page ID: (1200104)
- Название ***: Тест1
- Версия сайта**: Десктопная
- Уровень приватности**: Полностью

Below the "Уровень приватности" field, there is a note: "Как передавать информацию о блоке сторонним рекламным системам". At the bottom of the form, there are two buttons: "Сохранить" (yellow) and "Отменить" (grey).

💡 RTB-блоки могут активироваться долго. От 3-ёх часов. Это значит, что баннеры в черновике Вашей игры появятся не сразу.

💡 Общая площадь дополнительных рекламных блоков при постоянном отображении (в меню или в самом процессе игры) должна быть не более 20% размера игрового поля. Размер игрового поля соответствует непосредственно занимаемому игрой пространству на экране, рассчитывается без учета дополнительных неигровых областей (фона) и черного контура.

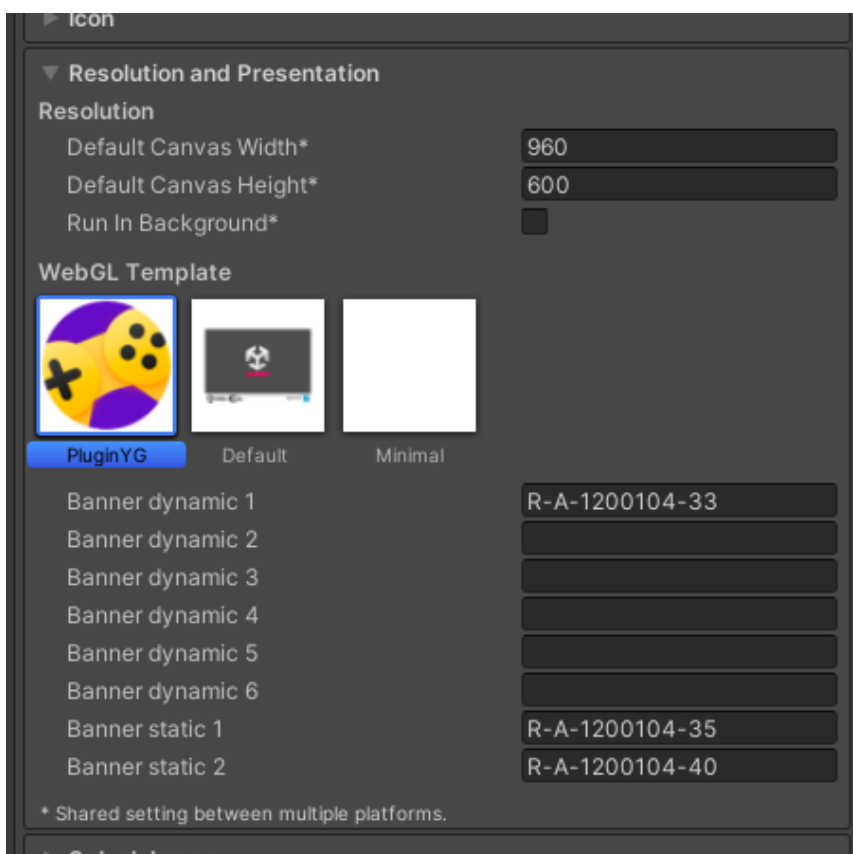
💡 Если широкий баннер снизу/сверху, то от краев экрана должно быть не менее 10%.

💡 Если имеется два широких баннер снизу и сверху, то их нужно делать симметричными.

💡 RTB-блоки Яндекса могут не всегда менять свою форму! Они могут долго не менять размер. Это значит, что, допустим, при переключении сцены один и тот же баннер, имеющий разную форму, не всегда будет её менять или полностью заполнять блок так, как вы настраивали в Unity. Если Вам нельзя допускать таких несоответствий, можете сделать несколько отдельных баннеров для разных ситуаций.

Настройки WebGL шаблона для баннеров

Вставьте скопированный ID баннера в соответствующее поле шаблона PluginYG следующим образом:



Вы можете сделать всего шесть динамических и два статических баннера. Если у Вас, допустим, всего один баннер, то заполните поле только для первого баннера, а остальные поля должны быть пустые.

- Возможные проблемы с шаблоном

Если у Вас не отображаются поля для ID блоков, потыкайте разные шаблоны и вернитесь к шаблону PluginYG. Важно отметить, что после переключений шаблонов поля ID блоков пропадают, и их приходится записывать заново. После обновления проекта до новой версии Unity или обновления плагина, также советую убедиться в том, что поля не пропали.

Типы баннеров

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

Статические баннеры — это баннеры, которые будут всегда зафиксированы в одном положении. Их размеры адаптируются под разные экраны. Статические баннеры нужны, по большей части, чтобы отобразиться только при загрузке игры. Их можно использовать и в процессе игры, но тогда их нельзя будет контролировать как динамические. Не получится поменять размер, положение, и нельзя будет их скрыть, они будут присутствовать в игре постоянно.

Создание динамического баннера

Чтобы создать динамический баннер, нужно указать ID баннера в шаблоне и перетащить готовый префаб с соответствующим номером на сцену. Префабы баннеров находятся по пути **YandexGame → Prefabs → Banners**. Если у Вас всего, допустим, один баннер, используйте префаб под номером один. Если, допустим, два баннера, то вы должны в шаблоне заполнить поле для второго баннера и использовать на сцене баннеры только под номерами 1 и 2. Если на сцене будет баннер под номером 4, а в шаблоне поле для четвертого баннера будет пустым, то это вызовет ошибку.

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

💡 *Canvas настраивать не нужно!*

Перетаскивать объект Render Block в другой Canvas не нужно!

Не делайте баннеры синглтоном!

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

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

Баннеры могут иногда быть меньше, чем вы создавали. Или наоборот выходить за рамки. Особенно, если блок будет меньше 50-ти пикселей по высоте, то баннер может выйти вниз блока. Обычно всё это не критично. Минимальный размер баннера 320x50 пикселей.

Параметры компонента **BannerYG**


RTB_Number — Номер баннера, ID которого вы указывали в шаблоне.

Device — Девайс, на котором будет отображаться баннер.

Desktop And Mobile - Отображение баннера на всех устройствах.

Only Desktop - Отображение баннера только на компьютере.

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

 *Вы можете расположить на сцене два баннера с одним и тем же номером для разных устройств. Чтобы, допустим, сделать разный размер блока или его якоря на разных устройствах для одного и того же баннера.*

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

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

Активация и деактивация динамических баннеров

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

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

Создание статических баннеров

Для создания статических баннеров нужно только лишь вписать ID в шаблоне.

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

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

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

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



Данные игрока

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`
тип `boolean`
 - `true` — Если SDK загрузился;
 - `false` — Если SDK не загрузился.
- `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.initializedLB`
тип `boolean`
 - `true` — Если лидерборды инициализированы;
 - `false` — Если лидерборды не инициализированы.
- `YandexGame.playerPhoto`
тип `stringify`
 - Ссылка **изображения аватарки игрока**
- `YandexGame.photoSize`
тип `stringify`
 - **Размер** подкачанного **изображения** пользователя. Возвращает значение параметра **Player Photo Size**, которое вы выбираете в InfoYG.
- `YandexGame.nowFullAd`
тип `boolean`
 - `true` — Полноэкранная реклама **открыта** в данный момент.

- `false` — Полноэкранная реклама **закрыта** в данный момент.
 - `YandexGame.nowVideoAd`
тип `boolean`
 - `true` — Видео реклама за вознаграждение **открыта** в данный момент.
 - `false` — Видео реклама за вознаграждение **закрыта** в данный момент.
-
- `YandexGame.EnvironmentData.language`
тип `stringify`
 - **Язык.** Поддерживаемые языки вы можете посмотреть в [официальной документации Яндекс Игр](#).
 - `YandexGame.EnvironmentData.domain`
тип `stringify`
 - **Домен.** Поддерживаемые домены вы можете посмотреть в [официальной документации Яндекс Игр](#).
 - `YandexGame.EnvironmentData.deviceType`
тип `stringify`
Устройство пользователя. Возвращает одно из значений:
 - `"desktop"` (компьютер)
 - `"mobile"` (мобильное устройство)
 - `"tablet"` (планшет)
 - `"tv"` (телевизор)
 - `YandexGame.EnvironmentData.isMobile`
тип `boolean`
 - `true` — мобильное устройство;
 - `false` — иное устройство.
 - `YandexGame.EnvironmentData.isDesktop`
тип `boolean`
 - `true` — компьютер;
 - `false` — иное устройство.
 - `YandexGame.EnvironmentData.isTablet`
тип `boolean`
 - `true` — планшет;
 - `false` — иное устройство.
 - `YandexGame.EnvironmentData.isTV`
тип `boolean`
 - `true` — телевизор;
 - `false` — иное устройство.
 - `YandexGame.EnvironmentData.appID`
тип `stringify`
 - **ID игры**
 - `YandexGame.EnvironmentData.browserLang`

тип stringify

- **Язык браузера**

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

- `YandexGame.EnvironmentData.payload`

тип stringify

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

- `YandexGame.EnvironmentData.promptCanShow`

тип boolean

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

-
- `YandexGame.savesData.isFirstSession`

тип boolean

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

- `YandexGame.savesData.language`

тип stringify

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

- `YandexGame.savesData.promptDone`

тип boolean

- Становится `true`, когда ярлык игры установлен.

-
- `YandexGame.PaymentsData.id[]`

тип stringify

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

- `YandexGame.PaymentsData.title[]`

тип boolean

- **Название** товара.

- `YandexGame.PaymentsData.description[]`

тип stringify

- **Описание** товара.

- `YandexGame.PaymentsData.imageURIimageURI[]`

тип stringify

- **URL изображения** товара.

- `YandexGame.PaymentsData.priceValue[]`

тип boolean

- **Стоимость** товара.

- `YandexGame.PaymentsData.purchased[]`

тип integer

Куплен товар или нет / Количество купленного товара. Допустим:

- 1 — Товар был куплен один раз; 0 — Товар **не** куплен.



Облачные сохранения

- Сохранения у Яндекс Игр есть как облачные, так и локальные. Яндекс сам сохранит данные в зависимости от авторизации.
- Сохранения работают в среде разработки Unity. Вне яндекса, данные сохраняются отдельно в файл. Так можно спокойно тестировать игру во время разработки.
- После обновления игры в Яндексе сохранения у игроков не слетят, чем не может похвастаться любой другой метод сохранения, который не привязан к Яндексу.
- Пользоваться сохранениями в плагине просто. Вы создаёте свой список данных для сохранения в отдельном классе. 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 будут равны дефолтным значениям.

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

Загрузка сохранений

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

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

Чтобы получить сохранения игры в старте, делайте это после инициализации в уже рассмотренном нами событии `GetDataEvent`.

Сохранение

Для сохранения 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 запускаем Ваш метод для загрузки
        GetLoad();

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

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

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

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

Сброс сохранений

Файл сохранений для тестирования игры сохраняется в папке `WorkingData`. Вы можете сбросить сохранения, для этого удалите файл `saveug.ug` в папке `WorkingData`.

Есть метод для сброса сохранений `YandexGame.ResetSaveProgress()`. Он не удаляет файл `saveug.ug`, но полностью устанавливает данные сохранений. Также данный метод сбрасывает сохранения и в готовом билде игры.



Лидерборды

Создание таблицы в консоли разработчика


Перейдите в раздел **Соревновательные таблицы** 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 `NewLeaderboardScores(string, int);`

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

При этом у Вас уже будет существовать соревновательная таблица, которая будет отображаться на странице Вашей игры. Но Вы также можете отобразить таблицы и в самой игре.

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

Создание таблицы в Unity

Отображением таблицы занимается скрипт **LeaderboardYG**. Все его опции имеют всплывающие подсказки.

Для **простого** описания рекордов достаточно просто указать в компоненте LeaderboardYG ссылку на компонент UI.Text в опции **Entries Text**. И скрипт сам будет записывать в него данные таблицы. Также можете взять префаб с настроенной таблицей по пути **YandexGame** → **Prefabs** → **Leaderboards** и изменить интерфейс таблицы.

Есть продвинутый режим (**Advanced**). Для него не нужно заполнять опцию Entries Text. Он ищет определенные дочерние объекты для заполнения в них данных. Он сделает все отдельными элементами. Рейтинговая позиция игрока, его аватарка, ник и рекорд будут в отдельных UI элементах. Поэтому просто возьмите настроенный префаб таблицы.

У скрипта LeaderboardYG есть следующие публичные методы:

- `UpdateLB()`

Обновление таблицы.

- `NewScore(int score)`

Запись нового рекорда.



Локализация

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

- `YandexGame.EnvironmentData.language`

тип `stringify`

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

Инструменты локализации плагина

Для начала работы зайдите в InfoYG и выберите языки, на которые будете переводить игру в опции **Languages**.

Параметр **Calling Language Check**.

- **First Launch Only** — язык будет проверяться только при первом запуске игры, и сохраняться. Последующие запуски игры будут с языком, который был выбран 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.

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

-
- **Auto Localization** даст возможность автоматически переводить текст через API Google Translate.
 - Проблемы авто-локализации

У Google Translate есть ограничения. Не получится переводить много текста в краткий промежуток времени. Если вы хотите пользоваться авто-локализацией, то я рекомендую переводить текста по маленьку. При создании игрового интерфейса, к примеру, как раз будут временные промежутки между переводами.

- Инструмент для перевода всей сцены

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

💡 *Этот и все инструменты, которые ищут объекты на сцене - игнорируют деактивированные объекты!*

-
- С **Manual** методом просто убираются излишние опции в компоненте. Это значит, Вы все будете переводить вручную.
-

- 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.
- 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.*

Шрифты

В InfoYG есть массивы шрифтов для каждого языка. Если вы перетяните 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 LanguagesYG вы укажете поле **Font Number** равное одному, то для этого текста будет ставиться шрифт из первого элемента массива.

Также, в компоненте LanguagesYG есть поле **Unicue Font**. Это означает **Уникальный Шрифт**. Если вы перетащите туда какой-либо шрифт, то данный текст всегда будет грузиться с указанным шрифтом.

Для того, чтобы кнопка **“Заменить шрифт на стандартный”** заработала, нужно в InfoYG указать дефолтный шрифт.

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

💡 *На данный момент не поддерживается TextMeshPro.*



Внутриигровые покупки

Для начала прочтите **документацию Яндекса по внутриигровым покупкам** до раздела “Условия подключения” включительно.

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

Скрипт PaymentsYG

Самый простой способ — использовать готовый скрипт PaymentsYG. Он автоматически отображает каталог всех покупок с выводом всей информации о товарах или отображает один товар по ID продукта. В папке **YandexGame** → **Prefabs** есть префабы **Payments Catalog** и **One Purchase**. Первый для отображения всего каталога товаров. Второй для одной покупки. Скрипт PaymentsYG автоматически обновляет и выводит всю информацию, и в нём уже есть кнопка для покупки.

Активация процесса покупки

Для совершения покупки используйте метод `YandexGame.BuyPayments(id)`. В перегрузку `id` указываете идентификатор товара, который Вы записывали при создании товара в консоли разработчика. После вызова данного метода откроется диалоговое окно для совершения покупки.

При совершении успешной покупки сработает событие `YandexGame.PurchaseSuccessEvent`. На него нужно подписаться для вручения товара пользователю или для вывода сообщения об успешной оплате.

В случае, если оплата не будет произведена, сработает событие `YandexGame.PurchaseFailedEvent`. Подпишитесь на него, чтобы выводить уведомление о том, что покупка не была произведена. 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.

Обновление информации о товарах

Вы можете обновить информацию о товарах с помощью метода `YandexGame.GetPayments()`. Данный метод будет брать свежие данные из SDK Яндекса.

После получения данных вызовется событие `YandexGame.GetPaymentsEvent`.

Метод `GetPayments` и соответственно вызов события `GetPaymentsEvent` происходят автоматически после инициализации SDK. Поэтому после запуска игры все данные покупок уже будут загружены.

После совершения покупки данные о приобретении товара обновляются (`YandexGame.PaymentsData.purchased[x] = true`). И срабатывает вышеописанное событие `GetPaymentsEvent`.

Получение информации о товарах

Существует следующая информация о товарах:

- `id`
тип `stringify`
 - **Идентификатор товара**, который Вы записывали при создании товара в консоли разработчика.
- `title`
тип `boolean`
 - **Название** товара.
- `description`
тип `stringify`
 - **Описание** товара.
- `imageURIimageURI`
тип `stringify`
 - **URL изображения** товара.
- `priceValue`
тип `boolean`
 - **Стоимость** товара.
- `purchased`
тип `integer`
Куплен товар или нет / Количество купленного товара.
Допустим:
 - 1 — Товар был куплен один раз;
 - 2 — Товар покупался два раза;
 - 0 — Товар **не** куплен.

Получать информацию можно сразу о всех товарах в массивах, или можно получать информацию об одном конкретном товаре по идентификатору.

Получение массивов данных всех товаров

Есть класс `YandexGame.PaymentsData`. В нём содержатся массивы данных.

Массивы хорошо подходят для создания каталога товаров. Допустим, вы хотите перечислить идентификаторы всех товаров в цикле. Это можно сделать следующим образом:

```
for (int i = 0; i < YandexGame.PaymentsData.id.length; i++)
{
    YandexGame.PaymentsData.id[i]
}
```

Получение данных одного товара по идентификатору

Метод `PurchaseByID(id)` возвращает класс `Purchase`, в котором содержатся все данные о товаре. В перегрузку `id` указываете идентификатор товара, данные которого хотите получить.

Создаём новый класс `Purchase` и сразу заполняем его значениями определенного товара таким образом: `Purchase purchase = PurchaseByID(id);`

Теперь у нас есть поле `purchase`, в котором содержатся все данные определенного товара.

И теперь мы можем брать данные таким образом:

```
// Допустим, мы хотим получить информацию о товаре под id "rifle"
Purchase purchaseRifle = PurchaseByID("rifle");
string idRifle = purchaseRifle.id;
string titleRifle = purchaseRifle.title;
string numArrayRifle = purchaseRifle.numArray;
...
```

В классе `Purchase` содержится поле `numArray` типа `int`. Оно содержит номер элемента массива. То есть, если товар, допустим, с `id` “rifle” в массиве `YandexGame.PaymentsData` будет элементом на третьей позиции, то `numArray` будет равен 3.

Еще пара примеров, как можно использовать всё в связке:

```
int rifleNum = PurchaseByID("rifle").numArray;
YandexGame.PaymentsData.purchased[rifleNum.numArray] += 1;

string[] descriptions = new string[YandexGame.PaymentsData.id.length];
descriptions[rifleNum] = YandexGame.PaymentsData.description[rifleNum];
```

Глубинное связывание (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, перейдя 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.



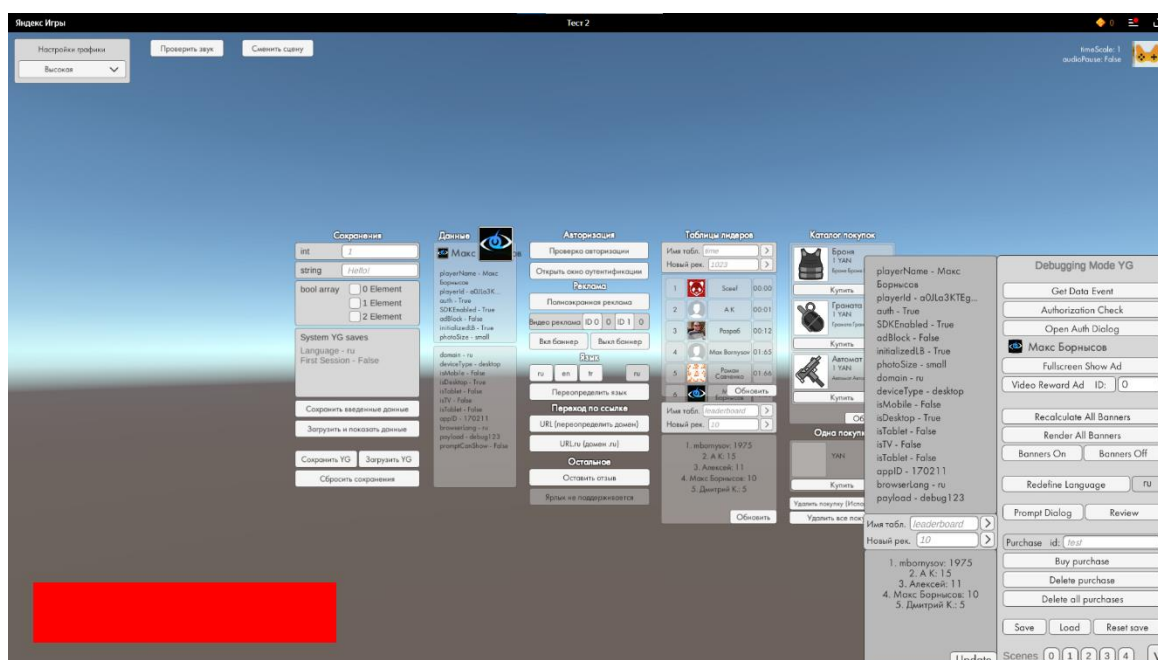
Инструмент для отладки

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.