УДК:004.023

**Интеграция Java плагина в среду разработки Unity с помощью AndroidJavaObject и AndroidJavaClass**

**Горбунов Никита Алексеевич**

Пензенский Государственный Университет, Пенза, Россия.

nik.gorbunov.02022002@mail.ru

**Аннотация.** В целях освоения новых инструментов разработки под операционную систему Android на мультиплатформенном движке Unity и дальнейшего их правильного применения, автором были проанализированы и протестированы встроенные возможности интеграции языка программирования Java в среду разработки проекта. При написании Android приложений довольно часто приходиться взаимодействовать с Android API. Большинство базовых используемых методов Android API уже встроено в инструментарий движка Unity, но в редких случаях его недостаточно, поэтому разработчики встроили возможность интеграции «JAR» и «AAR» плагинов с помощью классов «AndroidJavaObject» и «AndroidJavaClass», которые дают возможность вызова методов из скомпилированного файла. Для написания правильно работающего плагина нужно придерживаться определенного паттерна проектирования, изменить и настроить Android Manifest при компиляции JAR или AAR файла и правильно использовать классы из метаданных Unity.

**Ключевые слова:** Разработка, Unity, Android Application on Unity, Android Plugin for Unity, Java in Unity, JAR & AAR, Android API in Unity, Добавление Java плагина в Unity, Unity Activity.

**Для цитирования:** Горбунов Н.А. «Интеграция Java плагина в среду разработки Unity с помощью AndroidJavaObject и AndroidJavaClass» 2023.

**Integration of Java plugin into the Unity development environment using AndroidJavaObject and AndroidJavaClass**

**Gorbunov Nikita Alexeyevich**

Penza State University, Penza, Russia.

nik.gorbunov.02022002@mail.ru

Abstract. In order to master new development tools for the Android operating system on the multi-platform engine Unity and further apply them correctly, the author analyzed and tested the built-in possibilities of integrating the Java programming language into the project development environment. When developing Android applications, it is often necessary to interact with the Android API. Most of the basic methods used in the Android API are already integrated into the Unity engine toolkit, but in rare cases, it is not enough. Therefore, developers have incorporated the ability to integrate "JAR" and "AAR" plugins using the "AndroidJavaObject" and "AndroidJavaClass" classes, which allow calling methods from a compiled file. To write a properly functioning plugin, it is necessary to adhere to a certain design pattern, modify and configure the Android Manifest when compiling the JAR or AAR file, and correctly use classes from Unity metadata.

**Keywords:** Development, Unity, Android Application on Unity, Android Plugin for Unity, Java in Unity, JAR & AAR, Android API in Unity, Integration Java plugin into the Unity, Unity Activity.

**For citation:** Gorbunov N.A. «Integration of Java plugin into the Unity development environment using AndroidJavaObject and AndroidJavaClass» 2023.

Каждый разработчик сталкивался с задачей взаимодействия приложения и операционной системы, под которую оно собирается. Unity – кроссплатформенная среда разработки, с помощью которой можно создавать игры и приложения под разные операционные системы, но большая часть разработанных приложений на этом движке реализованы именно под ОС Android. Сам же Unity разработан на языке C#, поэтому многие неопытные разработчики могут задаваться вопросом – «Как же взаимодействовать с системными вызовами?». Большинство основных системных вызовов уже реализованы «под капотом» игрового движка, например: создание/удаление/запись файлов, запрос на разрешения приложения, геолокация, интернет-запросы, и так далее. Для написания простого приложения, которое должно будет сохранять что-либо или отправлять запросы на сервер, встроенного инструментария хватит, но если приложение должно работать с Bluetooth устройствами, сервисами, ИК-портом (если присутствует в телефоне), то встроенных методов Unity недостаточно. Разработчики движка спрогнозировали нехватку функционала и дали возможность интеграции Java в приложение. Код, написанный на Java, можно импортировать только в уже скомпилированном файле с расширением AAR или JAR, а взаимодействие с плагинами будет осуществляться через классы «AndroidJavaClass» и «AndroidJavaObject». «AndroidJavaClass» — это представление Unity универсального экземпляра «java.lang.Class»[[1]](#footnote-1), а «AndroidJavaObject» – «java.lang.Object»[[2]](#footnote-2). Его можно использовать как бестиповый интерфейс для экземпляра любого класса Java внутри C# скрипта.

При написании кода на языке Java открывается максимальный спектр возможностей взаимодействия с операционной системой Android. Но нужно придерживаться определенных правил, если написанный код, используется через собранную библиотеку, к которой идёт обращение из основного приложения.

Первым делом нужно разобраться с тем, как компилировать код на Java. При сборке библиотеки под OC Windows, и дальнейшим взаимодействии с ней, на выходе получается файл с расширением DLL. При компиляции под ОС Android возможны два вида файлов:

* JAR – Java Archive. Содержит файлы классов, JAVA-ресурсы, зависимые библиотеки, и другие необходимые для приложения файлы. JAR может использоваться как в Android-приложении, так и в приложении на чистой Java.
* AAR – Android Archive. Так же, как и JAR является zip-архивом, но может использоваться только в Android-приложении. AAR содержит скомпилированный исходный код в файле classes.jar, Android-ресурсы, AndroidManifest, и другие файлы, входящие в состав APK.[[3]](#footnote-3)

При компиляции библиотеки для интеграции в Unity-приложение подходят оба архива, но нужно учитывать особенности. В документации Unity, разработчики советуют импортировать AAR плагины из-за большей совместимости с движком. Некоторые API могут требовать определенные разрешения, которые нужно прописывать в «AndroidManifest.xml». Сторонние библиотеки добавляются в «build.Gradle». Если использовать AAR файл, то при компиляции не обязательно указывать все нужные ресурсы в плагине, достаточно будет добавить их в манифест вашего основного приложения, а движок сам сольёт всё в один Gradle файл. Также Unity не требует добавлять «3rd-party» в определенную папку, их можно будет расположить в наиболее удобной директории, находящейся в иерархии приложения. JAR файл, в свою очередь, обязывает прописывать все разрешения в манифесте при сборке, а библиотеки обязательно должны лежать в папке «Assets/Plugins/Android».

Прежде чем приступать к написанию кода, нужно определить, какие задачи будет решать плагин, написанный на Java. Если необходимо управлять уведомлениями, вызывать «AlertDialog» или «Pop-up» окна, то нужно будет обязательно взаимодействовать с Android Activity основного приложения. Android Activity – ключевой компонент для создания визуализации интерфейса, форма приложения, с которой взаимодействует пользователь. Активность основного приложения потребуется передать, как параметр в методе инициализации плагина из скрипта C#. Далее идет взаимодействие с базовыми функциями приложения, например, если задача требует доступ к файловой системе, запрос на разрешения, доступ к WIFI, Bluetooth, USB, то необходимо будет обращаться к контексту активности. Context – интерфейс для глобальной информации о среде приложения. Это абстрактный класс, реализация которого обеспечивается системой Android. Он предоставляет доступ к ресурсам и классам, специфичным для приложения, а также к вызовам для операций уровня приложения, таких как запуск действий, широковещательная рассылка, получение намерений (Intent), и так далее. Помимо всего вышеперечисленного, для опытных разработчиков есть возможность переопределения активности основного приложения. «UnityPlayerActivity» приложения отвечает за базовое взаимодействие между операционной системой Android и приложением. Предоставляется возможность использовать плагины, чтобы создать свою собственную активность, которая расширяет и переопределяет стандартную «UnityPlayerActivity», например, для запуска каких-либо сервисов, вместе с блокировкой экрана телефона.

После конкретизации поставленной задачи, нужно определиться с архитектурой написания плагина.

* Если важна точность исполнения всех вызовов плагина, сохранение состояний объектов и возврат результата, например, постоянное обращение к файловой системе или отправка данных по Bluetooth с сохранением подключения к устройству, то стоит использовать паттерн проектирования Singleton.
* Если сохранение состояния не настолько важно, и нужно лишь иногда обращаться к Android API, например, вывести «Pop-up» о какой-либо ошибке или вести «LogCat» с пользовательскими тегами, то достаточно будет реализовать всё в статических методах.
* Если нужно единожды вызвать API, который не отвечает за важные решения операционной системы, то пользовательский Java-плагин создавать не обязательно. Можно объявить и инициализировать какой-либо класс Android API в классе «AndroidJavaClass», вызвать нужный метод, после чего Garbage collector очистит выделенную память. Этот способ не оптимизированный, но быстрый для реализации.

1. Примечание – выбор архитектуры зависит от API, которые потребуется использовать.

Наконец, можно приступить к интеграции плагина в C# скрипты. Для начала нужно выбрать, какого типа будет переменная, которая хранит в себе библиотеку. Если было решено применить паттерн Singleton, то необходимо использовать «AndroidJavaObject», так как потребуется хранение экземпляра класса и вызов его методов. Иначе, если достаточно будет вызывать только статические методы класса, предполагается объявление переменной типа «AndroidJavaClass».

2. Примечание – во втором случае можно использовать «AndroidJavaObject», но логически данное решение будет неверным.

Далее, если присутствует метод инициализации в плагине, требуется его вызвать, передав в аргументах активность. Чтобы это сделать используется конструктор «AndroidJavaClass» / «AndroidJavaObject», который принимает строку с названием пространства имён в аргументах.

1. Сохраняется переменная с Unity Activity (Если имя активности не переопределено).
2. Объявляется переменная, в которой будет плагин.
3. Инициализируется плагин, передавая в параметры активность приложения.
4. Возвращается экземпляр класса библиотеки в отдельную переменную.

Пример кода:

**private** AndroidJavaClass unityClass;

**private** AndroidJavaObject unityActivity;

**private** AndroidJavaClass customClass;

**private** AndroidJavaObject instance;

**private** **const** **string** PackageName = “com.myPlugin.plugin.Manager”;

**private** **const** **string** UnityDefaultJavaClassName = “com.unity3d.player.UnityPlayer”;

**public** **void** **Init**()

{

AndroidJNI.AttachCurrentThread();

unityClass = **new** AndroidJavaClass(UnityDefaultJavaClassName);

unityActivity = unityClass.GetStatic<AndroidJavaObject>(“currentActivity”);

customClass = **new** AndroidJavaClass(PackageName);

customClass.CallStatic(“InitInstance”, unityActivity);

instance = customClass.GetStatic<AndroidJavaObject>(“instance”);

}

3. Примечание – Метод «AndroidJNI.AttachCurrentThread()» нужно включать при каждом обращении к плагину, если скрипт исполняется не из основного потока приложения.

Все методы библиотеки будут вызываться с помощью «Call» и «CallStatic», для обычных и static методов соответственно, которые принимают 2 параметра:

1. Имя метода из плагина (типа string)
2. Массив базовых классов object

При передаче параметров, для наибольшей производительности, следует использовать простые типы, такие как: int, float, string.

В результате проделанной работы, на выходе получается рабочий Java плагин для интеграции в среду разработки Unity. Для его реализации понадобилось:

* Настроить сборку Java кода в AAR файл.
* Проанализировать задачи, которые должна реализовывать библиотека.
* Исходя из задач, построить верную структуру программы.
* Интегрировать плагин в среду проекта и инициализировать его в скриптах.
* Для вызовов методов плагина использовать «Call» и «CallStatic», передавая базовые типы, чтобы не нагружать приложение.

Список литературы

1. Unity Documentation: AndroidJavaClass // Электронный ресурс: <https://docs.unity3d.com/ScriptReference/AndroidJavaClass.html>
2. Unity Documetation: AndroidJavaObject // Электронный ресурс: <https://docs.unity3d.com/ScriptReference/AndroidJavaObject.html>
3. Interview Review: В чем разница между jar и aar? // Электронный ресурс: <https://itsobes.ru/AndroidSobes/v-chem-raznitsa-mezhdu-jar-i-aar/>

1. Unity Documentation: AndroidJavaClass // Электронный ресурс: <https://docs.unity3d.com/ScriptReference/AndroidJavaClass.html> [↑](#footnote-ref-1)
2. Unity Documetation: AndroidJavaObject // Электронный ресурс: <https://docs.unity3d.com/ScriptReference/AndroidJavaObject.html> [↑](#footnote-ref-2)
3. Interview Review: В чем разница между jar и aar? // Электронный ресурс: <https://itsobes.ru/AndroidSobes/v-chem-raznitsa-mezhdu-jar-i-aar/> [↑](#footnote-ref-3)