Quick Actions IOS Plugin for Unity3D

1. Введение

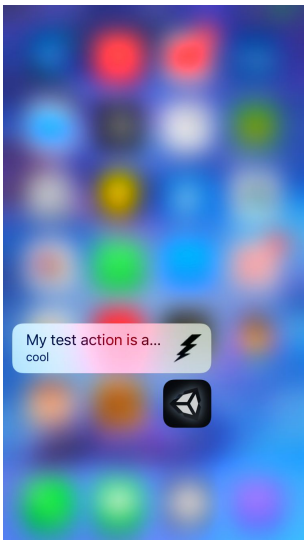
Данный плагин дает возможность работать с Home Screen Quick Actions (доступно начиная с iOS 9). За дополнительной информацией по Home Screen Quick Actions смотри официальную документацию Apple. Quick Actions можно разделить на 2 типа - Статические и Динамические. Главное различие между ними заключается в том что Статические доступны пользователю сразу после установки приложения, а Динамические будут доступны не раньше, чем после первого запуска приложения. Данный плагин поддерживает только Динамические Quick Actions.

Статические Quick Actions могут быть легко добавлены в приложения с помощью модифицирования файла Info.plist xCode проекта. Это можно сделать либо вручную, либо написать для этого скрипт пост-процесса. Если вас интересую статические Quick Actions обратите внимание на этот туториал. Этот туториал не имеет отношения к Unity3D, но даст вам понимание как интегрировать Статические Quick Actions.

Динамические Quick Actions полностью поддерживаются этим плагином, включая использование кастомных иконок или встроенных в систему. Полный список встроенных иконок для Quick Actions можно посмотреть в Apple's Human Interface Guidelines в разделе “Home Screen Quick Actions Icons”.

К несчастью из-за специфики работы Quick Actions их работу нельзять протестировать в редакторе Unity3D, по этому вам придется собирать билд вашей игры и запускать его на устройстве, чтобы протестировать данный функционал. Кроме то, вам понадобится iOS устройство с поддержкой 3D Touch. Несмотря на то, что на устройстве без поддержки 3D Touch Quick Actions работать не будут, приложение будет работать нормально и не будет вызывать ошибок или исключений..

Если вы не знаете, как протестировать Quick Actions - вам нужно установить придложение на устройство, сделать в нем необходимые действия (чтобы нужные Quick Actions установились в приложении), свернуть ваше приложение и сильно нажать на иконку приложения. После этого вы сможете увидеть список Quick Actions доступных вам..
Это может выглядеть примерно так:



2.Интеграция

Сначал распакуйте содержимое unitypackage в ваш проект. Данный плагин использует 2 скрипта пост-процесса для настройки xCode проекта.

Первый пост-процесс называется **ReplaceDelegatePostProcess.cs**. его едиснтвенная цель- заменить скрипт *UnityAppController.mm* в xCode на его измененную версию, которая входит в состав данного плагина. Этот плагин содержит свою версию скрипта *UnityAppController.mm* потому, что реализация Quick Actions требует прослушивания нескольких ивентов делегата iOS приложения о жизненном цикле самого приложения:

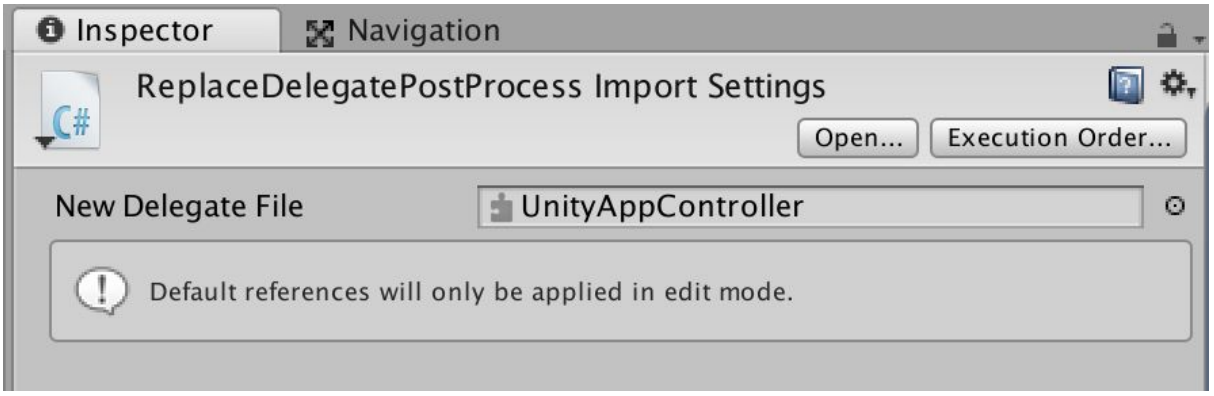
--performActionForShortcutItem - метод, который вызывается, когда приложение было запущено или его вернули из фонового режима с помощью Quick Action

--applicationDidEnterBackground - метод, который вызывается, когда приложение переходит в фоновый режим.

Замена скрипта UnityAppController не навредит фукнционалу Unity или любым другим функциями. Есть, конечно и другие пути подписывания на события приложения, например наследование от класса UnityAppController и реализация *IMPL_APP_CONTROLLER_SUBCLASS(ClassName)* в вашем кастомном классе. Но такой подход сработает только 1 раз в приложении.. По этому, если у вас в проекте будет 2 скрипта с подобной реализацией, то работать будет только 1 из них, а второй будет просто игнорироваться без каких-либо ошибок, связанных с *IMPL_APP_CONTROLLER_SUBCLASS(ClassName)*.

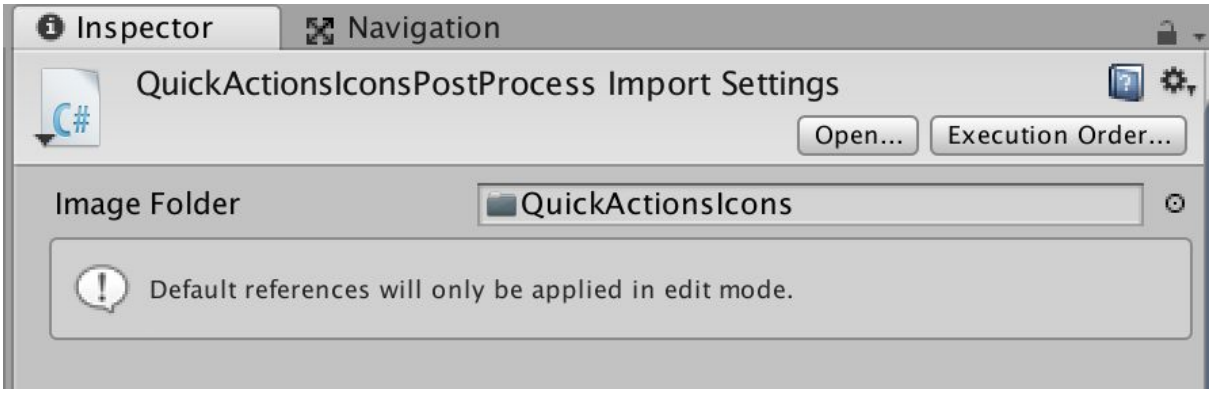
Чтобы настроить этот пост-процесс, выберите его в инспекторе и перетащите скрипт *UnityAppController.mm* (тот, который идет в данном плагине или

какой-либо свой) в поле инспектора пост-процесса - **New Delegate File**. Ваш пост-процесс в инспекторе должен выглядеть так:



Второй пост-процесс называется **QuickActionsIconsPostProcess.cs**. Его единственная цель - скопировать набор ваших кастомных иконок из Unity проекта в xCode проект и правильно настроить ссылки внутри проекта. Это делается потому, что если иконки попадут в билд, то будут перекодированы Unity в момент сборки билда в специальный бинарный формат, и вы не сможете использовать их внутри xCode проекта.

Чтобы настроить этот пост-процесс, поместите все нужные вам иконки внутрь одной папки в Unity проекте. Затем выберете в инспекторе **QuickActionsIconsPostProcess.cs** и перетащите саму папку с вашими иконками в единственное поле - **Image Folder.** Внутри пакета есть папка *QuickActionsIcons* которая содержит иконку молнии, которую можно использовать для теста. После настройки пост-процесс будет выглядеть вот так:



Это пост-процесс пройдется по всем .png файлам внутри заданной папки и скопирует их в xCode проект так, чтобы они попали в бандл приложения.

Кастомные иконки должны придерживаться следующих правил:

1. формат .png
2. разрешение 35*35 пикселей
3. квадратная картинка
4. только черно-белое изображение

Если какое-то из этих правил будет нарушено, при попытке установить “плохую” иконку к Quick Action iOS заменит ее на черный квадрат. Никаких ошибок при этом выведено не будет.

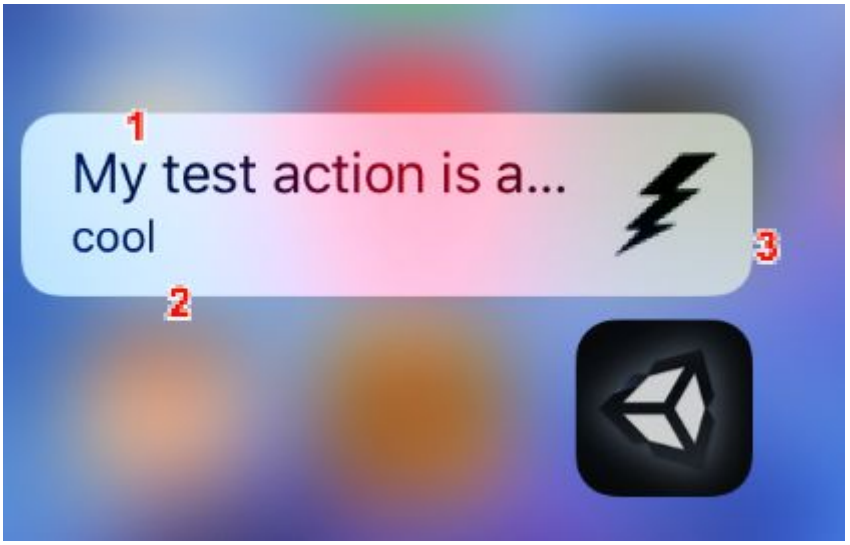
Теперь мы закончили настройку плагина.

3. API

Всю работу делает статический класс ***QuickActionsManager***. В нем есть 8 методов:

- **int GetNumberOfShortcuts()** **-** возвращает число Quick Actions доступных пользователю приложения в данный момент. Отображается не более 4.
- **QuickActionItem GetCurrentItem()** - когда приложение запускает или переходит из фонового режима при помощи Quick Actions, этот Action сохраняется в нативной части плагина. Вы можете вызывать этот метод внутри *OnApplicationPause(bool pauseStatus)* чтобы получить информацию о том, было ли приложение запущено с помощью Quick Action и какой именно Action это был. Эта информация сохраняется в виде экземпляра *QuickActionItem*. Quick Action уникально определяется полем *Type*. Хорошей практикой является устанавливать в это поле значчение в формате reverse DNS. Например: com.YouApp.YourAction. Если никакого Quick Action не было - вернется null.
- **QuickActionItem GetItemAtIndex(int i)** - все Quick Actions, доступные пользователю приложения в данный момент. Количество этих Quick Actions вы можете узнать вызвав метод *GetNumberOfShortcuts()*.
- **bool RemoveItem(string itemType)** - удалить Quick Action из списка доступных пользователю приложения по указанному *Type*. Возвращает статус выполнения (успех/неудача).
- **bool RemoveItem(QuickActionItem item)** - как и предыдущий метод, но принимает параметр типа QuickActionItem.
- **bool RemoveItemAtIndex(int i) -** удаляет Quick Action с заданным индексом (начиная с 0) из списка доступных пользователю.
- **void SetItem(string itemType, string localizedTitle, string localizedSubtitle, string customIconName, string builtinIconName) -** добавляет новый Quick Action в список доступных пользователю приложения. Первые 2 параметра обязательные, остальные опциональны и могут принимать значение null.
- **void SetItem(QuickActionItem item)** - тоже самое как и предыдущий метод, только с другим параметром.

Также в плагине есть класс ***QuickActionItem***, который служит как контейнер для хранения модели данны о Quick Action. Чтобы было понятней, давайте рассмотрим Quick Action подробней:



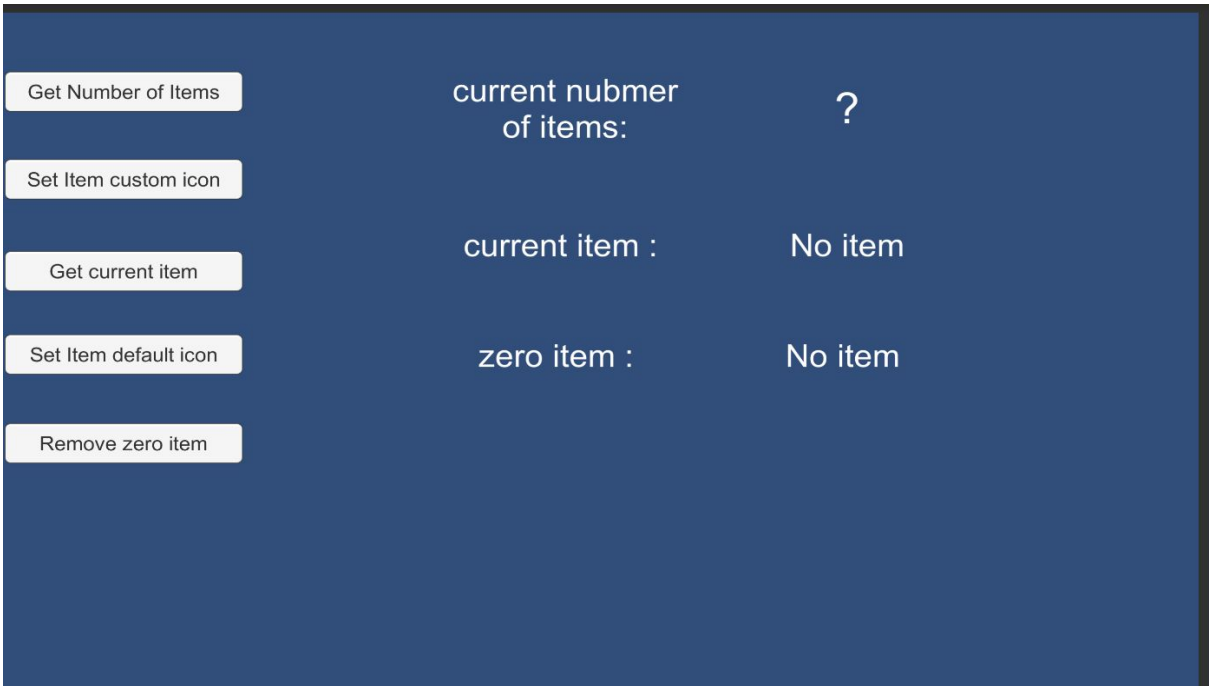
QuickActionItem имеет лишь несколько конструкторов и публичных полей:

- **string Type** - аналог id для Quick Action. Не виден для пользователя и служит только для внутренней идентификации. Обязательный параметр.
- **string Title** - под цифрой 1 на скрине выше. Заголовок, который видит пользователь. Шрифт и его размер изменить нельзя. Текст может быть любой длины, но обрежется где-то после 20 символа. Обязательный параметр.
- **string Subtitle** - под цифрой 2 на скрине выше. Описание, которое видит пользователь. Опциональный параметр. Размер шрифта меньше чем у заголовка.
- **string CustomIconName** - имя .png файла, который вы хотите использовать как иконку. Значение должно содержать только имя файла и его расширение. Путь не должен фигурировать. Например: “testAction.png”. Под цифрой 3 на скрине выше пример использования кастомной иконки;
- **QuickActionDefaultIcon DefaultIcon** - *QuickActionDefaultIcon* вспомогательный enum, инкапсулирующий работу со стандартными иконками Quick Action, встроенными в систему..

Что касается иконок - оба параметра опциональны. Иконка выбирается по следующему правилу: если указана кастомная иконка, она и будет использована, иначе смотрим значение DefaultIcon и ставим системную иконку. Если и стандартная иконка не будет указана, то iOS подставит иконку в виде черной точки.

4. Пример.

В пакете есть тестовая сцена, под названием ***QuickActionsExample.*** Вы можете запускать ее в редакторе Unity, однако, функциона Quick Actions работать не будет. Тем не менее никаких ошибок выведено не будет. Вы можете добавить эту сцену в ваш билд и запустить ее на девайсе.. Она выглядит следующим образом:



Нажатия на кнопки будут иметь мгновенный эффект, по этому можете добавлять и удалять action, сворачивать приложение и сразу наблюдать результат, сильно нажав на иконку приложения.