InAppBrowser Plugin

Basic usage

InAppBrowser.OpenURL("http://www.google.com");

Don't forget about protocol (*http://* or *https://*) in your URL!

Please keep in mind that browser **doesn't** work in Unity Editor - you have to run the app on actual Android or iOS device.

Setup

iOS

No steps required.

Android

Go to *Player Settings* (*File->Build Settings->Player Settings*) , click on Android icon and set *Internet Access* setting to *Require:*

Advanced usage

Customisation

You can specify back button text, page title and text and bar background color by using *DisplayOptions* struct:

```
DisplayOptions displayOptions = new DisplayOptions();
displayOptions.displayURLAsPageTitle = false;
displayOptions.backButtonText = "Go back!";
displayOptions.pageTitle = "My title";
displayOptions.barBackgroundColor = "#FF0000";
displayOptions.textColor = "#00FF00";
InAppBrowser.OpenURL(URL, displayOptions);
```

If *displayURLAsPageTitle* is set to *true*, URL is set as page title. That's default value.
In order to use custom title, set it to *false* and pass new title to *pageTitle*.
Colors should be in *"#RRGGBBAA"* or *"#RRGGBB"* format.

Hiding top bar

If you wish to hide top bar with back button & page title you can simply set option's *hidesTopBar* to *true:*

```
InAppBrowser.DisplayOptions options = new InAppBrowser.DisplayOptions();
options.hidesTopBar = true;
InAppBrowser.OpenURL("http://www.google.com", options);
```

Closing browser through code

If you wish to close browser programatically use InAppBrowser.CloseBrowser().

Example:

```
public void OnButtonClicked() {
  InAppBrowser.OpenURL(pageToOpen);
  StartCoroutine(CloseBrowserAfter5Seconds());
}

private IEnumerator CloseBrowserAfter5Seconds() {
  yield return new WaitForSeconds(5.0f);
  InAppBrowser.CloseBrowser();
}
```

Browser Lifecycle Events

In order to listen for browser lifecycle events simply drag&drop *InAppBrowserBridge* prefab on your scene. It contains script with the same name which will receive lifecycle events from browser and distribute it to your scripts through *UnityEvents:*

- *onJSCallback(message)*
Called when your JavaScript code sends message to your Unity app (see section below).

- *onBrowserFinishedLoading(url)*
Browser finished loading URL passed as parameter.

- *onBrowserFinishedLoadingWithError(url, error)*
Browser encountered error while loading URL passed as parameter. Keep in mind that this callback might be called when any resource on page failed to load, e.g. image, not only main page.

- *onBrowserClosed*
Browser has been closed (user clicked on back button).

You can subscribe to those events either directly from Editor, or code:

```
InAppBrowserBridge bridge = FindObjectOfType<InAppBrowserBridge>();
bridge.onJSCallback.AddListener(OnMessageFromJS);
```

```
void OnMessageFromJS(string jsMessage) {
      if (jsMessage.Equals("ping")) {
            Debug.Log("Ping message received!");
            InAppBrowser.ExecuteJS(javaScriptCode);
      }
}
```

Please keep in mind that *InAppBrowserBridge* object **HAS** to be on your active Unity's scene.

Communication between JavaScript and Unity

Messages between InAppBrowser and Unity are sent via *String* parameter.

- **Sending message from JavaScript to Unity**

Android
Put that line in your JS script:
UnityInAppBrowser.sendMessageFromJS('your message goes here');

iOS
It's more complicated as on Android, as you have to load *iframe* which will contain *inappbrowserbridge* scheme and message passed as a fragment, e.g. *inappbrowserbridge://your_message_goes_here*

Full example:
We will simply pass *'ping'* message to our Unity script, by creating *sendPing* function in our JavaScript file:

```
function sendPing() {
      sendMessageToUnity('ping');
}
```

Now, we have to create *sendMessageToUnity* function which will check platform and either use *UnityInAppBrowser.sendMessageFromJS* on Android or create *iframe* on iOS:

```
function sendMessageToUnity(message) {
      if (isIOS()) {
            appendIframeWithURL('inappbrowserbridge://' + message);
      }  else if (isAndroid()){
            UnityInAppBrowser.sendMessageFromJS(message);
      }
}
```

where *appendIframeWithURL* is :

```
function appendIframeWithURL(url) {
      var iframe = document.createElement("IFRAME");
      iframe.setAttribute("src", url);
```

```
        document.documentElement.appendChild(iframe);
        iframe.parentNode.removeChild(iframe);
        iframe = null;
}
```

You can use those platform-checking functions:

```
var userAgent = navigator.userAgent || navigator.vendor || window.opera;

function isIOS() {
        if (/iPad|iPhone|iPod/.test(userAgent) && !window.MSStream) {
                return true;
        } else {
                return false;
        }
}

function isAndroid() {
        return (/android/i.test(userAgent));
}
```

In order to receive that message from your Unity script check *onJSCallback* event from *InAppBrowserBridge* prefab.

- **Sending message from Unity to JavaScript**

Simply call:
```
InAppBrowser.ExecuteJS(…javaScriptCode…);
```

Example:
```
InAppBrowser.ExecuteJS("alert('pong!')");
```

Please check *ExampleJavaScriptCommunicationScene* to see bidirectional communication and sample website & JavaScript source code.

Using with Playmaker

Extract file from *InAppBrowser/InAppBrowserOpenUrl.cs.zip*. It contains open-browser action which you can use with Playmaker plugin.

Playing HTML5 Video

HTML5 Videos are supported on both iOS and Android, although on Android it requires one additional setup step.
You need to add hardware acceleration flag to your *AndroidManifest.xml* file, as specified here: http://developer.android.com/guide/topics/graphics/hardware-accel.html

If you already use custom AndroidManifest.xml:
If you already have *AndroidManifest.xml* file in your *Plugins/Android* directory, simply add *android:hardwareAccelerated="true"* on *Application* or *Activity* level:
<application android:hardwareAccelerated="true" android:theme…/>

If you don't use custom AndroidManifest.xml:
If you don't use custom custom manifest file, you can copy and paste generated manifest file from *Temp/StagingArea* (in your project's directory) into *Plugins/Android*. Then you can modify it by adding *hardwareAccelerated* flag. Keep in mind you have to build Android version in order to see generated file there.

Performance issues on Android

If you noticed that your page runs slow on Android try to set *hardwareAccelerated* flag to *true*. That step is described in *"Playing HTML5 Video"* section.

More info

Plugin supports Android 3.0+ and iOS7. It doesn't use any private API on iOS, so you can able to submit it to AppStore. Local files are not supported (yet).

Contact

In case of questions, please send me an email at: ptr.zmudzinski@gmail.com
Thanks for using my plugin!