iOS4Unity

Easy native iOS from C#

Introduction

iOS4Unity is an easy-to-use Unity plugin that exposes native iOS APIs to C#. The iOS APIs have been C#-ified and not just the method and property names! Types such as NSString and NSNumber are automatically converted to System.String and System.Double. Callbacks from Objective-C are exposed by simple C# events. Along with this goodness, iOS4Unity is developed entirely in managed C# code.

For complete documentation on the iOS APIs, refer to Apple's documentation here. In most cases, the APIs have been cleaned up for C# so Xamarin's documentation here will bear a closer resemblance to the types exposed by iOS4Unity.

How does it work? Should I be scared?

iOS4Unity uses p/invoke (platform invoke) to call the native Object-C runtime functions from C#. This is the same technique Xamarin uses for iOS development in C# (if you've heard of it), except iOS4Unity uses a much more stripped down version that is more optimal for game development. Callbacks are setup using Unity's MonoPInvokeCallbackAttribute and some cleverness to give Objective-C to callback into your C# code that works completely under Unity's AOT compilation.

Next, let's look at some sample code for a few scenarios.

Getting Unity's main UIViewController

```
private UIViewController GetUnityController()
{
        return UIApplication.SharedApplication.KeyWindow.RootViewController;
}
```

Display a UIAlertView

```
var alertView = new UIAlertView();
alertView.AddButton("Option 1");
alertView.AddButton("Option 2");
alertView.Title = "Title";
alertView.Message = "This is the UIAlertView message";

alertView.Clicked += (sender, e) =>
{
       int option = e.Index + 1;
       Debug.Log(string.Format ("Option {0} clicked!", option));
};
alertView.Dismissed += (sender, e) =>
{
    Debug.Log("Dismissed");
};

alertView.Show();
```

Display a UIActionSheet

```
var actionSheet = new UIActionSheet();
actionSheet.AddButton("Option 1");
actionSheet.AddButton("Option 2");
actionSheet.Title = "Title";

actionSheet.Clicked += (sender, e) =>
{
       int option = e.Index + 1;
       Debug.Log(string.Format ("Option {0} clicked!", option));
};
actionSheet.Dismissed += (sender, e) =>
{
       Debug.Log("Dismissed");
};

var controller = GetUnityController();
actionSheet.ShowInView(controller.View);
```

Open a URL in Safari

```
public void OpenUrl()
{
       UIApplication.SharedApplication.OpenUrl("http://www.google.com");
}
```

(Advanced) Bind a Third-Party Library

First, add the Objective-C library to your Unity project under Assets/Plugins/iOS. Next, you will need to review the Objective-C header files to translate the Objective-C classes to something that C# can understand. Here is a quick example of the UIAlertView class, and usage of iOS4Unity's ObjC and Callback classes for interacting with native code:

```
public class UIAlertView : NSObject
{
	private static readonly IntPtr _classHandle;

	static UIAlertView()
	{
		_classHandle = ObjC.GetClass("UIAlertView");
	}

	public override IntPtr ClassHandle
	{
		get { return _classHandle; }
	}

	public UIAlertView()
	{
		ObjC.MessageSendIntPtr(Handle, "init");
		ObjC.MessageSend(Handle, "setDelegate:", Handle);
	}

	public event EventHandler<ButtonEventArgs> Clicked
	{
		add
		{
		   Callbacks.Subscribe(this, "alertView:clickedButtonAtIndex:", value);
		}
		remove
		{
		    Callbacks.Unsubscribe(this, "alertView:clickedButtonAtIndex:", value);
		}
	}

	public bool Visible
	{
		get { return ObjC.MessageSendBool(Handle, "isVisible"); }
	}

	public string Message
	{
		get { return ObjC.MessageSendString(Handle, "message"); }
		set { ObjC.MessageSend(Handle, "setMessage:", value); }
	}

	public string Title
	{
		get { return ObjC.MessageSendString(Handle, "title"); }
		set { ObjC.MessageSend(Handle, "setTitle:", value); }
```

```
        }

        public void Show()
        {
                ObjC.MessageSend(Handle, "show");
        }

        public void Dismiss(int buttonIndex, bool animated = true)
        {
                ObjC.MessageSend(Handle,
                    "dismissWithClickedButtonIndex:animated:", buttonIndex, animated);
        }
}
```

Need Help?

If you are missing a native API, and would like it added to the iOS4Unity component, we are definitely interested in adding it so everyone has access to it. If you have any questions at all feel free to reach out to us at our website here or email to service@hitcents.com.