Universal Menu

v. 0.4

VrLib now implements two activities instead of one. The second activity is started when the user initiates any of the reserved button interactions detailed below. This activity is only responsible for displaying the Platform User Interface (sometimes referred to as the “global menu” or “meta UI”). The second activity allows non-native applications (currently Unity-based apps) to include the Platform UI with minimal effort. The older “AppMenu” for native applications that was previously included in AppMenu.cpp is now gone and has morphed into GlobalMenu.cpp, but without the ability to add additional menu items.

1. Reserved User Interactions

1.1 Back button/key interactions

Long-press

The user presses the button and holds it for > 0.75 seconds. *A long-press must always open the Universal Menu*.

- Apps must implement Universal Menu access through a user **long-press**. This will happen through integration with the latest SDK. The Universal Menu provides features such as the passthrough camera, a shortcut to Oculus Home, the ability to reorient, brightness, and do-not-disturb mode.

Short-press

The user presses the button and releases it for > 0.25 seconds.

- This is not the same as the user simply releasing the button before the **long-press** time (0.75 seconds) is exceeded. If a short press were anything < **long-press** time, then it would not be possible to abort a **long-press** in progress without resulting in a **short-press**.
- The way a “back” action is interpreted by the application depends on its current state, but generally it will retreat one level in an interface hierarchy. For example, if the app menu is up and at its initial, top-level screen, a short-press will exit the app menu. If no app menu or other satisfactory stateful condition is current (determined by the application), the **short-press** provides a confirmation dialog and then exits the app and returns to Oculus Home.

1.2 Volume button/key interactions

Volume buttons

Volume buttons must adjust the volume using the VR volume UI provided by the Oculus Mobile SDK.

2. Implementation Overview

The Back key

Back key behavior in this new environment is always associated with the Platform UI when the user initiates a long-press. The application is now free to treat a short press on the back key as a generic back action. In an app, a short-press on the back key may, for example, bring up the application's own menu. In other applications, the key could act as a generic back action until the root of the UI hierarchy is reached, at which point it would up the application specific menu if there is one, or exit the application to Oculus Home after calling a confirmation dialog.

Native apps

For native applications, see `ovr_StartPlatformUI()`. The UI may be started for various purposes; to bring up the global menu, a quit / home confirmation dialog, et cetera.

In native apps the application is responsible for hooking the back key short-presses by overloading `VrAppInterface::OnKeyEvent()` and deciding when the user is at the root of the application's UI, at which point it should ignore the back key event by returning false.This will allow VrLib to handle the back key and start the Platform UI quit confirmation dialog.

Unity apps

In Unity apps, the application still decides when a short-press opens the Platform UI's quit confirmation dialog, but the Unity scripts are responsible for starting the Platform UI by issuing a plugin event as `OVRPluginEvent.Issue( RenderEventType.PlatformUIConfirmQuit )`.

Unlike native applications using VrLib, which always intercept a back key long-press, Unity applications must handle all of their own input and start the platform UI with `OVRPluginEvent.Issue( RenderEventType.PlatformUI )` when a back key long-press is detected. See HomeMenu.cs in the SDK for an example.

Manifest file requirement

In order for the new Platform UI activity to work, the application's manifest must be updated to add the activity. Add the following to AndroidManifest.xml just after the existing <activity /> block inside of the <application /> block:

```
<activity android:name="com.oculusvr.vrlib.PlatformActivity"
android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen"
android:launchMode="singleTask" android:screenOrientation="landscape"
android:configChanges="screenSize|orientation|keyboardHidden|keyboard"
android:excludeFromRecents="true"> </activity>
```

For Unity apps, the Unity integration will now provide a res/raw folder located under Plugins/Android which contains the necessary resources for the platform activity.

While developing, it is often convenient to set excludeFromRecents to false. This will allow you to easily kill the app from the multitasking window when necessary.

Last Update: Nov 10, 2014