SDK Migration Guide

Moonlight SDK Migration To 0.4

We have made significant changes to the SDK structure since its limited release as the Moonlight SDK. This document discusses the most significant changes at a high level. If you began Gear VR development with the public 0.4 release, your project will not need to undergo any of the migration described in this doc.

For any migration process, **be sure to back up** your project prior to migration. If you are not using a source control solution to track changes to your project, you should start doing so before attempting to migrate. This will make all changes recoverable. Git and Perforce are both widely-used source control solutions.

When possible, test each significant change before moving onto the next one and submit to your source control application in stages. If something breaks, you can always go back to the last known good version of your application, or use that version as a base for diffing your most recent changes.

Migration from Moonlight SDKs Prior to 8-27-2014

Native projects no longer have global pointers to the App instance. This is partly due to cleaning up global state, and partly due to the implementation of the Platform UI. Instead of a single AppLocal instance, there is now one AppLocal instance per activity. The AppLocal instance is now created in VrAppInterface::SetActivity(), and stored on VrAppInterface.

In general, dealing with this requires passing the App pointer through functions. At times it may make sense to store a pointer to the App instance somewhere (such as in a class), but this is not recommended except as a temporary measure. In all instances, VrLib currently passes App down where it's needed. If you must store a pointer to App, be very sure not to do so in code that may be called by both the normal VrActivity activity and Platform UI activities, unless the storage is also instanced per-activity.

Most callbacks from Java to native code now take a jlong appPtr parameter, since the global AppLocal instance no longer exists.

Also due to the Platform UI changes, some initialization which was previously done per-activity is now done only once per application instance. nativeInit() no longer exists since AppLocal is instantiated in VrAppInterface::SetActivity(), which is called from nativeSetAppInterface(), called from the Java Activity object's onCreate().

VRMenu

The UI system has changed significantly. There is now a fully-realized event messaging and component system. Any menu object can have multiple components attached that can handle any messages sent to the object.

Events are set to menu objects in three possible ways: a broadcast event, which is sent to all objects starting with the root, but may be consumed or passed on; a focus event, which is passed to objects in the order their cull bounds are intersected, ending up at the focused object (if there is one); and a targeted event, which can be sent to a specific object, but which will go through its parent hierarchy first (such as an object that has just lost focus). Distributing events along these hierarchy paths allow parent objects to consume events before children, if necessary. An example of where this is useful is a swipe event that rotates a group of objects. The container object holding all of the rotatable objects can intercept and handle the event in its own components, as long as any of its children are gaze selected. See VRMenuEventHandler.cpp for more details on message propagation.

For examples of component implementation and usage, see DefaultComponent.cpp, GlobalMenu.cpp, FolderBrowser.cpp and PlatformActivity.cpp.

Last Update: Nov 10, 2014