OSVR Plugin Design and API Overview

Ryan A. Pavlik, PhD
Sensics, Inc.
July-2015

Links

- Dev Portal: http://osvr.github.io/build-with/#building-a-plugin
- Docs on "Writing a Device Plugin"
  - http://resource.osvr.com/docs/OSVR-Core/md_TopicWritingDevicePlugin.html
- Self-contained plugin example:
  - Cross-referenced source in Doxygen: http://resource.osvr.com/docs/OSVR-Core/com__osvr__example__selfcontainedDetectAndCreate_8cpp_source.html
  - https://github.com/OSVR/OSVR-Core/blob/master/examples/plugin/selfcontained/com_osvr_example_selfcontainedDetectAndCreate.cpp

Plugins

- Plugins can create/provide a device
  - Devices expose one or more interface classes
- Plugins can expose the ability to “detect hardware”
  - On callback from the server, look for the associated device and handle it if present

Process of Instantiating Device

- Create “DeviceInitOptions”
- Use to specify device interface classes
  - Returns an object used to send data on the interface.
- Create device
  - Trade your DeviceInitOptions for a DeviceToken
- Register callback
- Submit JSON device descriptor.
- Plugin structure intentionally flexible

Sync vs Async Devices

- A design distinction at the plugin level only: transparent to clients
- Most devices are “async”
  - your callback gets called on its own thread
  - you can block waiting for data
  - When sending data, behind the scenes will wait briefly for control of the transport

Sync devices

- Not as common, more complex
  - Typically used when building plugins for devices with a non-blocking SDK
  - You get frequently called, but any blocking adds to system latency
  - Not for most plugins

Suggested Structure

- Code:
  - A hardware detection object (functor)
    - Keeps track of what devices are already handled (will be added to core soon)
    - Creates device objects
  - Device objects
    - Holds on to the interface objects and device token, and any device handles/state needed
    - Device callback is a member function

Physical structure

- Most plugins maintained out of the OSVR-Core source tree.
- Compile/link against osvrPluginKit (and osvrUtil) headers and libraries
  - C API/ABI
  - Header-only C++ wrapper
  - Other libraries are used internally by PluginKit but not considered “public” APIs

Build Systems

- Cross-platform: CMake (self-contained sample)
  - Works for Android too with android-cmake toolchain
- Android:
  - basic ndk-build importable modules available
- Roll your own:
  - If you must, just need the include and libs for osvrClientKit and osvrUtil

For additional information:

- OSVR developer portal
  - http://osvr.github.io

- Sensics – Founding contributor to OSVR, experts working in VR/AR for over a decade
  - http://www.sensics.com

sensics

OSVR