OSVR Platforms and Portability

Ryan A. Pavlik, PhD
Sensics, Inc.

July-2015

Existing Platform Support

- Windows (Vista+, 32 and 64 bit)
    - Most users right now
- Android
    - Lot of interest
    - Major focus for us right now
- Linux
    - Primarily for dev, CI, and non-consumer applications

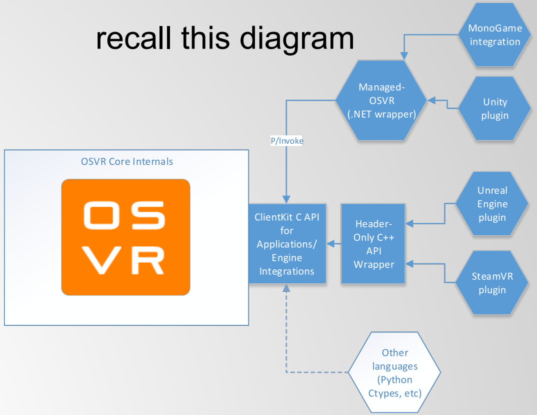
Portability

- OSVR-Core codebase
  - Written in C++11 with C public APIs
  - CMake build system for a single build system across all platforms
  - Most code is platform independent
- Potential
  - Underlying libraries used are all used on more platforms than OSVR supports right now
  - Portability a key design goal

Platform-specific adaptations

- Build on libraries (Boost, VRPN, HIDAPI, OpenCV, ...) with wide platform support already for minimum additional OSVR-specific adaptation
- Details of locating plugins, configuration files, and some hardware-interaction code varies between platforms

Terminology Notes


recall this diagram
MonoGame integration
Managed-OSVR (.NET wrapper)
Unity plugin
P/Invoke
OSVR Core Internals
OS VR
ClientKit C API for Applications/ Engine Integrations
Header-Only C++ API Wrapper
Unreal Engine plugin
SteamVR plugin
Other languages (Python Ctypes, etc)

- Terms used as follows:
  - Native code/API
    - C++ or C code writing against the main C++ or C APIs of OSVR
  - Managed code/API
    - Wrapping the OSVR native API for a VM/managed language
    - Includes .NET/Mono/CLR - Managed-OSVR (via p/Invoke)
    - Includes Android Java (via JNI)
  - Foreign Function Invoke (FFI)
    - C ABI/API-based invocation API from non-C-family language
    - .NET uses an FFI-based wrapper, but FFI != managed (e.g. Python CTypes, LuaJIT, etc. is FFI, while Java JNI is not FFI)

Android Port

- Initially built and used like a desktop Linux
  - Command-line server
  - Android permissions affecting direct USB access, so server run as root
  - Client applications using Unity were trivial to port
- Native libraries
  - Build against using CMake, now also ndk-build compatible packages
- JNI wrapper - TODO (prototype exists)

Android Deployment Plan

- One package with an Android app providing configuration and hosting the server
    - Server is actually a library, just replace command-line interface with Android interface
- Additional packages providing plugins for hardware support and GUI for authorization/permission as required
- Launch-on-demand vs launch-on-boot

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

OS VR