Display Handler

A framework and implementation for client virtualization interaction

Brendan Kerrigan

153 Brooks Road, Rome, NY | 315.336.3306 | http://ainfosec.com

Outline

- **Background**
- Goals
- Architecture
- Features
- Future Goals and Development

Background

- Previously AIS had made a secure compositing technology on top of surfman
- The architecture created challenges that impacted user experience
- Features our customers desired were not available in surfman, and we had no way to add them

Background

- Began design in Fall 2013 after a period of survey and experimentation with what existed in open source
- Several prototypes were made:
  - Wayland
  - Mir
  - eGL
  - DRM

Background

- We settled on a design that would utilize a plugin architecture
  - Minimize damage of lock-in to a specific technology
  - Allow extension by outside parties
  - LGPL was the intended license to allow for 3rd parties to develop proprietary bits that were interoperable
- Would utilize Qt libraries, along with two custom rolled libraries (svlib and libivc)
- Initially would have a DRM dumb buffer renderer

Outline

- Background
- **Goals**
- Architecture
- Features
- Future Goals and Development

Goals

- Modular, testable codebase
- Flexible multi-monitor support (extended, cloned, pinned, seamless)
- Intuitive monitor hotplugging
- Lower the surrounding code necessary to prototype new rendering technologies

Goals

- Up to 6 monitor support
- 4K resolution support
- At least feature parity with surfman, excluding zero copy
- Minimize the effort in bringing in new hardware compatibility

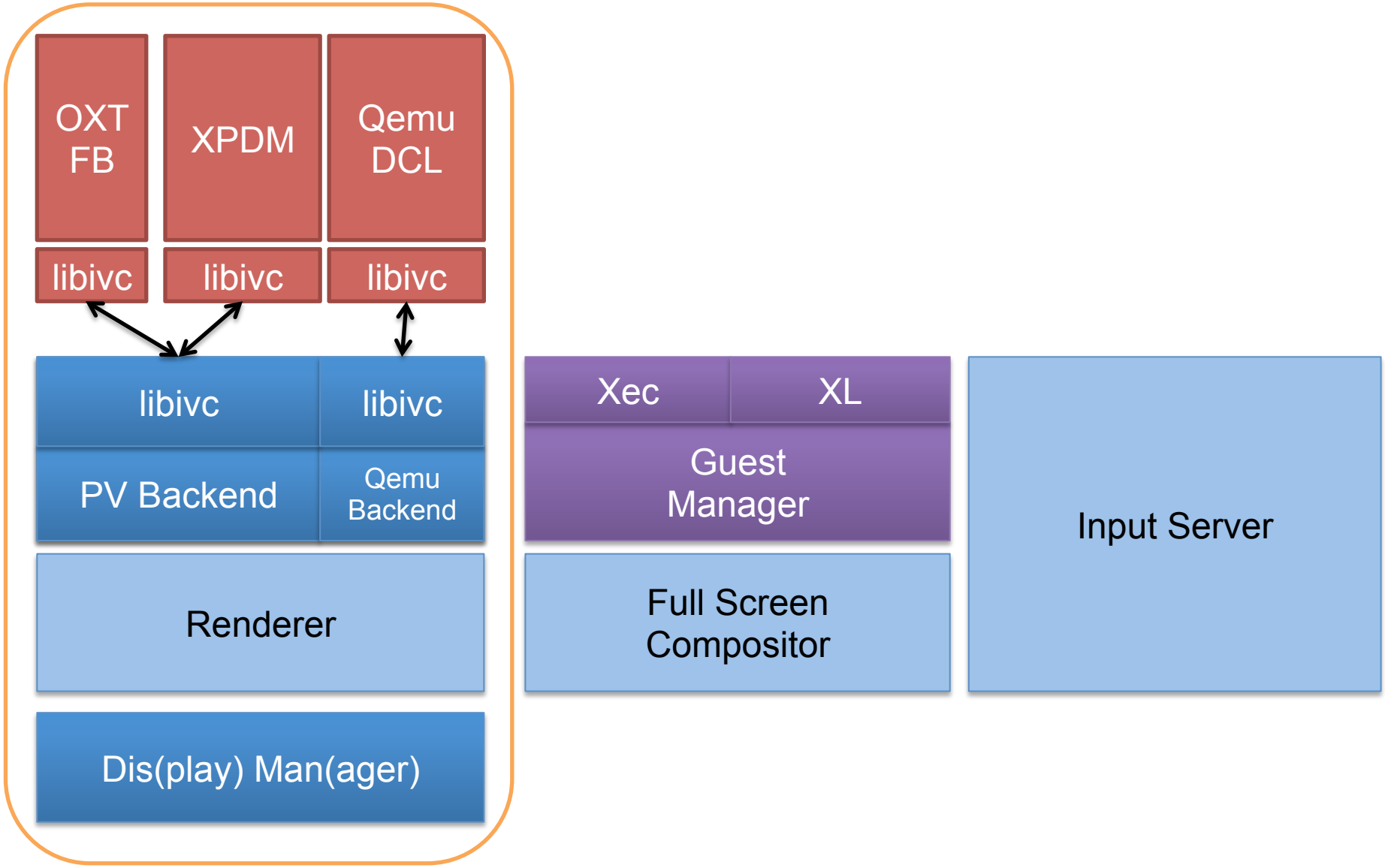
Outline

- Background
- Goals
- **Architecture**
- Features
- Future Goals and Development

Architecture

- C++ code base
- Utilizes the Qt framework (a very small subset of the libraries, ~8MB)
- Each set of modules is loaded as a plugin, and communicate via the signal/slot mechanism Qt provides

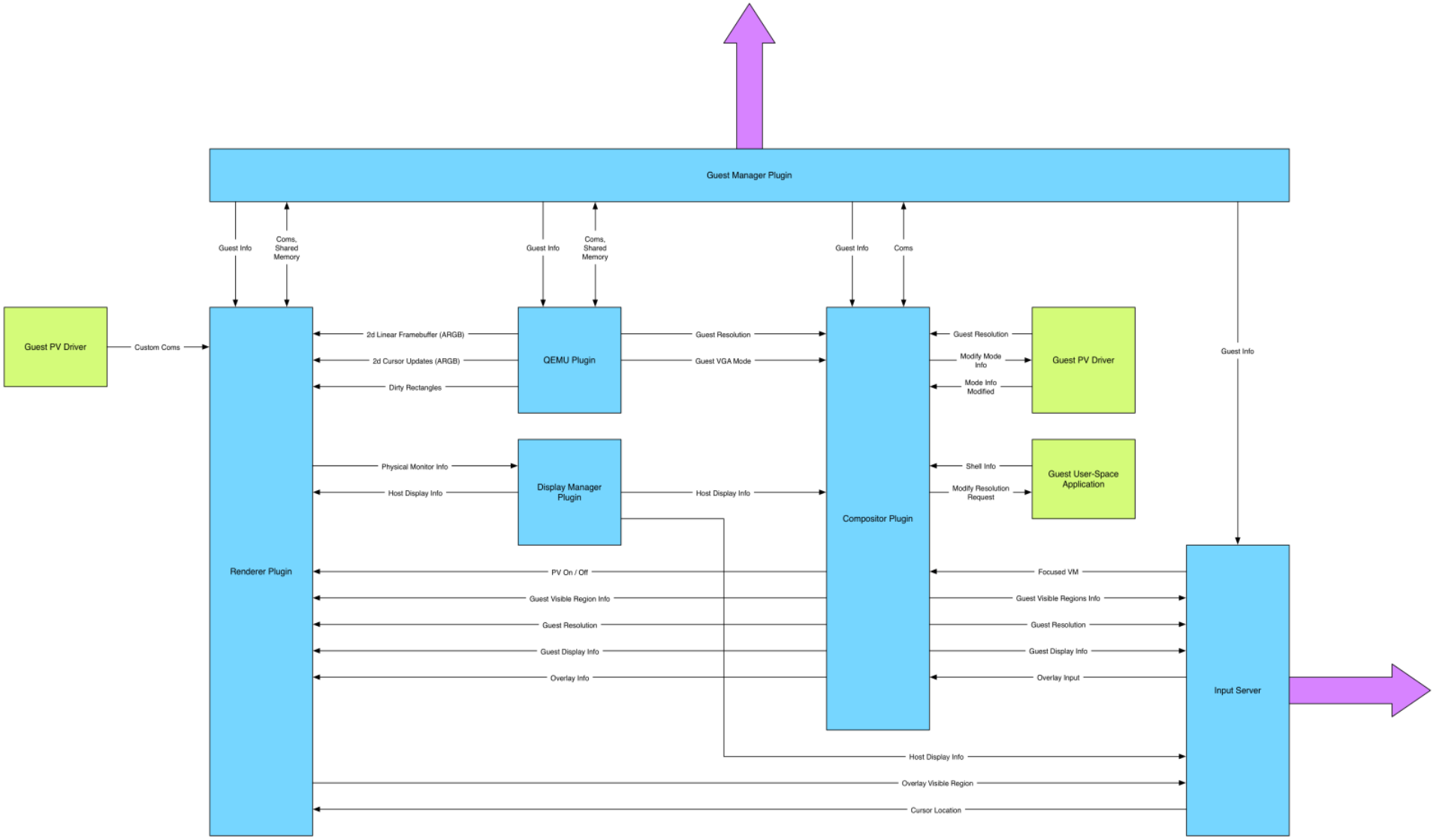
Architecture



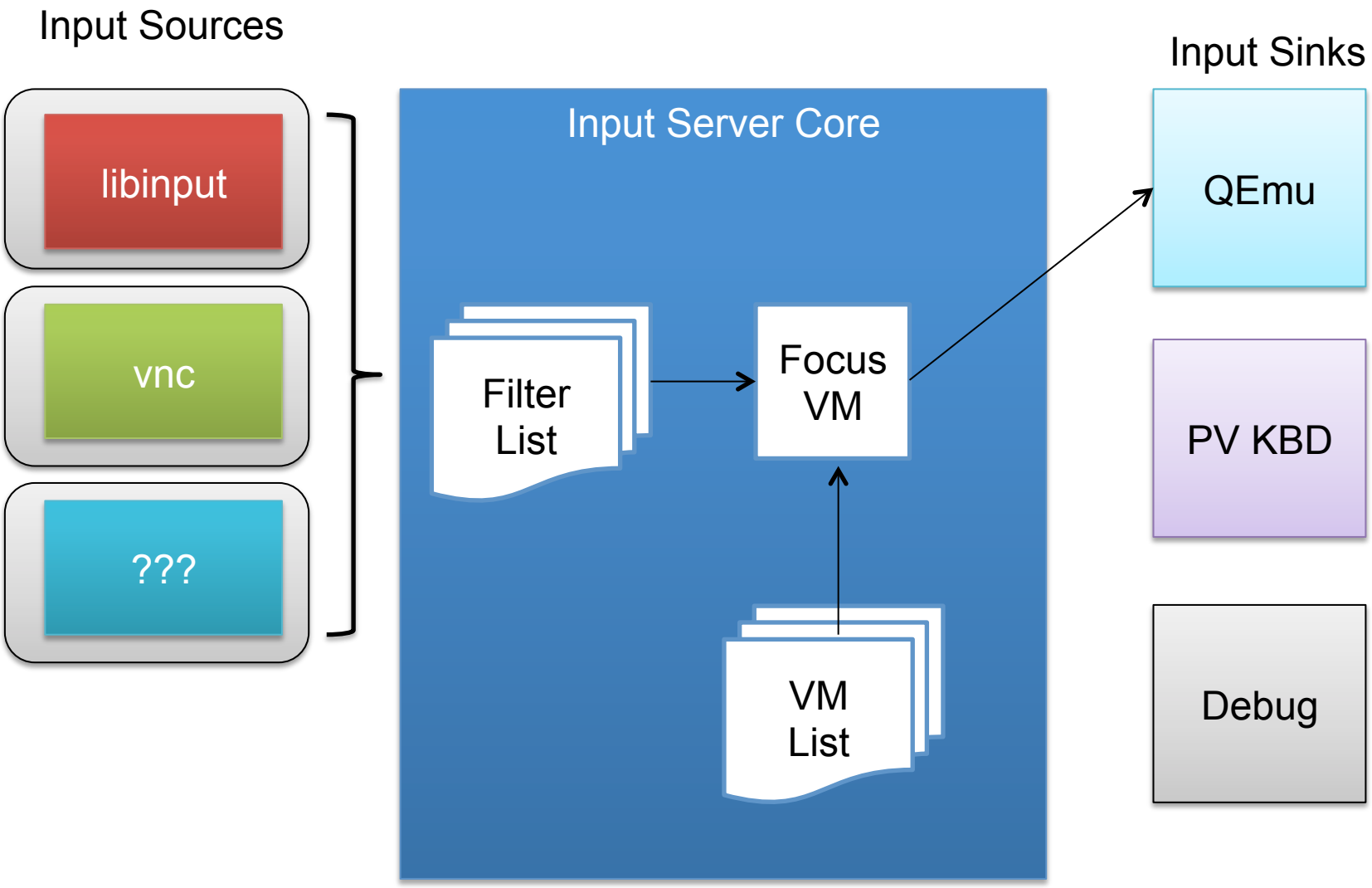
Architecture

- Modules:
  - Guestman – interface with the applicable toolstack module (currently support xec and xl)
  - Compositor – A mix between a compositor and a windowing manager (currently only full screen composition is supported)
  - Renderer – Does the final blitting to the scanout buffer (currently DRM dumb buffer support is included)
    - Custom renderers will have separate requirements for their ‘PV stack’
    - Current implementation has our PV stack based on libivc, and include generic Qemu based guest support, Linux FB driver support, and Windows XPDM support
    - I’ll discuss this a little later on with regards to other renderers
  - Display Manager – Provides information about currently available displays’ orientation, resolution, and layout
  - Input Server – A new input server module is included, and this brings the input server back into the same process space as the rest of the display code

Architecture



Architecture



Outline

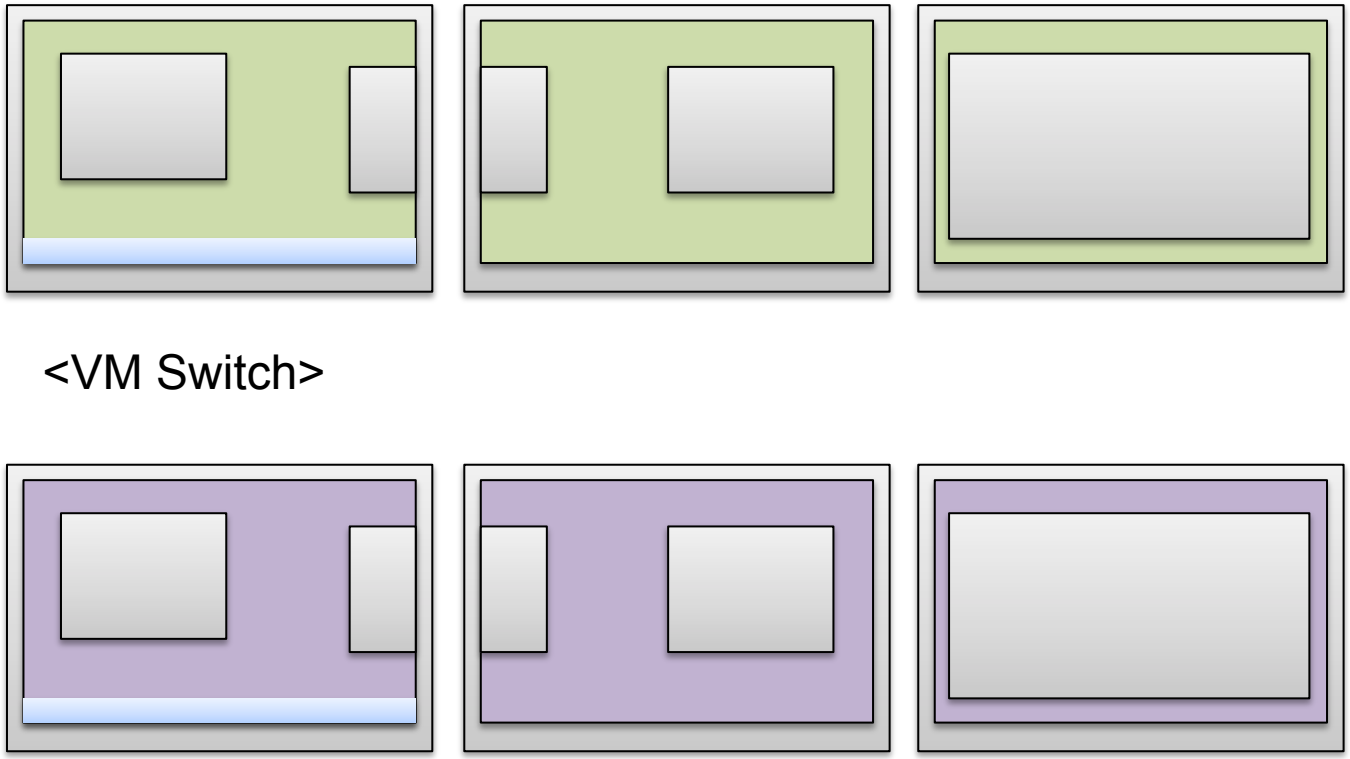
- Background
- Goals
- Architecture
- **Features**
- Future Goals and Development

Features

- Three modes of operation
  - Extended – in focus VM displays to all monitors
  - Cloned – in focus VM displays to all monitors, but the image is replicated on each monitor
  - Pinned – VMs can be assigned a subset of available displays

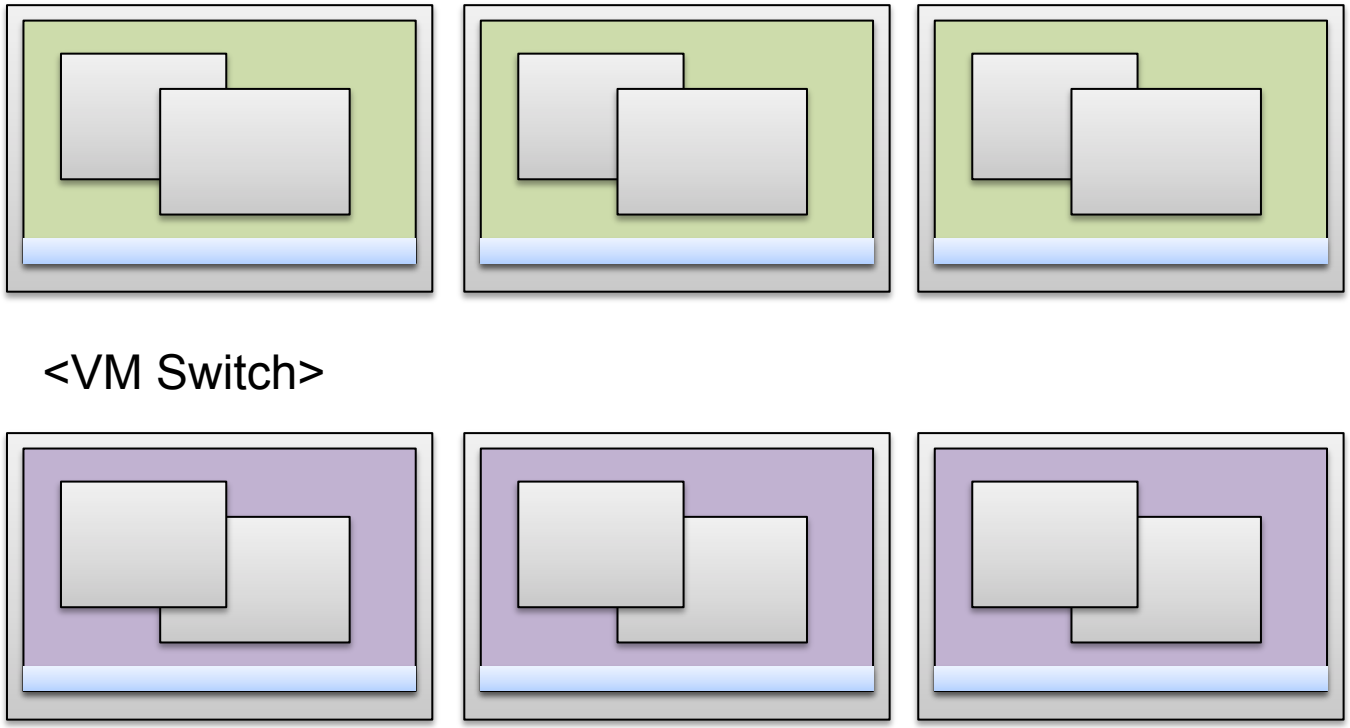
Features

- Extended mode



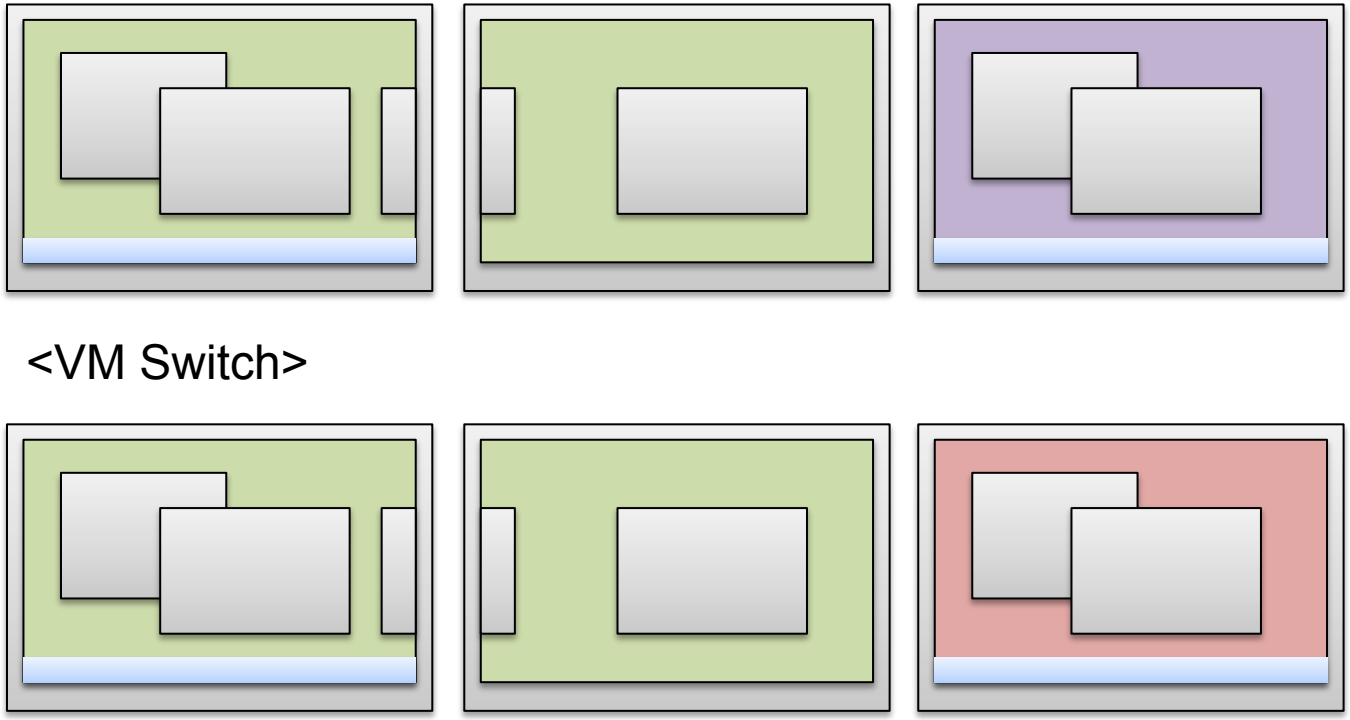
Features

- Cloned mode



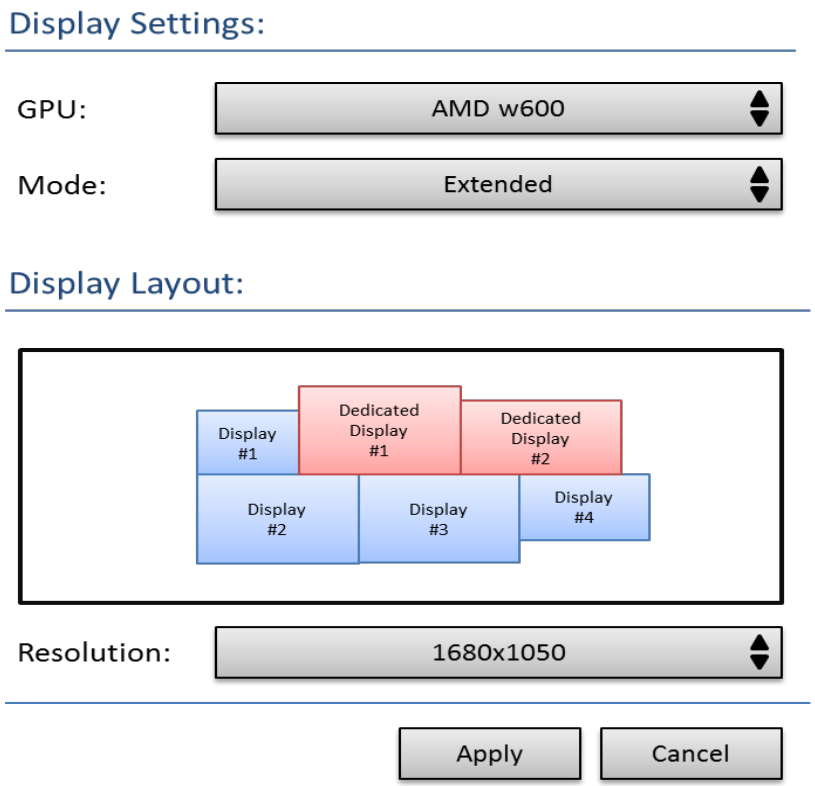
Features

- Pinned mode



Features

- Configure displays from the UIVM



Features

- Monitor hotplugging is well supported
- Feature parity with surfman
- Optional banner that displays VM name and administrator configurable string
- Guest/host resolution independence (auto scaling if necessary)

Features

- Performance impact is small
- With multiple HD videos playing full screen, CPU utilization sits at ~6%
- Audio processing actually uses more cycles during ad hoc video playback testing

Features

- Currently works on both OpenXT and upstream Xen
- Large set of unittests accompany the code

Outline

- Background
- Goals
- Architecture
- Features
- **Future Goals and Development**

Future Goals and Development

- First and foremost goal – open sourcing effort
  - This should be beginning soon and it is anticipated to be finish sometime during the summer
  - Unit testing coverage will be extended
  - Display Handler will most likely have it's own Github organization with the repositories necessary to build it
  - Packaging is a separate problem, though an OE layer will be provided

Future Goals and Development

- Additional renderers are being investigated:
  - Intel GVT-g as a renderer, providing mediated GPU acceleration using native Intel drivers in guest
  - GL based renderer, providing some improvement in performance and power consumption (especially in scaling cases)
  - virGL based renderer, providing API forwarding style 3D graphics to guests

Future Goals and Development

- Extending the input server to support some more modern input technologies
  - Better touch integration and passthrough
  - Supporting other sensor data being sent to guests
    - Accelerometer
    - GPS
  - UI configuration of gesture control