Monocle

Overview

Monocle is the implementation of the Glass windowing component of JavaFX for embedded systems. Monocle provides windowing functionality and access to native graphics for simple embedded systems that do not have an underlying window system.

Description of Monocle components

This is hard to read in the web interface, so you'll want to download the PDF if you are interested in Monocle's inner workings.

How to build Monocle

If you build OpenJFX for embedded Linux/ARM platforms from the 8u-dev repository then you get a functioning Monocle as well. You can also build it for Linux/x86 using the x86egl compile target (build with `-PCOMPILE_TARGETS=x86egl` on a system with GLESv2 and EGL development libraries installed). This builds the embedded stack for the Linux desktop; it is not the same as the GTK implementation of Glass. You cannot currently build any of the graphical implementations of Monocle for desktop windowing systems.

How to run Monocle

Monocle is now the default implementation of Glass on embedded platforms.

You can select one of the specific back-ends of Monocle with the system property `monocle.platform`. Some of the back-ends work only with hardware rendering (the `es2` pipeline); some work only with software rendering (the `sw` pipeline).

monocle.platform	prism.order options	Hardware on which this might work
`MX6`	es2 (default) or `sw`	Freescale i.MX6 SDP or similar boards. Needs accelerated Vivante graphics drivers for framebuffer; not all OS configurations have these.
`OMAP`	`es2` (default) or `sw`	BeagleBoard xM. Note that the es2 pipeline requires PowerVR graphics drivers, which are only available on soft float configurations of Linux on the BeagleBoard.
`OMAPX11`	`es2`	BeagleBoard xM. Renders the JavaFX window stack to a single X11 window.
`X11`	`es2`	BeagleBoard; Linux/x86 desktop
`Linux`	`sw`	Any Linux system; uses software rendering
`Headless`	`sw`	Any system
`VNC`	`sw`	Any system

If you are running the desktop build of JavaFX or OpenJFX then your only monocle option is Headless. Desktop JavaFX does not support the `javafx.platform` system property, but you can select Monocle with:
`-Dglass.platform=Monocle -Dmonocle.platform=Headless -Dprism.order=sw`

On MacOS and Windows, removing `-Dprism.order=sw` can be critical to prevent crashes.

Status

What's working:

- Accelerated rendering on Freescale i.MX6, BeagleBoard xM and Raspberry Pi
- Mouse, key and single-point/multi-point touch input with Linux device nodes
- Synthesis of mouse events from touch events
- Double-buffered software rendering to memory-mapped /dev/fb0, with a software cursor

- Hardware cursors on OMAP3, i.MX6 and Raspberry Pi
- Touch coordinate transformations for screen calibration
- Pluggable pipeline for touch event cleanup
- Nested event loops
- Robot input and capture
- HelloSanity is working on Freescale i.MX6, BeagleBoard xM and Raspberry Pi
- Headless implementation running on embedded and desktop platforms, passing unit and system tests (base/graphics tests, system tests and Linux input tests)
- VNC server mode with remote display and mouse input
- Mouse input and accelerated rendering on X11
- Drag and Drop
- Full screen and minimized windows

What's not done yet:

- Android port
- DirectFB port

See also the JIRA query for open issues on Monocle

How to port Monocle

See Porting JavaFX to additional embedded Linux devices