OpenJFX on the Raspberry Pi

You can run JavaFX on the Raspberry Pi, an inexpensive ARM development board. This page describes how to set up your board to run JavaFX.

Prerequisites

You will need:

- A Raspberry Pi device
- A PC to download and install the OS image for the Raspberry Pi
- An SD card with a capacity of 4GBytes or more. A fast card (such as class 10) works better. A larger card will usually last longer, because it has more blocks to degrade.
- Mini-USV 5V power adapter. Although the recommended minimum for the Raspberry Pi is 800mA, we find that a higher-rated power supply prevents some problems. For example, when using both the CPU and GPU at full speed with an 800mA power supply, you can get a sudden loss of power to USB ports, causing input devices to be disconnected. A 5V 2A power supply works well.
- Input devices (USB keyboard, mouse and/or touch screen). Use of a powered USB hub can prevent problems here, especially with a touch screen.
- A display with HDMI or composite input and appropriate cables. A display with DVI input and an HDMI-to-DVI cable also works well.

Touch screens known to work with JavaFX are:

- Screens from Chalkboard Electronics - both their older 10" 800x600 screen (2012 model) and their 10" 1280x800 screen (from the beginning of 2013)
- The M2256PW monitor from 3M
- The touch screen integrated into the Freescale i.MX6Q Sabre Device Platform.

In general a touch screen that is recognized by Linux and generates EV_ABS events will work with JavaFX.

Raspberry Pi OS

Tested:

- Raspbian Jessie
- Raspbian Jessie Lite
    - Note: is missing the Pango package for running OpenJFX. Easily fixed with "apt-get install libpango-1.0-0:armhf libpangoft2-1.0-0:armhf"
- NOOBS (v1_4_1) installing:
    - Raspbian Debian Wheezy (2015-05-05)

When the raspi_config options are displayed, there is one critical item for OpenJFX. Under "Advanced Option" choose "Memory Split" and set the value to 128 or 256. This is the memory allocated for GPU use. OpenJFX can use quite a bit of memory for textures and shaders. The amount available to OpenJFX cannot be queried from the OS, and OpenJFX cannot always handle the failure gracefully. You can always run the raspi_config tool after install to change the value.

The Jessie images were tested with jdk-8u65-linux-arm32-vfp-hflt.tar.gz, which is downloadable from the Oracle JDK8 download page. Note that this version of the JDK does not come with an integrated JavaFX. JavaFX can be obtained by building OpenJFX for ARM, or by using one of the community builds.

OLDER DISTRO NOTES

The configuration used by Oracle for testing is:

- Raspberry Pi model B
- The 2013-05-25 image of Raspbian Linux. You can get it from the Raspberry Pi downloads page.

Note that you need the hard-float Raspbian image. If you use the soft-float Debian "wheezy" image you will not be able to run JDK 8 (or any other software compiled for ARM hard float).

Raspbian setup instructions are at http://elinux.org/RPi_Easy_SD_Card_Setup.

When you first power up the board with Raspbian you will get the `raspi-config` tool. There are a few things to note here:

- Make sure to expand the filesystem, or you won't have enough room to install the JDK.
- If you are using the device primarily with JavaFX, you will probably not want to select "Enable Boot to Desktop". JavaFX on the Pi takes over the whole screen and does not interact with the Linux desktop.

- Under "Advanced Options" you can select "Memory Split" to allocate memory to the graphics processor. A 50/50 split will let you get most use out of JavaFX accelerated graphics.
- If you are using JavaFX over ethernet, it can be convenient to turn on the SSH server under "Advanced Options". Since JavaFX takes over input devices, it can be hard to stop an application if you don't have an SSH connection to it from another machine.

Because of the way JavaFX is run, it is sensitive to the system screen blank timer. To disable the screen blank timer, edit the file */etc/kbd/config* and change the lines for BLANK_TIME and POWERDOWN_TIME to be 0 (disable), and reboot. If not disabled, many applications will appear to stop after 30 minutes or so of use.

If you are connecting to your Pi over the network, you can save some time by setting a host name, and enabling ssh access for it in `raspi-config`, then installing the `avahi-daemon` package (`sudo apt-get install avahi-daemon` from the command line). You'll then be able to reach the Pi from a Linux PC with the command: `ssh pi@<host name>.local`

If the Raspberry Pi isn't detecting the screen size correctly, you might need to tweak video mode settings and maybe tell the Pi to ignore the capabilities reported by the display. For example, some of the Chalkboard Electronics screens require installing an EDID file and/or changing the boot configuration.

If you run into problems with input events being dropped, you should try reducing the USB bus speed. You need recent firmware to do this, so first you should update firmware:

```
sudo apt-get update
sudo apt-get install raspberrypi-bootloader --reinstall
```

Then open `/boot/cmdline.txt` in an editor and add on the same line as the other options `dwc_otg.speed=1`. Run `sudo sync` and reboot. This drops USB speeds from 480Mbits/s to 12Mbits/s, which is known to resolve issues with a variety of USB devices on the Raspberry Pi.

Running a JDK

Raspbian has Java SE 7 preinstalled on the image. This version does not contain JavaFX. To obtain JavaFX download either:

- Java SE 8. (Linux ARM v6/v7 Hard Float ABI).
- The development mainline JDK9 early access.

This downloaded bundle should be unpacked on the Pi. For example,

```
sudo tar zxvf jdk-8-linux-arm-vfp-hflt.tar.gz -C /opt
```

To check that the JDK is installed correctly, run:

```
/opt/jdk1.8.0/bin/java -version
```

This should show that you are running JDK 8. If the VM won't even start, you might be running a hard-float VM on a soft-float system.

To avoid confusion, change your PATH variable so that the newer version is earlier in the path.

You can use the samples bundle from https://jdk8.java.net/download.html on the Raspberry Pi. Not all the samples will work on the Pi; here are some that will:

- Stopwatch
- BouncingBalls
- Calculator
- BrickBreaker (requires display resolution of at least 1280x720)

You can run these applications without any additional parameters. For example,

```
/opt/jdk1.8.0/bin/java -cp Stopwatch.jar stopwatch.MainScreen
```

Note that the default configuration of JavaFX on the Raspberry Pi does not use X11. Instead JavaFX works directly with the display framebuffer and input devices. So you should not have the X11 desktop running when starting JavaFX.

JDK 8 EA builds for the Raspberry Pi include full support for hardware accelerated graphics, with everything from the base, graphics, controls and FXML modules. Media and Web modules are not included.

Stopping an application

JavaFX on the Raspberry Pi takes over the whole screen and captures all Linux input devices. While this will generally be the behavior you want in a deployed application, it is less convenient for development because you can't stop an application using control-C unless the JavaFX application has a KeyEvent handler that listens for control-C and calls `Platform.exit()`. There's nothing unusual about this - many Linux full-screen console applications have the same behavior - but it is often useful to have a quick way to end an an application during development without changing the application code.

There are two ways to run applications with the ability to be terminated by control-C:

- Run applications over an SSH connection from a PC. This gives most control over the device, because once you have SSH connections set up then you can use them for other purposes as well.
- Alternatively, you can use a built-in debugging feature to trap control-C. If you set the environment variable `JAVA_DEBUG=1` before starting Java then JavaFX will exit when you press control-C. For example

```
JAVAFX_DEBUG=1 /opt/jdk1.8.0/bin/java -cp Stopwatch.jar stopwatch.MainScreen
```

The `JAVAFX_DEBUG` environment variable is only for use in development and you shouldn't rely on it for deployment of your application. In the future this functionality might be specified differently or be removed.

Raspberry Pi Memory Split

The older versions Raspberry Pi Raspian preallocate a fixed amount of the system memory for use by the video engine (VRAM). The utility `raspi_config` can be used to alter how much memory is allocated to VRAM.

Newer versions of Raspian can dynamically allocate system memory for use as VRAM.

The minimum recommended memory split for JFX on the Pi is 128 MBytes, with many applications requiring 256 MBytes.

This JavaFX texture caching mechanism currently defaults to 512 MBytes - a value that will likely exceed what is available on the Pi. Given the limited amount of VRAM on the Pi, it is quite possible that an image intensive application might fail when the cache exceeds the available system limit.

Read this page for more details on VRAM and tuning.

Touch events

A touch screen attached to the Raspberry Pi generates both `TouchEvents` and `MouseEvents`. The Monocle subsystem in the latest OpenJFX sources supports a wider range of touch screens than the 8u6 release does and also supports calibration.