colord: color daemon

Richard Hughes

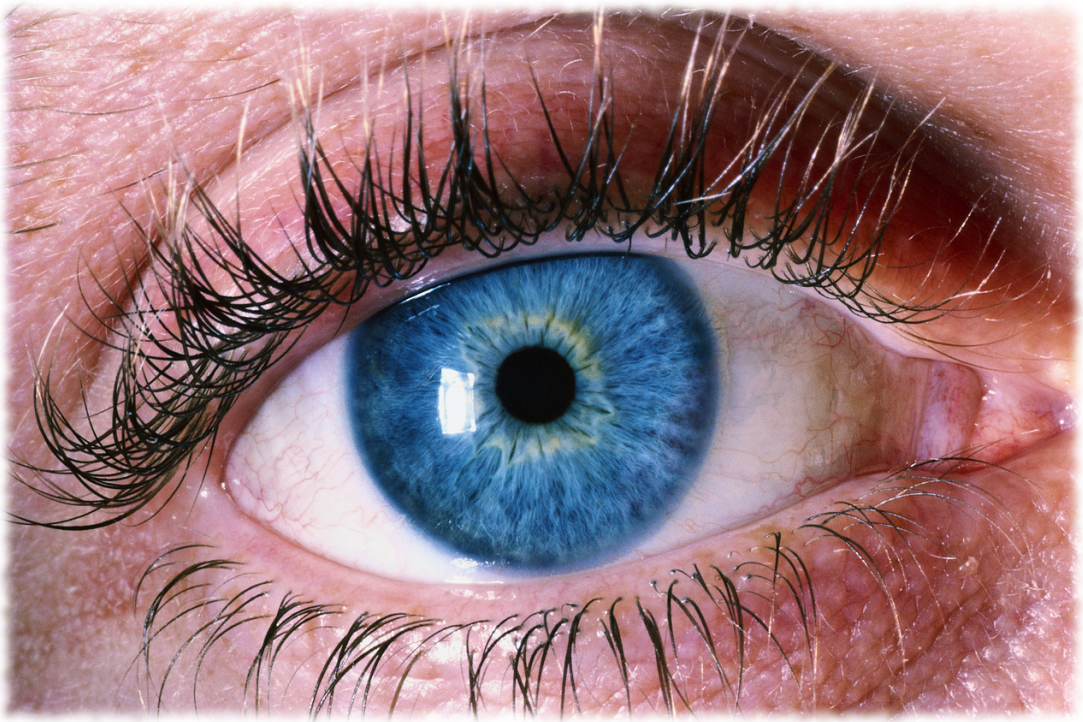
April 8, 2011

Who am I?

- My name is Richard Hughes
- Maintainer of GNOME PackageKit and PackageKit
- Maintainer of GNOME Power Manager and UPower
- Maintainer of GNOME Color Manager and colord
- I work for Red Hat in the desktop team and live in London
- I've been working on colour management for a few months

Basic principles: Gamut

- Human eye can only capture a certain range of colors



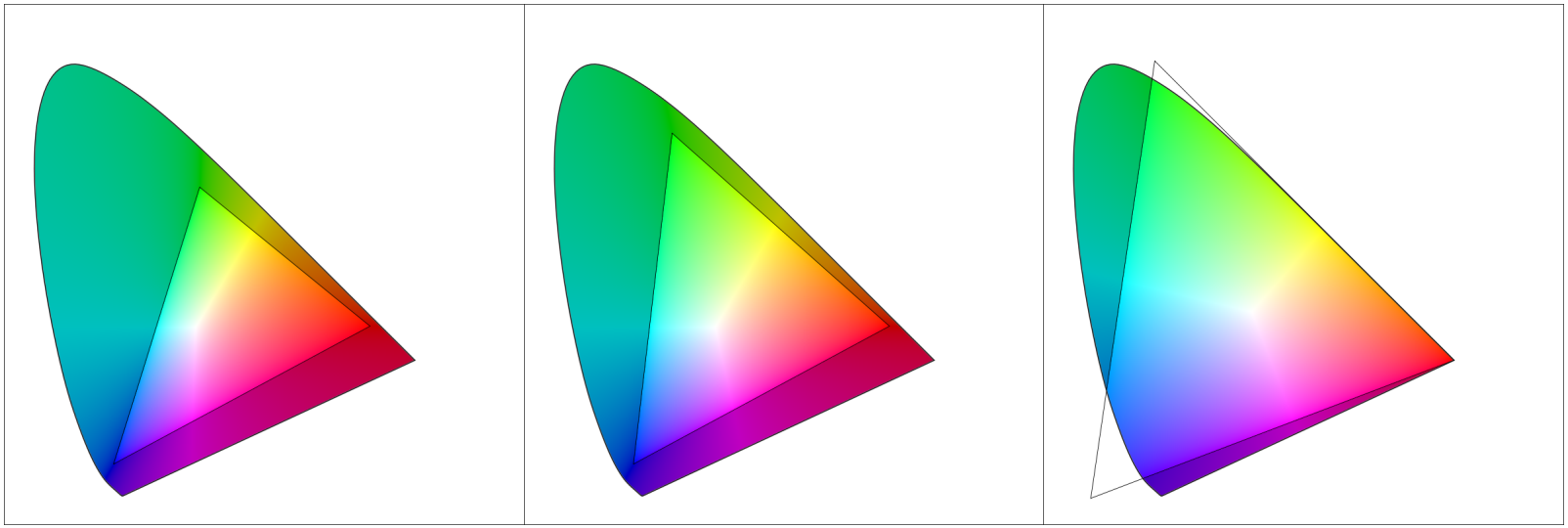
Basic principles: Gamut

- Devices can only capture or produce a certain range of colors

- Mapping from one color-space to another (RGB to CMYK)

Basic principles: Gamut

- sRGB vs AdobeRGB vs ProPhotoRGB

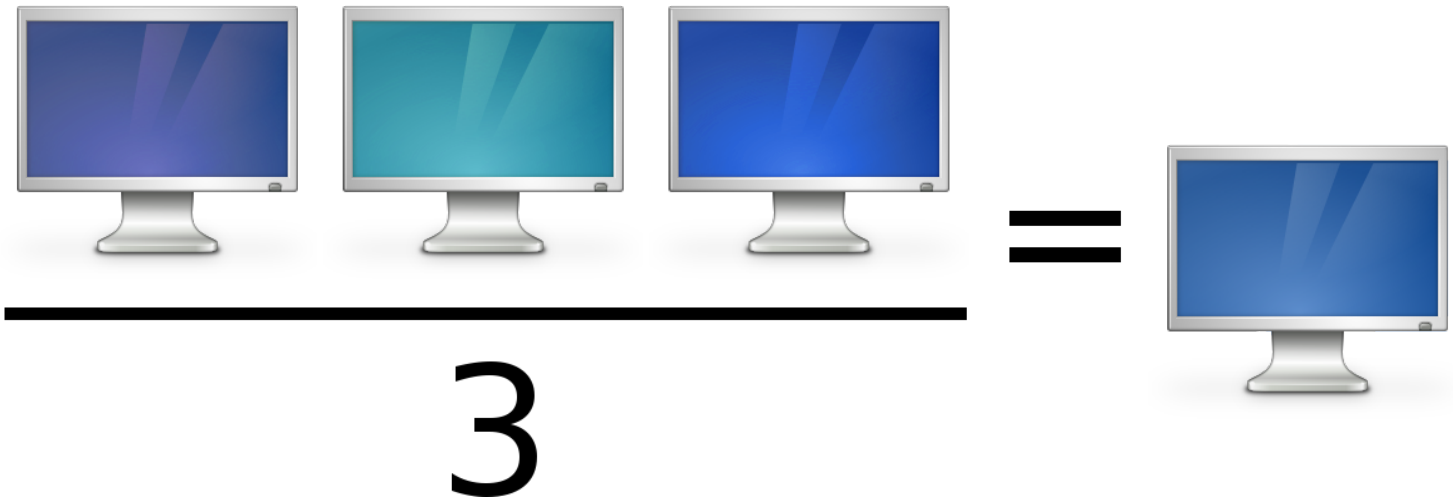


The basic problem

- Camera (14bit RAW RGB), Display (8bit PNG sRGBish), Printer (CMYK)

ICC Profiles

- Set of data that characterizes a device or color space
- Generic profiles are bad...



- "End to end color managed workflow"

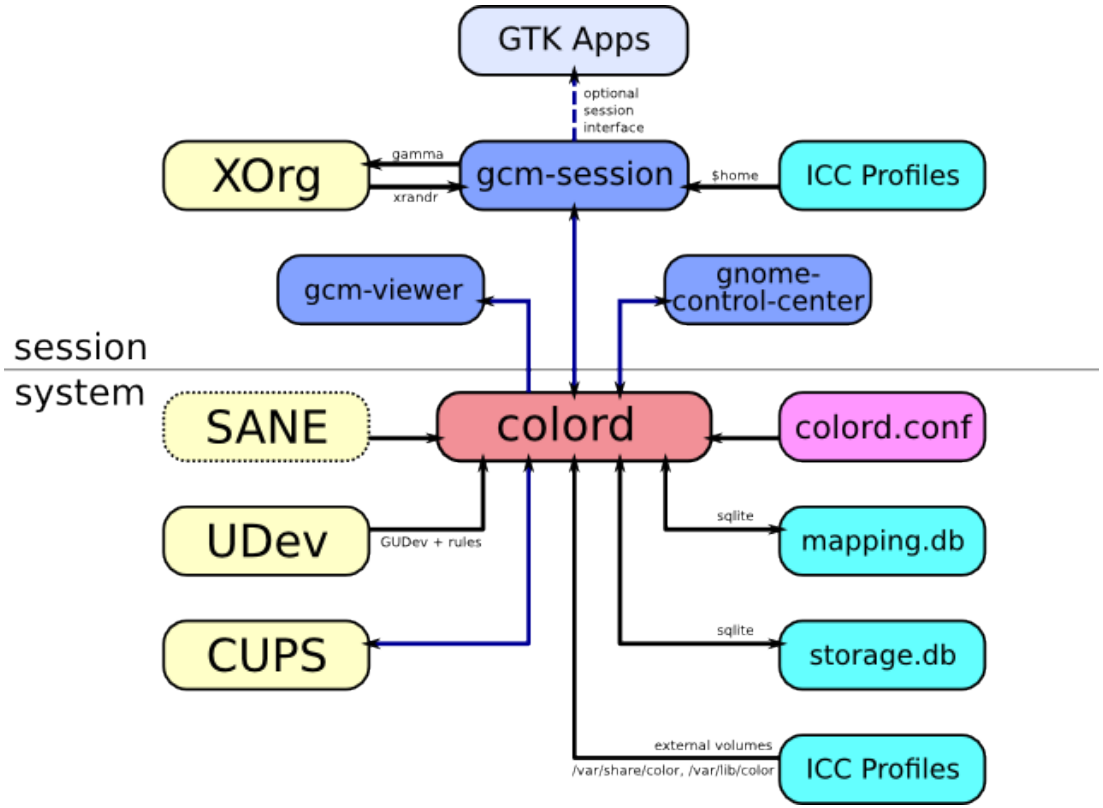
OSX and Windows 7

- OSX has color management turned on by default
- Windows support pretty half-arsed

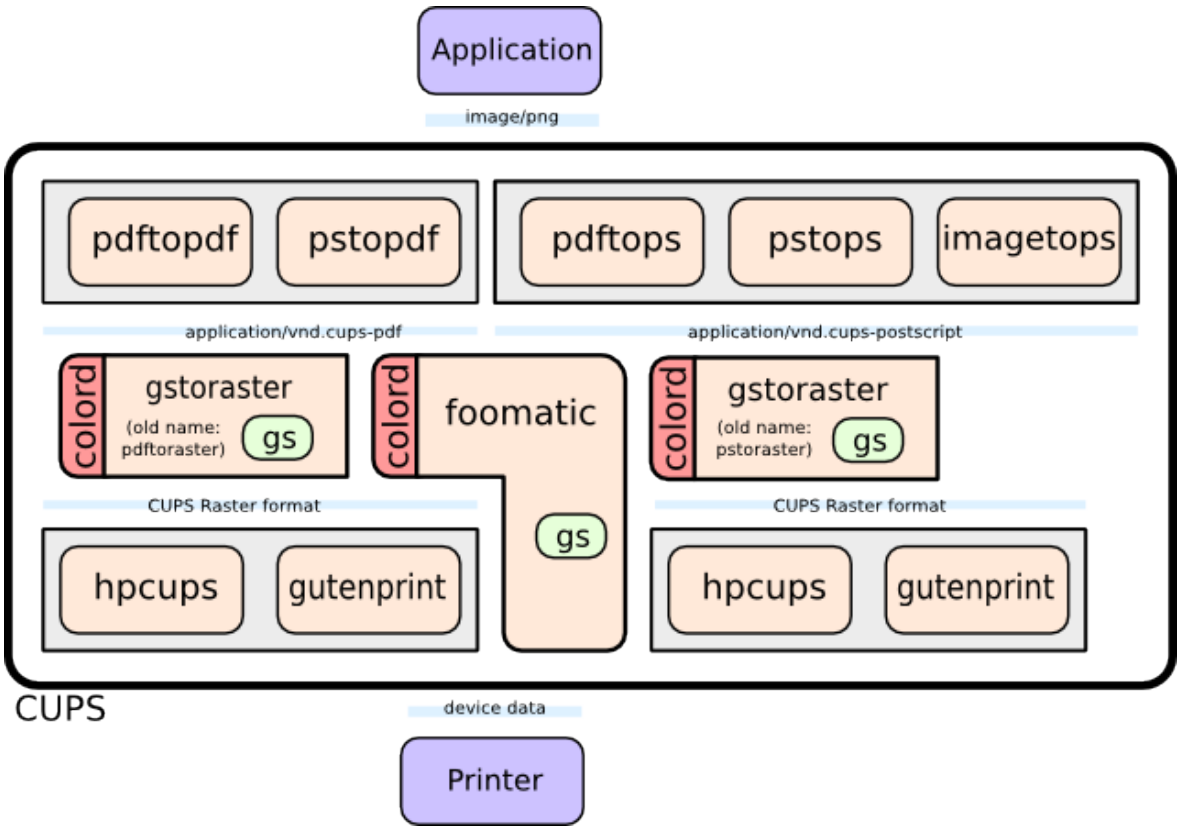
What is colord: High Level Architecture

- Really, only dealing with device to profile mapping.
- Provides a DBus API for system frameworks to query, e.g. *"Get me the profiles for device* `foo`*"* or *"Create a device* `bar` *and assign it profile* `baz`*"*.
- Provides a persistant database backed store that is preserved across reboots.
- Provides the session for a way to set system settings, for instance setting the display profile for all users and all sessions.

What is colord: High Level Architecture



What is colord: Printing Architecture



Key Concepts: Qualifiers

- `RGB.Glossy.300dpi`
- `RGB.Glossy.*`
- `RGB.*.*`
- Already defined by Apple for use in CUPS

Key Concepts: hard and soft relationships

- hard = user set mapping, and default
- soft = autogenerated mapping, and used as fallback
- examples...

Key Concepts: DeviceId and 'object paths'

- DeviceId is unique to the device, e.g. `xrandr-LVDS1`
- Object path is an actual remote DBus object for the device, e.g. `/org/freedesktop/ColorManager/devices/xrandr_LVDS1`
- So we can use any language with a DBus binding to interact with colord devices and profiles: C, Python, Perl, PHP, Pascal, Ruby, Scheme, Haskell, OCaml, Java, Javascript, C#, C++, Qt4, etc.

How It Works: Overview

- System daemon
- System activated when required
- PolicyKit to control access to privileged operations
- One SQLite database for the persistent device to profile mappings
- One SQLite database for the virtual devices

Licensing: The Basics

- daemon is `GPLv2+`
- libcolord (requires GObject and GIO) is `LGPLv2+`
- DBus interface has no ‘linking’, so possible for use in proprietary software

How To Use colord: Using the CLI

- The colormgr program allows you to interact with colord on the command line

```
[hughsie@hughsie-work ~]$ colormgr get-devices
Object Path: /org/fdo/ColorManager/devices/Photosmart_B109a_m
Created: 1296825095
Kind: printer
Model: Photosmart b109a-m
Vendor: HP
Serial: 123456789
Colorspace: rgb
Device ID: Photosmart-B109a-m
Profile 1: /org/fdo/ColorManager/profiles/HP 109 (custom).icc
```

How To Use colord: Using the CLI (2)

- Designed for administrators...

```
[hughsie@hughsie-work ~]$ colormgr --help
```

- `create-device`, `create-profile`, `delete-device`, `delete-profile`, `device-add-profile`, `device-get-default-profile`, `device-get-profile-for-qualifier`, `device-make-profile-default`, `device-set-kind`, `device-set-model`, `device-set-serial`, `device-set-vendor`, `find-device`, `find-profile`, `get-devices`, `get-devices-by-kind`, `get-profiles`, `profile-set-filename`, `profile-set-qualifier`

How To Use colord: libcolord

- The libcolord gobject library wraps the DBUS interface in a nice GLib-style API.

```
CdClient *client;
GError *error;
GPtrArray *devices;
client = cd_client_new ();
devices = cd_client_get_devices_sync (client,
                                      cancellable,
                                      &error);
g_object_unref (client);
g_ptr_array_unref (devices);
```

How To Use colord: raw DBus

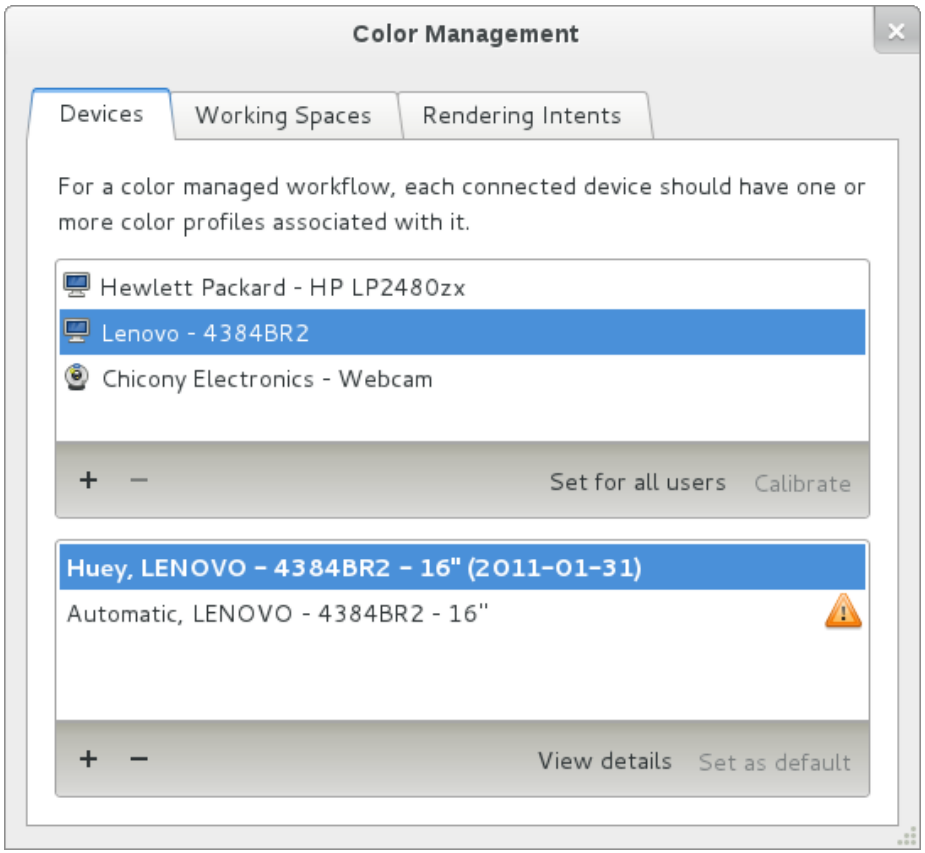
- Using the DBUS methods and signals directly means that no GObject dependency is needed.

```
con = dbus_bus_get(DBUS_BUS_SYSTEM, NULL);
message = dbus_message_new_method_call("org.freedesktop.ColorManager",
                                       "/org/freedesktop/ColorManager",
                                       "org.freedesktop.ColorManager",
                                       "CreateDevice");
dbus_message_iter_init_append(message, &args);
dbus_message_iter_append_basic(&args, DBUS_TYPE_STRING, &device_id);
dbus_message_iter_append_basic(&args, DBUS_TYPE_STRING, "temp");
reply = dbus_connection_send_with_reply_and_block(con,
                                                  message,
                                                  -1,
                                                  &error);
```

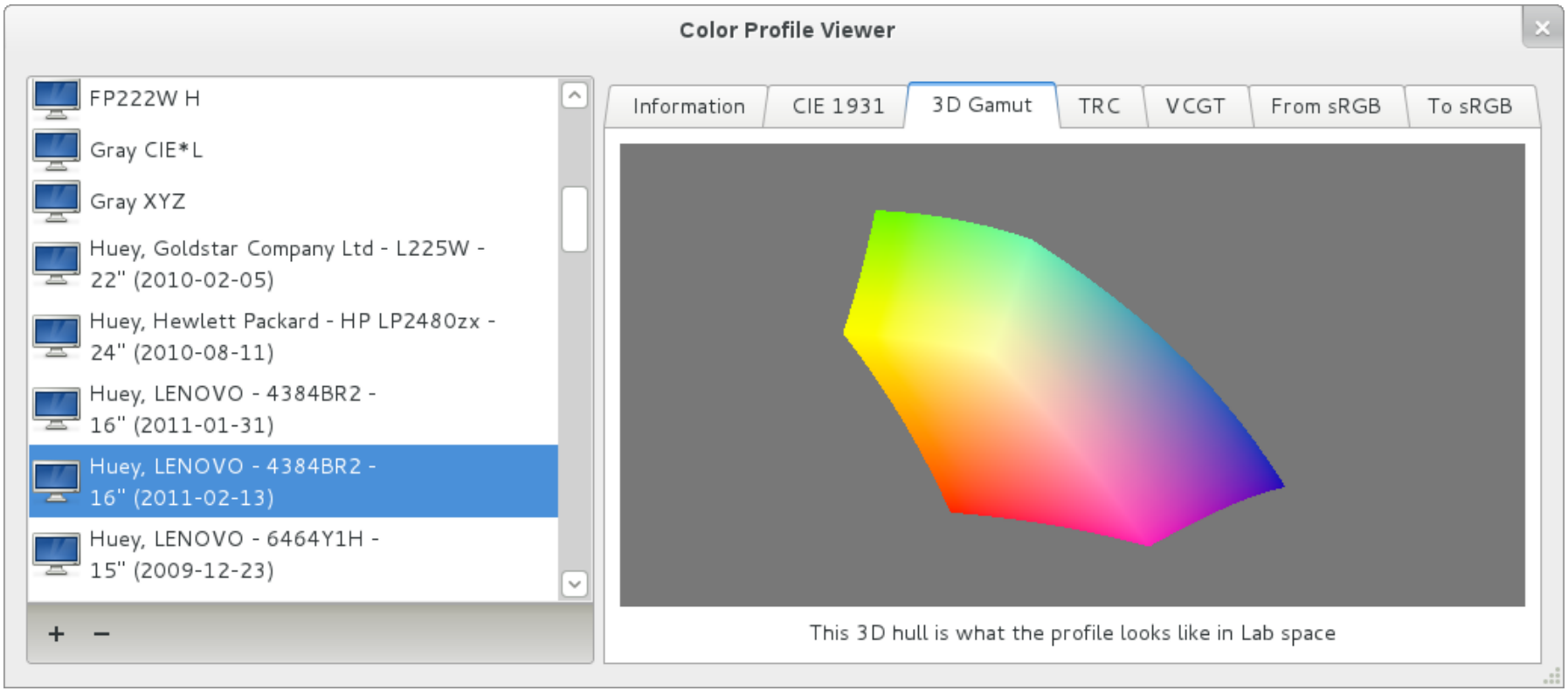
What does GNOME Color Manager do?

- Call `CreateDevice` for each connected XRandr screen.
- Create an ICC profile file for each Xrandr device using the EDID (optional).
- Call `CreateProfile` for each profile found in the home directory.
- For each `::profile-added` event check if the `EDID_md5` metadata matches.
- For each `::device-added` event check the device modified property (optional).
- For each `::device-added` event from a Xrandr device, send the gamma ramp to X.

Preferences



Profile Viewer



Recap

- Working spaces and gamut
- Per-device ICC profiles
- xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
- GNOME Color Manager UI

colord questions

- Thanks to Red Hat for sponsoring development
- I'm **expecting** questions!