Best Practices for Color Management

What you need to know about color on OS X and iOS

Session 523

Ken Greenebaum and Luke Wallis
Graphics and Imaging

Introduction to Color Management

What You Will Learn

- How color is managed on iOS and Mac OS X
  - Implication for your applications
- How to
  - Control color using high and low-level frameworks
  - Create and modify video/image content
  - Verify the results

Introduction

Introduction

- Apple color manages video, still image, graphics
  - Consistent high quality results
    - Across devices and environments
  - Preserves ‘author’s intent’
  - Not just for pros
    - Great for content authoring and consumption

Introduction

- Apple color manages video, still image, graphics
  - Consistent high quality results
    - Across devices and environments
  - Preserves 'author's intent'
  - Not just for pros
    - Great for content authoring and consumption
- The rest of the industry largely does not
  - Some high end drawing or photo packages
  - Video industry instead relies on 'Broadcast' displays in consistent environments

Color Management Philosophy

- Film, images, media are creative endeavors
  - Camera != Colorimeter
  - Not *scene referred*
- We attempt to reproduce 'author's intent'
  - What is proofed
  - *Output (display) referred*
- Content is reproduced on different devices and environments
  - Requiring color matching, gamma conversion, etc.

Creative Endeavor

Creative Endeavor

Creative Endeavor

Bright sunlit environment

Creative Endeavor

Bright sunlit environment

Creative Endeavor

Cinematographer changing knobs

Bright sunlit environment

Display Referred Environment

The author's intent

Display Referred Environment

The author's intent

Bright sun

Display Referred Environment

The author's intent

Broadcast monitor

Bright sun

Display Referred Environment

The author's intent

Display Referred Environment

The author's intent

Display Referred Environment

The author's intent

Display Referred Environment

The author's intent

Display Referred Environment

The author's intent

Rendering Environment

Modern usage

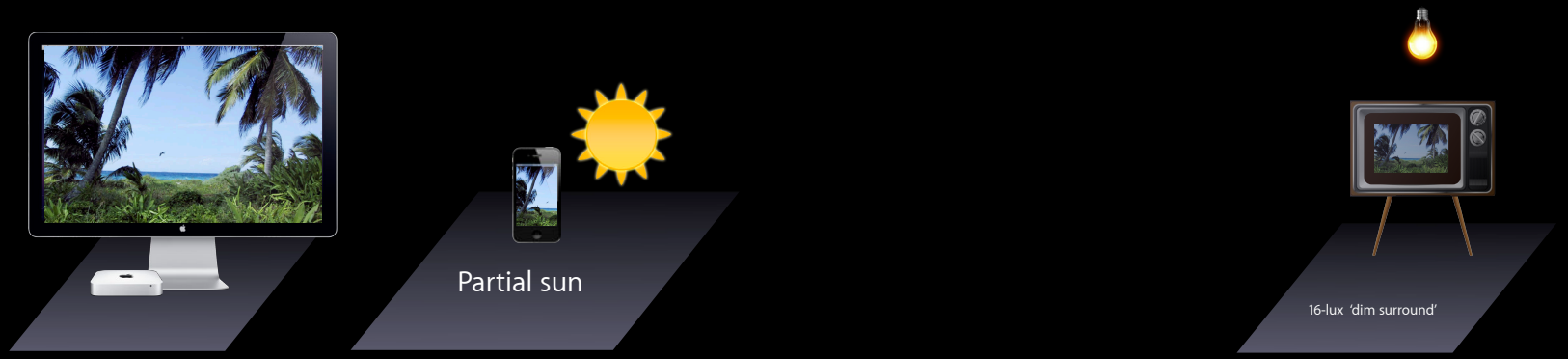
!=

Rendering Environment

Modern usage

Rendering Environment

Modern usage



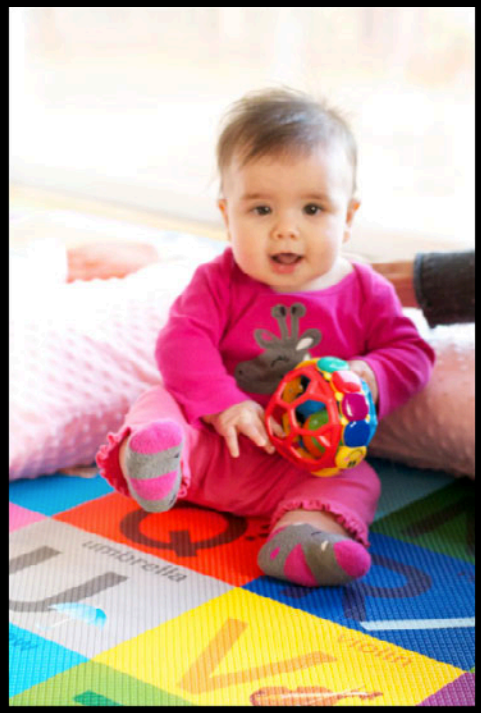
Rendering Environment

Modern usage

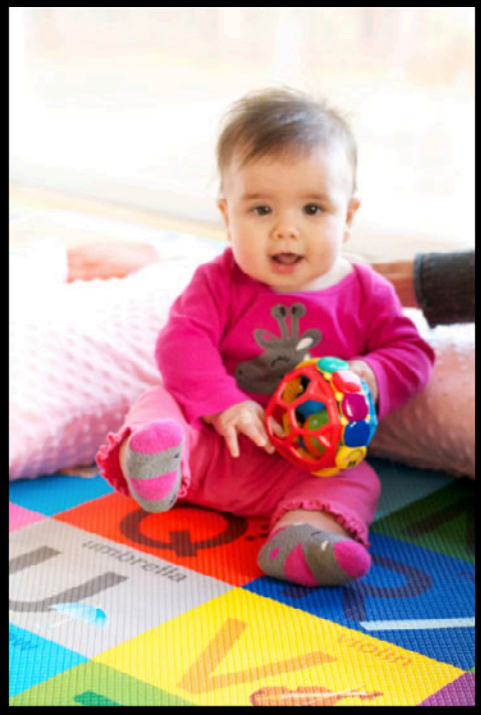
Why Color Manage

Why Color Manage

- ProPhoto JPEG
- Preview
- Color Managed



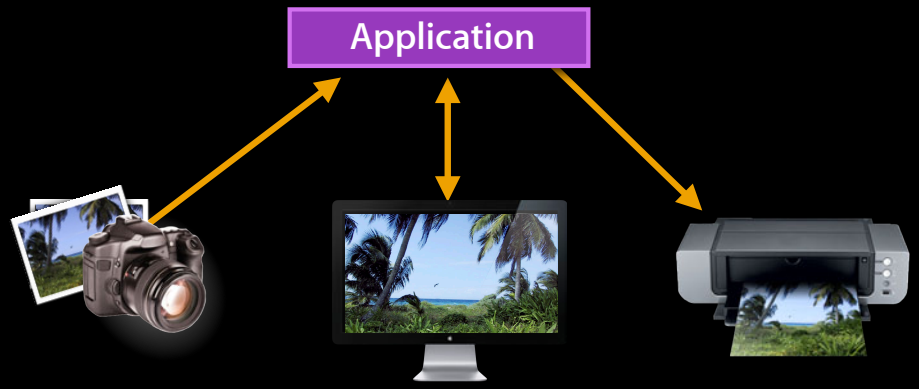
Why Color Manage



Controlling Color Conversions

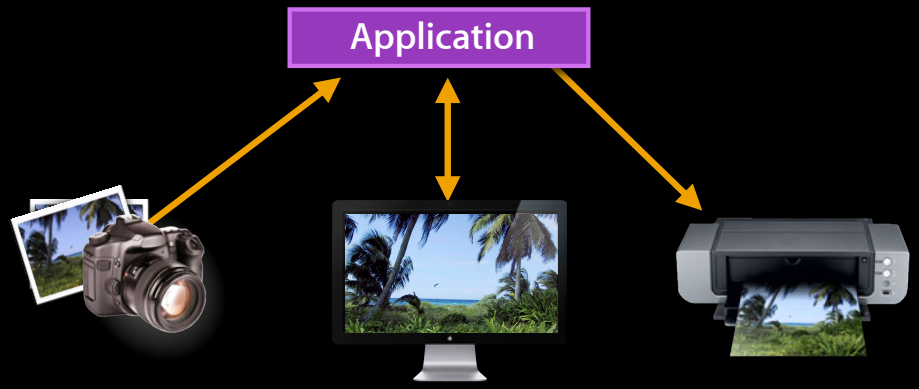
Color Management Workflows

Consistent color appearance across devices



Color Management Workflows

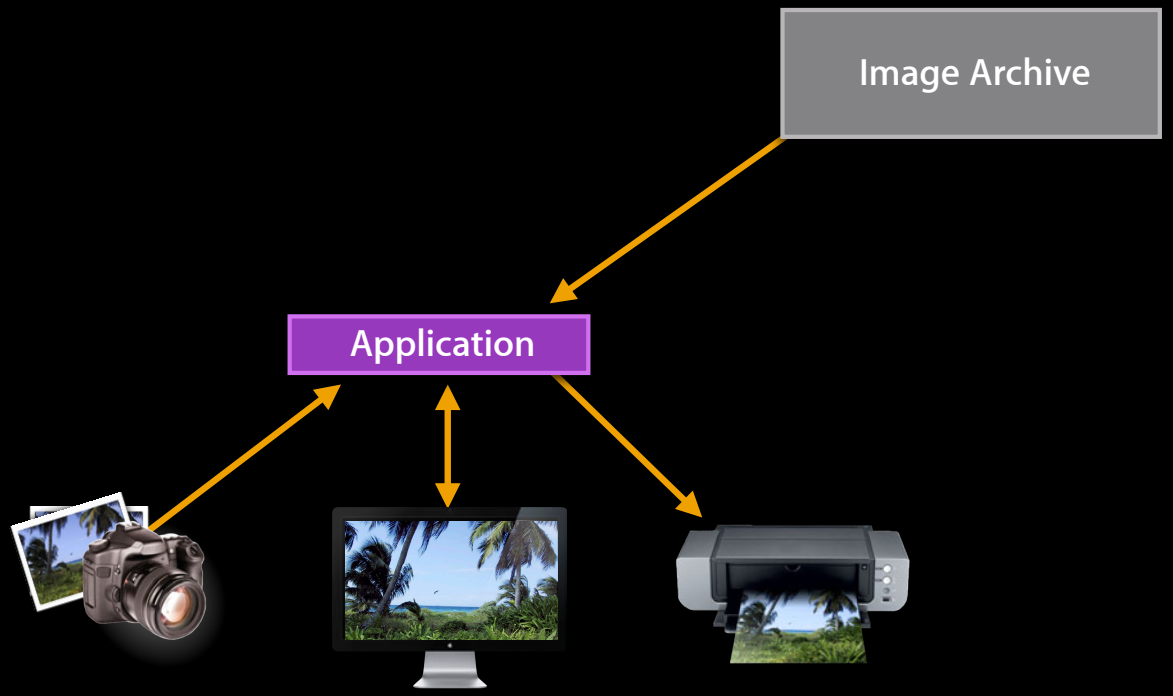
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

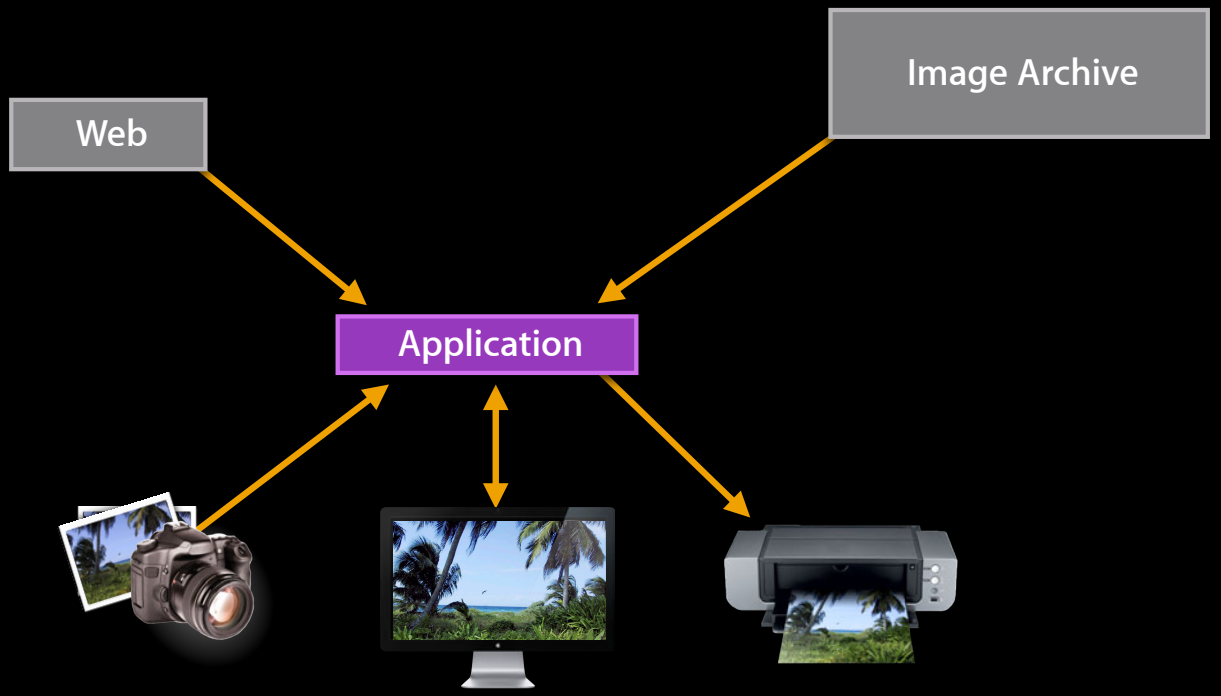
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

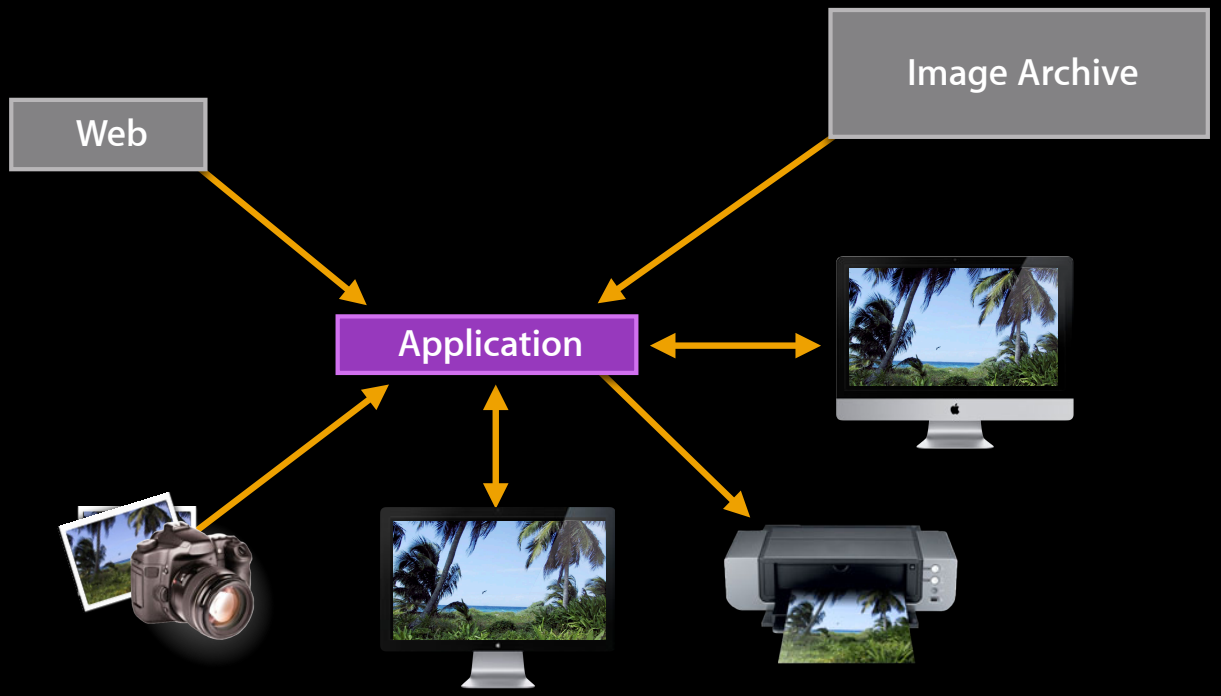
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

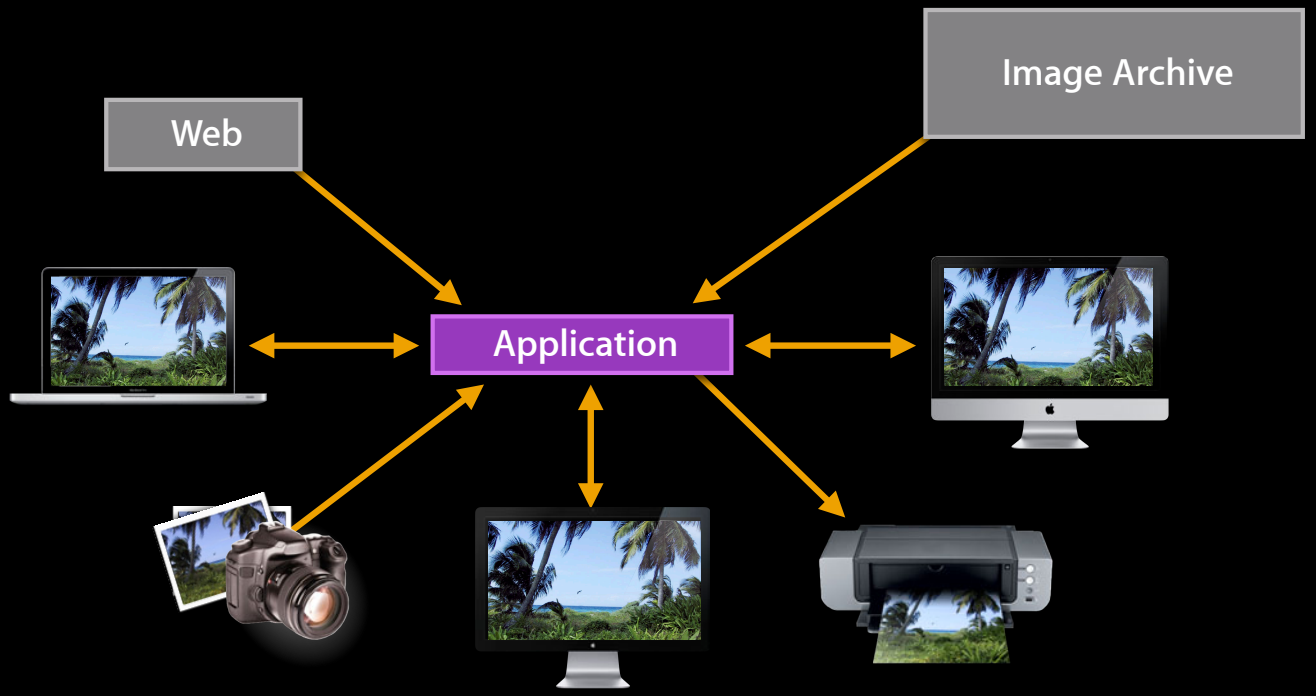
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

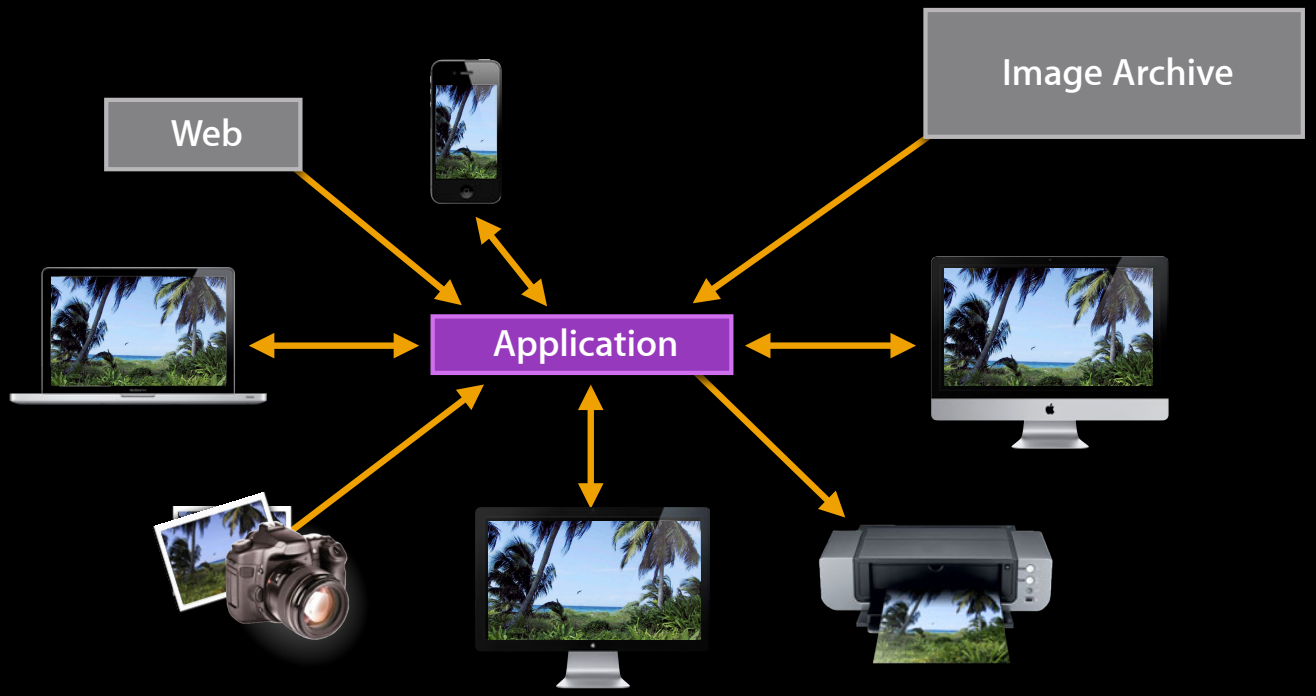
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

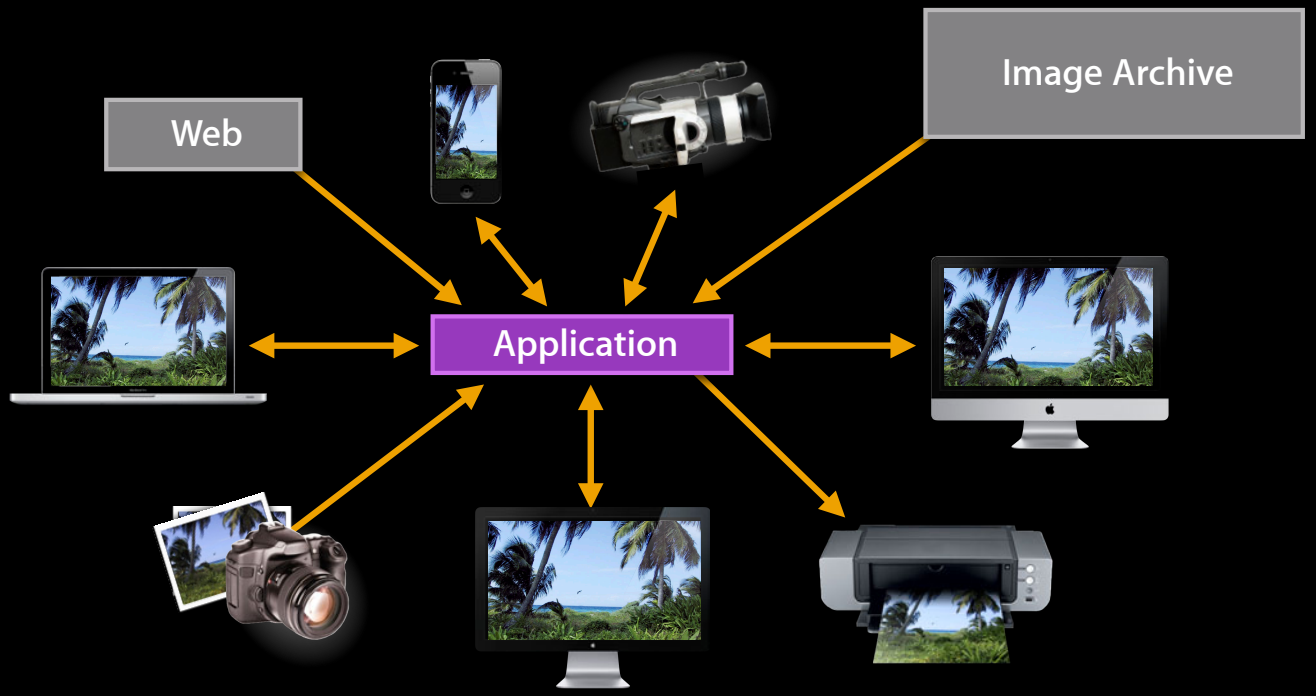
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Color Management Workflows

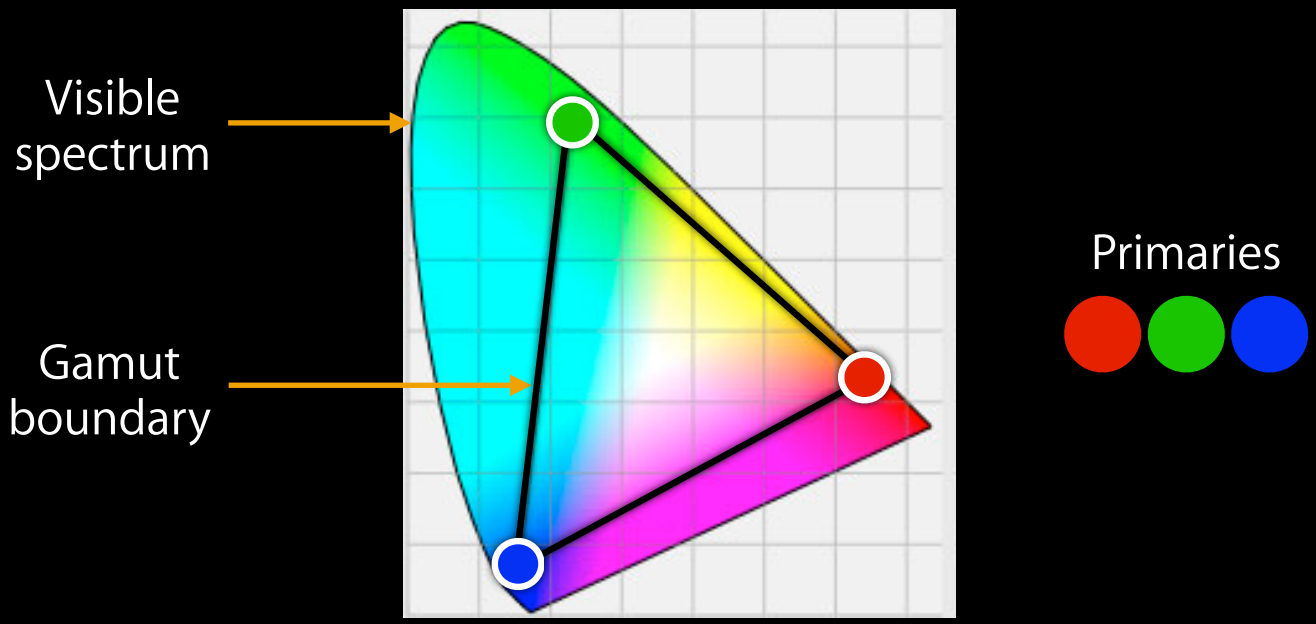
Consistent color appearance across devices



Color Management = Controlled conversion between different color representation

Device Color Capabilities

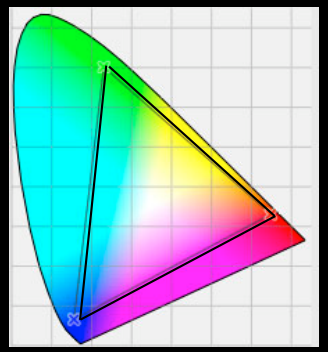
Device gamut



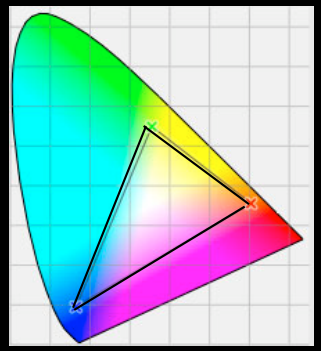
Adobe RGB—Typical color space of a DSLR camera

Problem to Solve

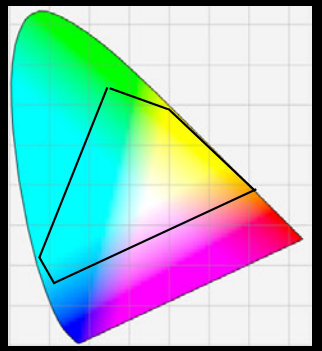
Reproducing color on different device



Adobe RGB Camera



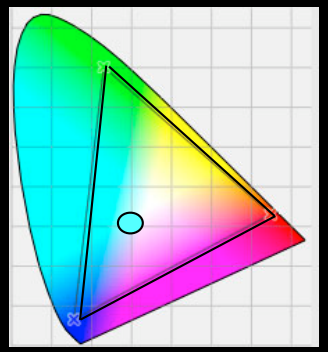
Laptop Display



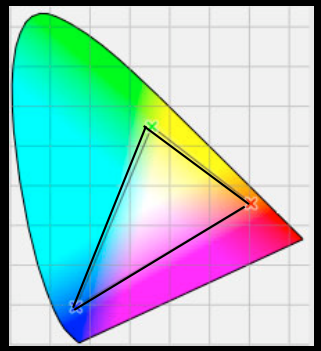
Inkjet Printer

Problem to Solve

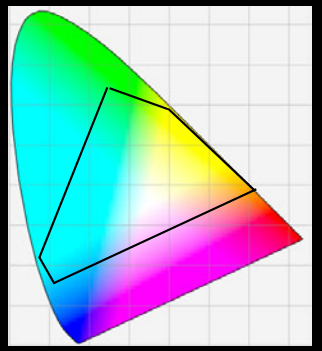
Reproducing color on different device



Adobe RGB Camera



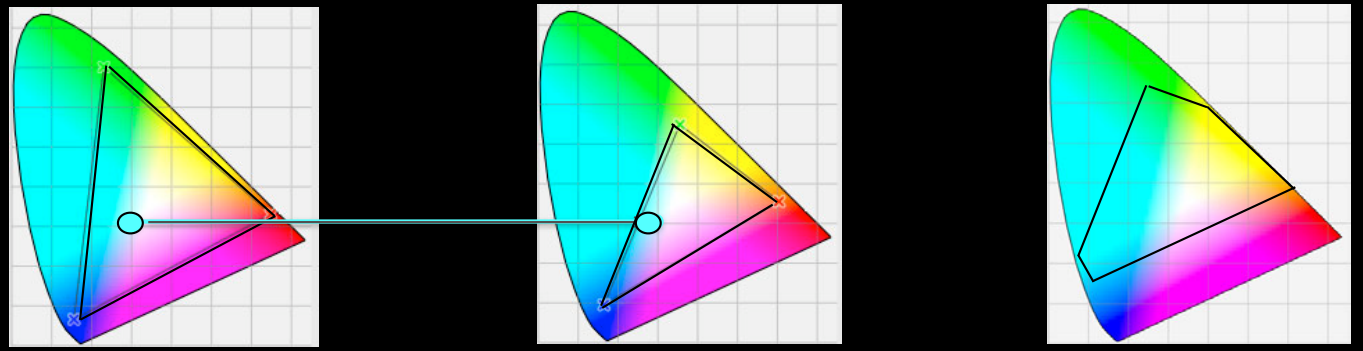
Laptop Display



Inkjet Printer

Problem to Solve

Reproducing color on different device



Adobe RGB Camera

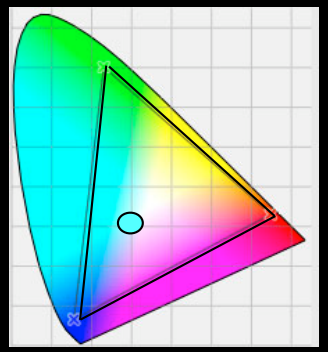
Laptop Display

Inkjet Printer

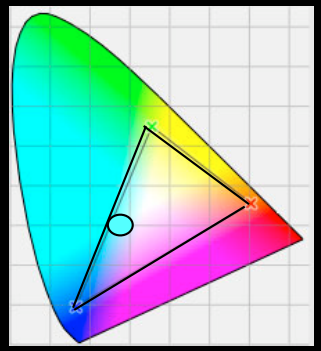
- In gamut, no color shift

Problem to Solve

Reproducing color on different device

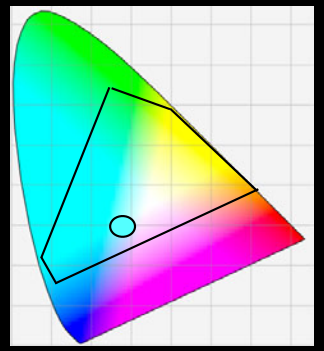


Adobe RGB Camera



Laptop Display

- In gamut, no color shift

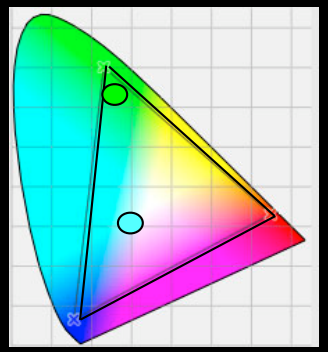


Inkjet Printer

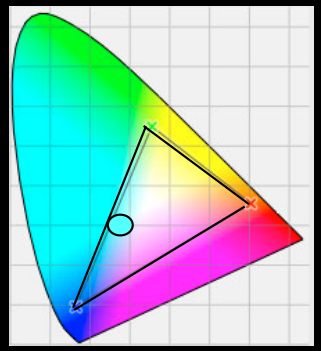
- In gamut, no color shift

Problem to Solve

Reproducing color on different device

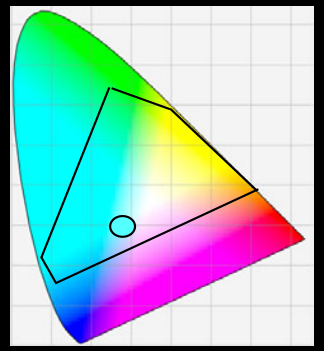


Adobe RGB Camera



Laptop Display

- In gamut, no color shift

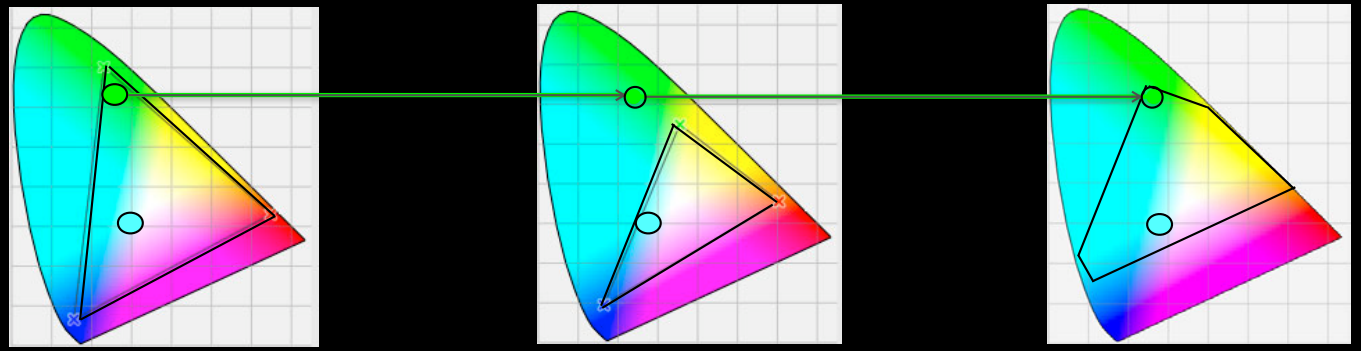


Inkjet Printer

- In gamut, no color shift

Problem to Solve

Reproducing color on different device



Adobe RGB Camera

Laptop Display

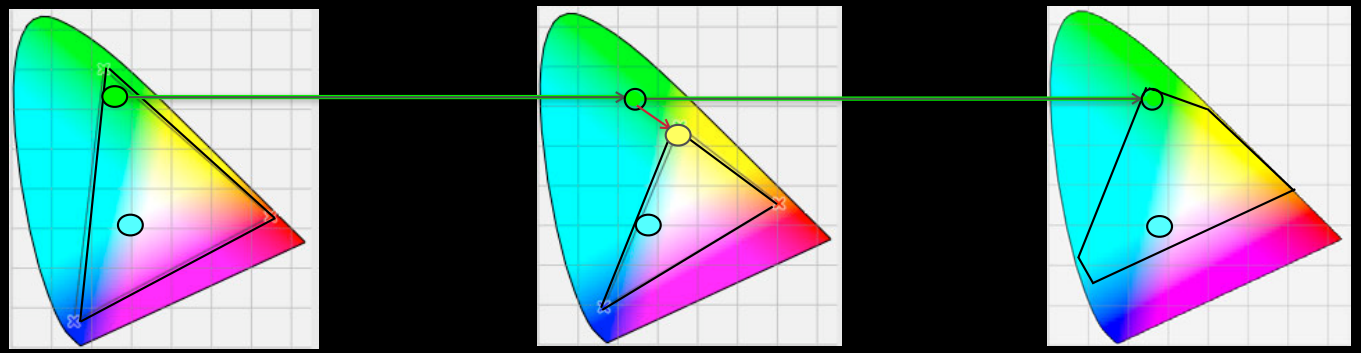
Inkjet Printer

- In gamut, no color shift

- In gamut, no color shift

Problem to Solve

Reproducing color on different device



Adobe RGB Camera

Laptop Display

Inkjet Printer

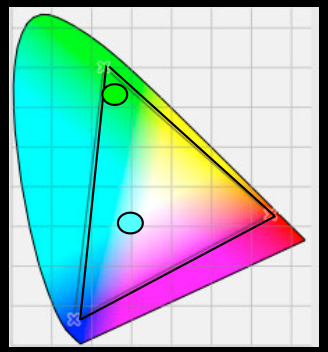
- In gamut, no color shift

- Significant color shift, different appearance

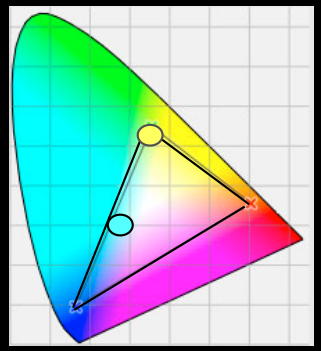
- In gamut, no color shift

Problem to Solve

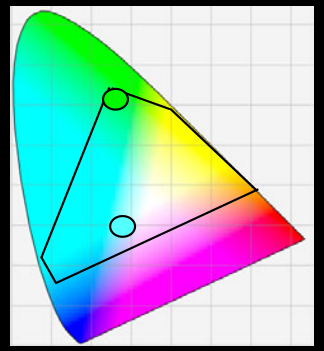
Reproducing color on different device



Adobe RGB Camera



Laptop Display



Inkjet Printer

- In gamut, no color shift

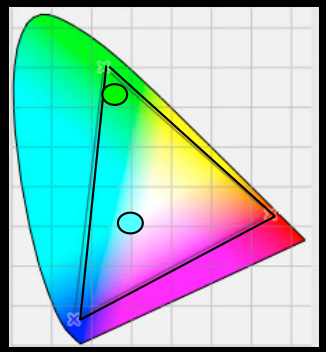
- Significant color shift, different appearance

- In gamut, no color shift

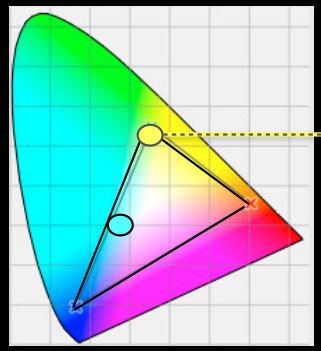
- Small color shift, similar appearance

Problem to Solve

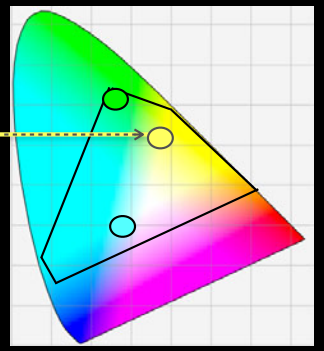
Reproducing color on different device



Adobe RGB Camera



Laptop Display

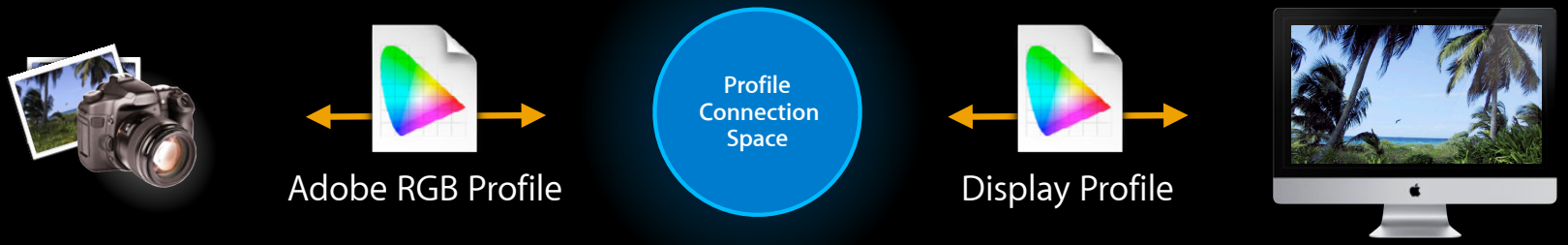


Inkjet Printer

- In gamut, no color shift
- Significant color shift, different appearance
- Conversion to a smaller gamut can cause irreversible damage

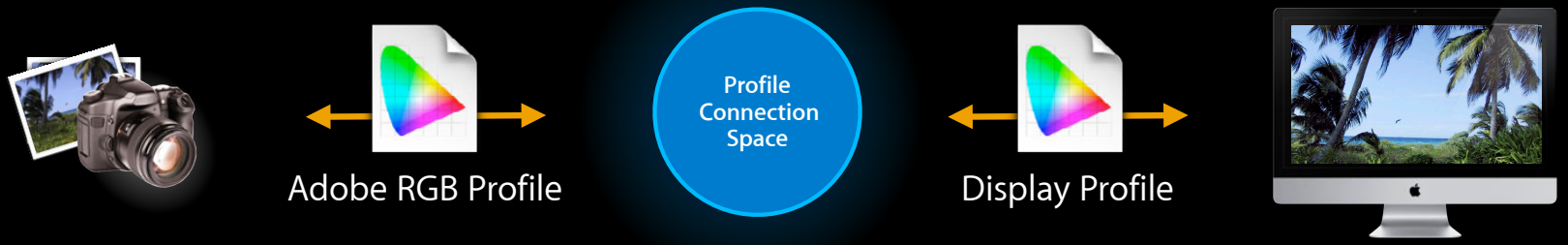
- In gamut, no color shift
- Small color shift, similar appearance

ICC Profiles Control the Match



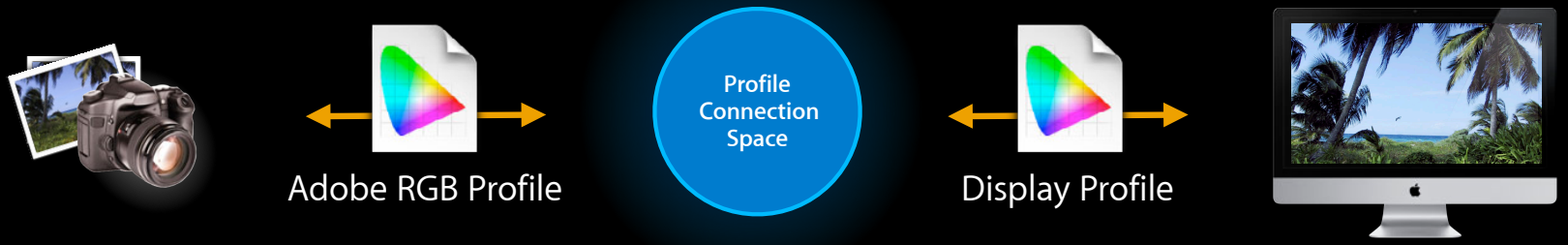
ICC Profiles Control the Match

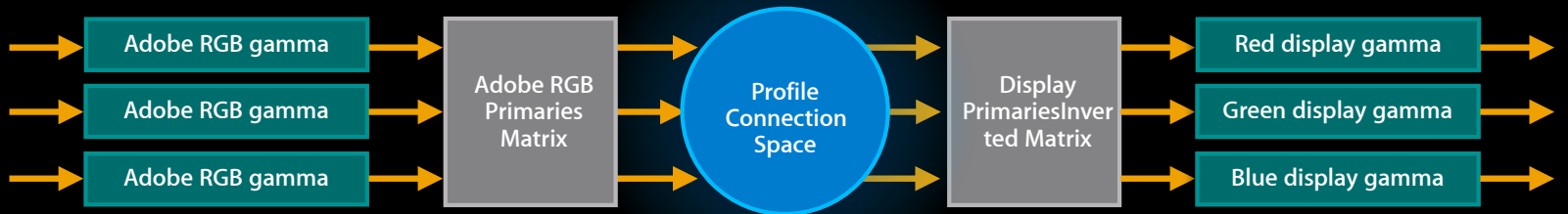
Transform created from source and destination profiles



ICC Profiles Control the Match

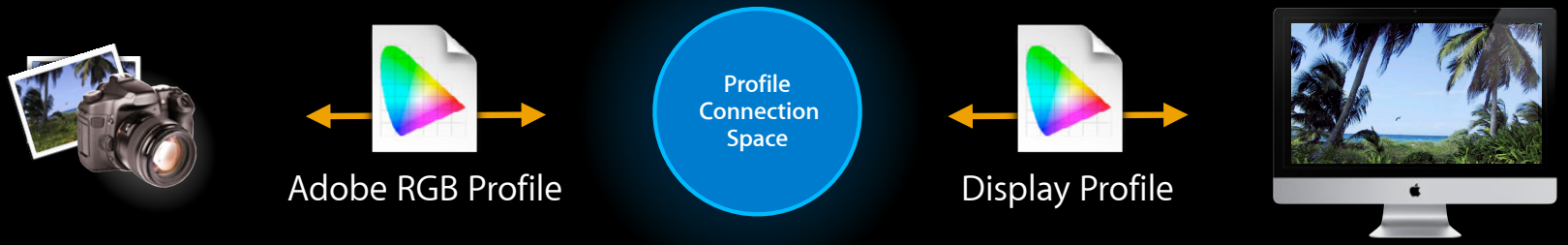
Transform created from source and destination profiles

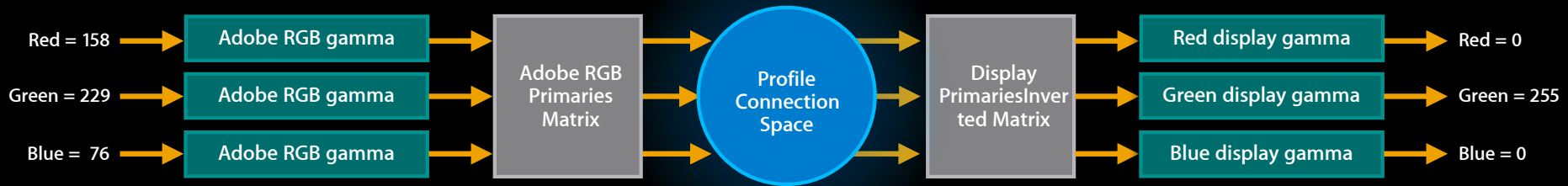




ICC Profiles Control the Match

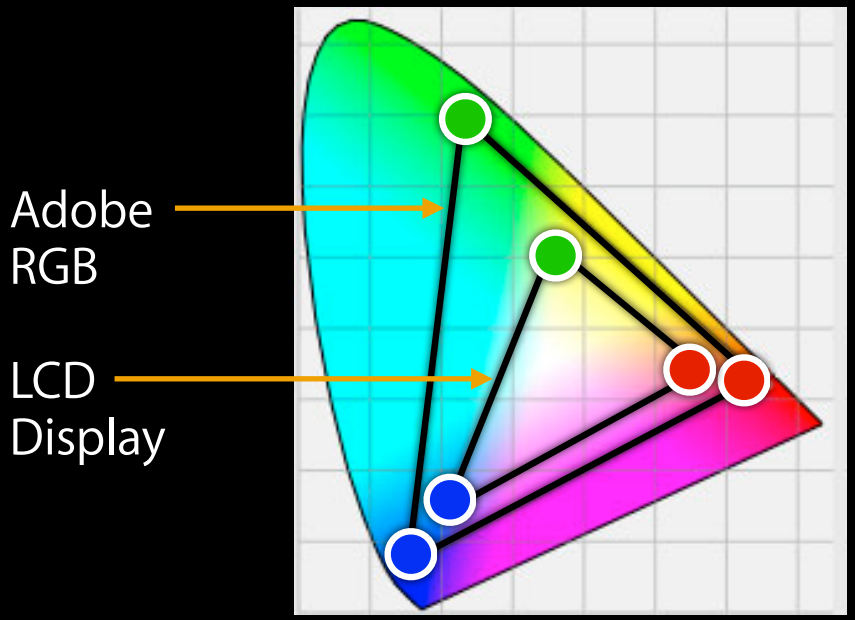
Transform created from source and destination profiles





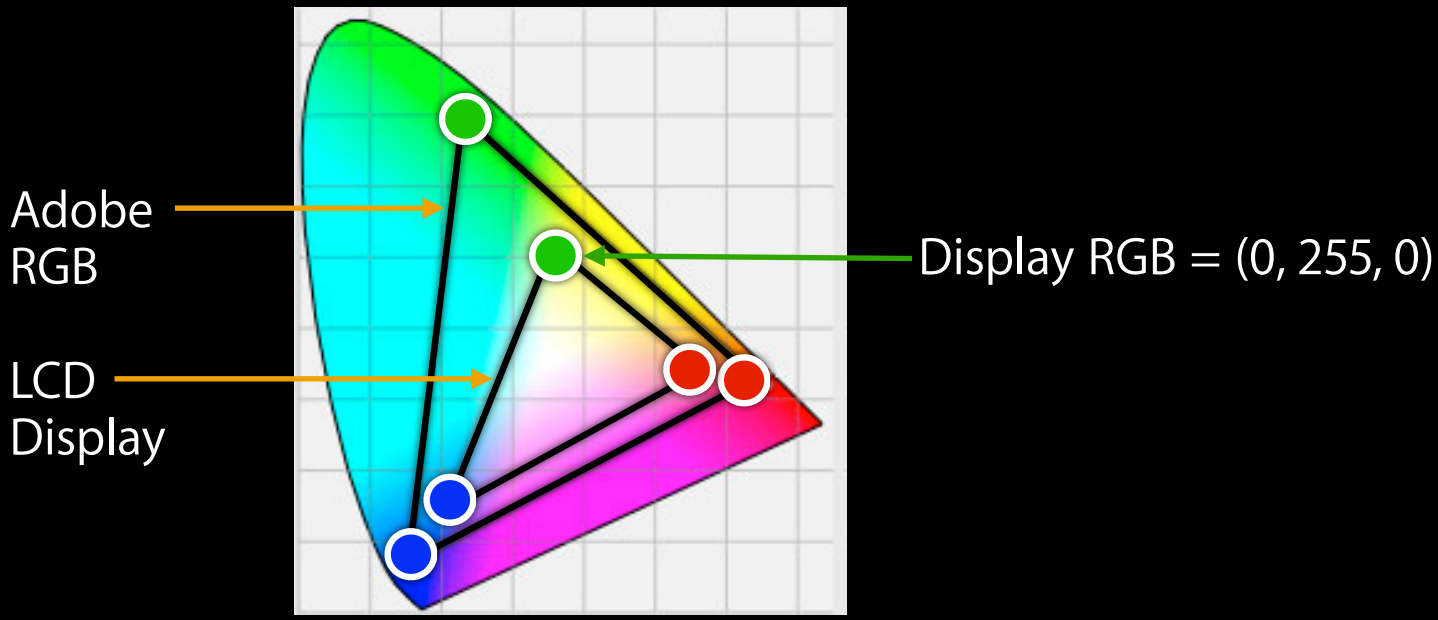
Color Values in Perspective

When RGB != RGB



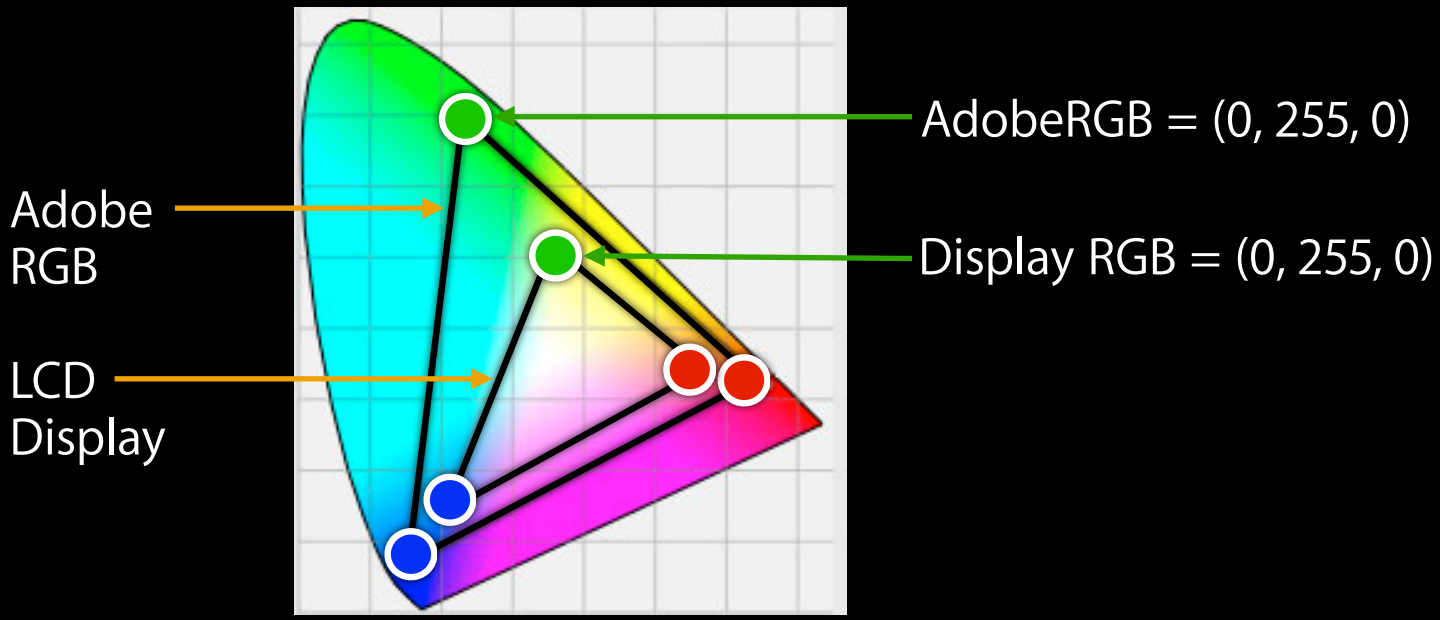
Color Values in Perspective

When RGB != RGB



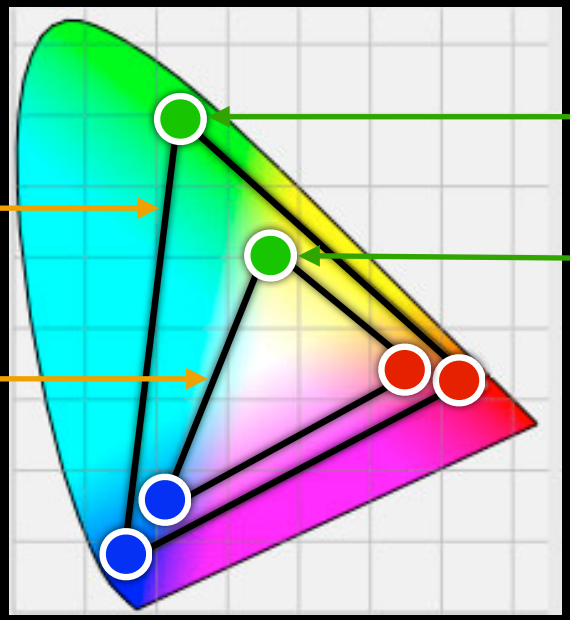
Color Values in Perspective

When RGB != RGB



Color Values in Perspective

When RGB != RGB



Adobe RGB

LCD Display

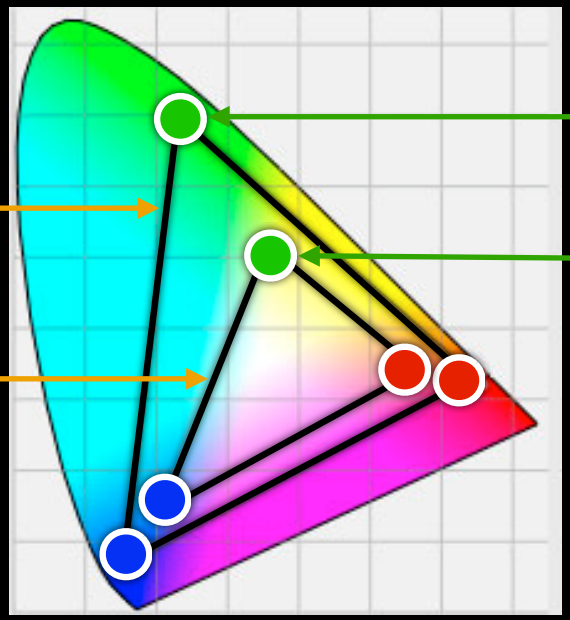
AdobeRGB = (0, 255, 0)

Display RGB = (0, 255, 0)

Display RGB (0, 255, 0) = (158, 229, 76) in Adobe RGB

Color Values in Perspective

When RGB != RGB



Adobe RGB

LCD Display

AdobeRGB = (0, 255, 0)

Display RGB = (0, 255, 0)

Display RGB (0, 255, 0) = (158, 229, 76) in Adobe RGB

Color values meaningful only if tagged with proper profile!

Active and Targeted Color Management

Active Color Management

- Real-time, per-pixel, per-frame
- Color Match
  - From Content's (Source) Profile
  - To Display's (Destination) Profile
    - Possibly via Working Space or intermediate Profile
  - Both gamut and gamma match applied

- Still image can be CPU based
- Video/Graphics must be GPU (or HW) accelerated

Targeted Color Management

- One target color space is used
- Media is color matched at authoring
- Hardware is used to ensure display's fidelity
- Similar to video distribution targeting SD or HD video

Targeted Color Management

- One target color space is used
- Media is color matched at authoring
- Hardware is used to ensure display's fidelity
- Similar to video distribution targeting SD or HD video

OS X

Active color management

- Logic provided by ColorSync
  - GPU accelerated color match provided by
    - Window Server, CoreImage, or CoreAnimation
- Provides flexibility for Authoring/Proofing
  - Any content
    - Self described via ICC Profile
  - Any Display/Output attached to computer
    - Calibrated and described via ICC Profile

iOS

sRGB Targeted color management

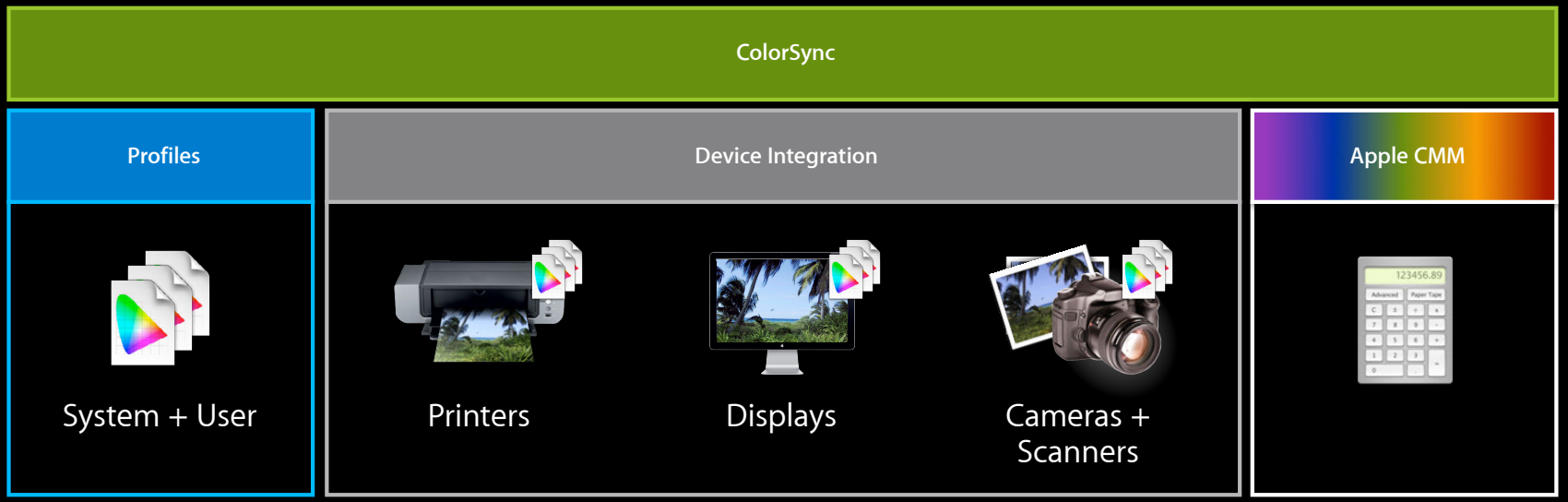
- Content matched to sRGB target colorspace
  - During authoring—on OS X desktop via ColorSync
  - During sync—in iTunes via ColorSync
- Advantages for mobile
  - Conserves power/runtime
  - Unburdens valuable resources for other processing
  - Equivalent quality result (as Active Management)

Active Color Management

OS X

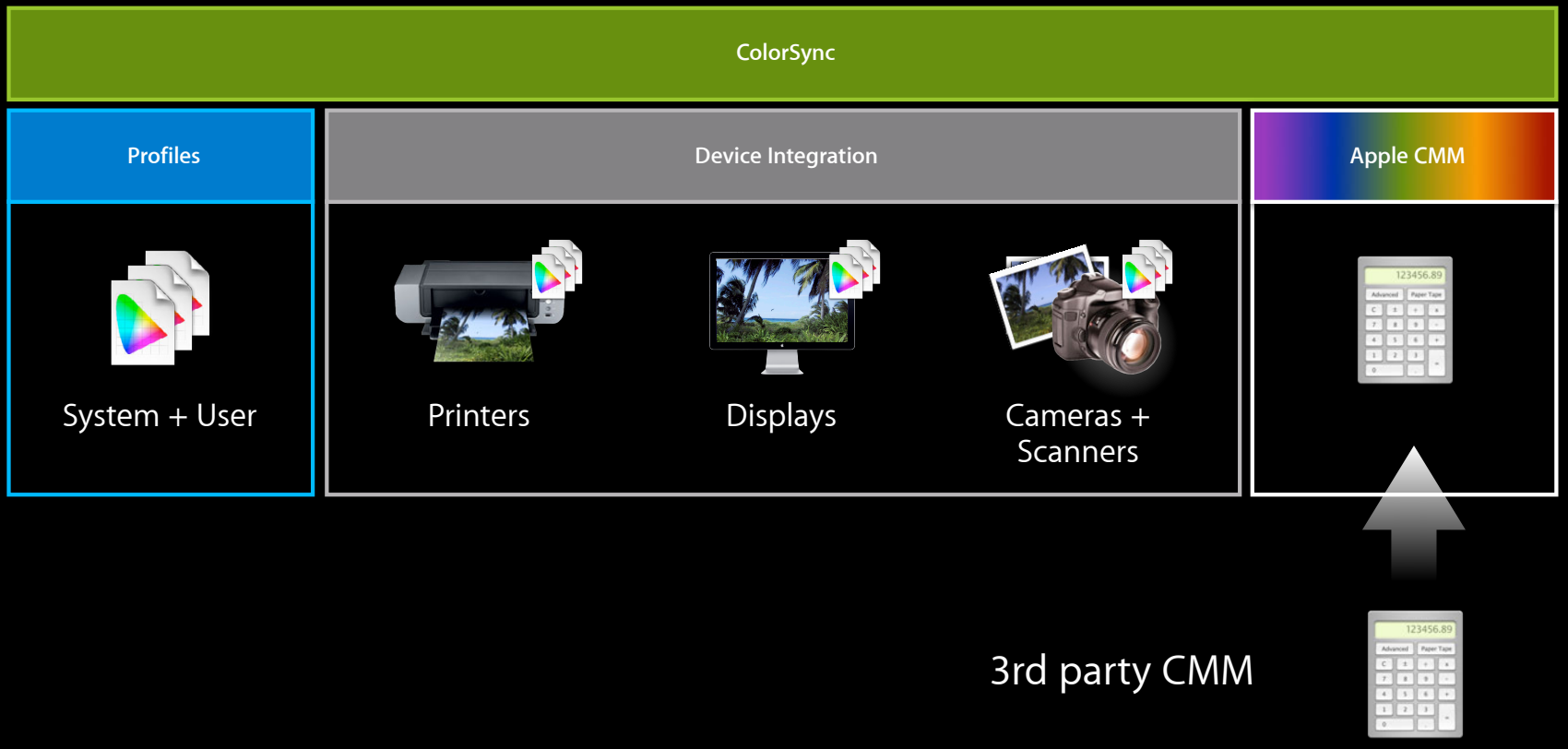
Color Management Technology

ColorSync



Color Management Technology

ColorSync



Color Managed Frameworks

Integrated with ColorSync

- Modern graphics frameworks to control graphics content
  - Quartz
  - ImageIO
  - ImageCapture
  - AVFoundation
  - Printing

- AppKit to create high level abstracts for displaying media
  - NSView, NSWindow, etc.

Quartz

CoreGraphics

- Objects pertaining to color
  - CGColorSpaceRef
  - CGColorRef
  - CGImageRef
  - CGContextRef

CGColorSpaceRef

- CGColorSpace specifies how Quartz interprets color data
  - e.g. `CGColorSpaceCreateWithName` (kCGColorSpaceSRGB)
- Can be created directly from ICC profiles
  - `CGColorSpaceCreateWithICCProfile` (...)

CGColorRef

- CGColor consists of
  - CGColorSpaceRef
  - Set of component values
- Component values correspond to color space primaries
  - e.g.

```
CGColorSpaceRef space = CGColorSpaceCreateWithName(kCGColorSpaceSRGB);
CGFloat comp[4] = {0.5, 1.0, 0.7, 1.0}; // last component is alpha
CGColorRef space = CGColorCreate(space, comp);
```

CGImageRef

- CGImage consists of CGColorSpaceRef and array of component values organized in rows and columns
- Component values correspond to color space primaries

```
CGImageRef image = CGImageCreate(width,
                                 height,
                                 ...
                                 ...
                                 colorSpace,
                                 ...
                                 renderingIntent);
```

CGContextRef

- CGContext represents drawing destination

```
CGContextDrawImage(...)
CGContextStrokeRect(...), CGContextFillRect(...)
```

- Different types:
  - With CGColorSpace
    - CGBitmapContext, CGWindowContext
  - Without CGColorSpace
    - CGPDFContext, CGPostScriptContext

CGContextRef

Automatic color management

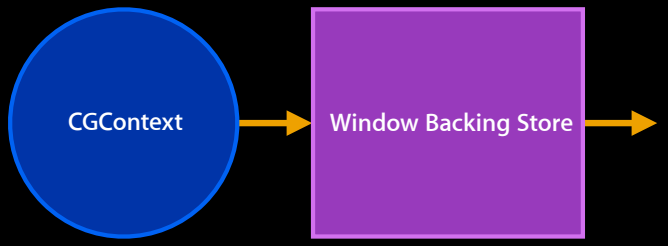
- If context's CGColorSpace != source's CGColorSpace
  - CoreGraphics invokes ColorsSync to convert color

- Reference: Quartz 2D Programming Guide
- Sample Code: ImageApp

Window Backing Store Color Space

Window Backing Store Color Space

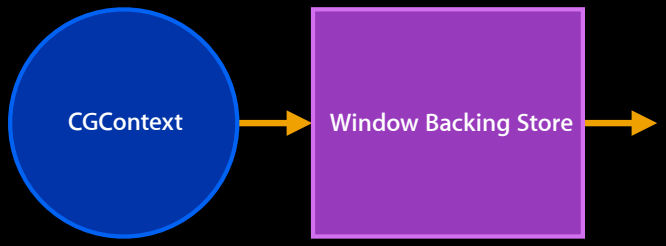
ColorSpace:
Current Display Profile



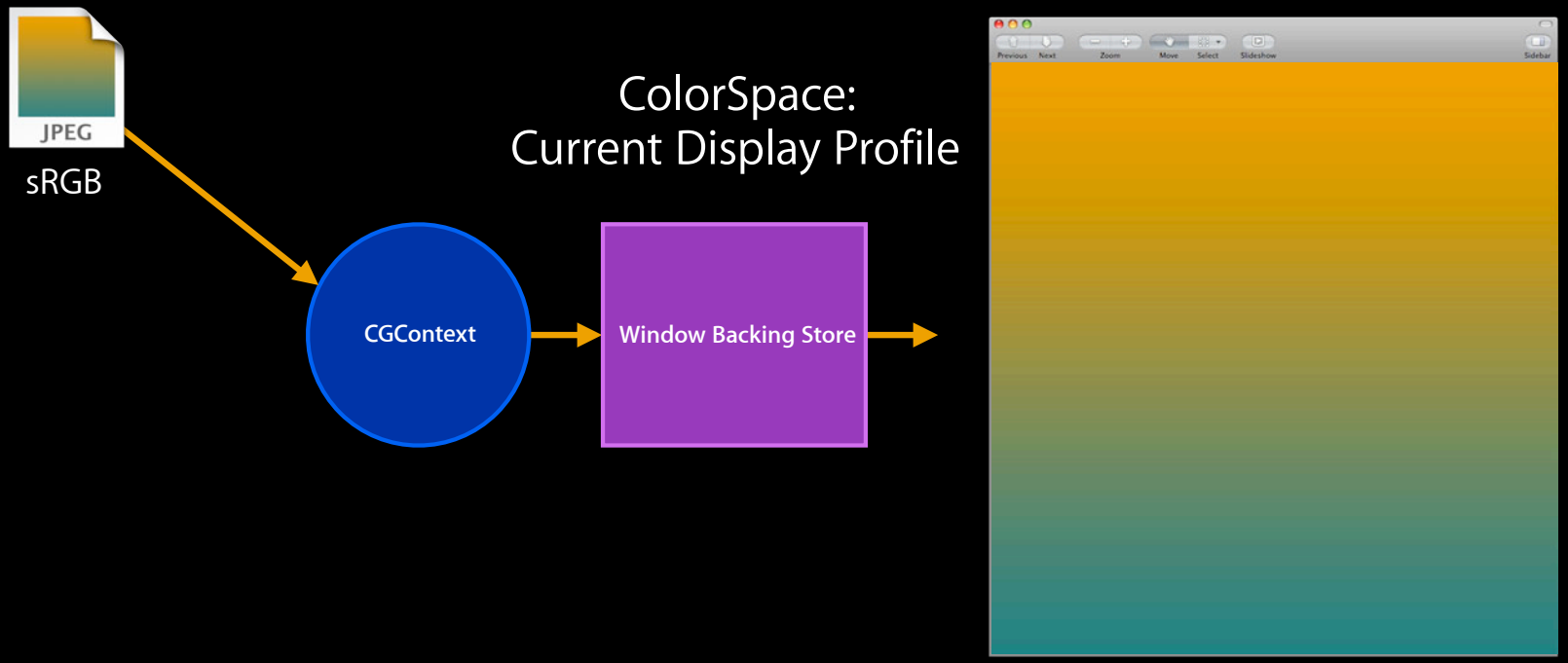
Window Backing Store Color Space

sRGB

ColorSpace:
Current Display Profile

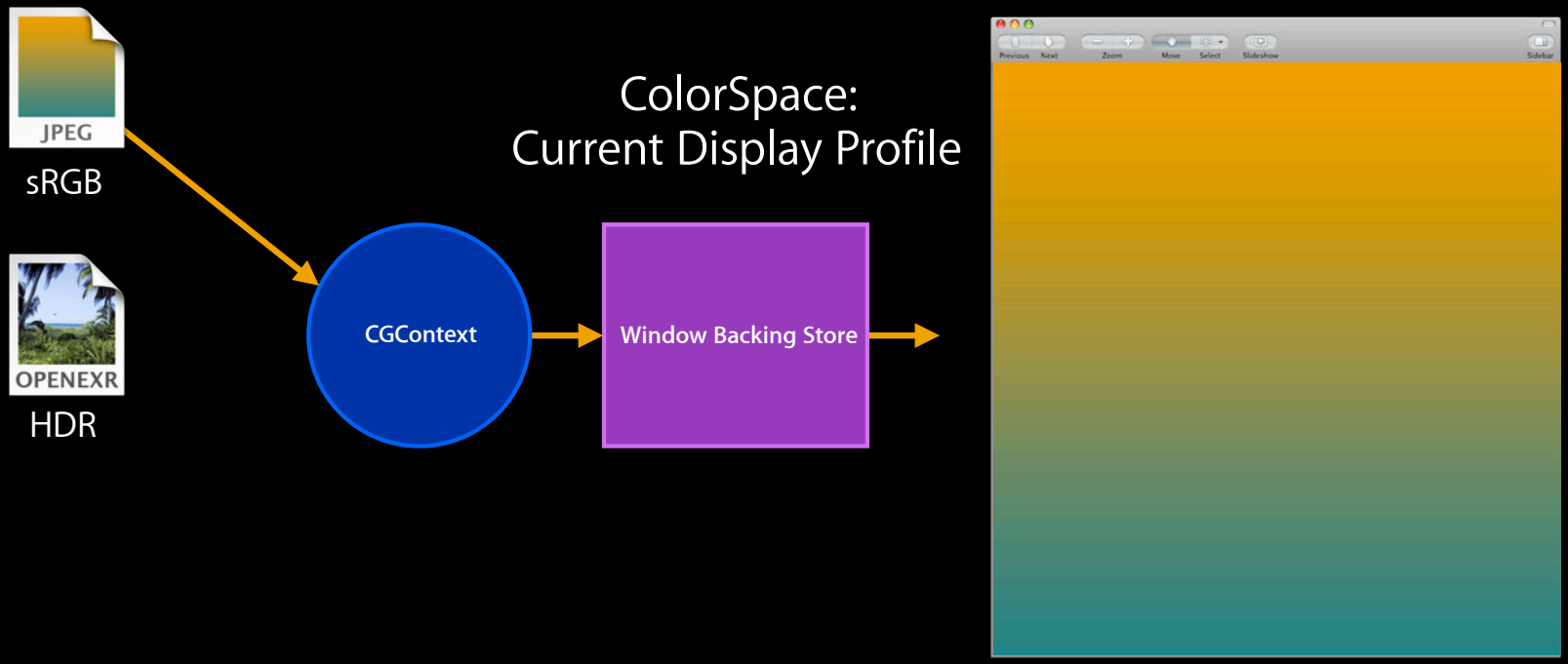


Window Backing Store Color Space



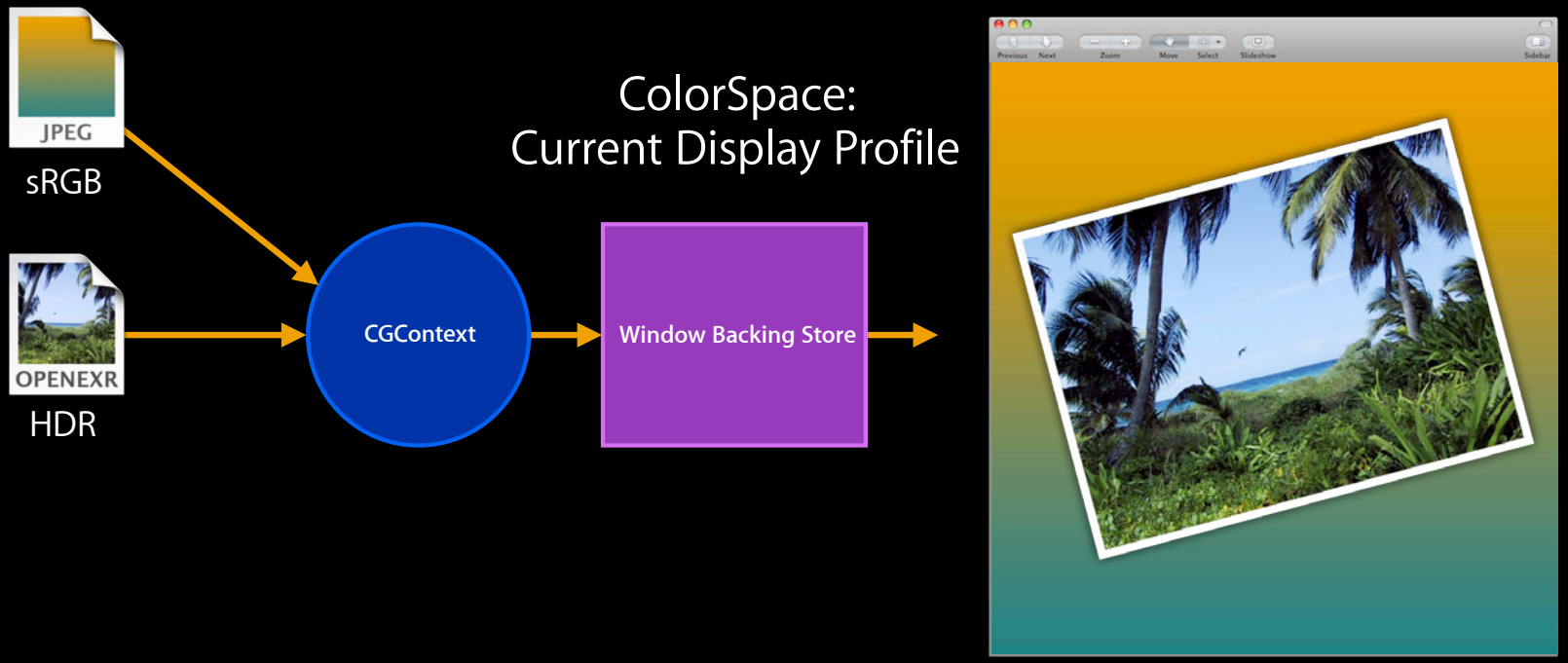
CGContextDrawImage

Window Backing Store Color Space



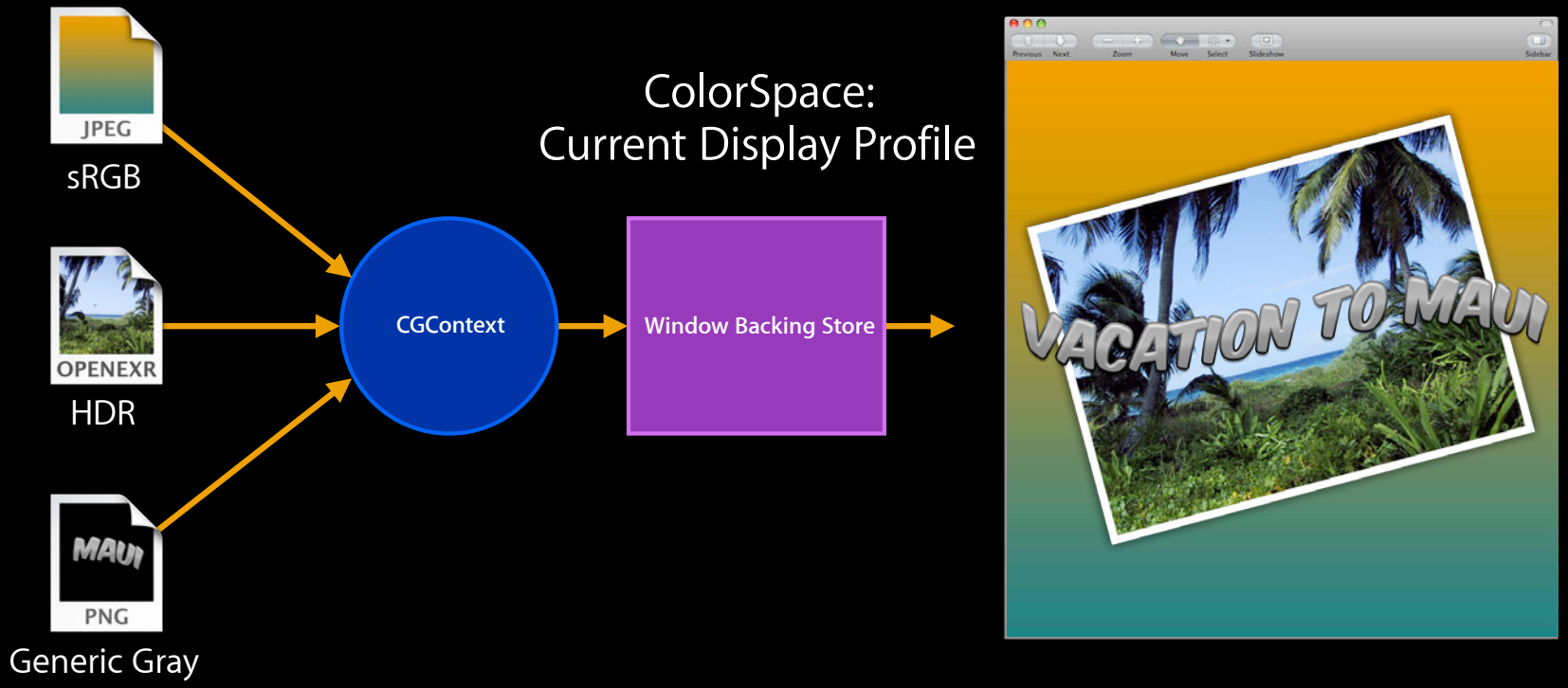
CGContextDrawImage

Window Backing Store Color Space



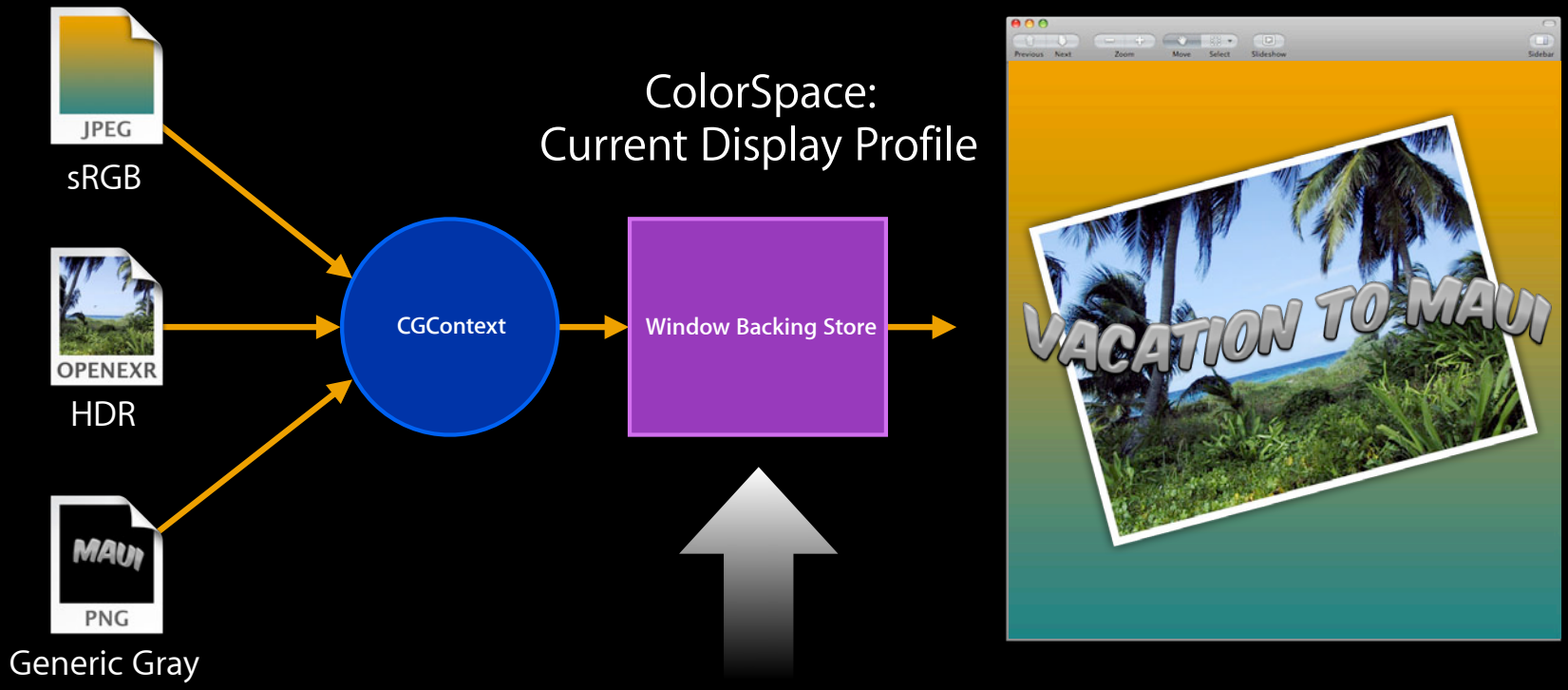
CGContextDrawImage

Window Backing Store Color Space



CGContextDrawImage

Window Backing Store Color Space



CGContextDrawImage

Window Backing Store Color Space can be set by the application to any arbitrary matrix based RGB profile, e.g. sRGB, Adobe RGB, etc…

Other frameworks in Quartz

- CoreAnimation
  - Use GPU when drawing CGImageRef into a window (NSView)

```
[NSView layer].contents = mCGImage; // set layer contents to CGImageRef
```

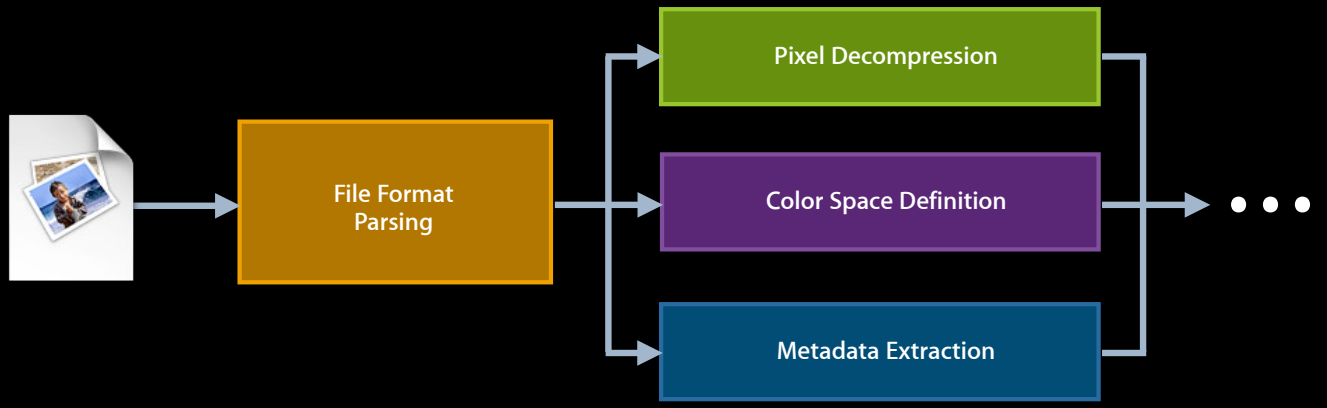
- CoreImage
  - Low level framework that leverages GPU to process images

```
[CIImage imageWithCGImage:]
```

- Reference:
  - Core Image Reference Collection
  - Core Animation Programming Guide
- Sample Code: ImageApp, CoreAnimationKioskStyleMenu

ImageIO

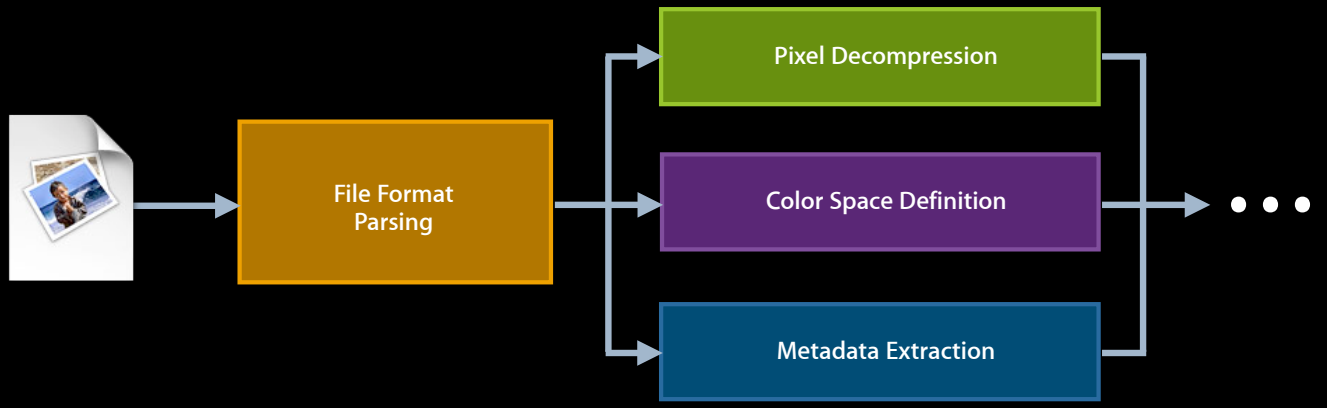
What is involved in reading image file



ImageIO

What is involved in reading image file

- Automatic Color Management
  - CGColorSpace created from metadata or embedded profile



ImageIO continued

Supports 20+ file formats including RAW camera

- Reading: create CGImageSource and acquire CGImage

```
CGImageSourceCreate(...), CGImageSourceCreateWithURL(...), ...
CGImageSourceCreateImageAtIndex()
```

- Writing: create CGImageDestination in a specified file format and add CGImage

```
CGImageDestinationCreateWithData(...),
CGImageDestinationCreateWithURL(...), ...
CGImageDestinationAddImage(...)
```

- Manual: Image I/O Programming Guide on developer.apple.com
- Sample Code: ImageApp

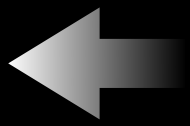
ImageCapture

- Acquire images directly from cameras and scanners
  - Based on ImageIO
  - Uses ColorSync if needed to
    - Determine ICC profile from Device integration database
    - Color convert images for display

ImageCapture

- ImageKit for complete UI
  - Views of available cameras, scanners, and images

```
IKDeviceBrowserView
IKCameraDeviceView, IKScannerDeviceView
IKImageView
```



Color managed to current display

- ImageCaptureCore for lower level control

```
ICDeviceBrowserDelegate
ICCameraDeviceDelegate, ICScannerDeviceDelegate
ICCameraDeviceDownloadDelegate (e.g. for downloading images)
```

- Manual: Image Capture Applications Programming Guide

AVFoundation

- Open, display, and save video content

```
AVAssetReader, AVPlayer, AVAssetWriter
```

- Critical to create output settings dictionary to tag video content on saving

```
AVVideoColorPrimariesKey
AVVideoTransferFunctionKey
AVVideoYCbCrMatrixKey

[AVAssetWriterInput assetWriterInputWithMediaType:outputSettings:]
```

AVFoundation

- Open, display, and save video content

```
AVAssetReader, AVPlayer, AVAssetWriter
```

- Critical to create output settings dictionary to tag video content on saving

```
AVVideoColorPrimariesKey
AVVideoTransferFunctionKey
AVVideoYCbCrMatrixKey

[AVAssetWriterInput assetWriterInputWithMediaType:outputSettings:]
```

- New in Mountain Lion for power users
  - VTPixel Transfer Session and VTCompression Session

AppKit

- NSWindow automatically sets Backing Store color space at creation
- Acquire CGContext for the window

```
[[NSGraphicsContext currentConext] graphicsPort]
```

- For special cases set backing store colorspace

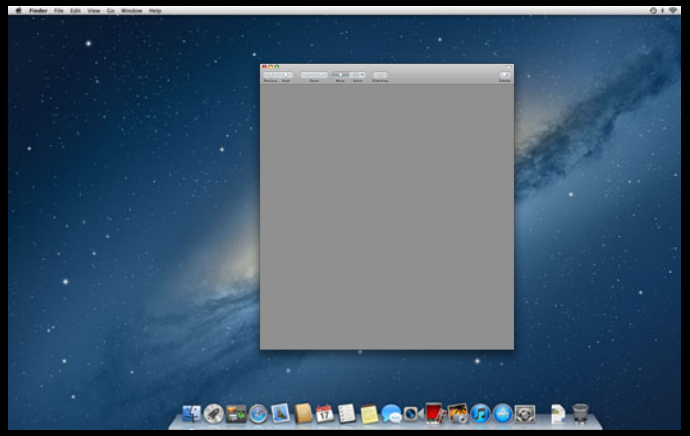
```
[NSWindow setColorSpace:]
```

- If needed register for Display Change Notification

```
[NSWindow setDisplaysWhenScreenProfileChanges:YES]
```

- Sample Code: ImageApp

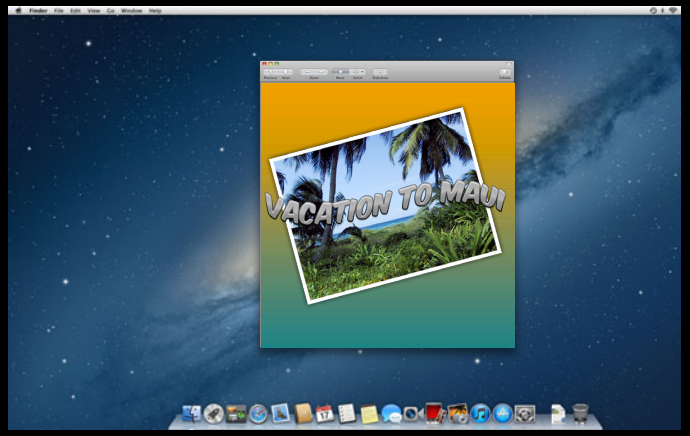
Display Change Notification



Main Display

Secondary Display

Display Change Notification

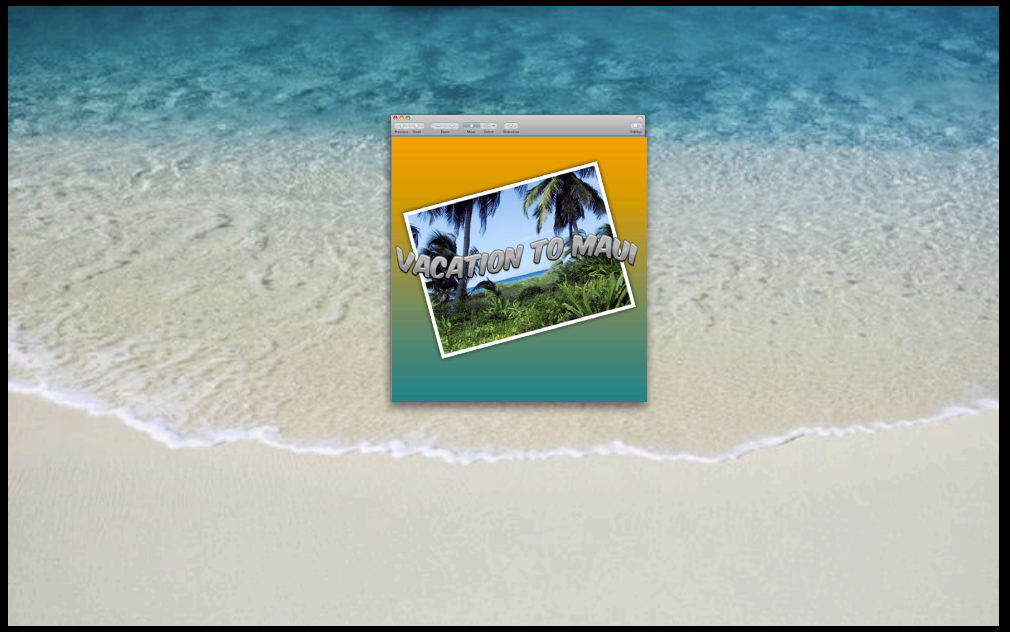


Main Display

Secondary Display

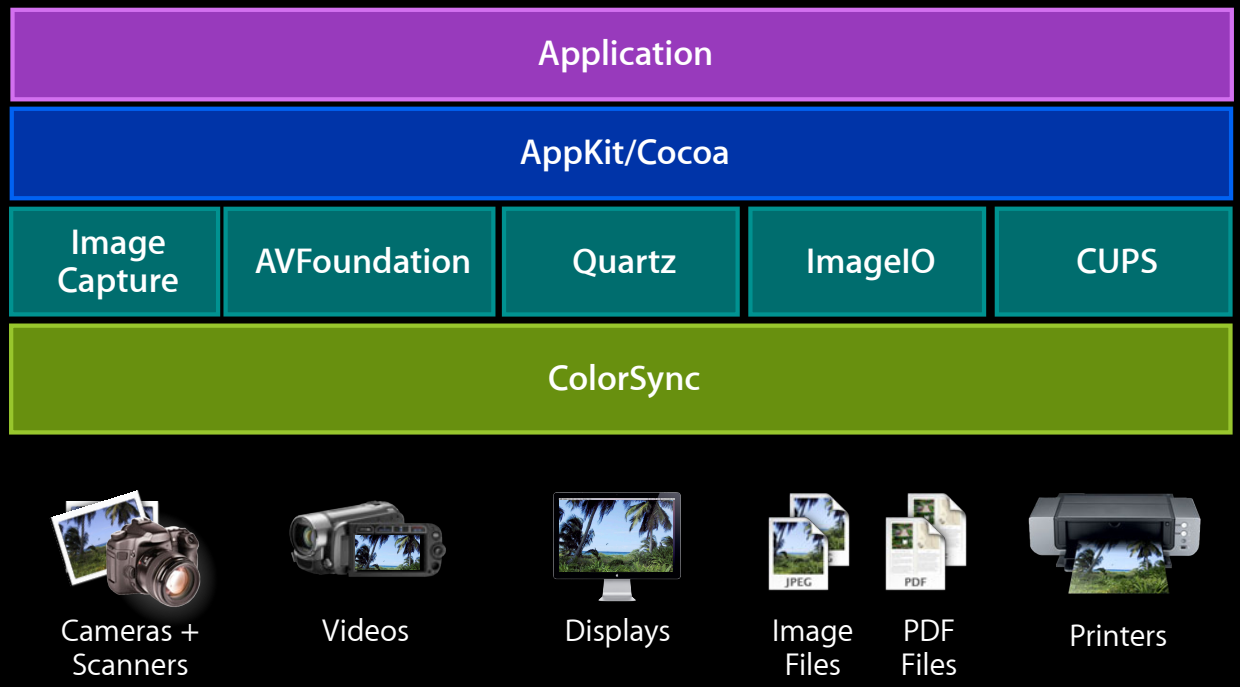
Display Change Notification

Main Display

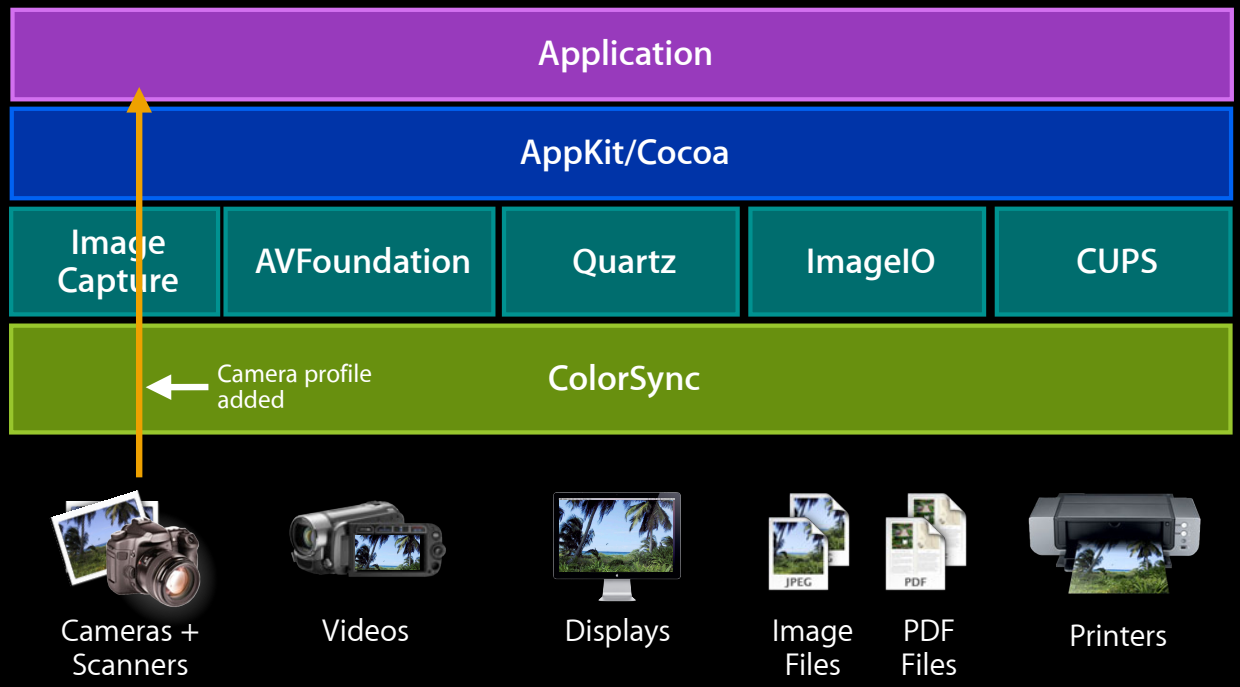


Secondary Display

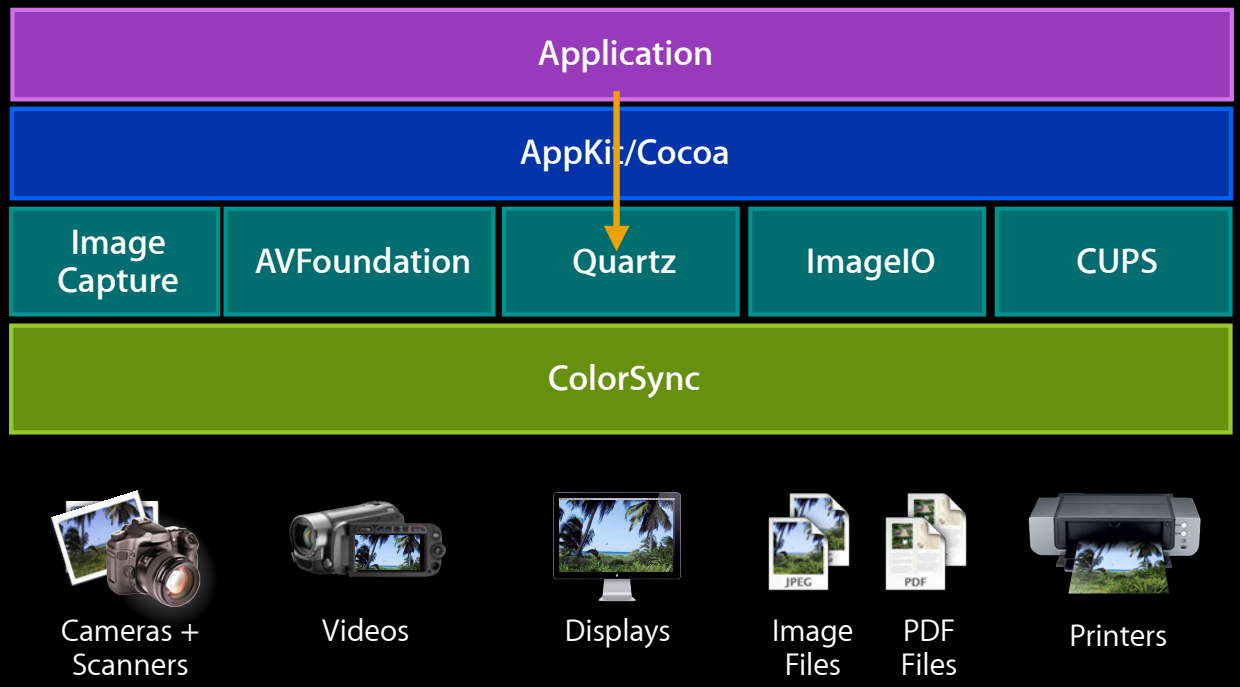
What happens in the application?



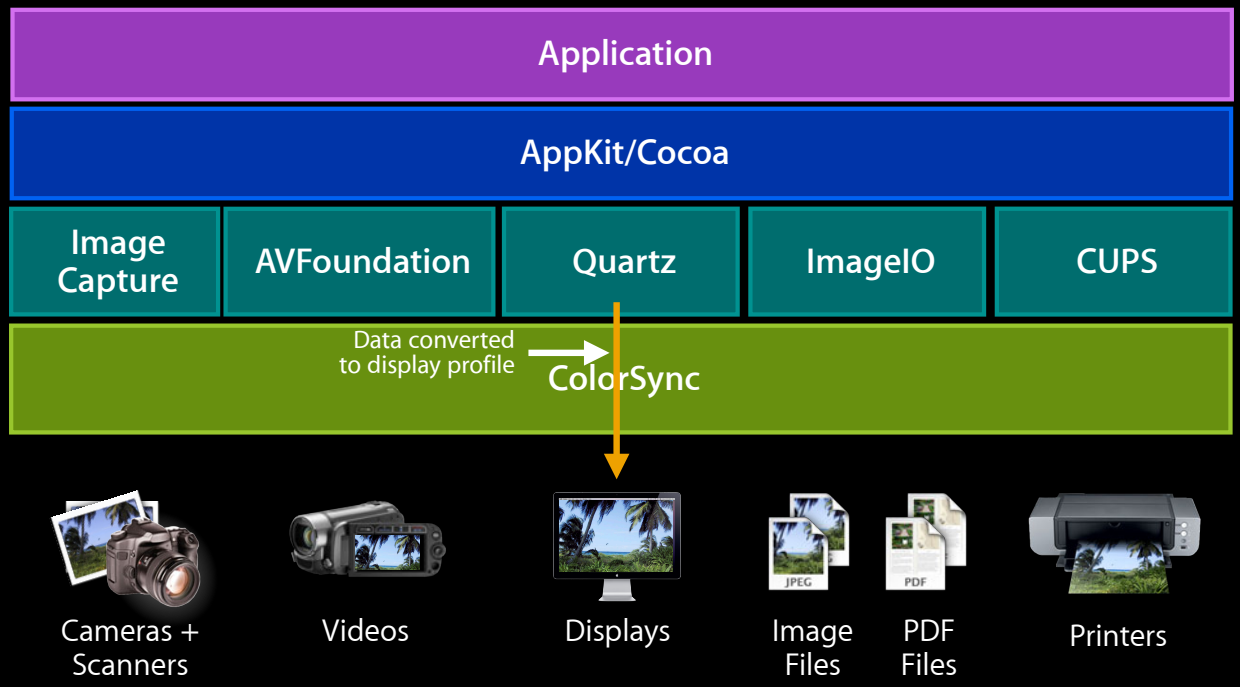
What happens in the application?



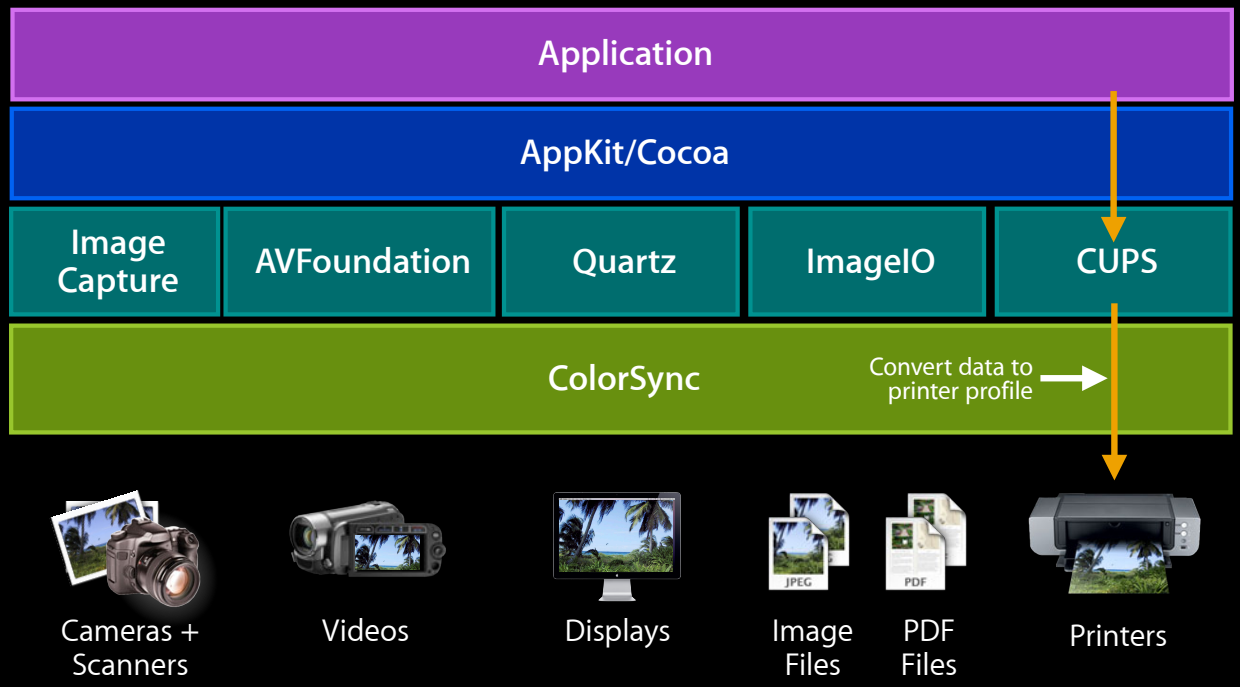
What happens in the application?



What happens in the application?

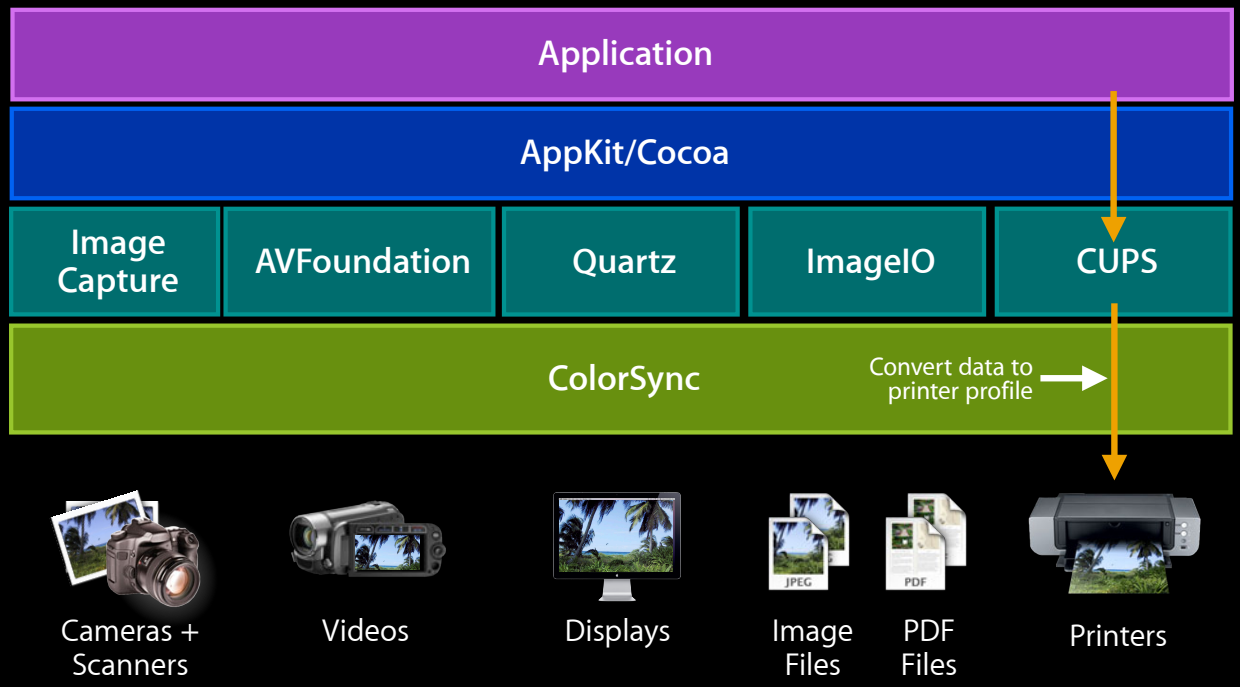


What happens in the application?



What happens in the application?

OS X offers implicit color management
through frameworks integrated with ColorSync



Color Managed Frameworks

You still have responsibilities

- Ensure
  - Buffers (CGImages) you create are tagged
  - Contexts are fully specified
  - Use of *DeviceRGB* problematic 

Non-Color Managed Frameworks

- Low-level frameworks and libraries
  - OpenGL
- Require developer to explicitly
  - Tag buffers
  - Provide color matches before rendering

Color Managing OpenGL

- OpenGL
  - Assumes linear-light (for 3D rendering)
  - No explicit color primaries
- DisplayBuffer
  - sRGB 2.2-ish gamma
  - Display's primaries (are approaching sRGB)

OpenGL Options

Simple Color Management

- What happens if you do nothing
  - Implicit 1.0->2.2 gamma boost
  - More contrasty than intended

- Provide 1.0->2.2 gamma-only conversion via shader
  - result = pow(fvalue, (source / destination));
  - result = sqrt(fvalue); // approximation

OpenGL Options

Correct Color Match

- Create shader based on code ColorSync code fragment

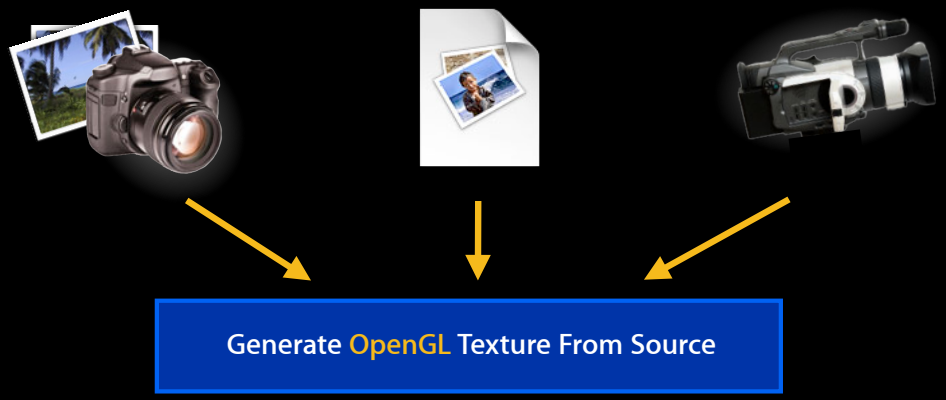
```
ColorSyncTransformCopyProperty(...,
kColorSyncTransformPrametricConversionData, NULL);
```

- Provides formula for a parametric match
  - For performance can be approximated as 3D-LUT

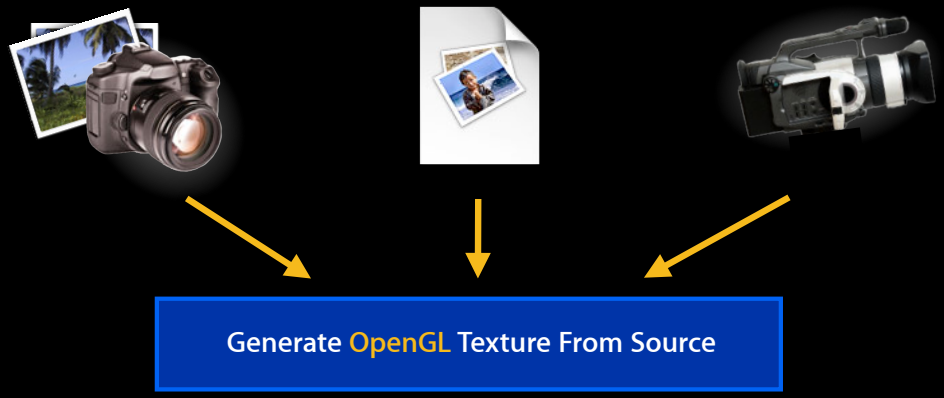
OpenGL Example

OpenGL Example

OpenGL Example



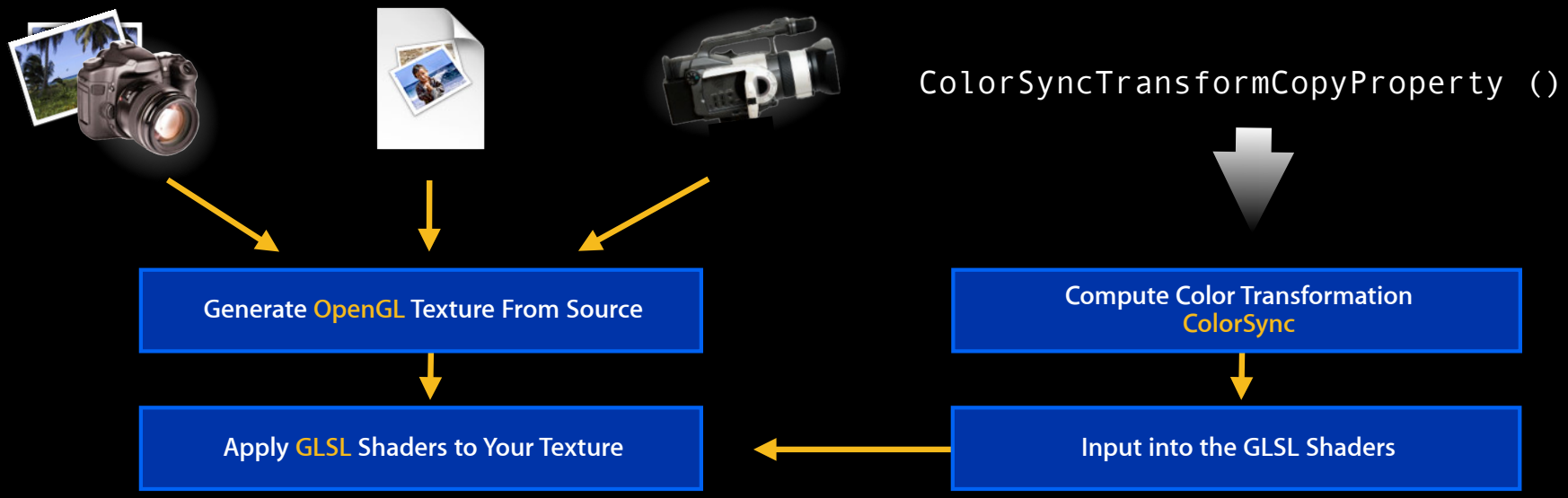
OpenGL Example



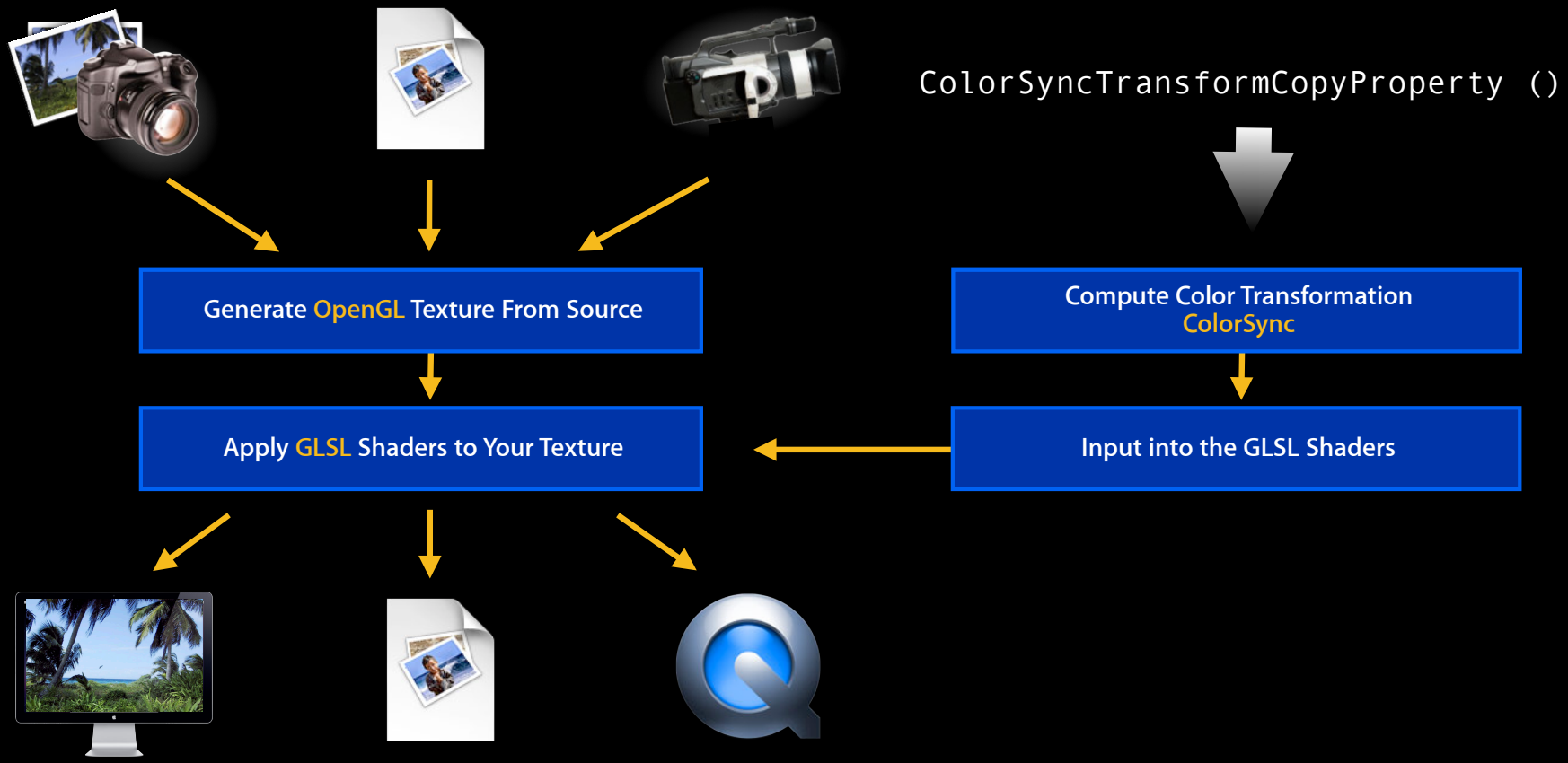

```
ColorSyncTransformCopyProperty ()
```

Compute Color Transformation
ColorSync

OpenGL Example



OpenGL Example



Authoring Image/Video Content for Mac/iOS

Authoring/Proofing Workstation

- Use Apple desktop displays
  - 3rd-party wide-gamut displays contraindicated
- Strive for consistent viewing environment
  - Avoid windows
  - Backlight displays
- Calibration not required

Content Authoring/Proofing

- Use color managed authoring tools
- Most straightforward to use sRGB for everything
  - Author in sRGB
  - Export to sRGB
  - Source material can be anything
- High value content on Mac
  - Keep source in native format

Tagging Content

- Tagging content makes it self-describing
  - Otherwise *garbage-in*
- Video tagged with *nclc*
- Images tagged with ICC profiles
  - Sometimes colorspace described in meta-information
- Image buffers tagged with CGColorSpaces (programmatically)

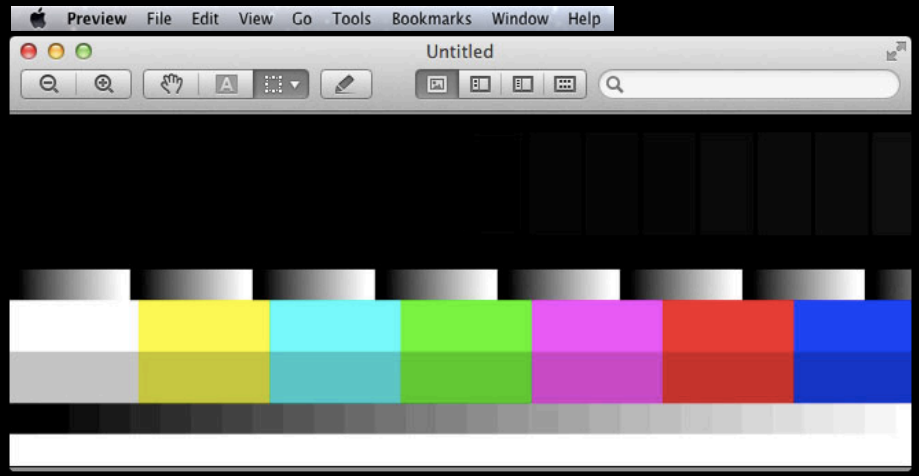
Tagging on OS X

- Apple tools automatically tag content
- Other tools do not
- Make sure to explicitly tag and verify content

- Mountain Lion includes the following tools
  - Preview
  - ColorSync Utility
  - QuickTime X Automator Action

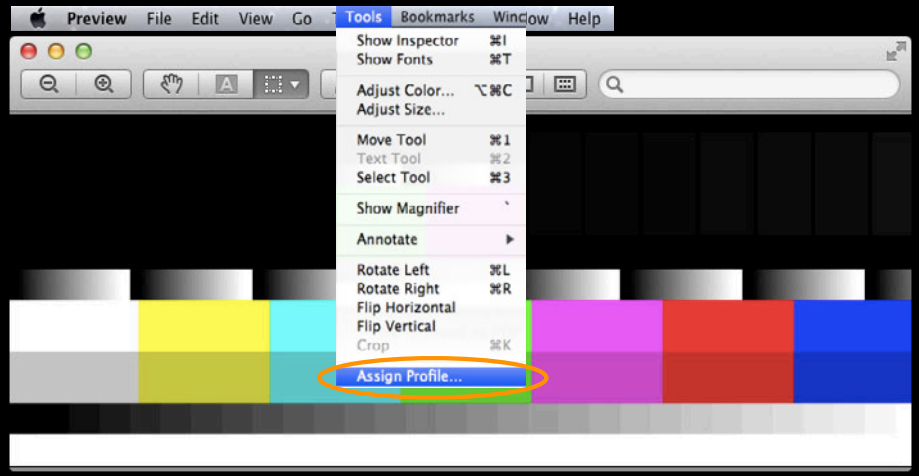
Preview

- Use Preview to assign profile to any image



Preview

- Use Preview to assign profile to any image



ColorSync Utility

ColorSync Utility

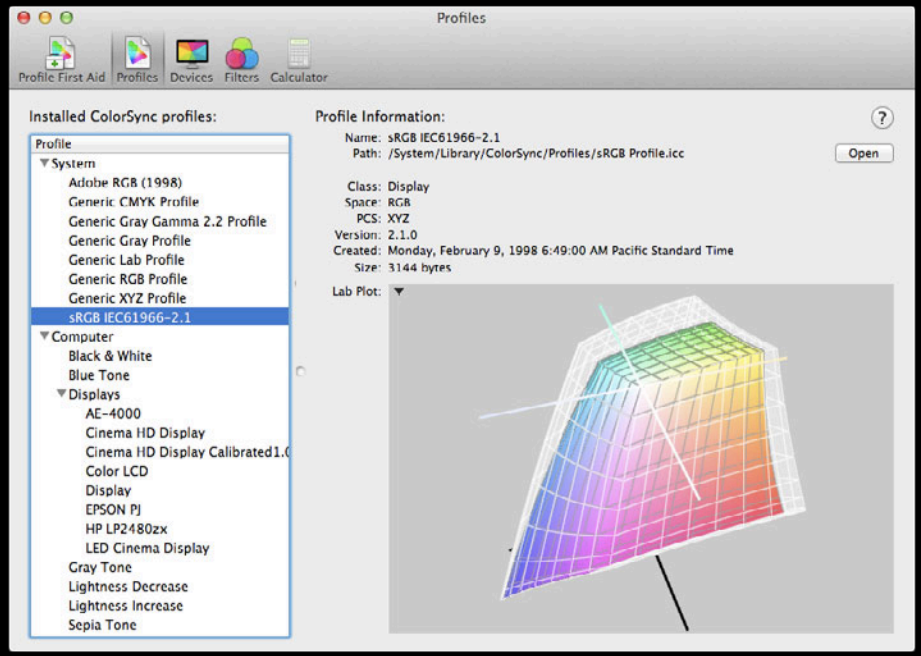
- Assign profiles to images

ColorSync Utility

- Assign profiles to images
- Match images to profiles

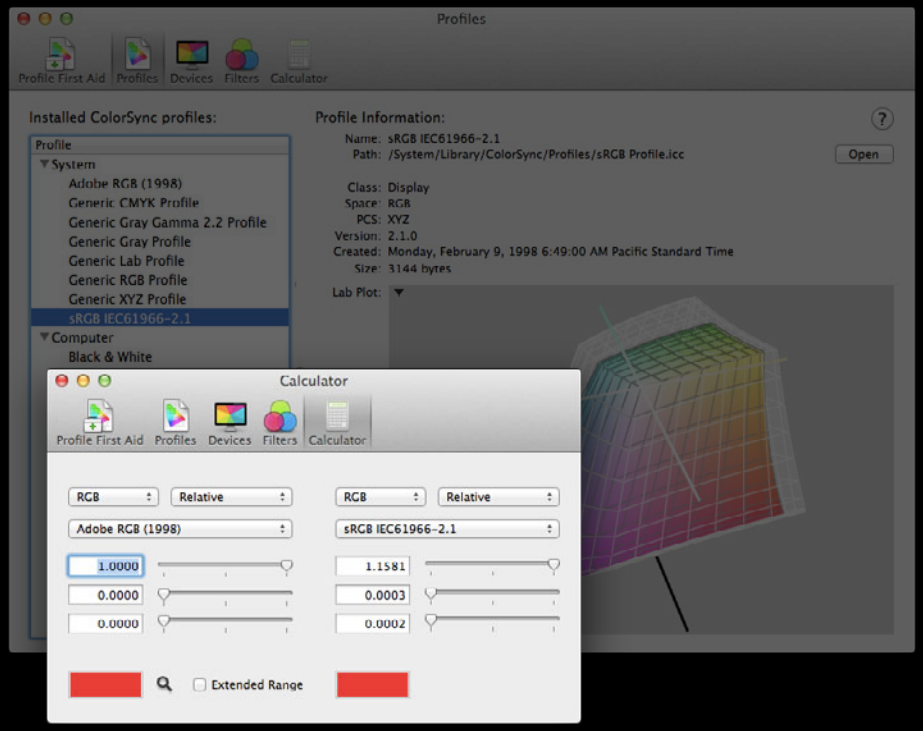
ColorSync Utility

- Assign profiles to images
- Match images to profiles
- Compare profiles in 3D!



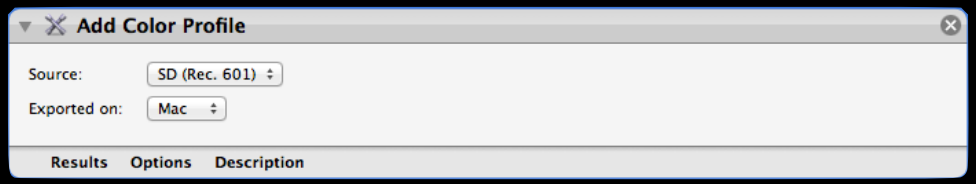
ColorSync Utility

- Assign profiles to images
- Match images to profiles
- Compare profiles in 3D!
- Perform *color math*



Add Color Profile Automator Action

- Sets *nclc* tags on video
- Updated for Mountain Lion
  - In addition to describing video source
    - Now describes RGB
    - New P22 phosphor



SIPS: Scriptable Image Processing System

- Batch color manage image files from command line
  - Example: Convert all jpegs in directory to sRGB

```
sips --matchto "/System/Library/ColorSync/Profiles/sRGB Profile.icc" *.jpg
```

  - Can also perform other tagging and image manipulation
- Man sips (for details)

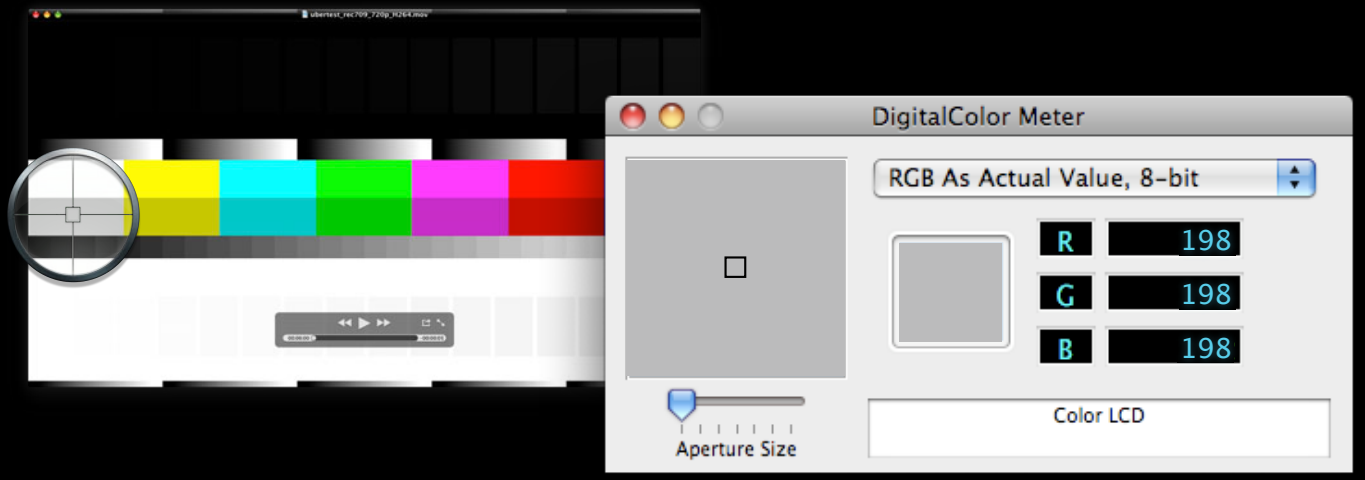
Evaluating Results

Evaluating Results

- Tools
  - Test patterns
  - Trick profiles
  - Trick content

Interpreting 75% White Levels

Values and scenarios



198, 198, 198	1.96->2.2	Standard: QT X, FCP X, iMovie, etc.
191, 191, 191	2.2->2.2 or NOP	The RGB values encoded in the file

Trick Profile

Display profile

Trick Profile

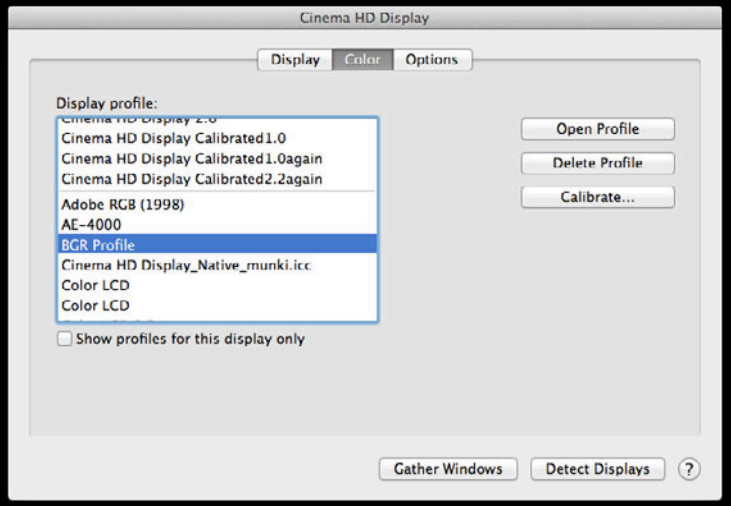
Display profile

- cp BGR.icc ~/Library/ColorSync/Profiles

Trick Profile

Display profile

- cp BGR.icc ~/Library/ColorSync/Profiles
- Select in display preferences pane



Trick Profile

Display profile

- cp BGR.icc ~/Library/ColorSync/Profiles
- Select in display preferences pane
- Enjoy

Safari color manages tagged content

Trick Profile

Display profile

- cp BGR.icc ~/Library/ColorSync/Profiles
- Select in display preferences pane
- Enjoy

Safari color manages tagged content

No color management during zooming

Trick Profile

Display profile

- cp BGR.icc ~/Library/ColorSync/Profiles
- Select in display preferences pane
- Enjoy

Safari color manages tagged content

No color management during zooming

Trick Content

With embedded profile

Trick Content

With embedded profile

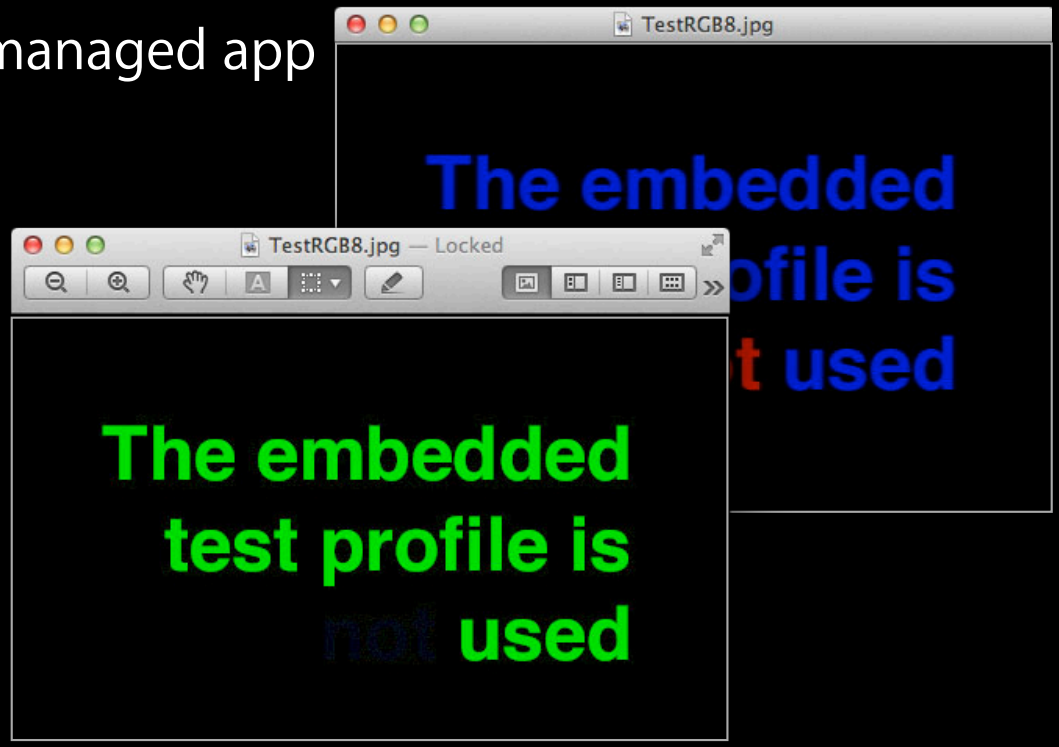
Non-color managed app

Trick Content

With embedded profile

Non-color managed app

Preview

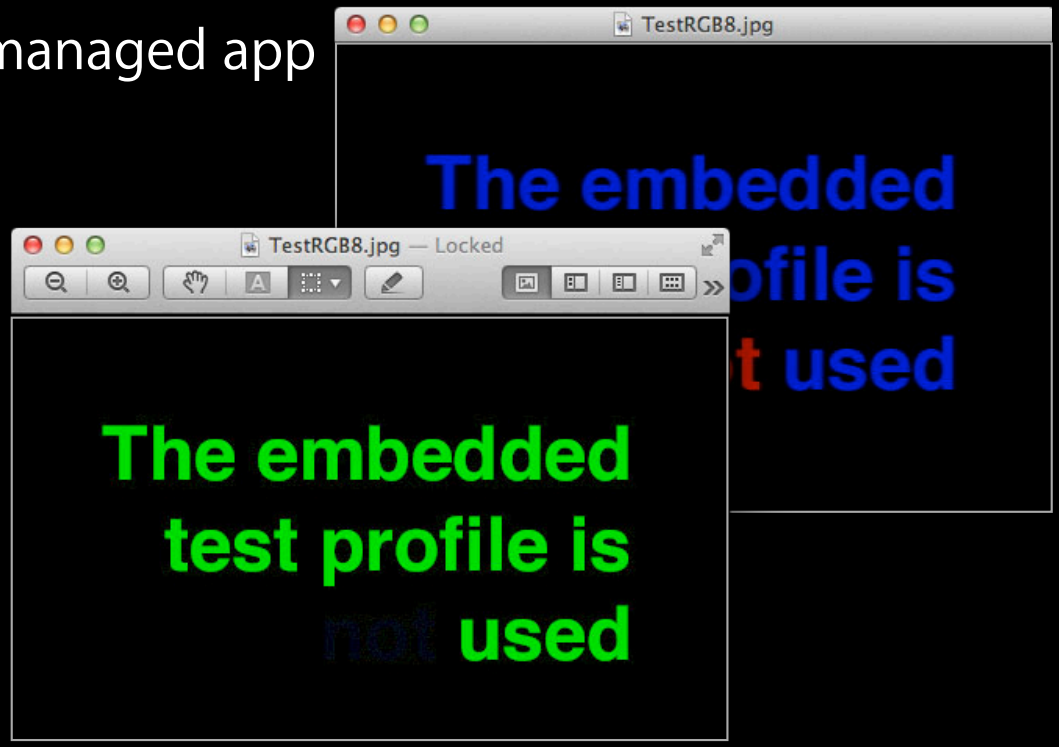


Trick Content

With embedded profile

Non-color managed app

Preview



Content available in ImageApp sample

Mountain Lion Changes

Changes to *Device Color* Interpretation

- Mountain Lion
  - `DeviceRGB`: Always interpreted as sRGB
- Pre-Mountain Lion
  - `DeviceRGB`: Dependent on context
    - If window -> Avoid color match
    - If other -> Use default profile (i.e. sRGB)

QuickTime Player X

Color manages all content

QuickTime Player X

Color manages all content

- Now automatically selects *nclc*
  - Assumes content was video exported via QuickTime
  - Picks appropriate colorspace and gamma based on this
  - Fixes confusion of untagged classic content exported to modern codec had different appearances

QuickTime Player X

Color manages all content

- Now automatically selects *nclc*
  - Assumes content was video exported via QuickTime
  - Picks appropriate colorspace and gamma based on this
  - Fixes confusion of untagged classic content exported to modern codec had different appearances

- Use updated automator action to
  - Tune or make the assumptions permanent

Final Cut Pro X

- Now color managed
  - Provides best match to given display
- Same video appearance as
  - QuickTime Player X and all Apple video apps

- Eliminates different appearance of video
  - Authored in previous FCP and viewed in QTPX

Summary

- Color Management is Powerful
  - Verification can be subtle
- Developers need to
  - Author for sRGB
  - Ensure all content is tagged
  - Use test content and *trick* ICC profiles to verify results

More Information

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Eryk Vershen
Media Technologies Evangelist
evershen@apple.com

More Information

Documentation
Technical Note TN2227 Video Color Management in AVFoundation and QT Kit
http://developer.apple.com/library/mac/#technotes/tn2227

Image I/O Programming Guide
https://developer.apple.com/library/mac/#documentation/GraphicsImaging/Conceptual/ImageIOGuide/imageio_intro/ikpg_intro.html#//apple_ref/doc/uid/TP40005462

Trick Profiles, Test content
https://developer.apple.com/library/mac/#samplecode/ImageApp

Apple Developer Forums
http://devforums.apple.com
http://developer.apple.com

Related Sessions

Getting Started with Core Image	Pacific Heights Wednesday 10:15AM
Real-Time Media Effects and Processing during Playback	Presidio Thursday 9:00AM

Lab

Color Management Lab	Graphics, Media & Games Lab B Friday 10:15AM
iOS Camera Capture Lab	Graphics, Media & Games Lab D Friday 9:00AM


WWDC2012