Working with Wide Color

Understanding and optimizing for Wide Color Gamut Displays

Session 712

Justin Stoyles Graphics and Media
Patrick Heynen Cocoa Frameworks
Steve Holt UIKit

Agenda

Core Color Concepts

What is wide color and why does it matter?

Adapting your content workflow

Implications on app colors

Optimizing your app drawing for wide color displays

Transforming color on macOS and iOS

Brief Overview

ColorSync

International Color Consortium (ICC)

Built in to OS X

Brief Overview

Retina Display

Brief Overview

Core Color Concepts

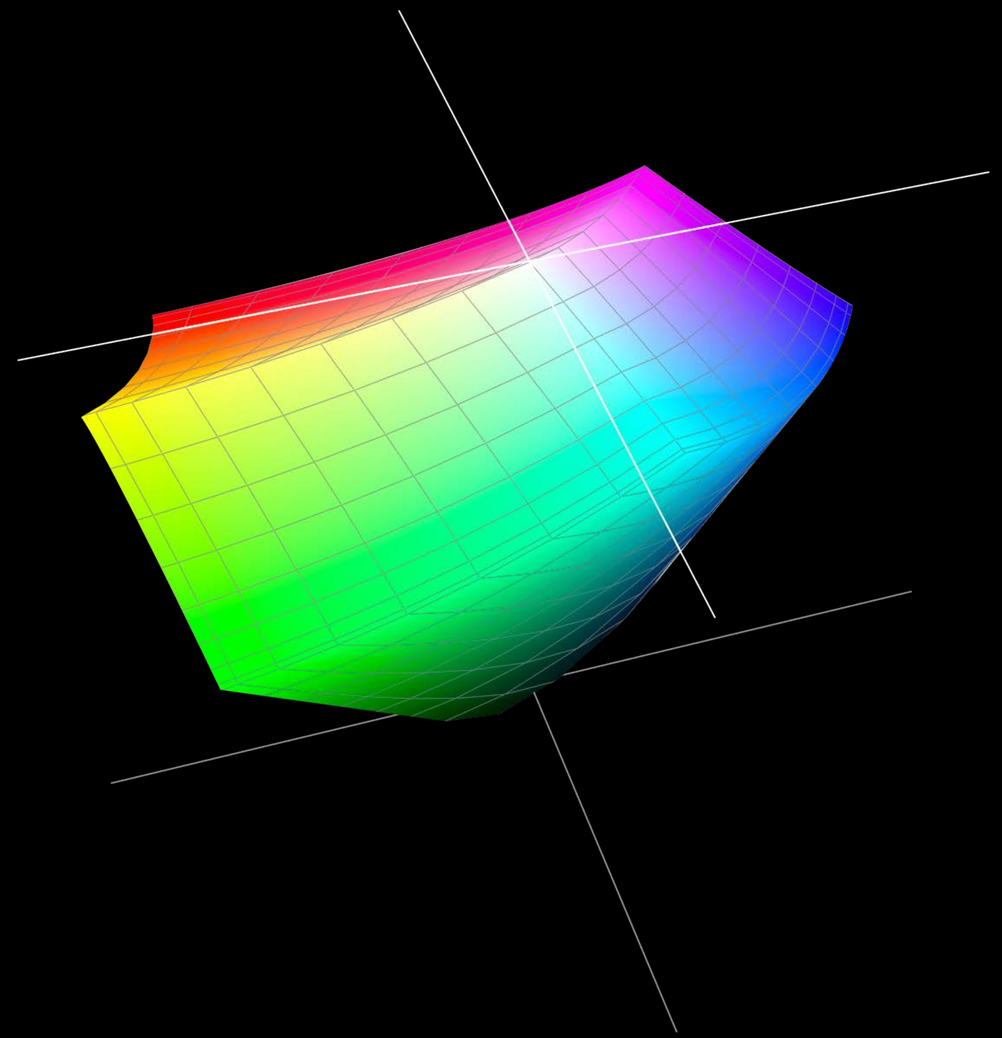
Core Color Concepts

color space

Color Channels

Color Primaries

Color Gamut



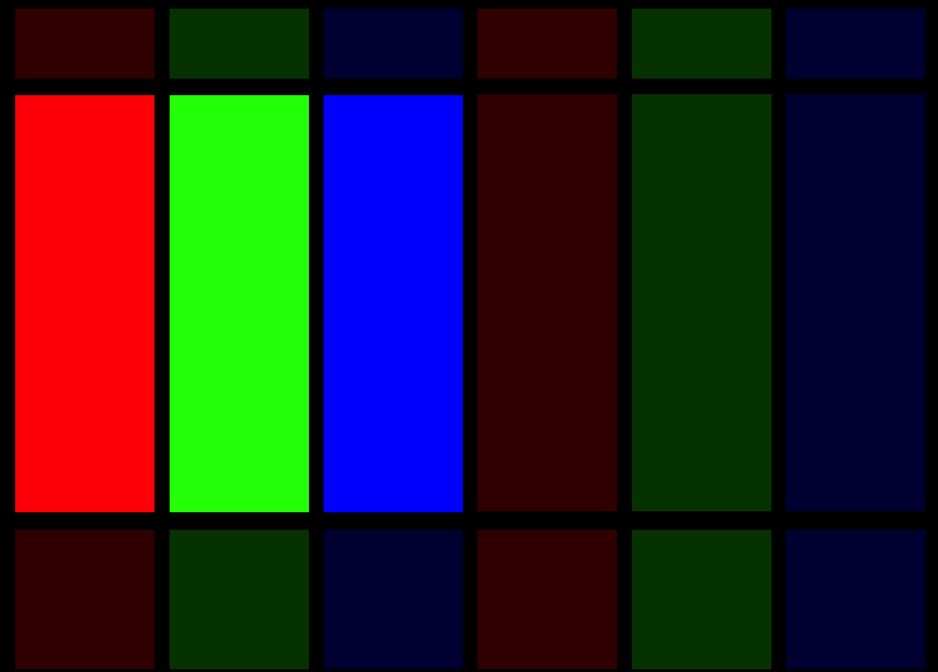
Core Color Concepts

color space

Color Channels

Color Primaries

Color Gamut



Core Color Concepts

color space

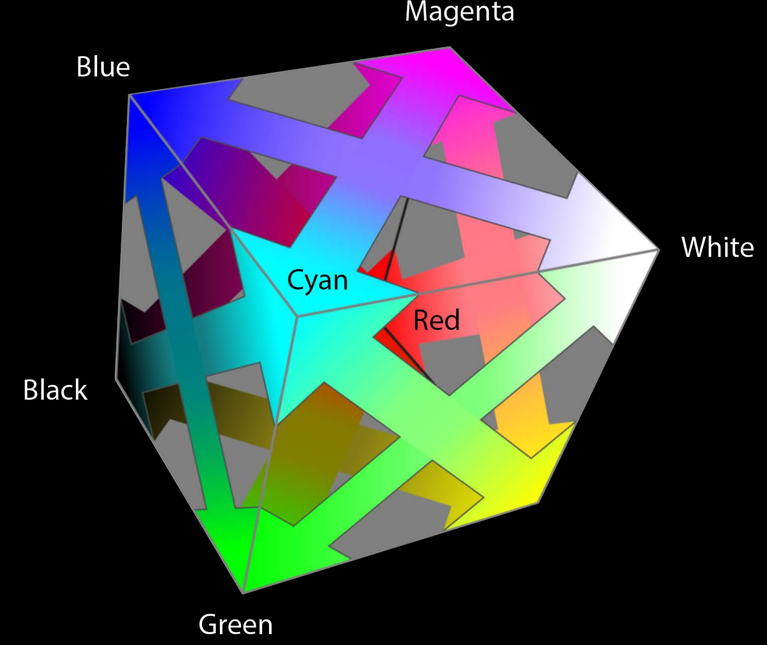
Color Channels

Color Primaries

Color Gamut

Gray Spaces

RGB Spaces



Core Color Concepts

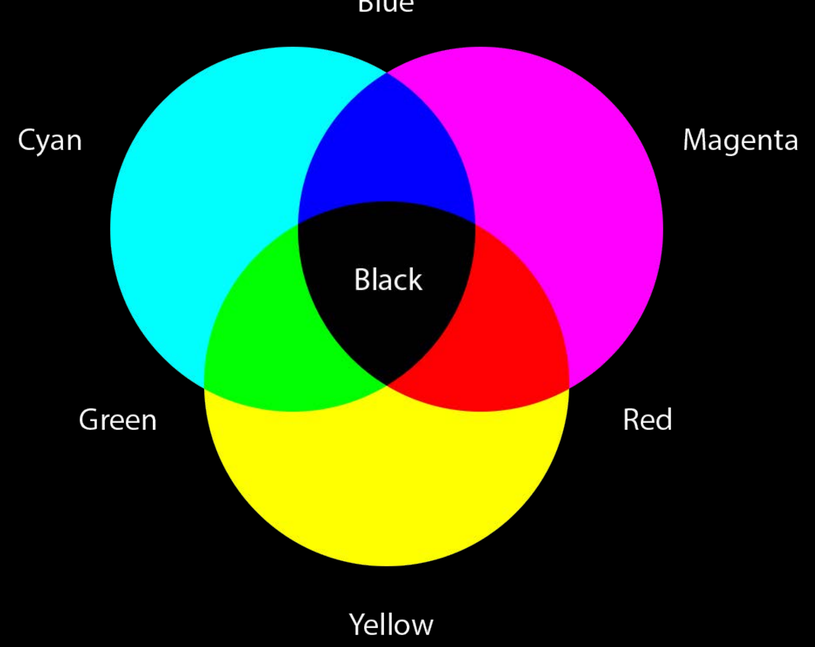
color space

Color Channels

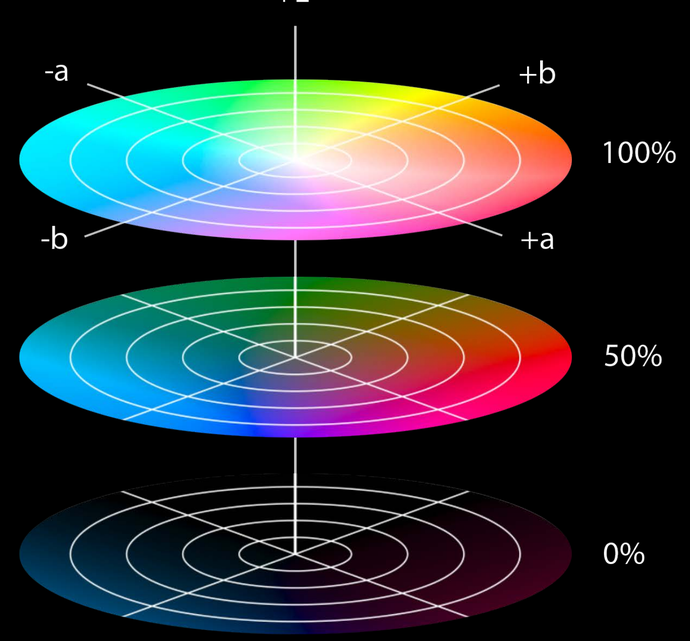
Color Primaries

Color Gamut

CMYK Spaces



Device Independent Spaces



Core Color Concepts

color space

Color Channels

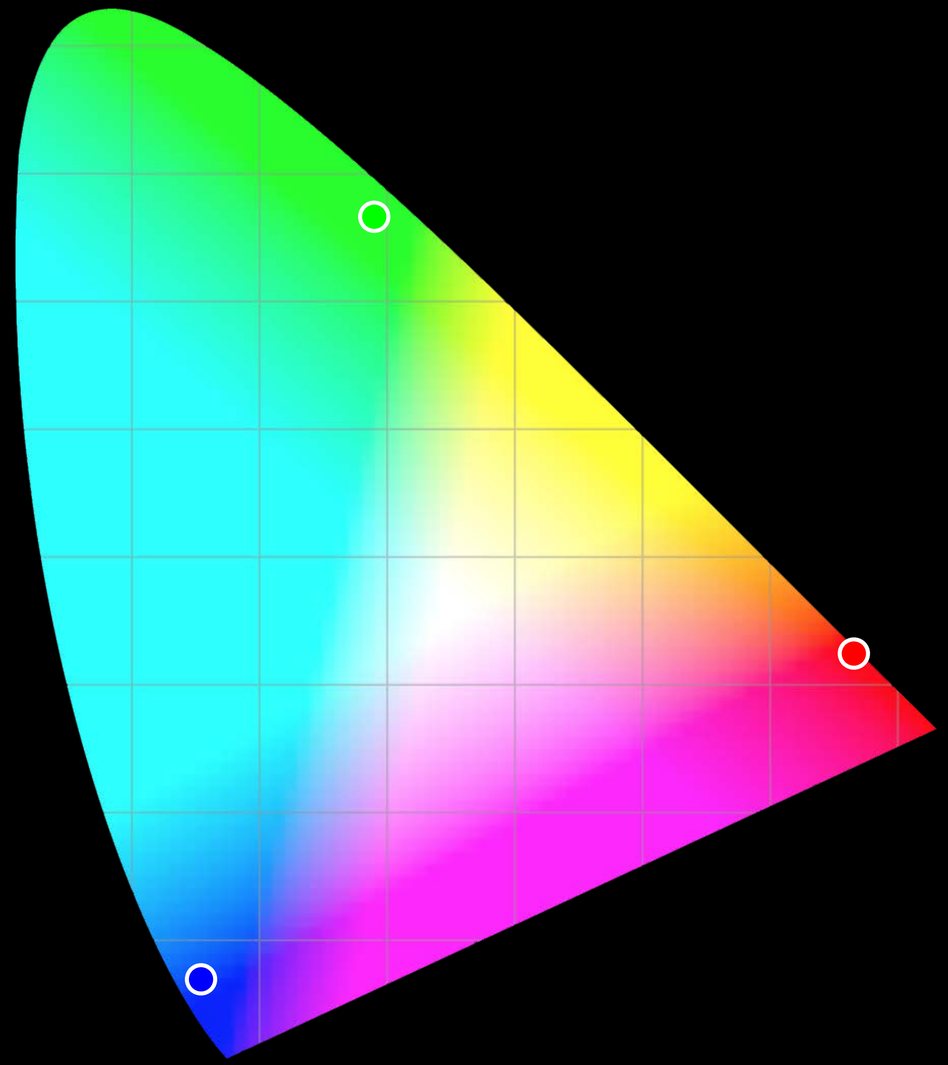
Color Primaries

Color Gamut

RGB {0.0, 0.0, 0.0} = Black

RGB {1.0, 1.0, 1.0} = White

RGB {1.0, 0.0, 0.0} = Red



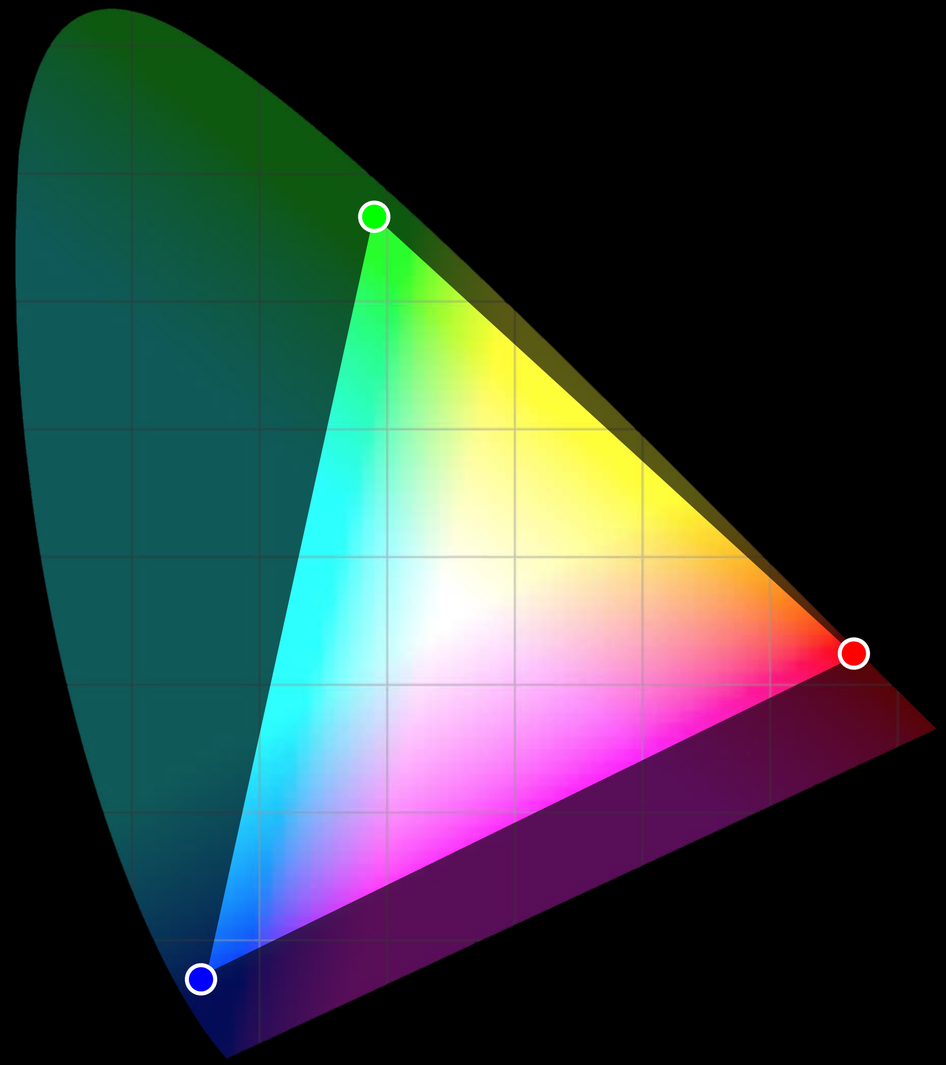
Core Color Concepts

color space

Color Channels

Color Primaries

Color Gamut



What is wide color?

Standard RGB Color Space (sRGB)

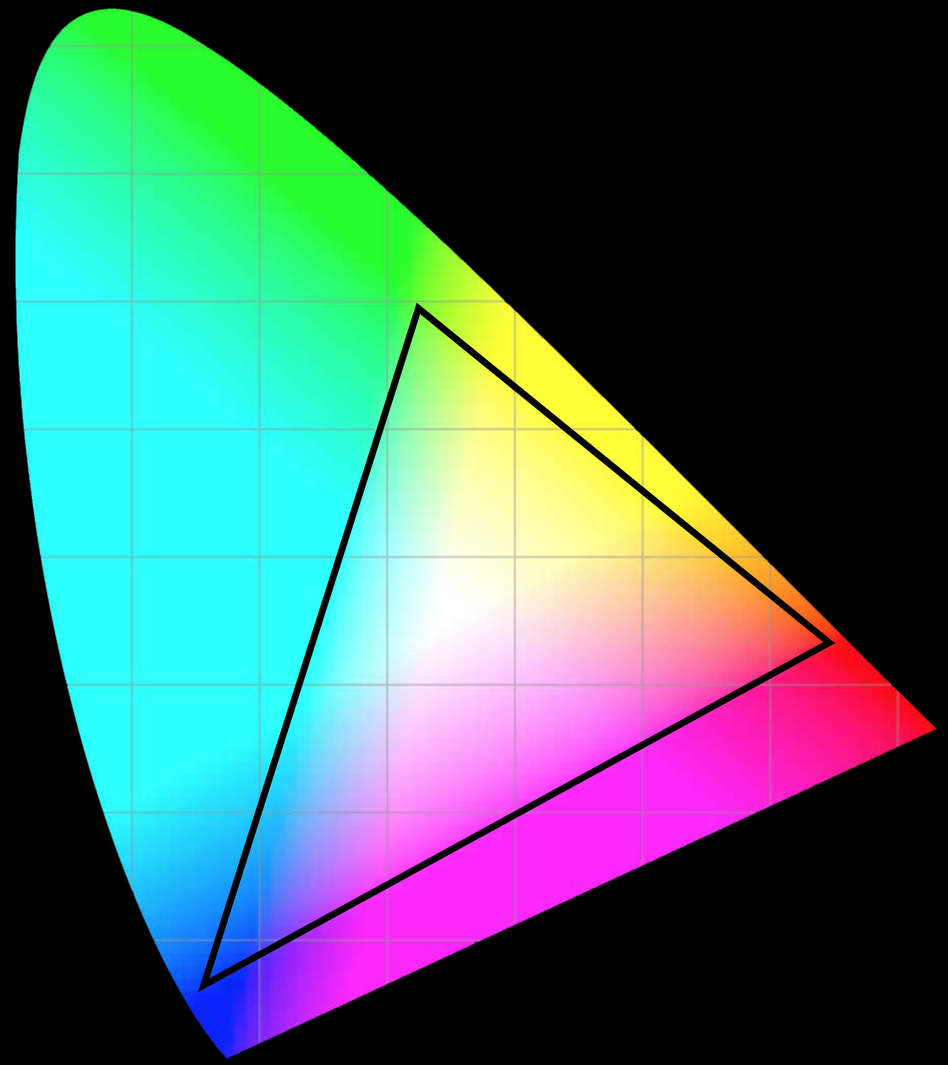
Widely used

Based on ITU-R BT.709

Gamma ≈ 2.2

Typical lighting conditions

Default color space for iOS



What do we do about this?

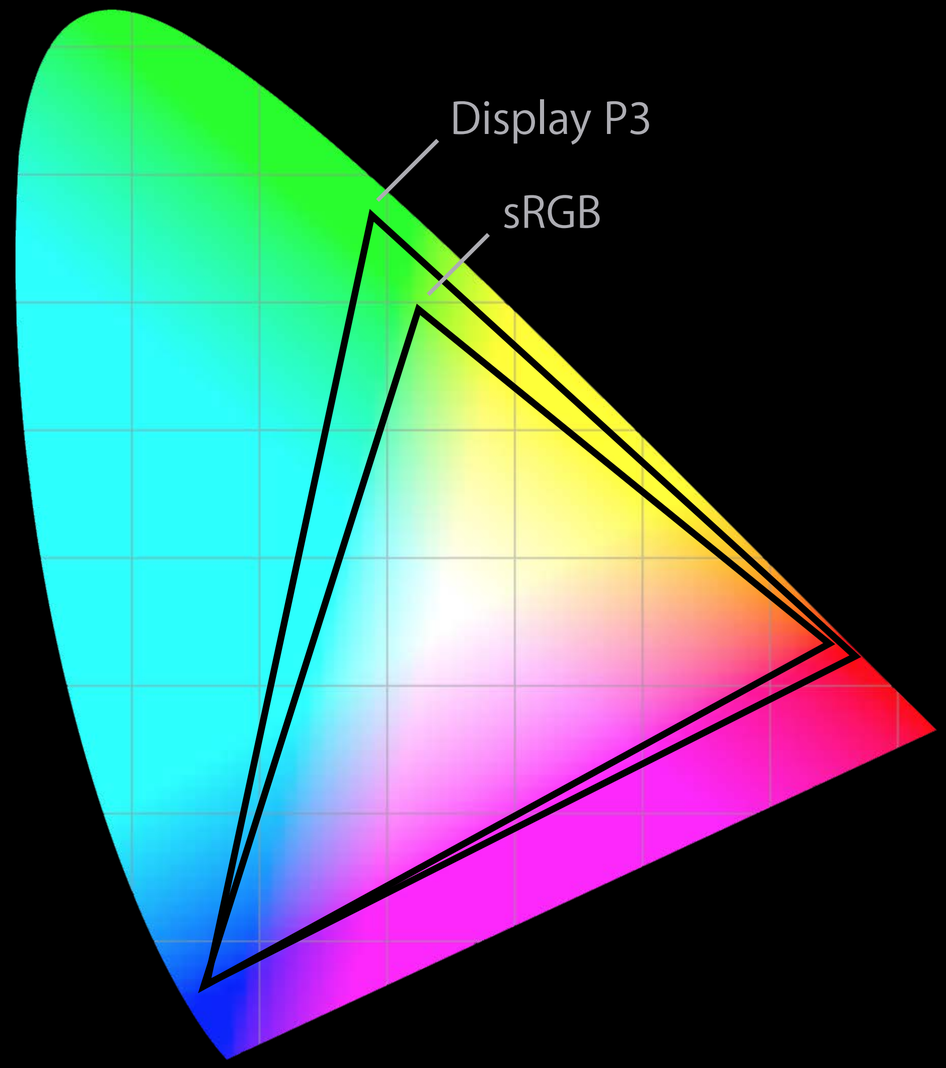
Introducing Display P3

Wide color space for today's platforms

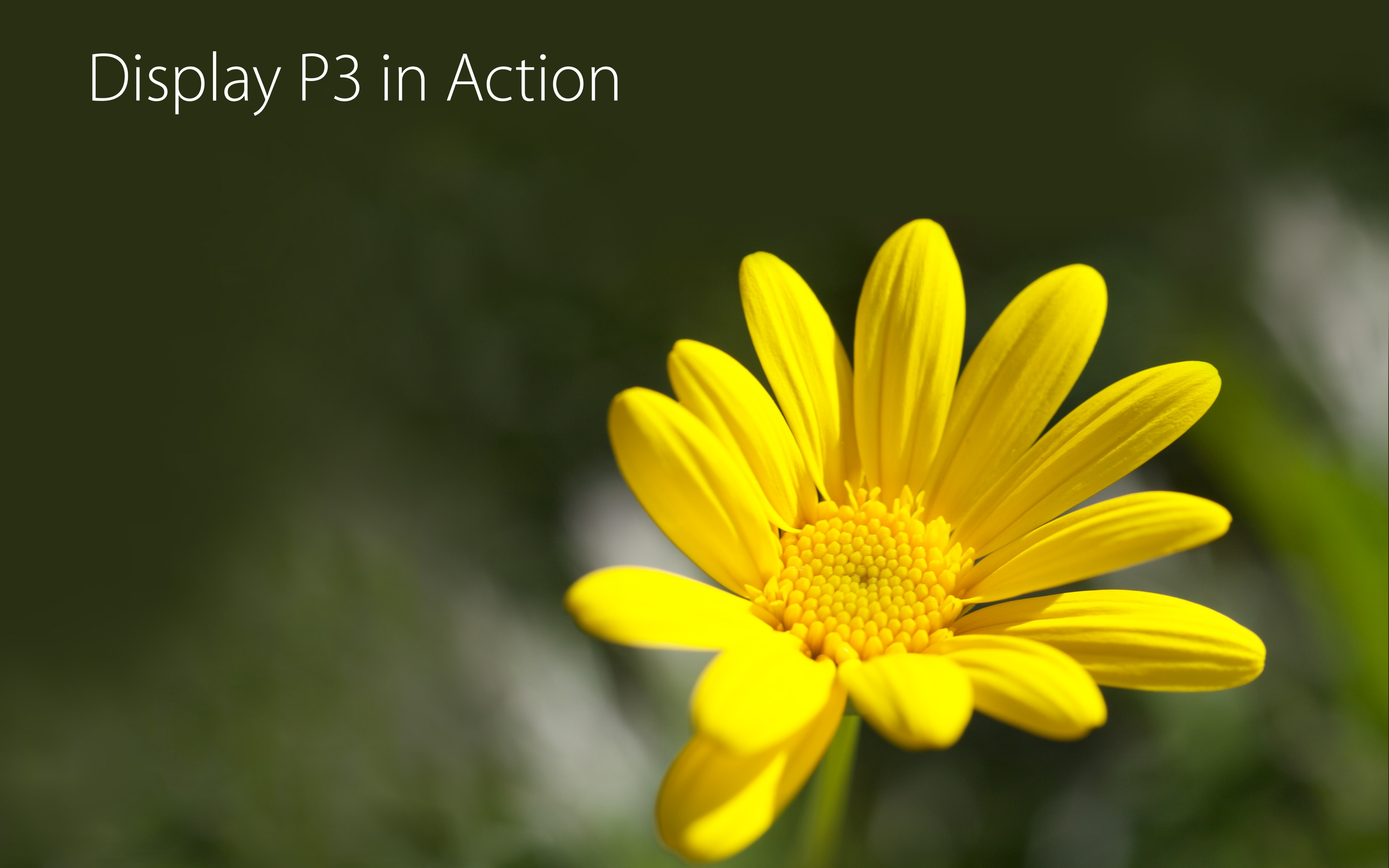
Based on the SMPTE DCI-P3

Gamma ≈ 2.2

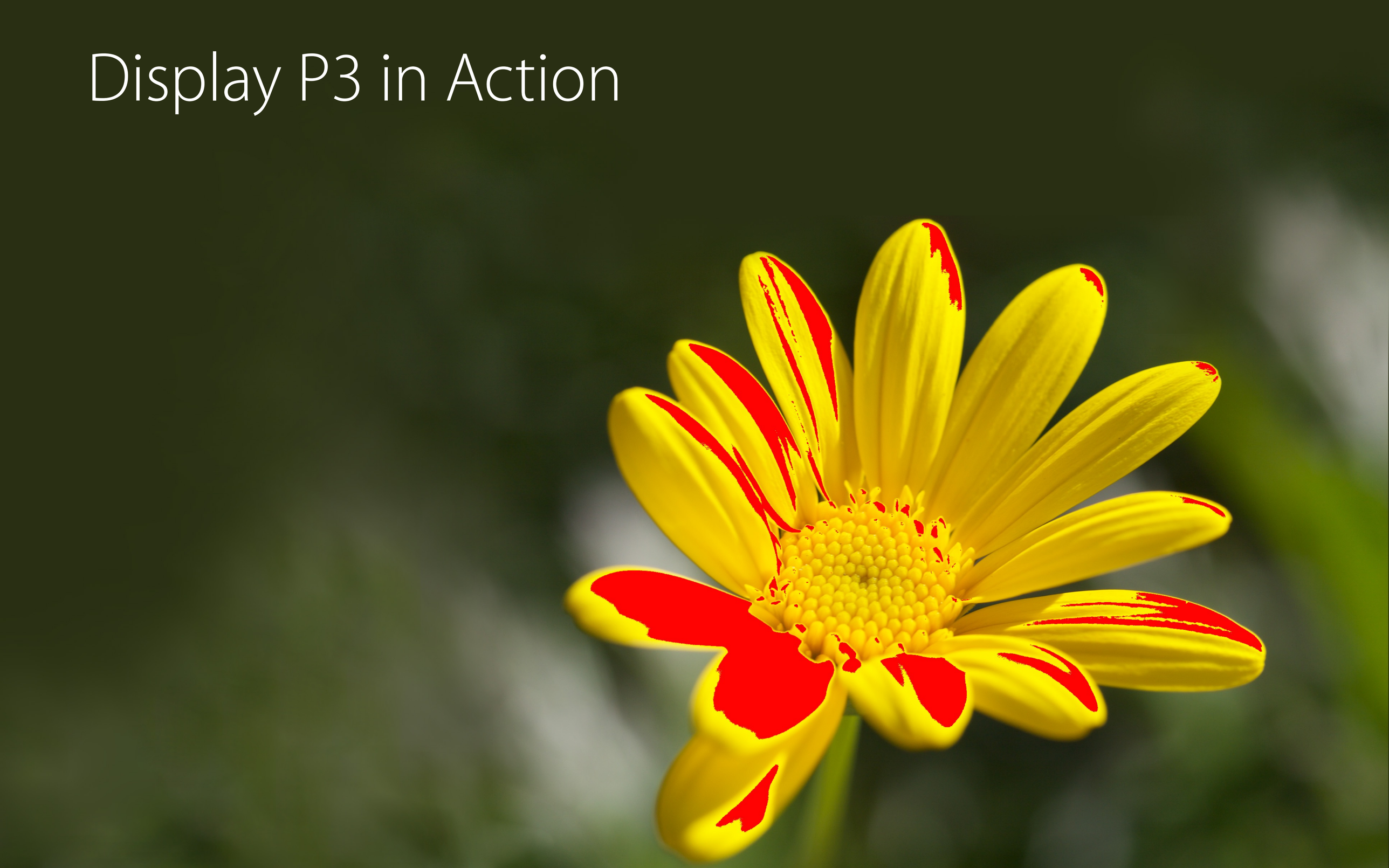
Typical lighting conditions



Display P3 in Action



Display P3 in Action



Color Management on iOS 10

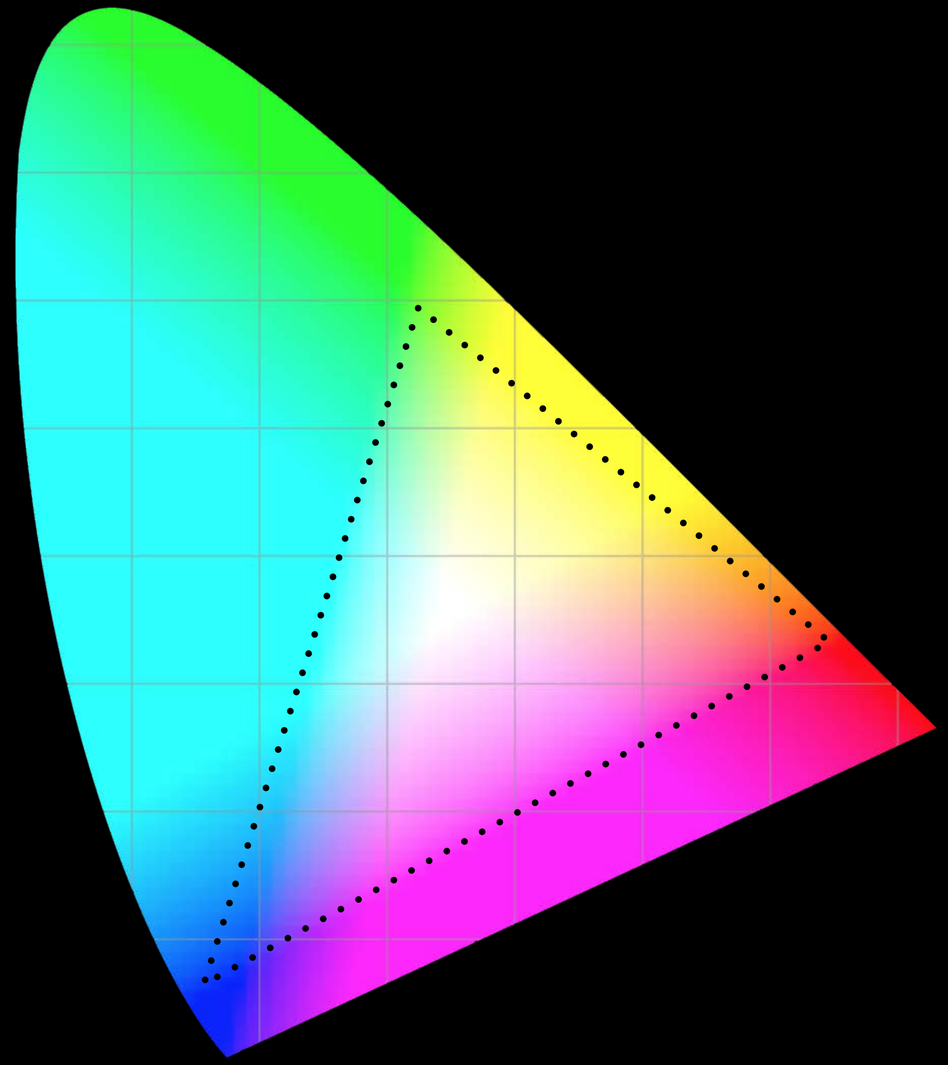
Introducing Extended Range sRGB

Same sRGB Primaries

Gamma ≈ 2.2

Typical lighting conditions

Negative values and values greater than 1



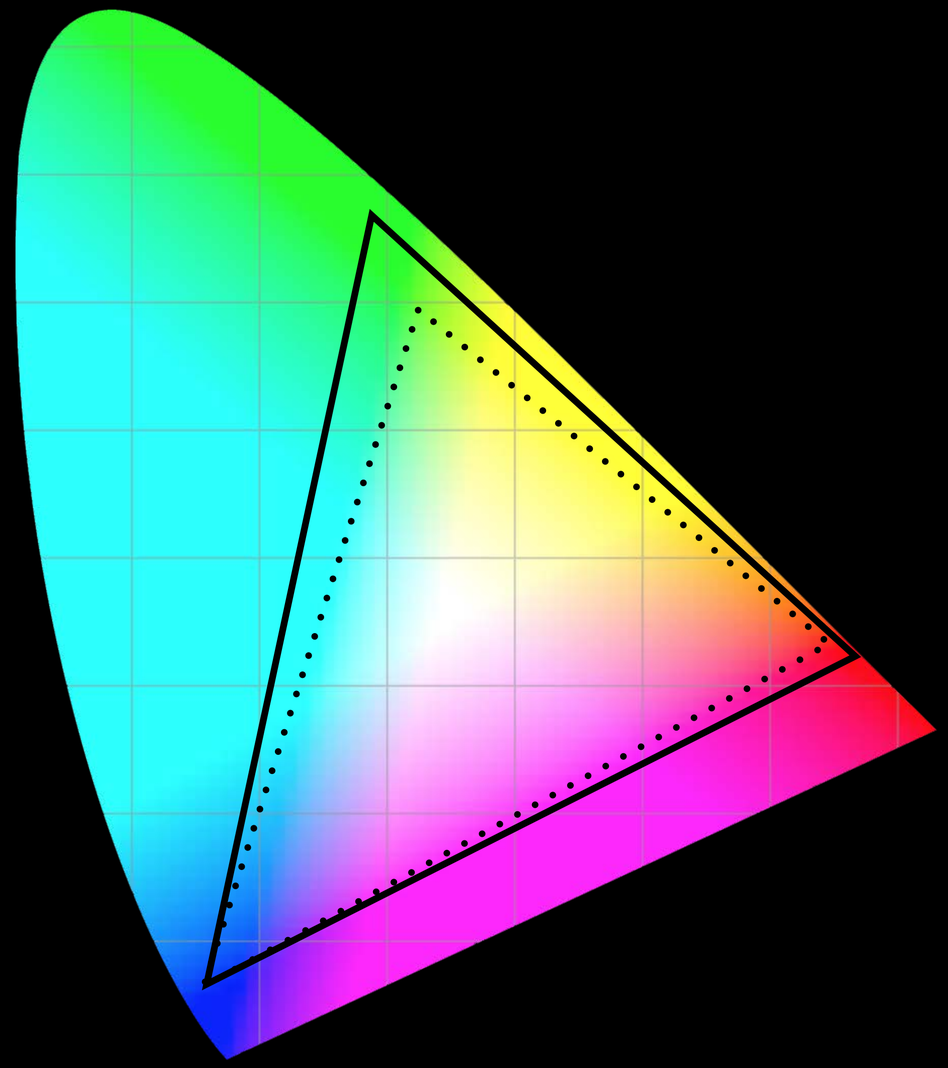
Extended Range sRGB in Action

Display P3

{1.0, 0.0, 0.0}

Extended Range sRGB

{1.358, -0.074, -0.012}



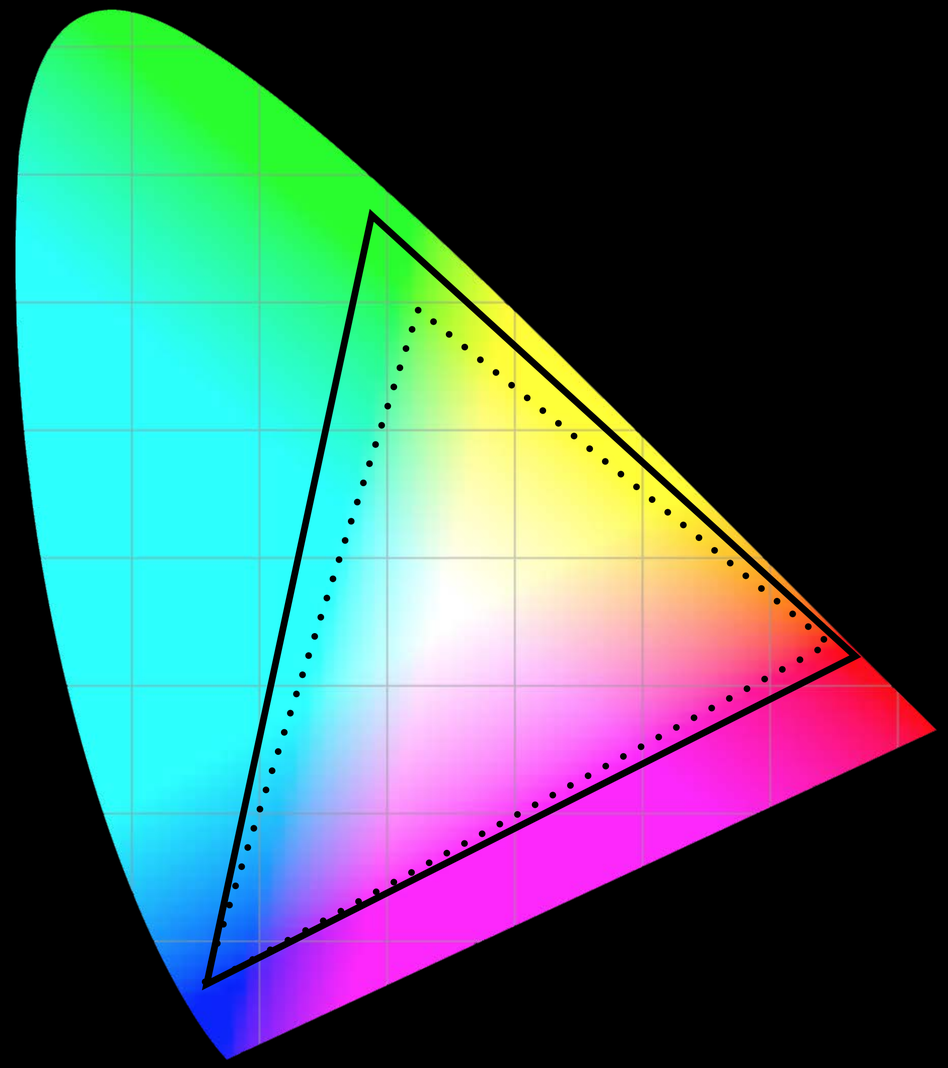
Extended Range sRGB in Action

Display P3

{1.0, 0.0, 0.0}

Extended Range sRGB

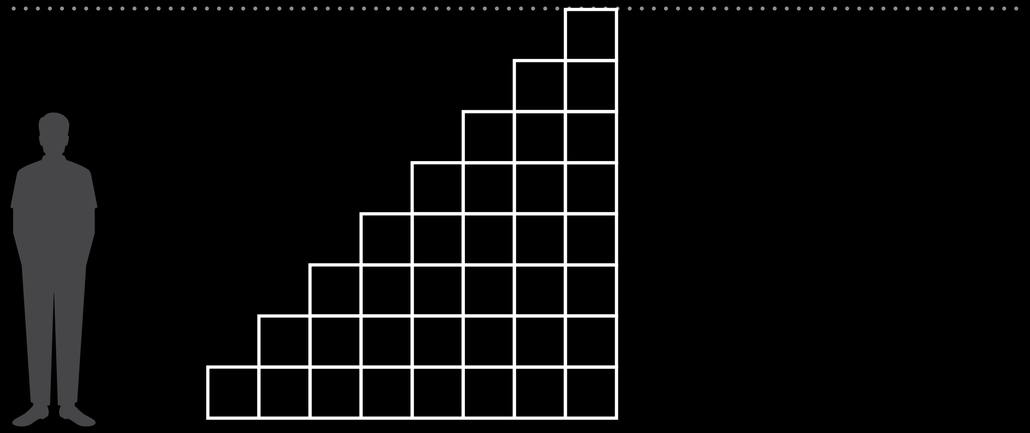
{1.358, -0.074, -0.012}



Precision and Color

2nd Floor

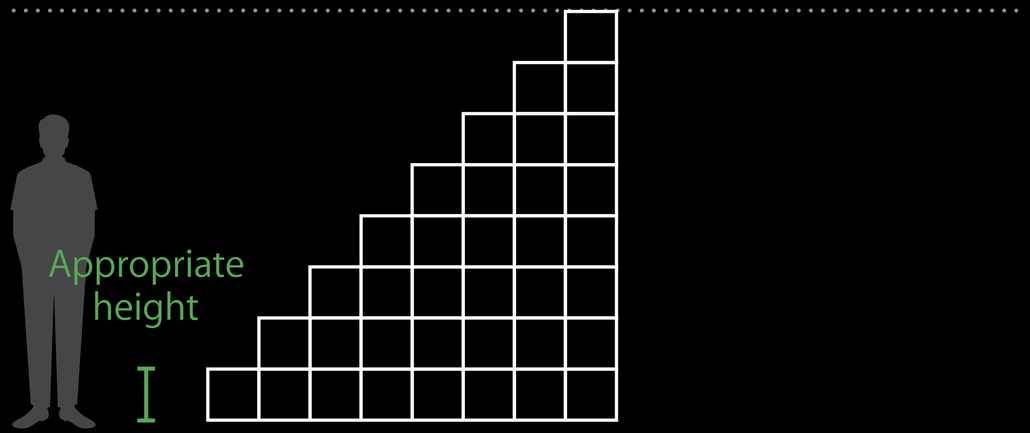
1st Floor



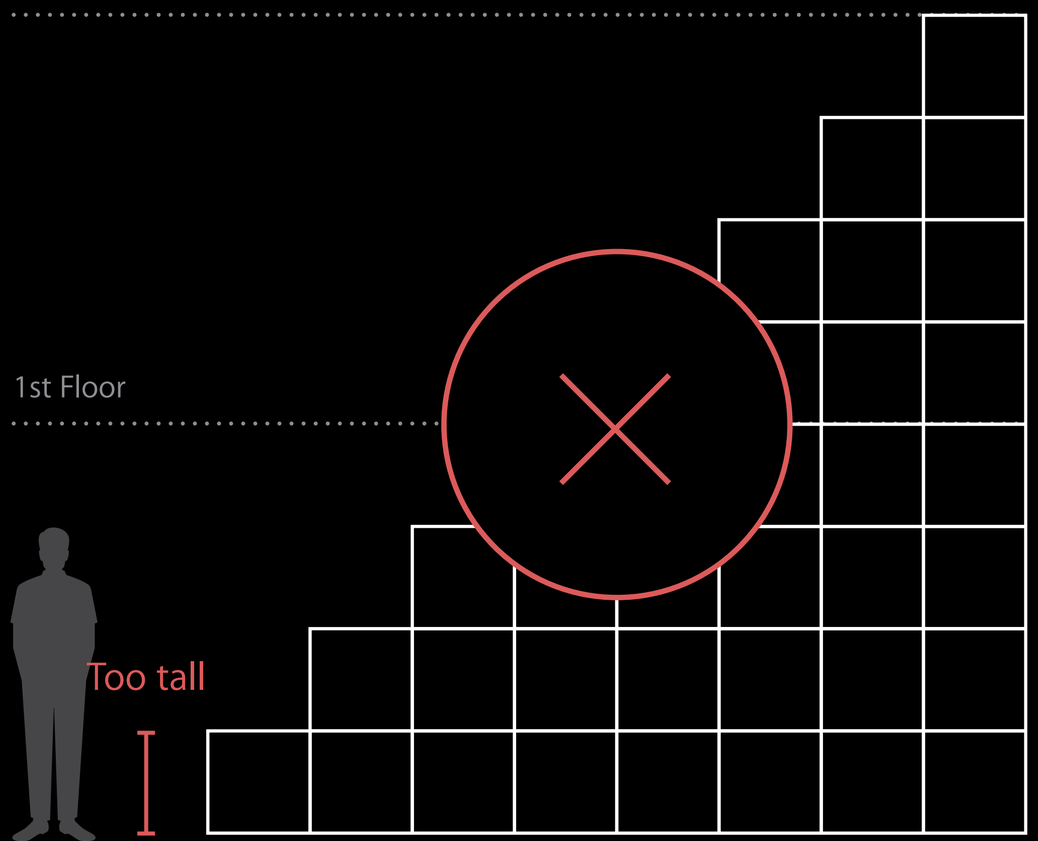
Precision and Color

2nd Floor

1st Floor



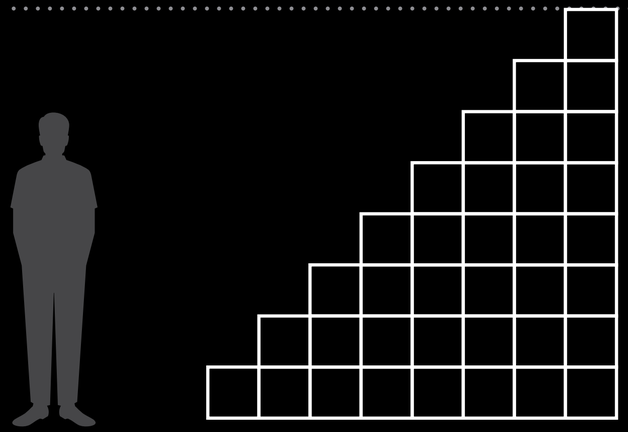
Precision and Color



Precision and Color

2nd Floor

1st Floor

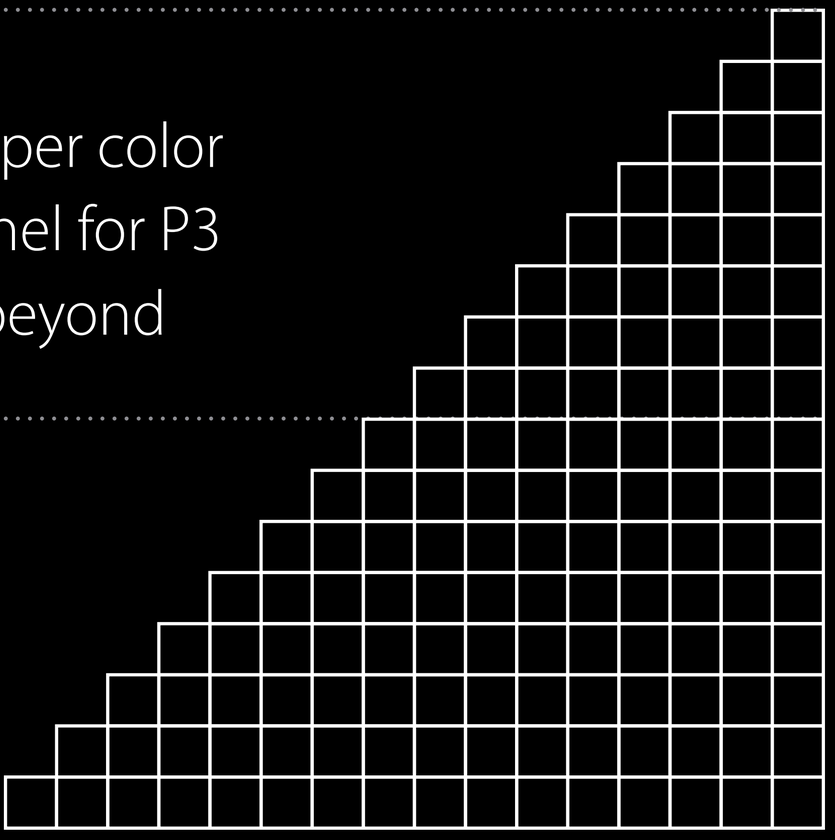


8bit per color
channel works
well for sRGB

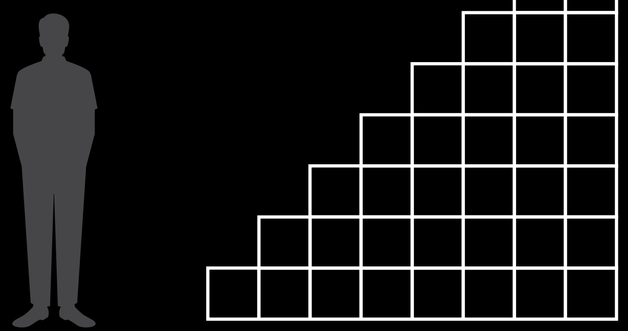
Precision and Color

2nd Floor

16bit per color
channel for P3
and beyond



1st Floor



8bit per color
channel works
well for sRGB

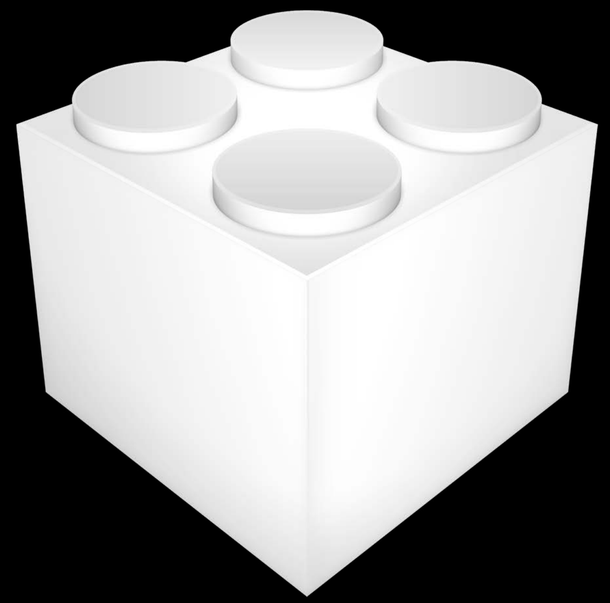
We've Got You Covered

Core Image

WebKit

ImageIO

SpriteKit



Core Graphics

Retina Display

SceneKit

Core Animation

UIKit

AppKit

Wide Gamut Content

Adapting your content workflow

Patrick Heynen
Cocoa Frameworks

Where does Wide Color come from?

It comes from you!

Apps and their Content

Application Content Types

Application Content Types

Static image resources

Application Content Types

Static image resources

Document and network image resources

Application Content Types

Static image resources

Document and network image resources

Advanced Media

Advances in iOS Photography	Mission	Tuesday 11:00AM
Editing Live Photos and Raw on iOS	Marina	Wednesday 3:15PM

Application Content Types

Static image resources

Document and network image resources

Advanced Media

GPU Textures

What's New in Metal, Part 2	Mission	Wednesday 11:00AM

Framing the Color Problem

Framing the Color Problem

App Content can come in a broad range of color richness from many sources

Framing the Color Problem

App Content can come in a broad range of color richness from many sources

Devices and Displays come in a broad range of color capabilities

Framing the Color Problem

App Content can come in a broad range of color richness from many sources

Devices and Displays come in a broad range of color capabilities

How do we bridge the differences?

Solving the Color Problem

Color Management

Solving the Color Problem

Color Management

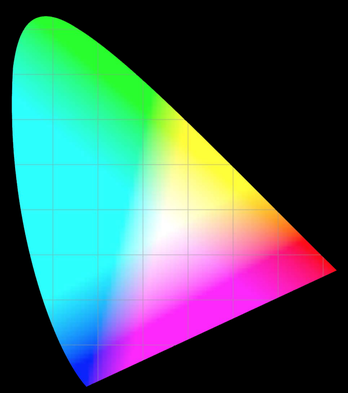
The job of color management is to ensure that an image looks the same on any output device no matter what color space it is encoded in.

How does it work?

Color Management

Color Management

Every image has an associated color space (color profile)



Color Management

Every image has an associated color space (color profile)

Color matching maps image colors to output device

Color Management

Every image has an associated color space (color profile)

Color matching maps image colors to output device

Not for free: Every pixel needs to be color matched

Color Management

Every image has an associated color space (color profile)

Color matching maps image colors to output device

Not for free: Every pixel needs to be color matched

Potentially lossy: Color fidelity is lost when output has smaller gamut

Good News!

Color Management

Color Management

Color matching operations are easily hardware accelerated

Color Management

Color matching operations are easily hardware accelerated

Works automatically via Quartz 2D, ColorSync and Core Animation

Color Management

Color matching operations are easily hardware accelerated

Works automatically via Quartz 2D, ColorSync and Core Animation

Properly tagged content requires no code to display properly

Platform Color Management

Platform Color Management

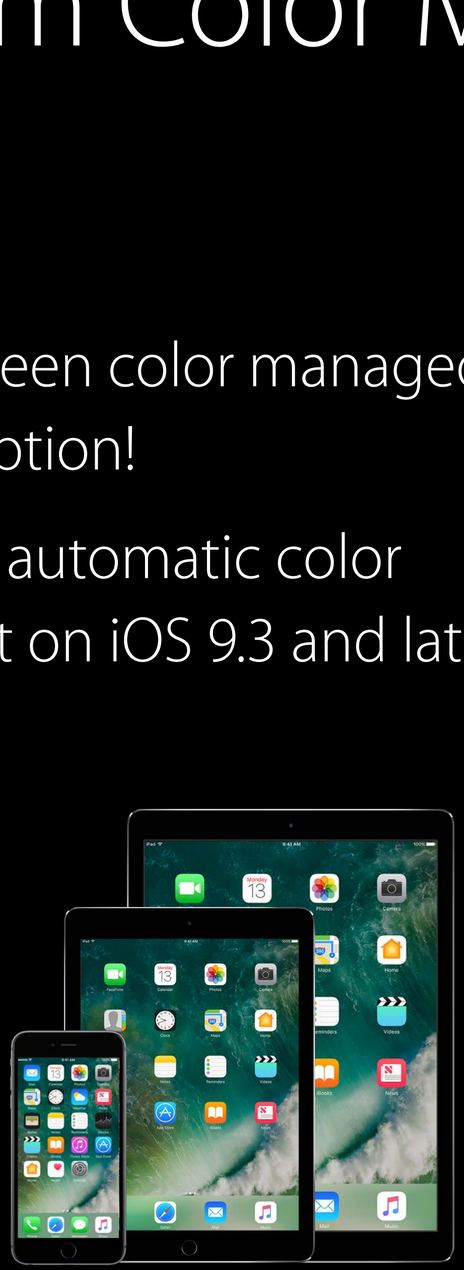
macOS has been color managed
since its inception!

Platform Color Management

NEW

macOS has been color managed
since its inception!

iOS supports automatic color
management on iOS 9.3 and later



Solving the Color Problem

Color Management


Design
Tools
Deploy


Design
Tools
Deploy


Design
Tools
Deploy

Design

Design Considerations for Wide Gamut

Design Considerations for Wide Gamut

Use wide gamut content where it makes sense

Design Considerations for Wide Gamut

Use wide gamut content where it makes sense

Use where vivid colors enhance the user experience

Design Considerations for Wide Gamut

Use wide gamut content where it makes sense

Use where vivid colors enhance the user experience

No need to change all content to P3

Design Considerations for Wide Gamut

Use wide gamut content where it makes sense

Use where vivid colors enhance the user experience

No need to change all content to P3

Toolchain support makes gradual opt-in of wide gamut content possible

Upgrading Content to Wide Color

Be careful when promoting an existing design file to wide color!

Upgrading Content to Wide Color

Be careful when promoting an existing design file to wide color!

- Don’t “assign” P3 profile

Upgrading Content to Wide Color

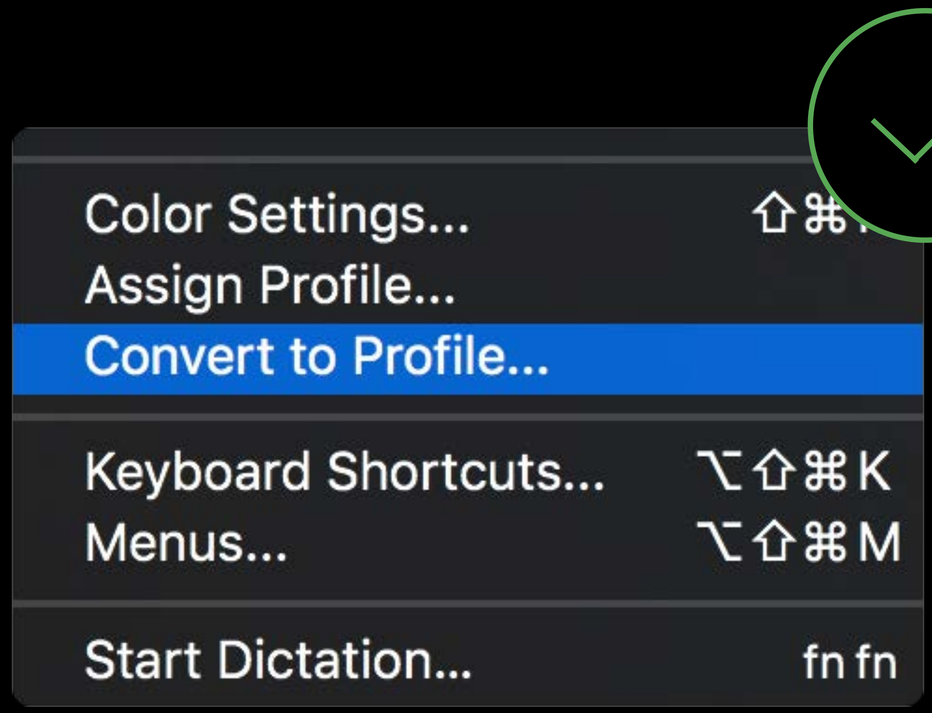
Be careful when promoting an existing design file to wide color!

- Don’t “assign” P3 profile

Upgrading Content to Wide Color

Be careful when promoting an existing design file to wide color!

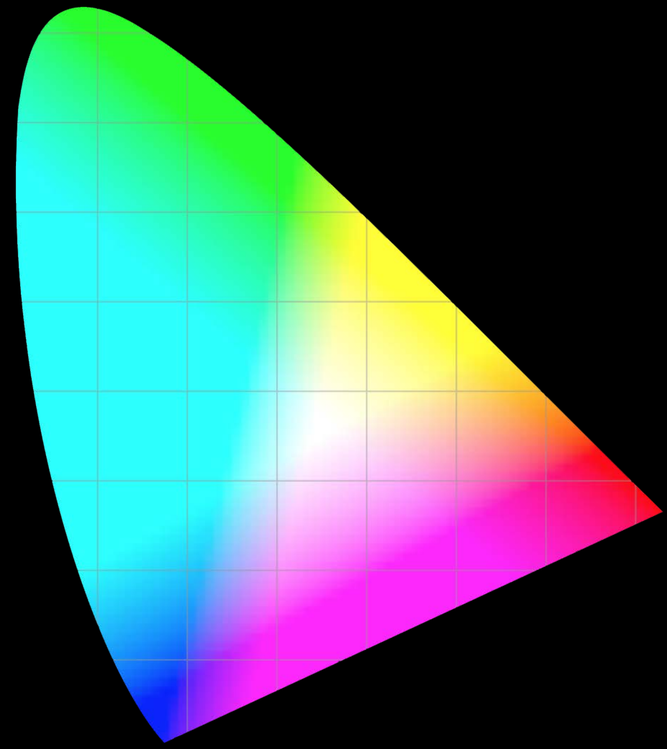
- Don't "assign" P3 profile
- Convert to P3 instead



File Formats and Color Profiles

File Formats and Color Profiles

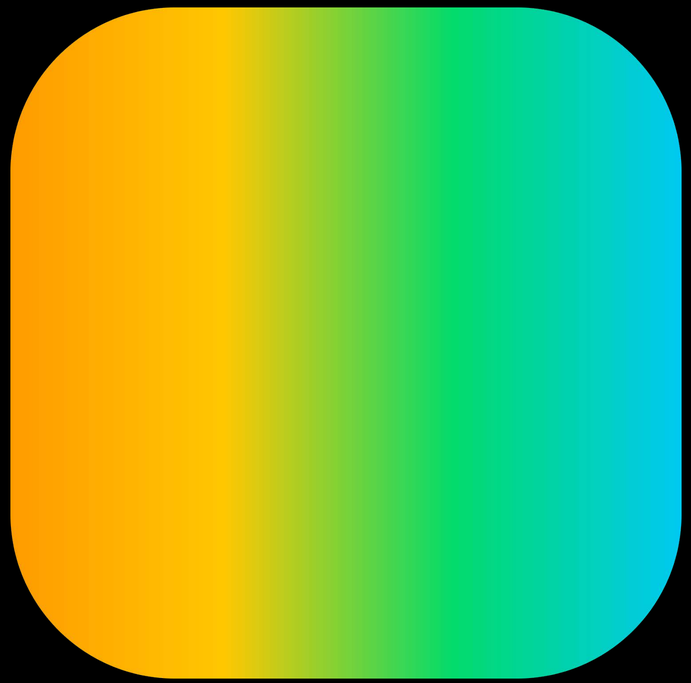
Use “Display P3” color profile for RGB working space



File Formats and Color Profiles

Use “Display P3” color profile for RGB working space

Use 16 bit per channel color mode

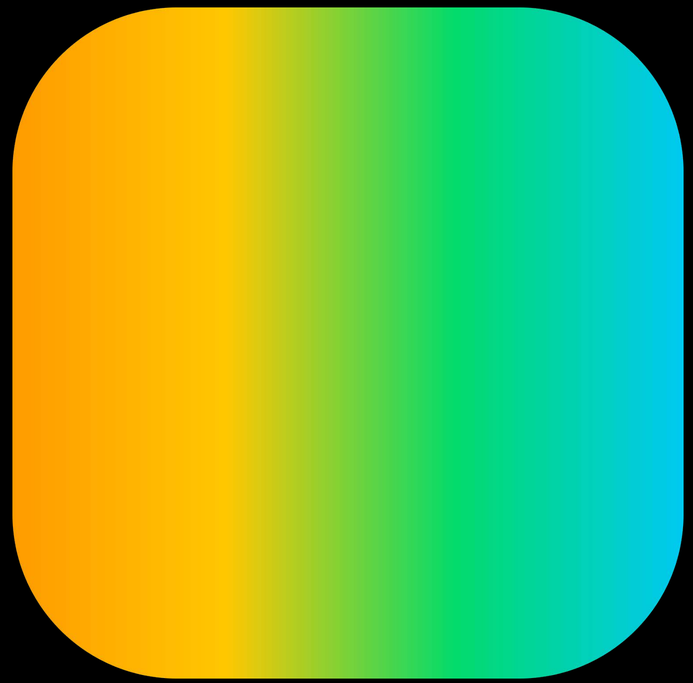


File Formats and Color Profiles

Use “Display P3” color profile for RGB working space

Use 16 bit per channel color mode

Use iMac (Late 2015) for accurate preview



File Formats and Color Profiles

Use “Display P3” color profile for RGB working space

Use 16 bit per channel color mode

Use iMac (Late 2015) for accurate preview

Export assets as 16bit PNG files with embedded “Display P3” ICC profile

File Formats and Color Profiles

Use “Display P3” color profile for RGB working space

Use 16 bit per channel color mode

Use iMac (Late 2015) for accurate preview

Export assets as 16bit PNG files with embedded “Display P3” ICC profile

Note: Save for Web and Export Assets do not support wide color!

File Formats and Color Profiles

Use “Display P3” color profile for RGB working space

Use 16 bit per channel color mode

Use iMac (Late 2015) for accurate preview

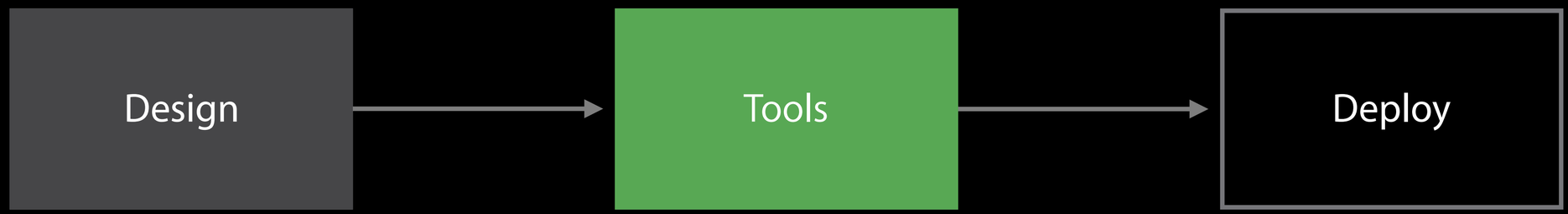
Export assets as 16bit PNG files with embedded “Display P3” ICC profile

Note: Save for Web and Export Assets do not support wide color!

Use Save As -> PNG format from 16bit source document as a workaround

Solving the Wide Color Problem

Solving the Wide Color Problem



Xcode Asset Catalogs

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help Tue 5:44 PM

D...mo 〉 Patrick's iPad Pro (9.7-inch) | DeepColorDemo | Build DeepColorDemo: **Succeeded** | Today at 5:08 PM

DeepColorDemo 〉 DeepColorDemo 〉 Assets.xcassets 〉 HueWheel 〉 Universal 2x

- DeepColorDemo
  - DeepColorDemo
    - Image Assets
    - AppDelegate.h
    - AppDelegate.m
    - ViewController.h
    - ViewController.m
    - Main.storyboard
    - Assets.xcassets
    - LaunchScreen.storyboard
    - Info.plist
    - Supporting Files
  - Products
    - DeepColorDemo.app

Filter

- AppIcon
- HueWheel
- Sweep

Filter

HueWheel

1x 2x 3x

Universal

Show Slicing

Image Set

Name: HueWheel

Render As: Default

Compression: Basic

Devices

All: Universal
iOS: iPhone
iPad
OS X: Mac
tvOS: Apple TV
watchOS: Apple Watch

Scale Factors: Individual Scales

Width: Any

Height: Any

Direction: Fixed

Color: Any

Memory: 1 GB
2 GB
4 GB

Graphics: Metal 1v2
Metal 2v2
Metal 3v1
Metal 3v2

Image

File Name: HueWheel-sRGB@2x.png

Compression: Basic (Inherited)

Image Size: 1024 × 1018 pixels

Color Space: sRGB IEC61966-2.1

Idiom: Universal

Asset Catalogs

Asset Catalogs

Best deployment vehicle for static assets

Asset Catalogs

Best deployment vehicle for static assets

Automatic color correction

Asset Catalogs

Best deployment vehicle for static assets

Automatic color correction

Automatic pixel format optimization

Asset Catalogs

Best deployment vehicle for static assets

Automatic color correction

Automatic pixel format optimization

App Slicing

Asset Catalog Enhancements

NEW

Asset Catalog Enhancements

NEW

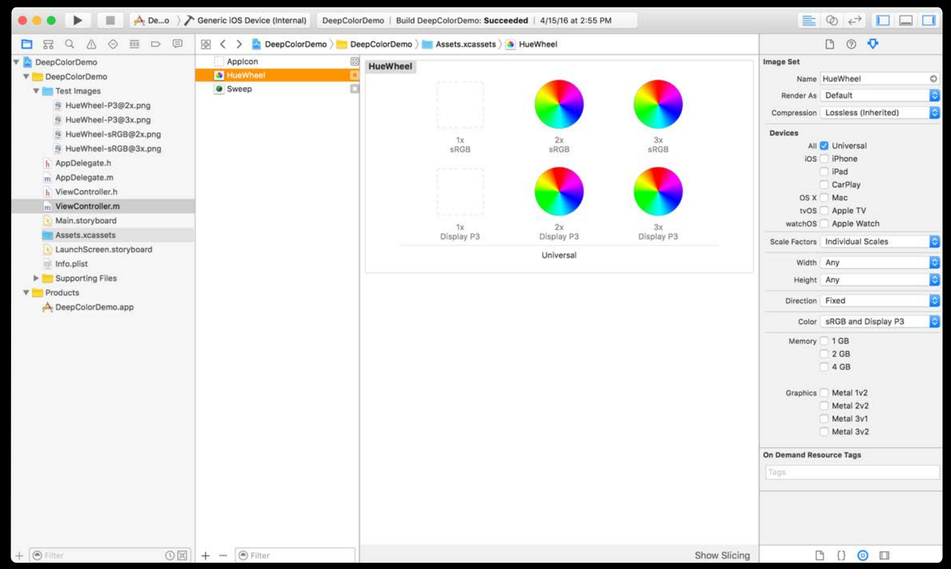
Support for 16 bit source content

Asset Catalog Enhancements

NEW

Support for 16 bit source content

Cataloging by display gamut

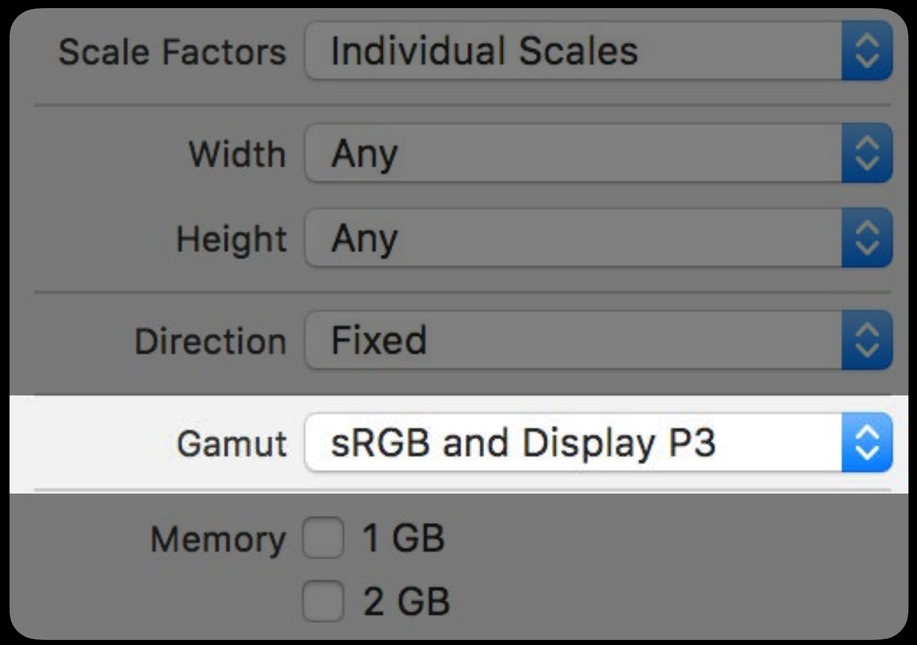


Asset Catalog Enhancements

NEW

Support for 16 bit source content

Cataloging by display gamut



Three Easy Choices

Choice #1:

Choice #1: Do Nothing!

No Changes to Asset Catalog

No Changes to Asset Catalog

8 bit sRGB

No Changes to Asset Catalog

Appearance will be preserved

8 bit sRGB

No Changes to Asset Catalog

Appearance will be preserved

No wide gamut colors

8 bit sRGB

Choice #2:

Choice #2: Upgrade to P3

Universal P3 Asset

Universal P3 Asset

Replace existing asset with upgraded asset

16 bit Display P3

Universal P3 Asset

Replace existing asset with upgraded asset

Automatic generation of sRGB derivative

16 bit Display P3

8bit sRGB

Universal P3 Asset

Replace existing asset with upgraded asset

Automatic generation of sRGB derivative

High quality color match and dither to 8bit

16 bit Display P3

8bit sRGB

Universal P3 Asset

Replace existing asset with upgraded asset

Automatic generation of sRGB derivative

High quality color match and dither to 8bit

Content selected according to display class

Choice #3:

Choice #3: Optimized Assets

Wide Color Optimized Assets

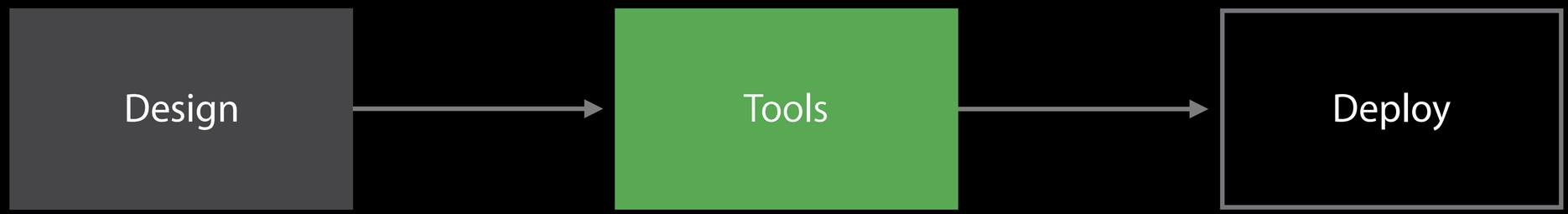
Wide Color Optimized Assets

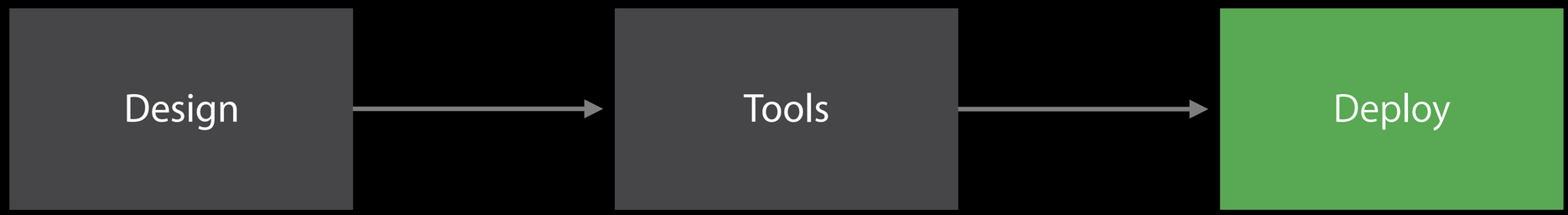
16 bit Display P3

8 bit sRGB

Demo

Creating a wide color asset


Design
Tools
Deploy


Design
Tools
Deploy

Asset Catalog Deployment

iOS

Asset Catalog Deployment

iOS

App Slicing will ensure that appropriate variant is delivered to a given device

Asset Catalog Deployment

iOS

App Slicing will ensure that appropriate variant is delivered to a given device

No payload cost for wide gamut content on sRGB devices

Asset Catalog Deployment

macOS

NSImage automatically selects best representation
for target display

Asset Catalog Deployment

macOS

NSImage automatically selects best representation
for target display

Content is refreshed when display changes
or properties change

Asset Catalog Storage

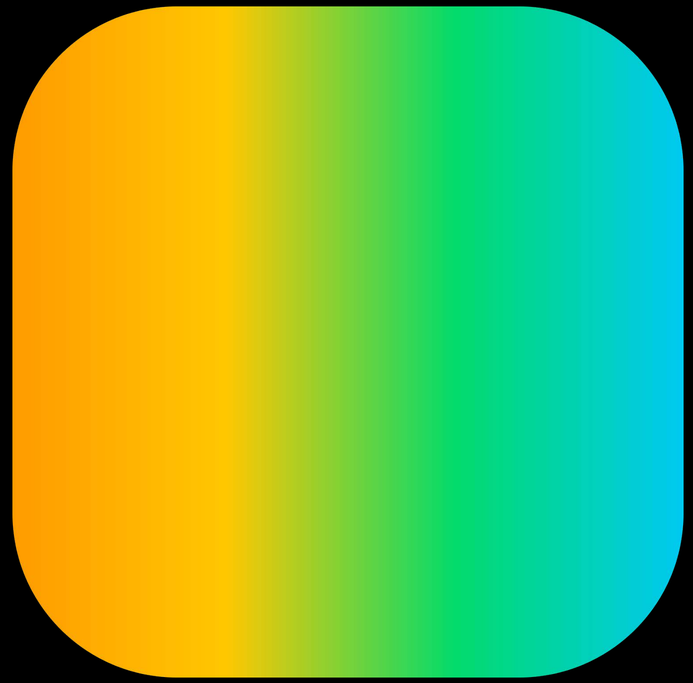
Asset Catalog Storage

High quality image conversions and
efficient storage

Asset Catalog Storage

High quality image conversions and efficient storage

16 bit per component storage for wide color content

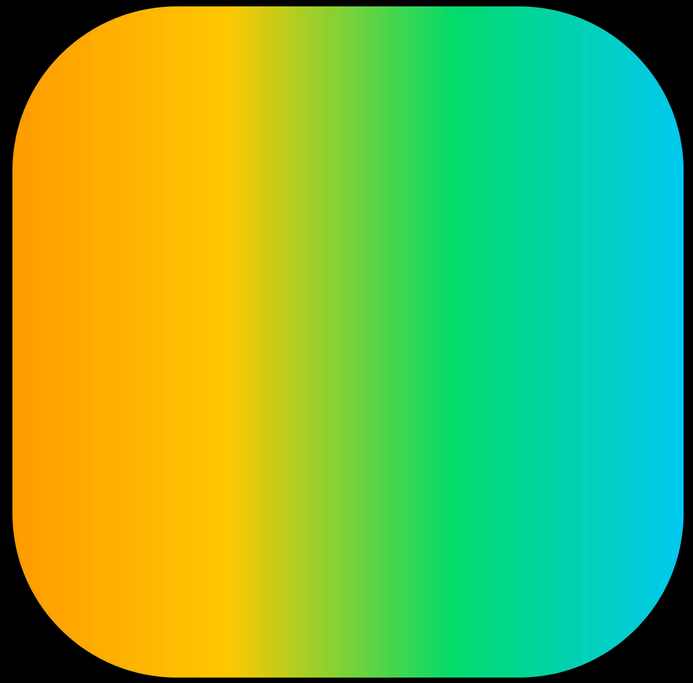


Asset Catalog Storage

High quality image conversions and efficient storage

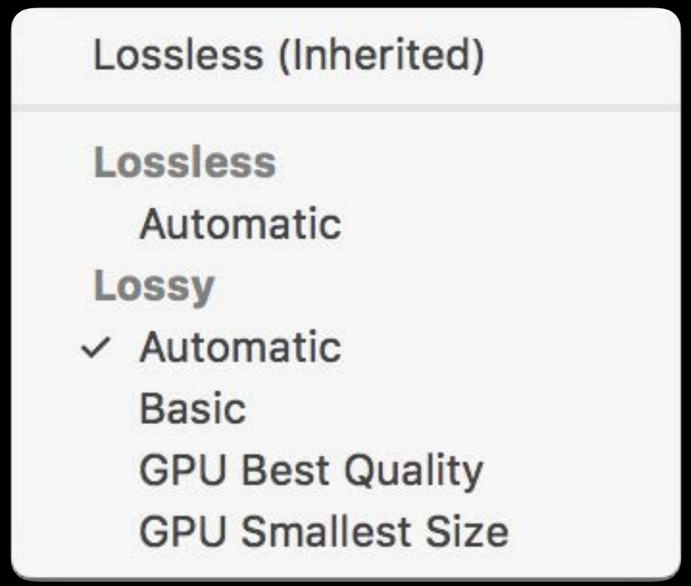
16 bit per component storage for wide color content

Content dependent image compression



Lossy Compression

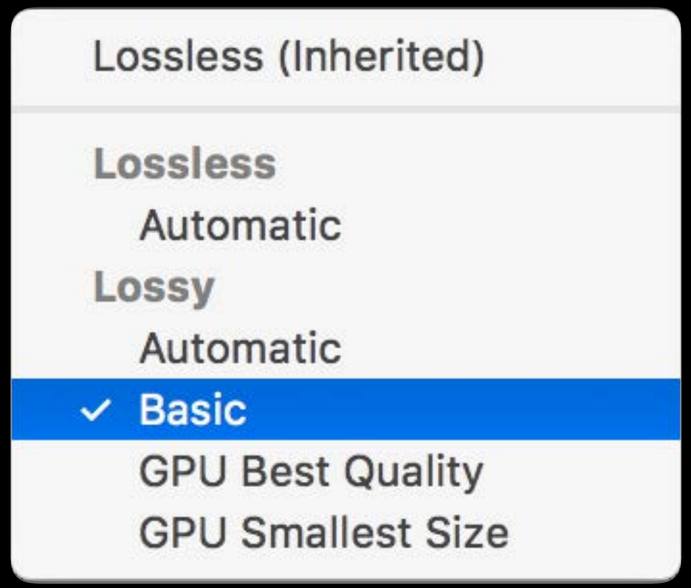
NEW



Lossy Compression

NEW

Basic compression (JPEG + alpha)

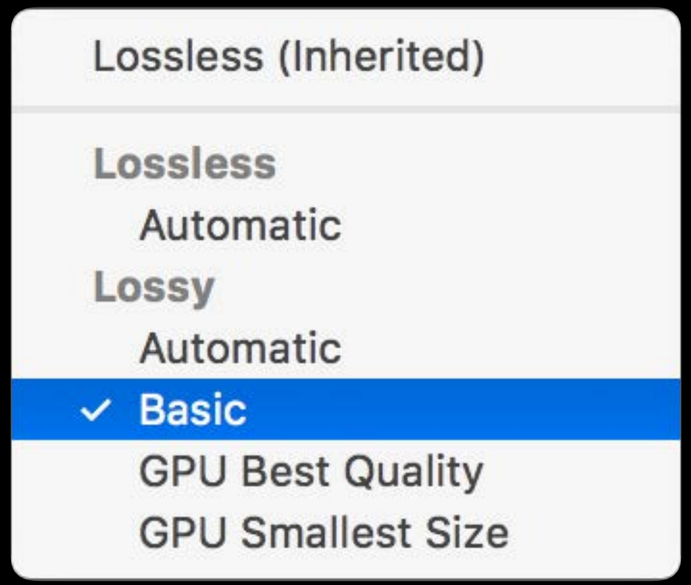


Lossy Compression

NEW

Basic compression (JPEG + alpha)

Excellent performance across all devices



GPU Compression (ASTC)

NEW

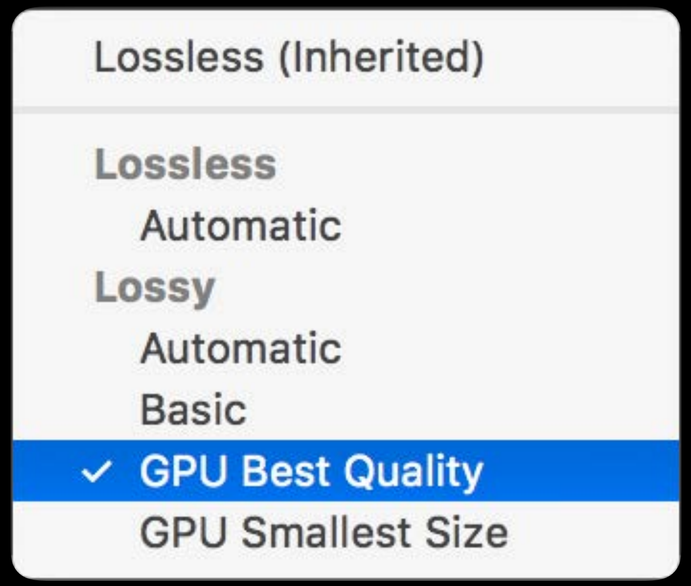
Advanced Scalable Texture Compression

GPU Compression (ASTC)

NEW

Advanced Scalable Texture Compression

- Optimize for best quality or smallest size (4bpp or 1bpp)

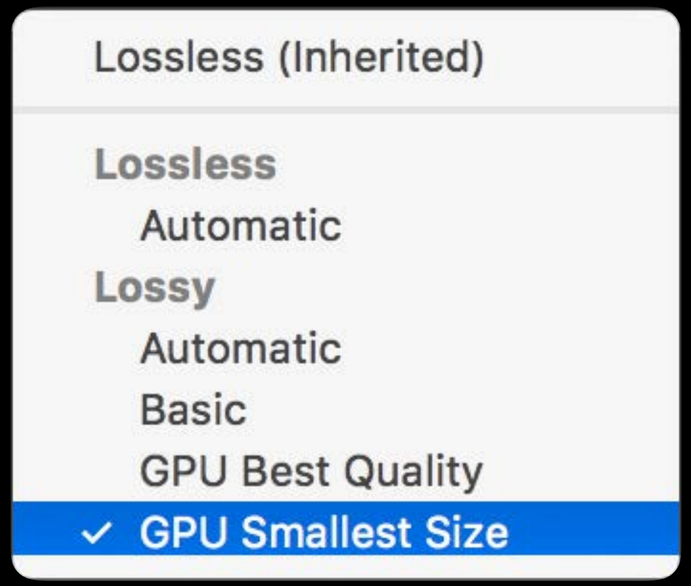


GPU Compression (ASTC)

NEW

Advanced Scalable Texture Compression

- Optimize for best quality or smallest size (4bpp or 1bpp)

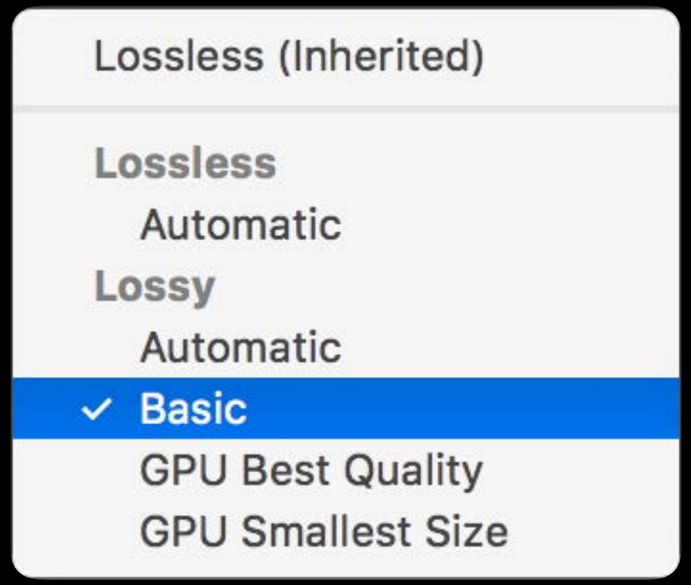


GPU Compression (ASTC)

NEW

Advanced Scalable Texture Compression

- Optimize for best quality or smallest size (4bpp or 1bpp)
- Automatic software fallback for devices lacking ASTC graphics capability



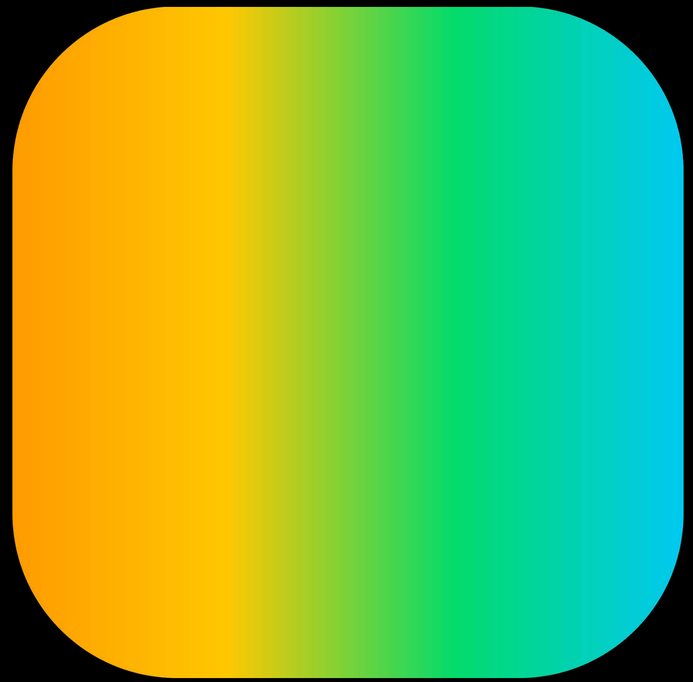
GPU Compression and Wide Color Assets

GPU compression uses ASTC LDR
texture compression

GPU Compression and Wide Color Assets

GPU compression uses ASTC LDR texture compression

Wide content needs to be reduced to 8bits before compression



GPU Compression and Wide Color Assets

GPU compression uses ASTC LDR texture compression

Wide content needs to be reduced to 8bits before compression

Xcode performs a high quality dither automatically

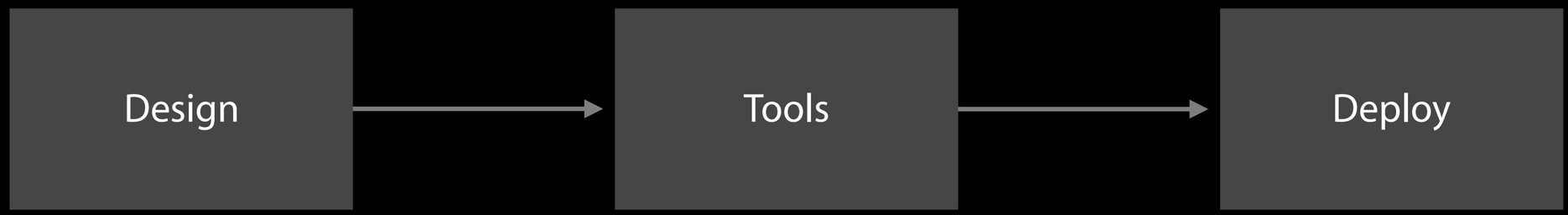
GPU Compression and Wide Color Assets

GPU compression uses ASTC LDR texture compression

Wide content needs to be reduced to 8bits before compression

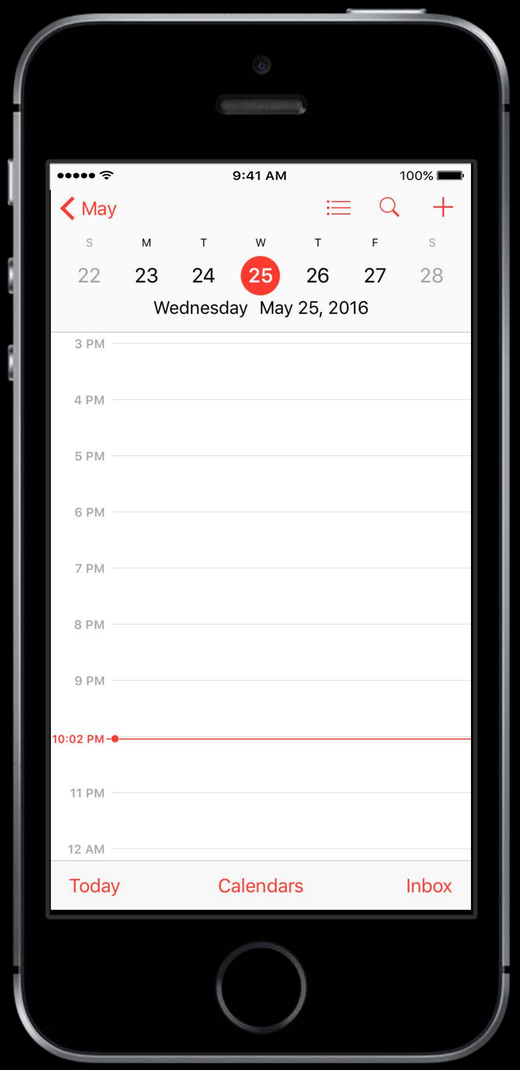
Xcode performs a high quality dither automatically

Wide gamut colors preserved by encoding in Display P3 color space


Design
Tools
Deploy

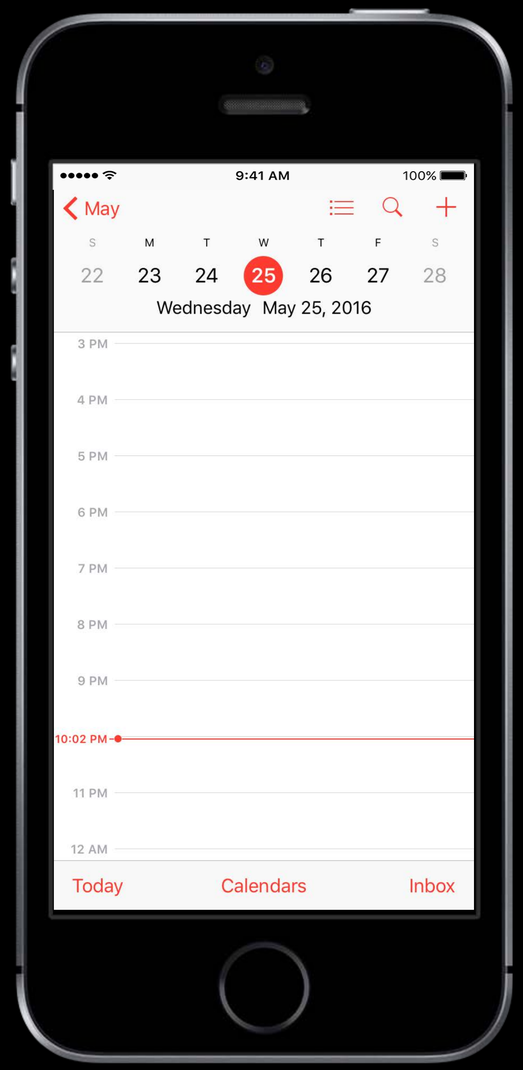
What about Colors?

Colors in UI


9:41 AM
100%
May
S M T W T F S
22 23 24 25 26 27 28
Wednesday May 25, 2016
3 PM
4 PM
5 PM
6 PM
7 PM
8 PM
9 PM
10:02 PM
11 PM
12 AM
Today
Calendars
Inbox

Colors in UI

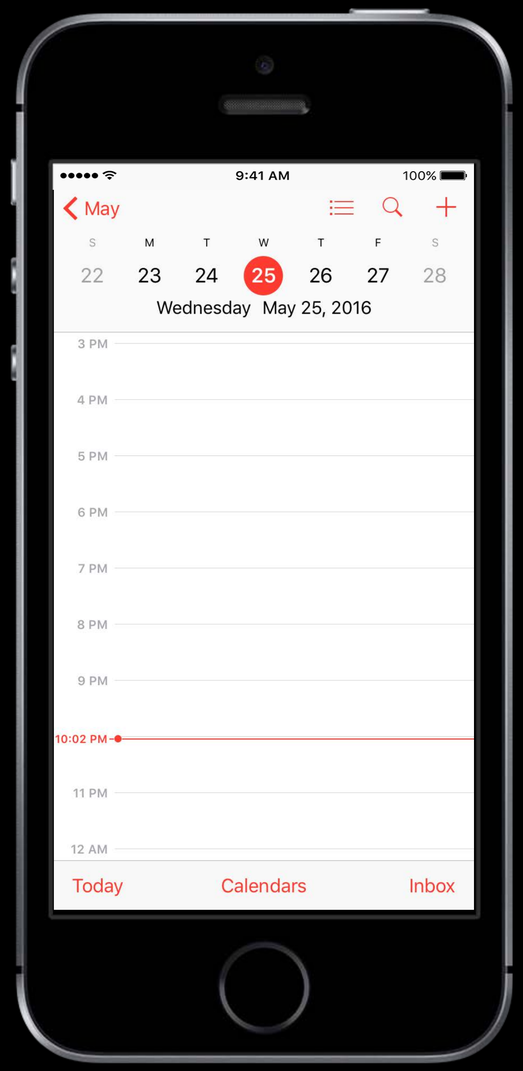
Most pixels on screen are solid colors



Colors in UI

Most pixels on screen are solid colors

Wide gamut colors present new challenges



Talking Colors

Color Specification

Color Specification

Colors are usually communicated with an assumed sRGB color space

Color Specification

Colors are usually communicated with an assumed sRGB color space

RGB (128, 45, 56)

#FF0456

This is no longer sufficient
for Wide Gamut colors

Color Specification

Be specific about color space!

Color Specification

Be specific about color space!

Use Display P3 instead of sRGB when working with wide gamut designs

Color Specification

Be specific about color space!

Use Display P3 instead of sRGB when working with wide gamut designs

Use floating point for more precision

Color Specification

Be specific about color space!

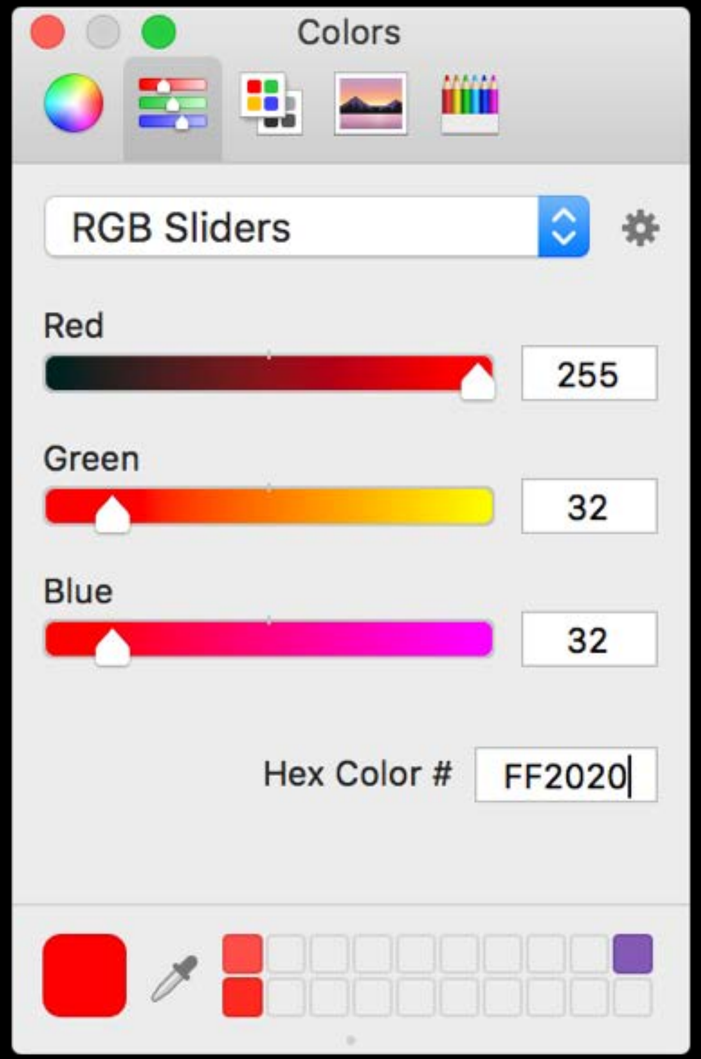
Use Display P3 instead of sRGB when working with wide gamut designs

Use floating point for more precision

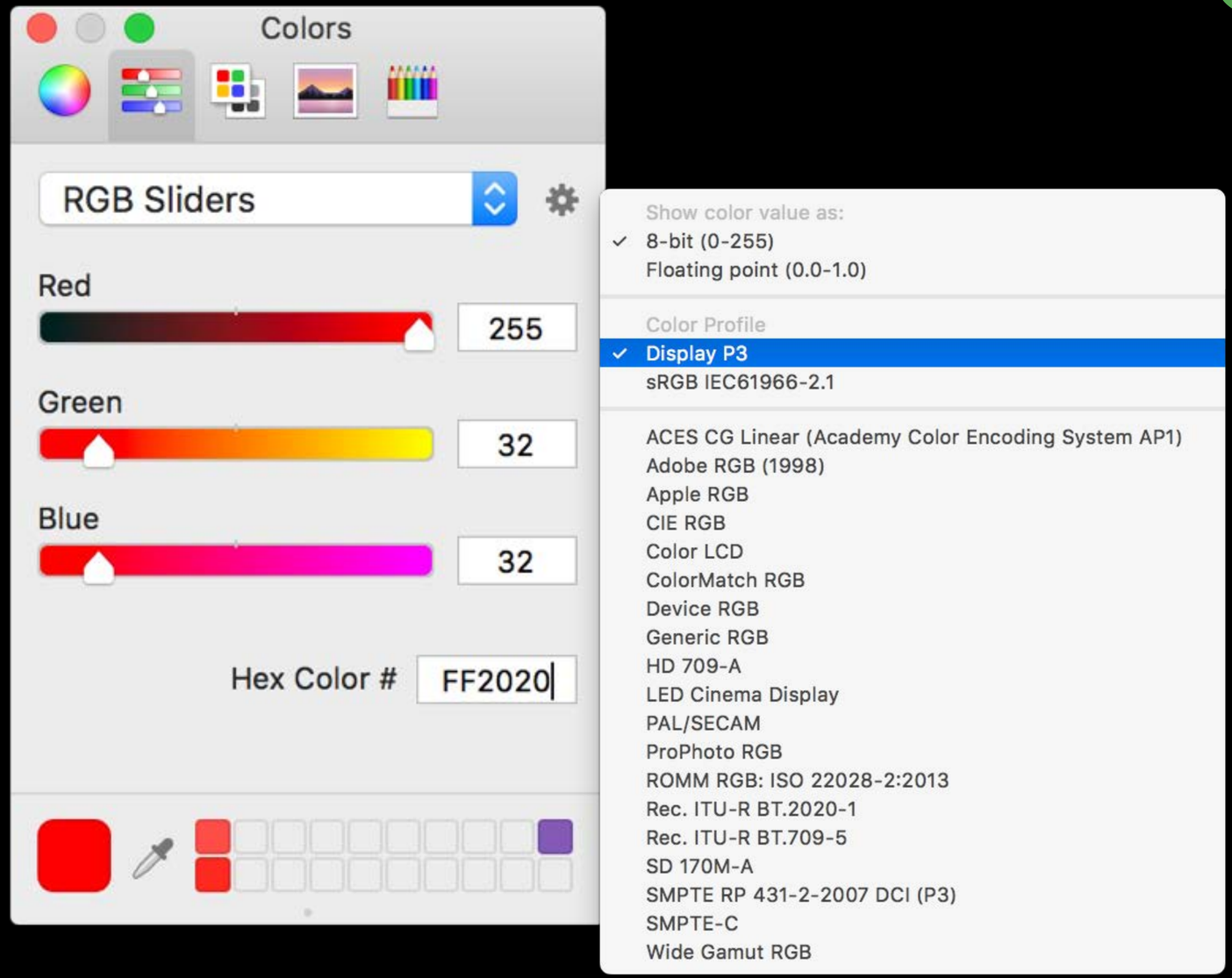
P3 (255, 128, 191)

P3 (1.0, 0.5, 0.75)

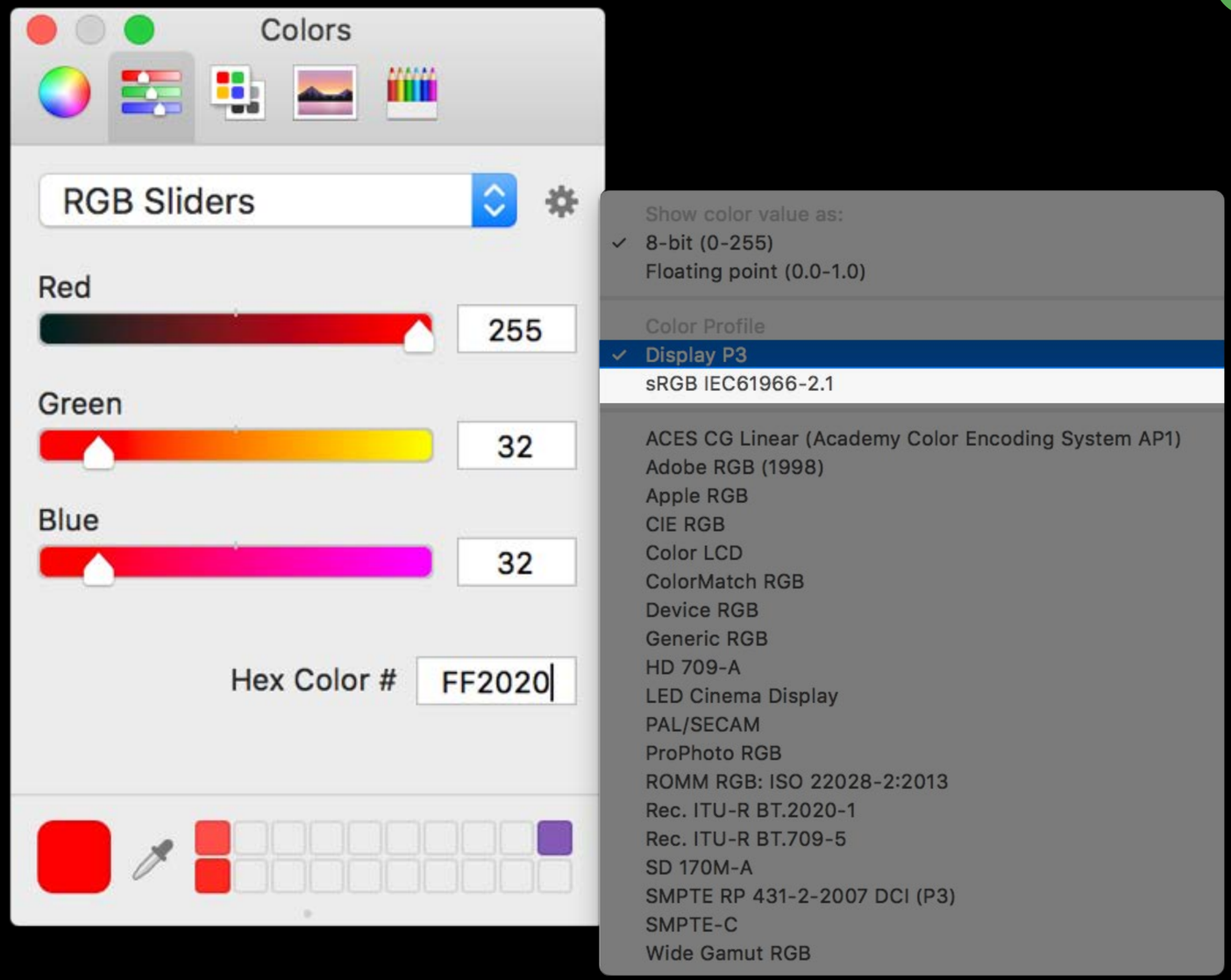
Picking Colors


Colors
RGB Sliders
Red
255
Green
32
Blue
32
Hex Color #
FF2020

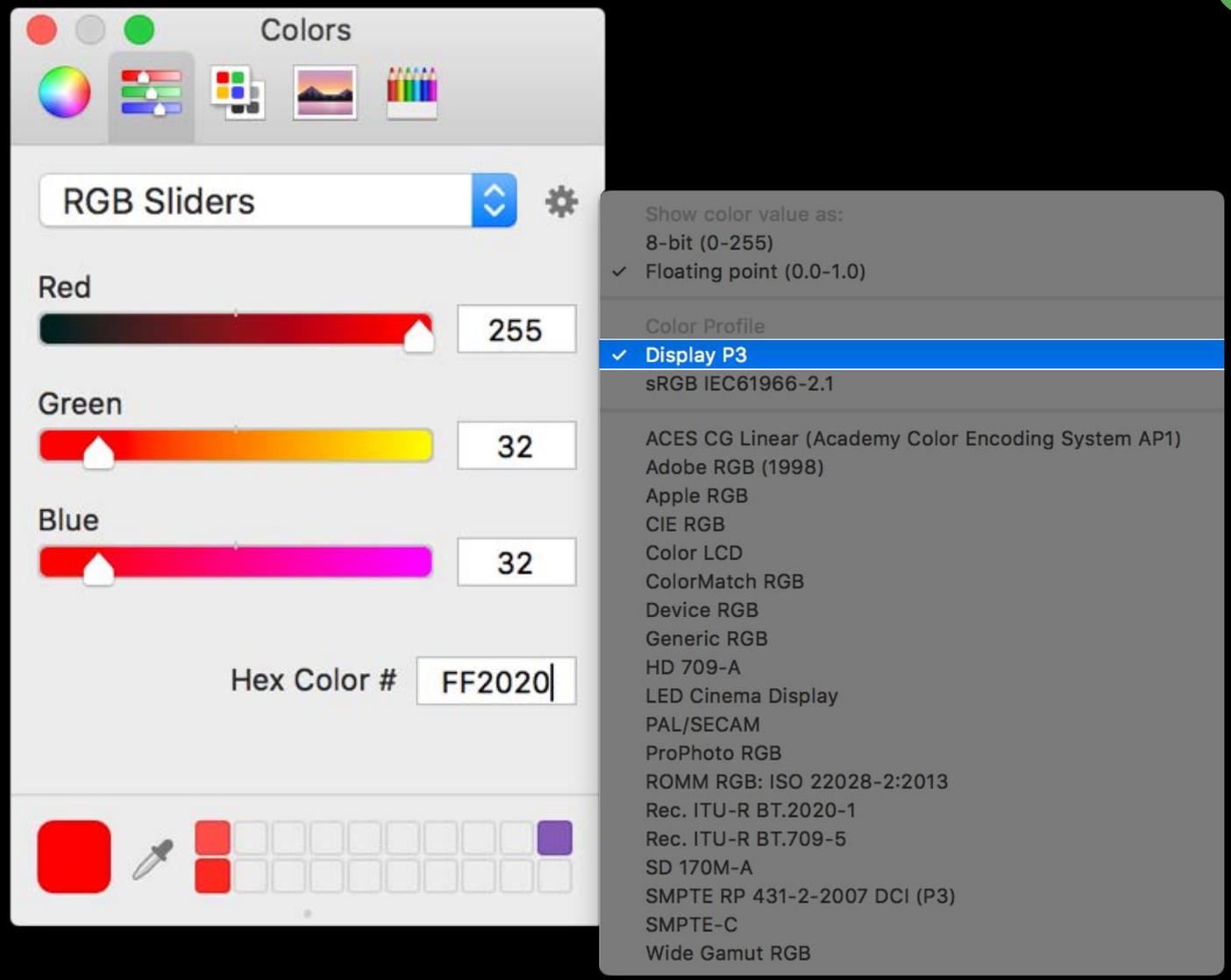
NEW


Colors
RGB Sliders
Red
255
Green
32
Blue
32
Hex Color #
FF2020
Show color value as:
8-bit (0-255)
Floating point (0.0-1.0)
Color Profile
Display P3
sRGB IEC61966-2.1
ACES CG Linear (Academy Color Encoding System AP1)
Adobe RGB (1998)
Apple RGB
CIE RGB
Color LCD
ColorMatch RGB
Device RGB
Generic RGB
HD 709-A
LED Cinema Display
PAL/SECAM
ProPhoto RGB
ROMM RGB: ISO 22028-2:2013
Rec. ITU-R BT.2020-1
Rec. ITU-R BT.709-5
SD 170M-A
SMPTE RP 431-2-2007 DCI (P3)
SMPTE-C
Wide Gamut RGB

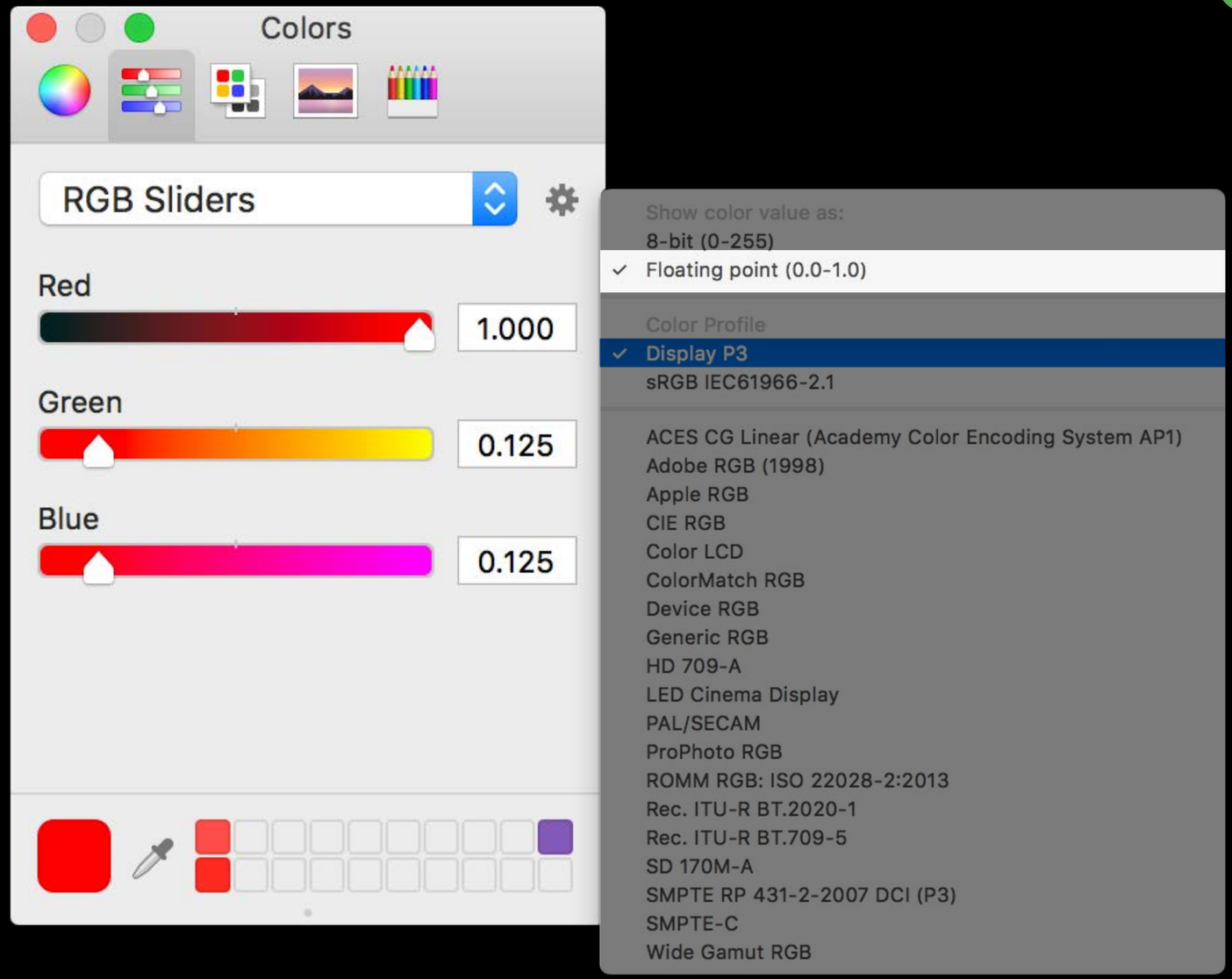
NEW


Colors
RGB Sliders
Red
255
Green
32
Blue
32
Hex Color #
FF2020
Show color value as:
8-bit (0-255)
Floating point (0.0-1.0)
Color Profile
Display P3
sRGB IEC61966-2.1
ACES CG Linear (Academy Color Encoding System AP1)
Adobe RGB (1998)
Apple RGB
CIE RGB
Color LCD
ColorMatch RGB
Device RGB
Generic RGB
HD 709-A
LED Cinema Display
PAL/SECAM
ProPhoto RGB
ROMM RGB: ISO 22028-2:2013
Rec. ITU-R BT.2020-1
Rec. ITU-R BT.709-5
SD 170M-A
SMPTE RP 431-2-2007 DCI (P3)
SMPTE-C
Wide Gamut RGB

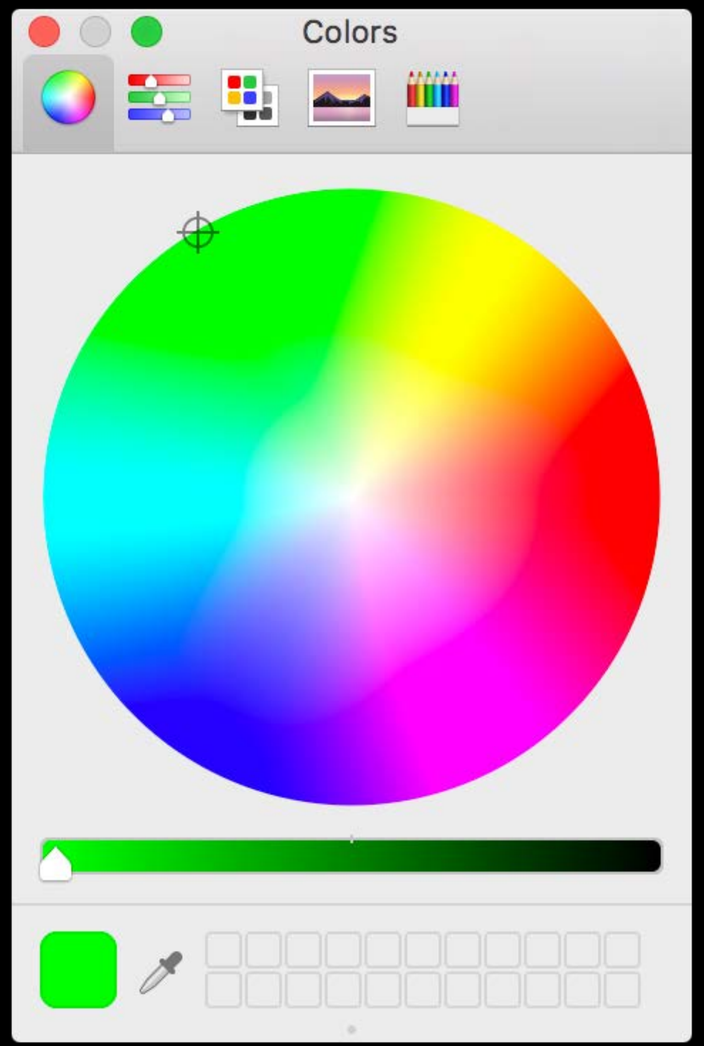
NEW


Colors
RGB Sliders
Red
255
Green
32
Blue
32
Hex Color #
FF2020
Show color value as:
8-bit (0-255)
Floating point (0.0-1.0)
Color Profile
Display P3
sRGB IEC61966-2.1
ACES CG Linear (Academy Color Encoding System AP1)
Adobe RGB (1998)
Apple RGB
CIE RGB
Color LCD
ColorMatch RGB
Device RGB
Generic RGB
HD 709-A
LED Cinema Display
PAL/SECAM
ProPhoto RGB
ROMM RGB: ISO 22028-2:2013
Rec. ITU-R BT.2020-1
Rec. ITU-R BT.709-5
SD 170M-A
SMPTE RP 431-2-2007 DCI (P3)
SMPTE-C
Wide Gamut RGB

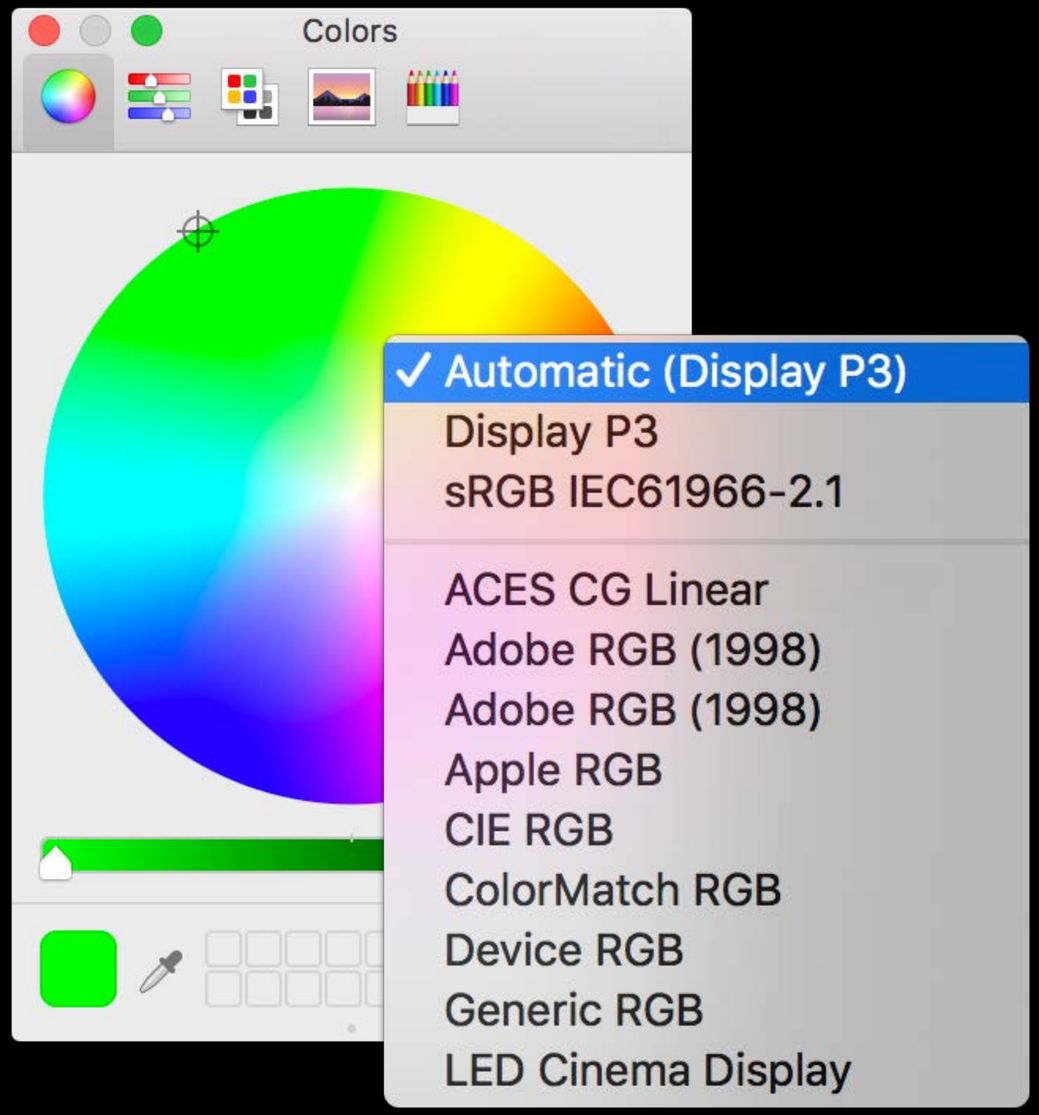
NEW


Colors
RGB Sliders
Red
1.000
Green
0.125
Blue
0.125
Show color value as:
8-bit (0-255)
Floating point (0.0-1.0)
Color Profile
Display P3
sRGB IEC61966-2.1
ACES CG Linear (Academy Color Encoding System AP1)
Adobe RGB (1998)
Apple RGB
CIE RGB
Color LCD
ColorMatch RGB
Device RGB
Generic RGB
HD 709-A
LED Cinema Display
PAL/SECAM
ProPhoto RGB
ROMM RGB: ISO 22028-2:2013
Rec. ITU-R BT.2020-1
Rec. ITU-R BT.709-5
SD 170M-A
SMPTE RP 431-2-2007 DCI (P3)
SMPTE-C
Wide Gamut RGB

NEW



NEW


Colors
✓ Automatic (Display P3)
Display P3
sRGB IEC61966-2.1
ACES CG Linear
Adobe RGB (1998)
Adobe RGB (1998)
Apple RGB
CIE RGB
ColorMatch RGB
Device RGB
Generic RGB
LED Cinema Display

Coding Colors

Constructing Wide Gamut Colors

NEW

Constructing Wide Gamut Colors

NEW

```
NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0)

UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0)
```

Constructing Extended Range sRGB Colors

NEW

Constructing Extended Range sRGB Colors

NEW

```
NSColor(red: 1.1, green: -.25, blue: 0.0, alpha: 1.0)

UIColor(red: 1.1, green: -.25, blue: 0.0, alpha: 1.0)
```

Storing Colors

Focus: Archiving Colors

Special care needs to be taken when storing wide gamut colors in document data

Focus: Archiving Colors

Special care needs to be taken when storing wide gamut colors in document data

Beware of assumed color spaces!

Focus: Archiving Colors

Special care needs to be taken when storing wide gamut colors in document data

Beware of assumed color spaces!

Consider encoding “compatible” sRGB color alongside new wide gamut colors

Focus: Archiving Colors

Special care needs to be taken when storing wide gamut colors in document data

Beware of assumed color spaces!

Consider encoding "compatible" sRGB color alongside new wide gamut colors

iOS: Use `CGColor.convert()` API to convert colors to sRGB

Focus: Archiving Colors

Special care needs to be taken when storing wide gamut colors in document data

Beware of assumed color spaces!

Consider encoding "compatible" sRGB color alongside new wide gamut colors

iOS: Use `CGColor.convert()` API to convert colors to sRGB

macOS: Use `NSColor.usingColorSpace()` API to convert colors to sRGB

Case Study: RTF / TextEdit

Case Study: RTF / TextEdit

Existing

```
{\colortbl;\red0\green0\blue255;\red0\green128\blue255;\red255\green14\blue12;}
```

Case Study: RTF / TextEdit

Existing

```
{\colortbl;\red0\green0\blue255;\red0\green128\blue255;\red255\green14\blue12;}
```

Case Study: RTF / TextEdit

Existing

```
{\colortbl;\red0\green0\blue255;\red0\green128\blue255;\red255\green14\blue12;}
```

New

```
{\*\expandedcolortbl;\cssrgb\c50000\c100000\c0;\cspthree\c100000\c50000\
c0;\cssrgb\c-20000\c140000\c0;}
```

Case Study: RTF / TextEdit

Existing

```
{\colortbl;\red0\green0\blue255;\red0\green128\blue255;\red255\green14\blue12;}
```

New

```
{\*\expandedcolortbl;\cssrgb\c50000\c100000\c0;\cspthree\c100000\c50000\
c0;\cssrgb\c-20000\c140000\c0;}
```

Surfing with Colors

Colors in the Web

All tagged images are color matched

Colors in the Web

All tagged images are color matched

Media Queries to resolve assets between P3 and sRGB capable devices

```
<picture>
    <source srcset="flower-p3.jpg" media="(color-gamut: p3)">
    <source srcset="flower-rgb.jpg">
</picture>
```

Colors in the Web

All tagged images are color matched

Media Queries to resolve assets between P3 and sRGB capable devices

```
<picture>
    <source srcset="flower-p3.jpg" media="(color-gamut: p3)">
    <source srcset="flower-rgb.jpg">
</picture>
```

WebKit proposal for CSS colors in other color spaces

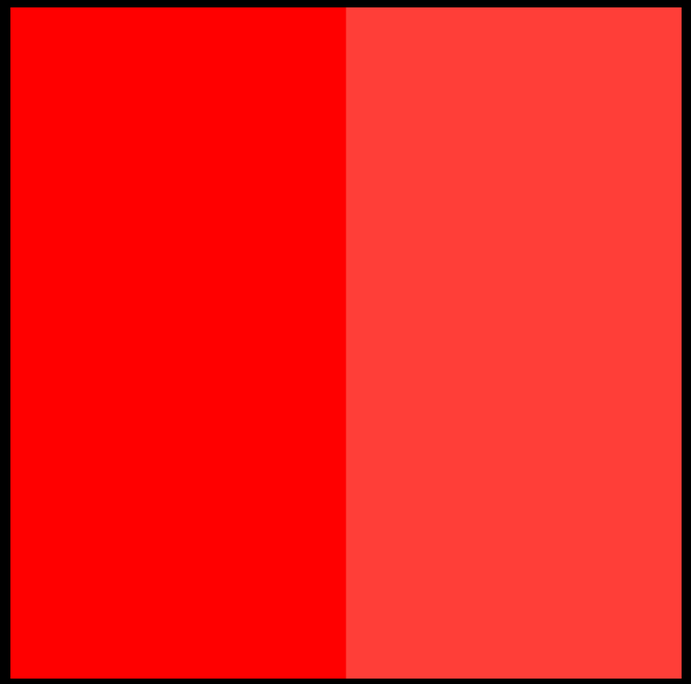
Wide Gamut Rendering

Optimizing your app's drawing for wide gamut displays

Steve Holt UIKit

Drawing with Wide Color

Drawing with Wide Color



(Output simulated)

Drawing with Wide Color

Cocoa

Drawing with Wide Color

Cocoa

```
NSImage.init(size: NSSize, flipped drawingHandlerShouldBeCalledWithFlippedContext: Bool,
drawingHandler: (NSRect) -> Bool)
```

Drawing with Wide Color

Cocoa

```
NSImage.init(size: NSSize, flipped drawingHandlerShouldBeCalledWithFlippedContext: Bool,
drawingHandler: (NSRect) -> Bool)
```

- DrawingHandler called with current NSGraphicsContext instance

Drawing with Wide Color

Cocoa

```
NSImage.init(size: NSSize, flipped drawingHandlerShouldBeCalledWithFlippedContext: Bool,
drawingHandler: (NSRect) -> Bool)
```

- DrawingHandler called with current NSGraphicsContext instance
- Use anywhere you would NSImage

```
// Drawing with Wide Color
// Cocoa

import Cocoa

let size = CGSize(width: 250, height: 250)
let image = NSImage(size: size, flipped: false) { drawRect -> Bool in
    let rects = drawRect.divide(drawRect.size.width/2, fromEdge: .minXEdge)

    NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.slice).fill()

    NSColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.remainder).fill()
    return true
}
```

```
// Drawing with Wide Color
// Cocoa

import Cocoa

let size = CGSize(width: 250, height: 250)
let image = NSImage(size: size, flipped: false) { drawRect -> Bool in
    let rects = drawRect.divide(drawRect.size.width/2, fromEdge: .minXEdge)

    NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.slice).fill()

    NSColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.remainder).fill()
    return true
}
```

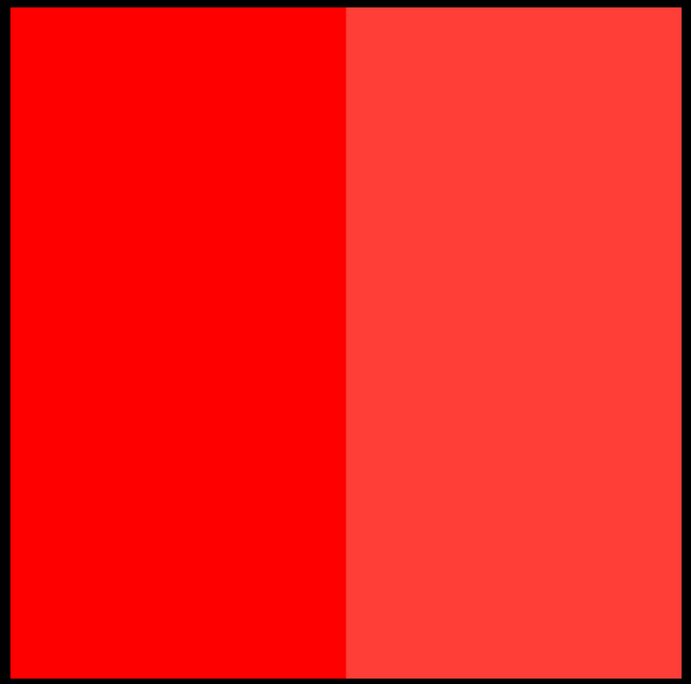
```
// Drawing with Wide Color
// Cocoa

import Cocoa

let size = CGSize(width: 250, height: 250)
let image = NSImage(size: size, flipped: false) { drawRect -> Bool in
    let rects = drawRect.divide(drawRect.size.width/2, fromEdge: .minXEdge)

    NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.slice).fill()

    NSColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.remainder).fill()
    return true
}
```

```
// Drawing with Wide Color
// Cocoa

import Cocoa

let size = CGSize(width: 250, height: 250)
let image = NSImage(size: size, flipped: false) { drawRect -> Bool in
    let rects = drawRect.divide(drawRect.size.width/2, fromEdge: .minXEdge)

    NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.slice).fill()

    NSColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.remainder).fill()
    return true
}
```

```swift
// Drawing with Wide Color
// Cocoa

import Cocoa

let size = CGSize(width: 250, height: 250)
let image = NSImage(size: size, flipped: false) { drawRect -> Bool in
    let rects = drawRect.divide(drawRect.size.width/2, fromEdge: .minXEdge)

    NSColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.slice).fill()

    NSColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    NSBezierPath(rect: rects.remainder).fill()
    return true
}
```

Drawing with Wide Color

Cocoa



(Output simulated)

Drawing with Wide Color

CocoaTouch

```
UIGraphicsBeginImageContext(_ size: CGSize)
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

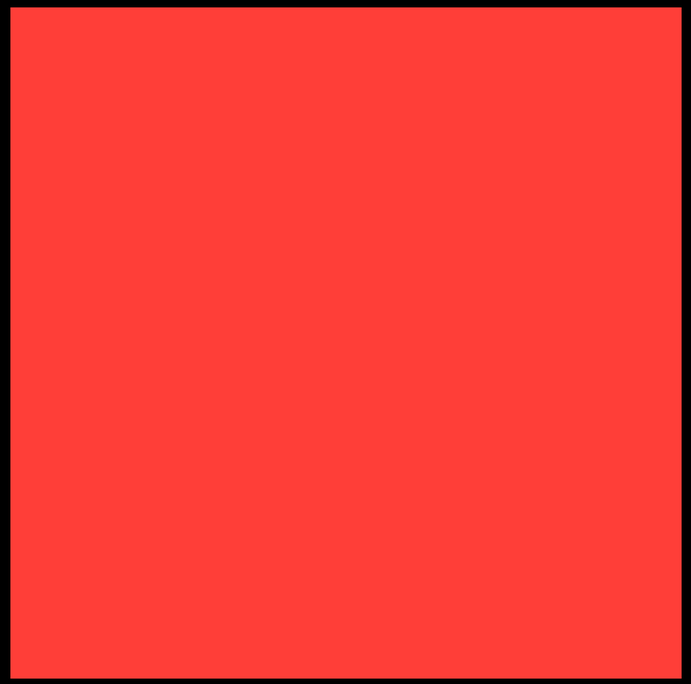
```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

UIGraphicsEndImageContext()
let image = UIGraphicsGetImageFromCurrentImageContext()
```

```swift
// Drawing with Wide Color
// CocoaTouch

import UIKit

let size = CGSize(width: 250, height: 250)
UIGraphicsBeginImageContext(size)

let rects = CGRect(origin: .zero, size: size).divide(size.width/2, fromEdge: .minXEdge)
UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
UIRectFill(rects.slice)

UIColor.red().set()
UIRectFill(rects.remainder)

let image = UIGraphicsGetImageFromCurrentImageContext()
UIGraphicsEndImageContext()
```

Drawing with Wide Color

CocoaTouch



(Output simulated)

“The format for the bitmap is a ARGB 32-bit integer pixel format using host-byte order.”

From the UIKit Function Reference

Drawing with Wide Color

CocoaTouch

```
UIGraphicsBeginImageContext(_ size: CGSize)
```

Drawing with Wide Color

CocoaTouch

```
UIGraphicsBeginImageContext(_ size: CGSize)
```

- Cannot create contexts with more than 8 bits per color channel

Drawing with Wide Color

CocoaTouch

```
UIGraphicsBeginImageContext(_ size: CGSize)
```

- Cannot create contexts with more than 8 bits per color channel
- Cannot represent colors in extended range sRGB

Drawing with Wide Color

CocoaTouch

```
UIGraphicsBeginImageContext(_ size: CGSize)
```

- Cannot create contexts with more than 8 bits per color channel
- Cannot represent colors in extended range sRGB
- Existing interface has no ability to create a context in non-sRGB color space

```
- UIGraphicsBeginImageContextWithOptions(_ size: CGSize, _ opaque: Bool, _ scale: CGFloat)
```

Drawing with Wide Color

CocoaTouch

NEW

```
UIGraphicsImageRenderer(size: CGSize)
```

```swift
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

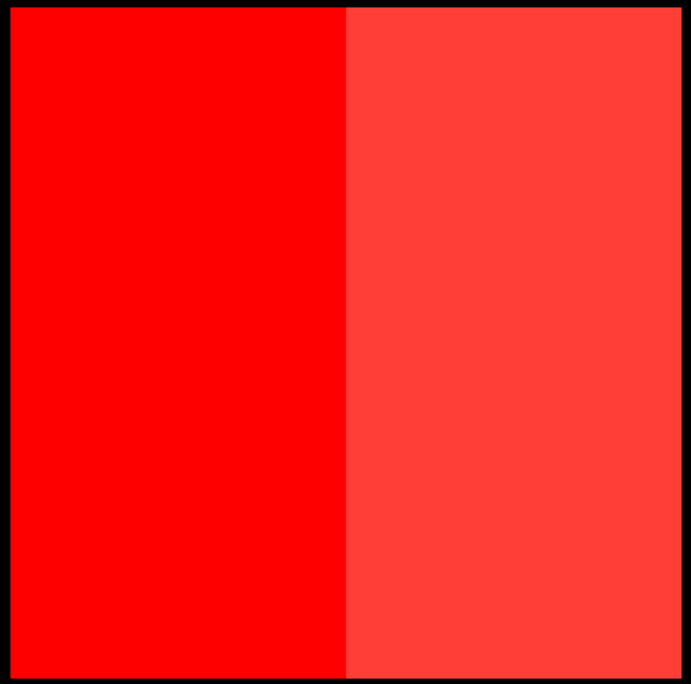
```
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

```swift
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

```swift
// Drawing with Wide Color
// CocoaTouch

import UIKit

let renderer = UIGraphicsImageRenderer(size: CGSize(width: 250, height: 250))

let image = renderer.image { rendererContext in
    let bounds = rendererContext.format.bounds
    let rects = bounds.divide(bounds.size.width/2, fromEdge: .minXEdge)

    UIColor(displayP3Red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.slice)

    UIColor(red: 1.0, green: 0.0, blue: 0.0, alpha: 1.0).set()
    rendererContext.fill(rects.remainder)
}
```

Drawing with Wide Color

Cocoa



(Output simulated)

Drawing with Wide Color

CocoaTouch

Drawing with Wide Color

CocoaTouch

```
UIGraphicsImageRenderer(size: CGSize)
```

Drawing with Wide Color

CocoaTouch

```
UIGraphicsImageRenderer(size: CGSize)
```

- Fully color managed by default

Drawing with Wide Color

CocoaTouch

```
UIGraphicsImageRenderer(size: CGSize)
```

- Fully color managed by default
- Supports extended range sRGB color space

Drawing with Wide Color

CocoaTouch

```
UIGraphicsImageRenderer(size: CGSize)
```

- Fully color managed by default
- Supports extended range sRGB color space
- Manages CGContext lifetime

Drawing with Wide Color

CocoaTouch

```
UIGraphicsImageRenderer(size: CGSize)
```

- Fully color managed by default
- Supports extended range sRGB color space
- Manages CGContext lifetime
- Works with your existing code using UIGraphicsGetCurrentContext()

Wide Color on the Screen

Wide Color on the Screen

CocoaTouch

NEW

Wide Color on the Screen

NEW

CocoaTouch

UIView

```
draw(_ rect: CGRect) // called in the extended sRGB color space
```

Wide Color on the Screen

NEW

CocoaTouch

UIView

```
draw(_ rect: CGRect) // called in the extended sRGB color space
```

UIImageView

Color managed since iOS 9.3

Wide Color on the Screen

CocoaTouch

NEW

Wide Color on the Screen

CocoaTouch

NEW

```
UITraitCollection.displayGamut
```

Wide Color on the Screen

NEW

CocoaTouch

```
UITraitCollection.displayGamut
```

UIDisplayGamut

Wide Color on the Screen

NEW

CocoaTouch

```
UITraitCollection.displayGamut
```

UIDisplayGamut

```
• .P3
```

Wide Color on the Screen

NEW

CocoaTouch

```
UITraitCollection.displayGamut
```

UIDisplayGamut

- .P3
- .SRGB

Wide Color on the Screen

Controlling the depth—CocoaTouch

NEW

For UIView, use the view's layer's contentsFormat property

Wide Color on the Screen

NEW

Controlling the depth—CocoaTouch

For UIView, use the view's layer's contentsFormat property

Valid CALayer contents formats:

- kCAContentsFormatRGBA8Uint
- kCAContentsFormatRGBA16Float
- kCAContentsFormatGray8Uint

Wide Color on the Screen

Cocoa

NSView

Wide Color on the Screen

Cocoa

NSView

- `draw(_ dirtyRect: NSRect)`

Wide Color on the Screen

Cocoa

NSView

- `draw(_ dirtyRect: NSRect)`
- `viewDidChangeBackingProperties()`

Wide Color on the Screen

Cocoa

NSView

- `draw(_ dirtyRect: NSRect)`
- `viewDidChangeBackingProperties()`
- `NSWindowDidChangeBackingPropertiesNotification //from your view’s window`

Wide Color on the Screen

Cocoa

NSView

```
• draw(_ dirtyRect: NSRect)
• viewDidChangeBackingProperties()
• NSWindowDidChangeBackingPropertiesNotification //from your view’s window
```

- Backing properties include scale, color space and output display gamut

Wide Color on the Screen

Controlling the Depth—Cocoa

Use NSWindow's depthLimit property to control how large the NSWindow's backing store will be

Wide Color on the Screen

Controlling the Depth—Cocoa

Use NSWindow's depthLimit property to control how large the NSWindow's backing store will be

NSWindowDepth

- NSWindowDepthTwentyfourBitRGB
- NSWindowDepthSixtyfourBitRGB
- NSWindowDepthOnehundredtwentyeightBitRGB

Summary

Summary

Wide Color is technology for the next generation of displays

Summary

Wide Color is technology for the next generation of displays

Color Gamuts and Color Management

Summary

Wide Color is technology for the next generation of displays

Color Gamuts and Color Management

Working with wide gamut content

Summary

Wide Color is technology for the next generation of displays

Color Gamuts and Color Management

Working with wide gamut content

Using colors in your application

Summary

Wide Color is technology for the next generation of displays

Color Gamuts and Color Management

Working with wide gamut content

Using colors in your application

Adapting your drawing code to wide color gamut

More Information

https://developer.apple.com/wwdc16/712

Related Sessions

Advances in iOS Photography	Pacific Heights	Tuesday 11:00AM
What's New in Cocoa	Nob Hill	Tuesday 11:00AM
What's New in Cocoa Touch	Presidio	Tuesday 1:40PM
What's New in Metal, Part 2	Pacific Heights	Wednesday 1:40PM
Live Photo Editing and RAW Processing with Core Image	Nob Hill	Thursday 11:00AM
Advances in SceneKit Rendering	Mission	Thursday 11:00AM

Related Sessions

What's New in SpriteKit	Presidio	Thursday 5:00PM
Crafting Modern Cocoa Apps	Pacific Heights	Friday 5:00PM

Labs

Live Photo and Core Image Lab	Graphics, Games, and Media Lab C	Thursday 1:30PM
Cocoa Lab	Frameworks Lab D	Thursday 2:00PM
SceneKit Lab	Graphics, Games, and Media Lab A	Thursday 3:00PM
Live Photo and Core Image Lab	Graphics, Games, and Media Lab D	Friday 9:00AM
Cocoa Touch and 3D Touch Lab	Frameworks Lab C	Friday 10:30AM

Labs

SpriteKit Lab	Graphics, Games, and Media Lab B	Friday 12:00PM
Color Lab	Graphics, Games, and Media Lab C	Friday 4:00PM
Safari and WebKit Lab	Frameworks Lab C	Friday 4:30PM

WWDC16