#WWDC18

# Optimizing App Assets

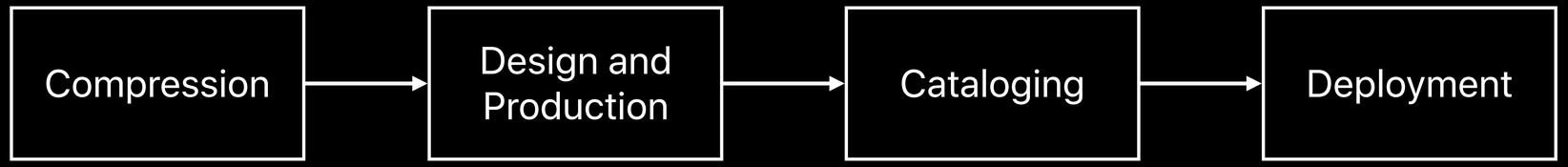
Session 227

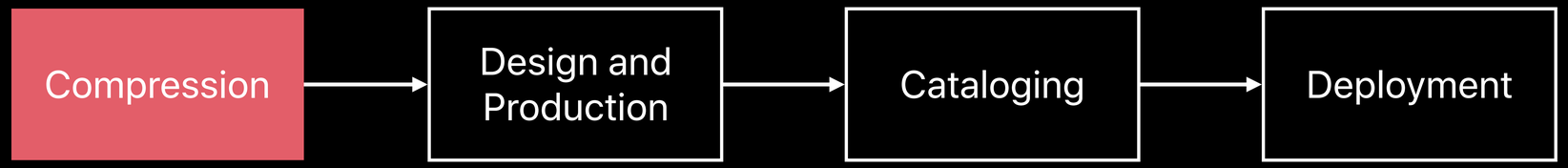
Will Li, Cocoa Frameworks
Patrick Heynen, Cocoa Frameworks


9:41
FaceTime
Monday
4
Calendar
Photos
Camera
Mail
Clock
Maps
280
Weather
Notes
Reminders
News
Stocks
TV
iTunes Store
App Store
Books
Health
Home
Wallet
Settings

# Optimizing App Assets

Use new features of Asset Catalogs to optimize the deployment of your application assets

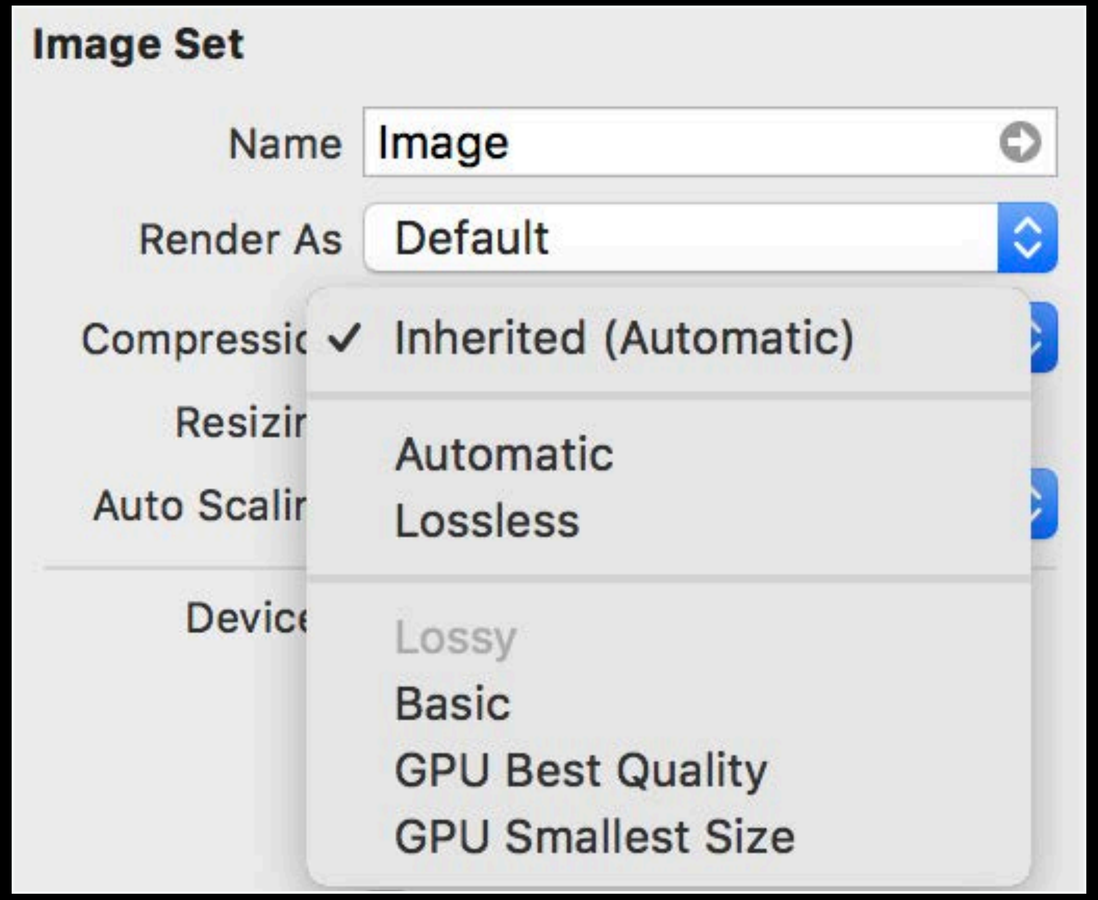

Compression
Design and Production
Cataloging
Deployment


Compression
Design and Production
Cataloging
Deployment

# Compression

# Image Compression

Choose the appropriate compression type



Automatic image packing

Lossy compression

Lossless compression

Deployment target and app thinning

App variant export

# Automatic Image Packing

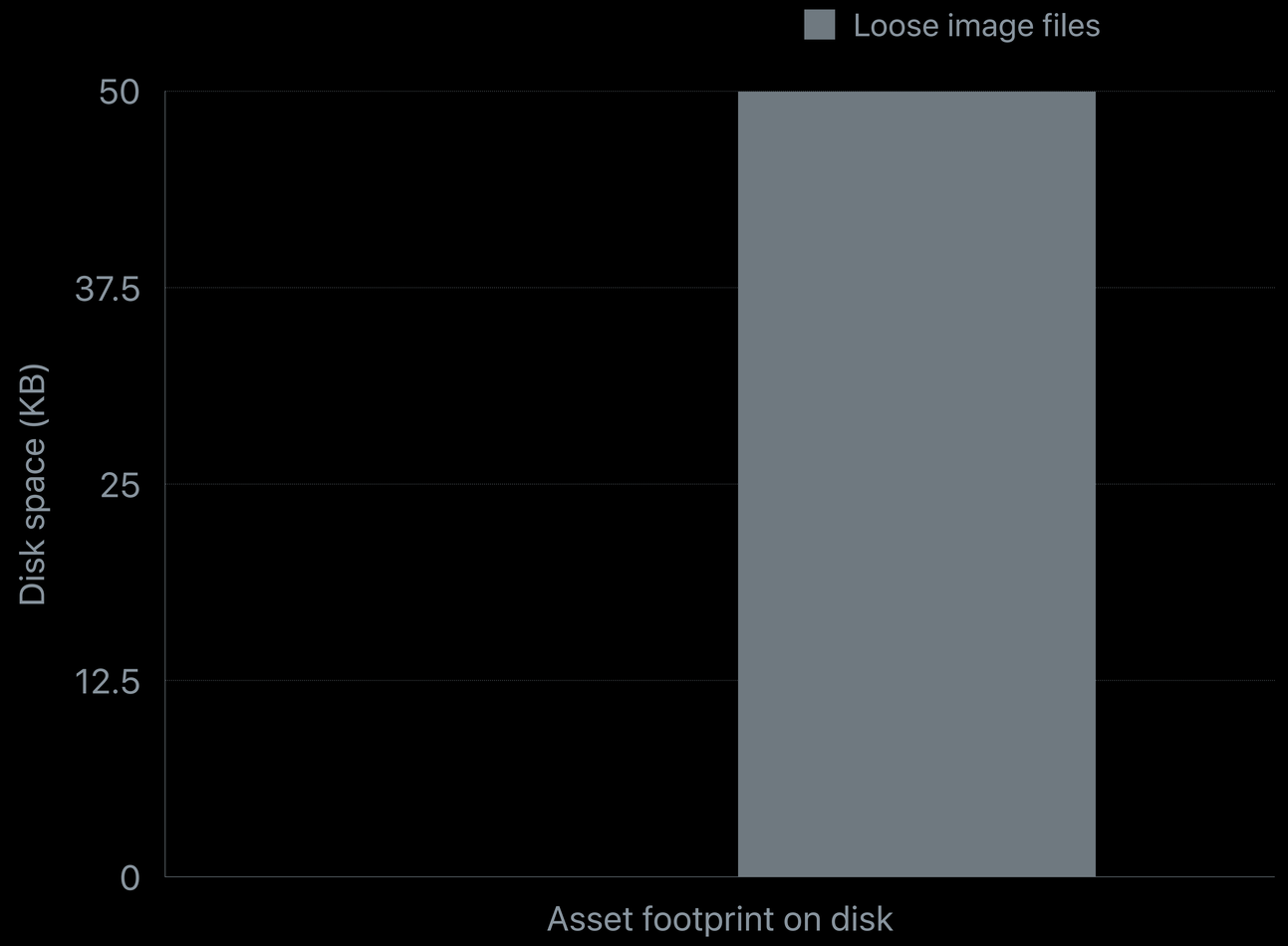
Loose images have hidden costs

Difficult to manage large number of files

Inconsistent and uncertain image formats

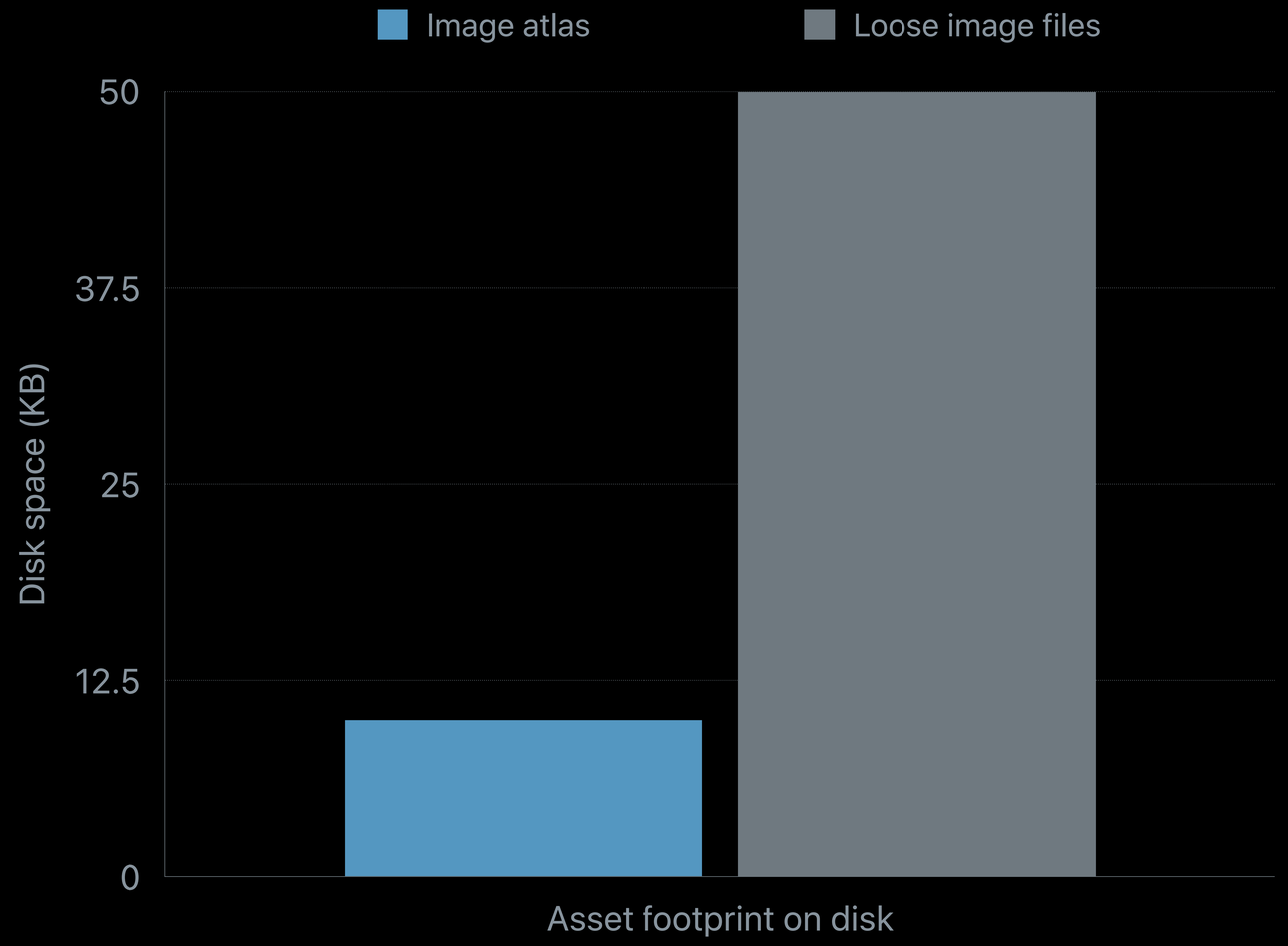
# Automatic Image Packing

Loose image files



# Automatic Image Packing

Image atlas



# Automatic Image Packing

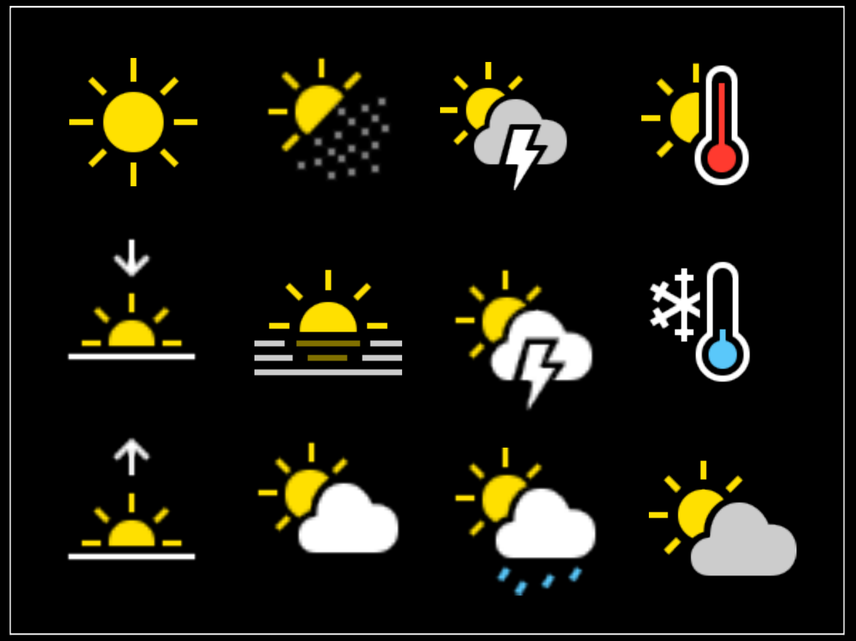
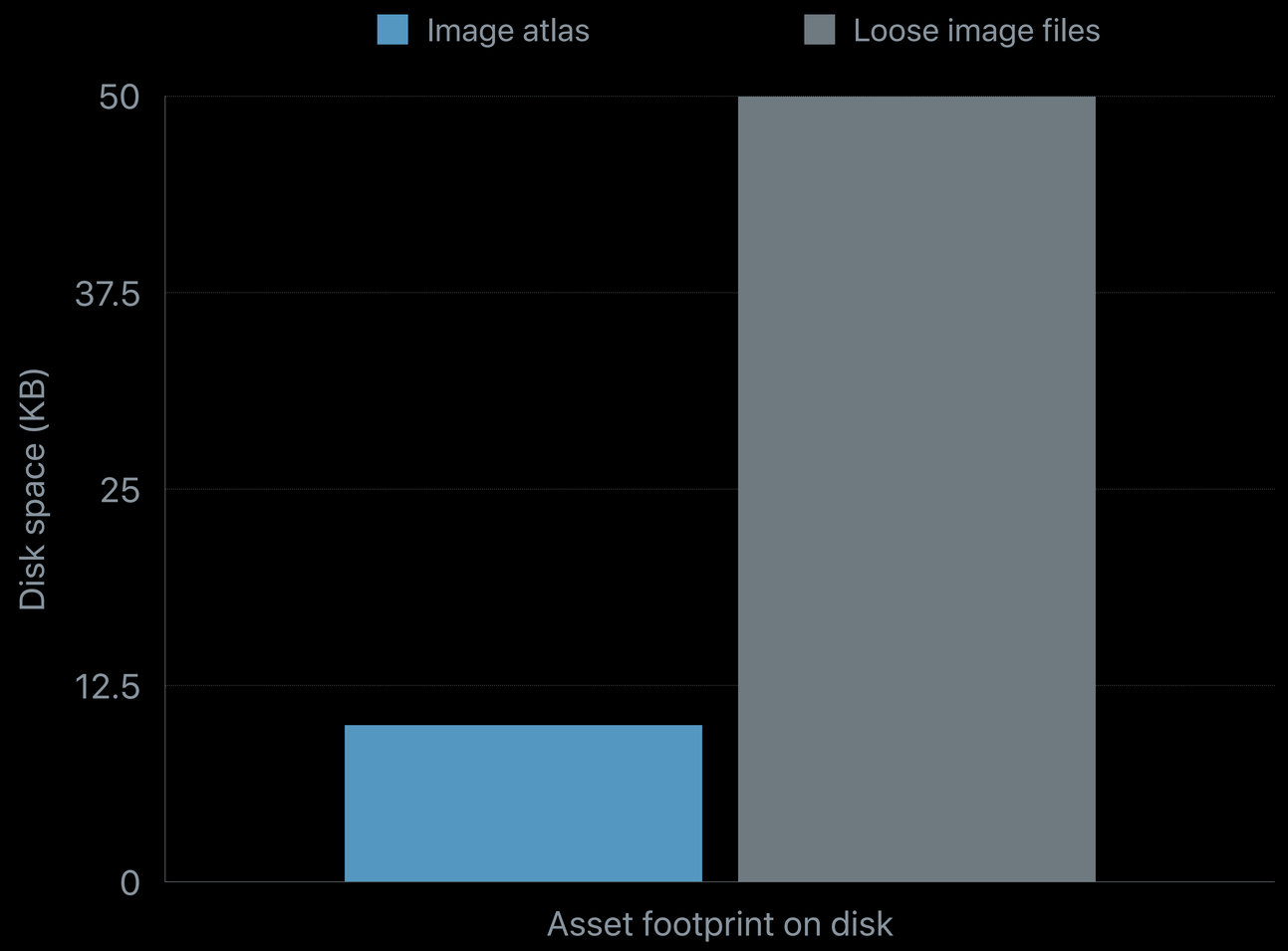


Image atlas



Automatic image packing

Lossy compression

Lossless compression

Deployment target and app thinning

App variant export

# Lossy Compression

Quality versus size tradeoff

Best suited for certain scenarios

# High Efficiency Image File Format

# High Efficiency Image File Format

Better compression ratio than JPEG

Supports transparency

Automatic conversion from other formats

Automatic image packing

Lossy compression

Lossless compression

Deployment target and app thinning

App variant export

# Lossless Compression

Icons and simple artwork

Complex artwork

# Lossless Compression

Icons and simple artwork

Complex artwork

# Apple Deep Pixel Image Compression

# Apple Deep Pixel Image Compression

NEW

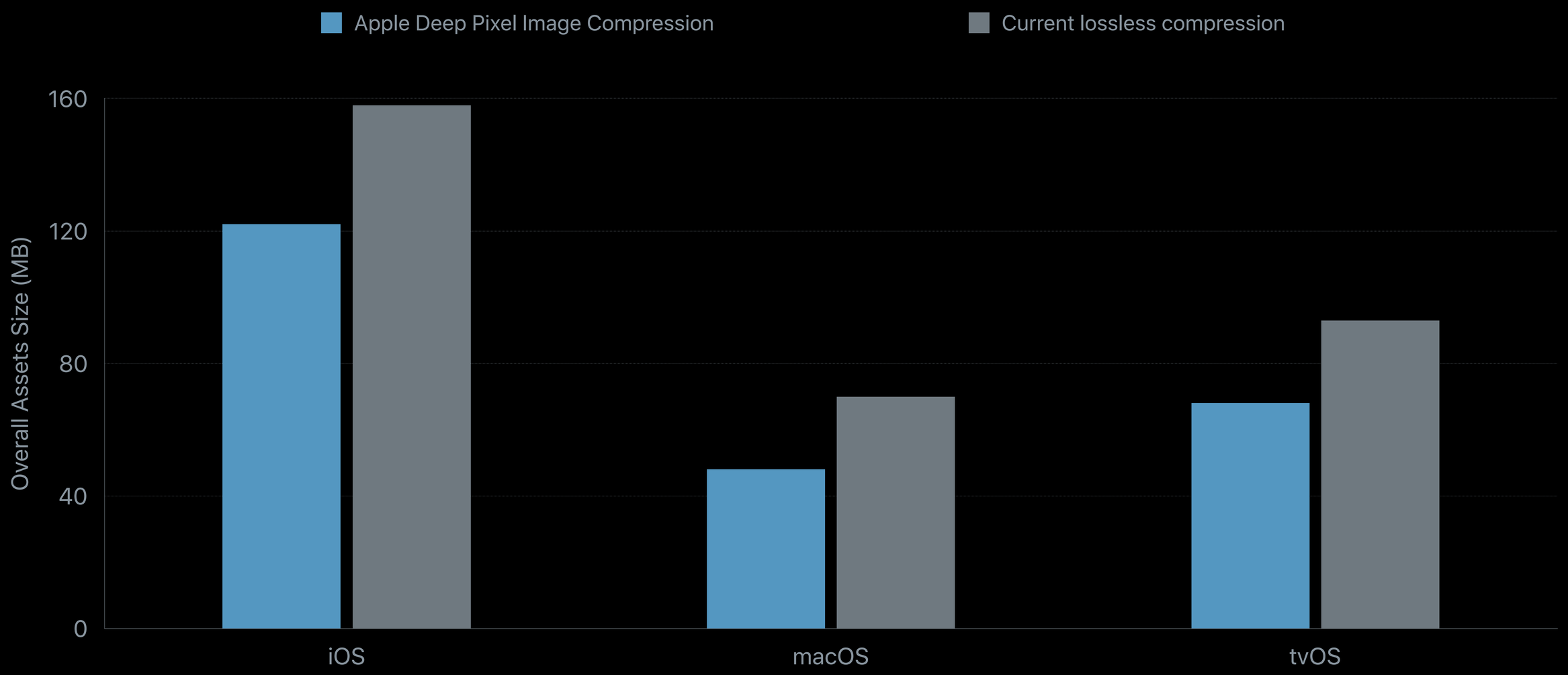
Adaptive to image color spectrum

Selects optimal compression algorithm

15–20% size improvement

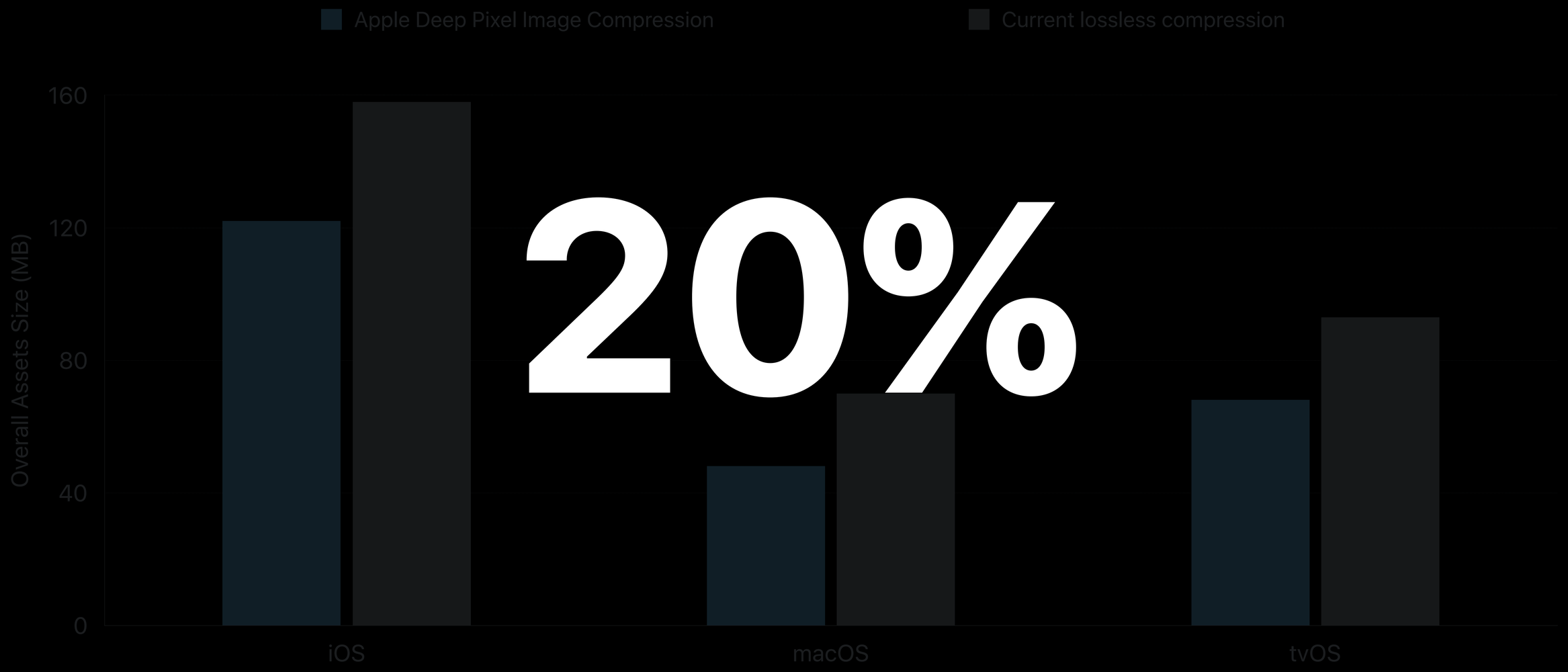
# Apple Deep Pixel Image Compression

## Size improvement



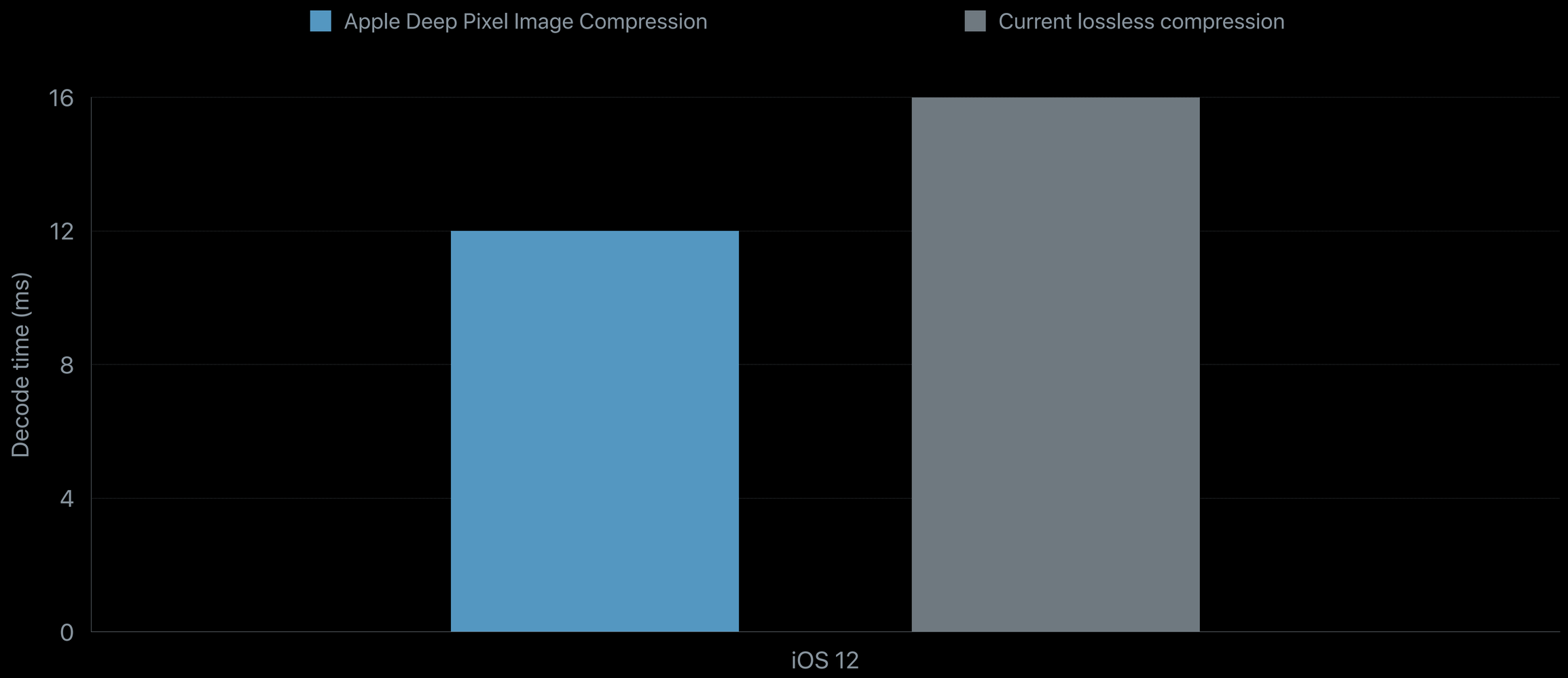
# Apple Deep Pixel Image Compression

## Size improvement



# Apple Deep Pixel Image Compression

## Decode time improvement



Automatic image packing

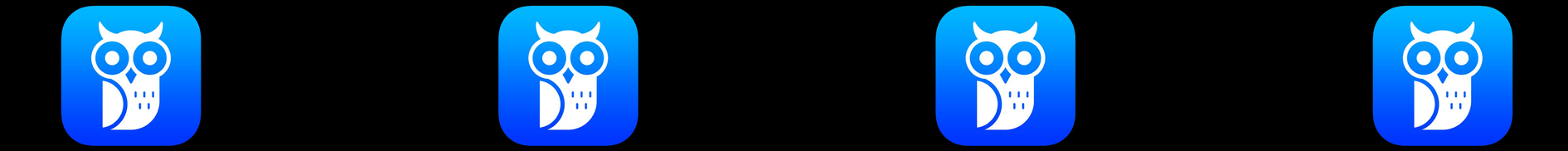
Lossy compression

Lossless compression

Deployment target and app thinning

App variant export

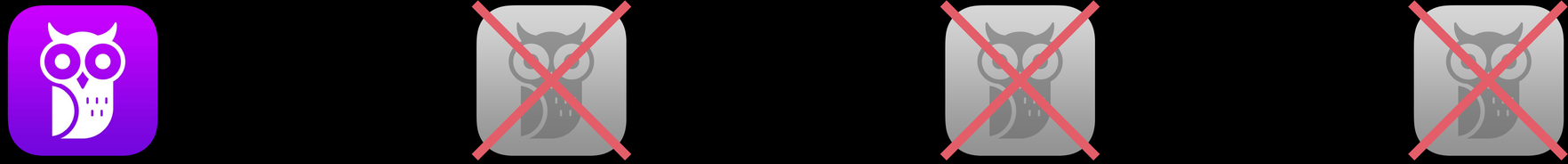
# Deployment Target and App Thinning



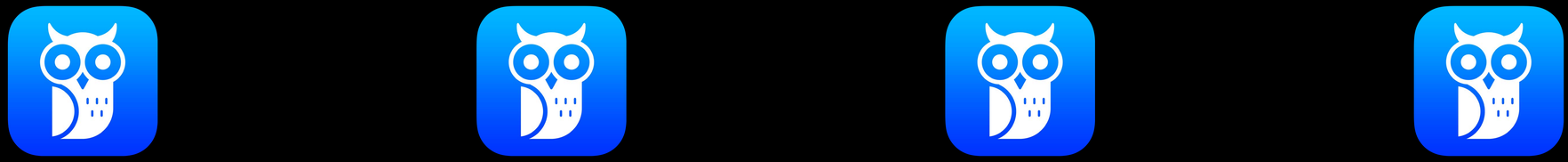

Traditional Backward Deployment

# Deployment Target and App Thinning

# Deployment Target and App Thinning



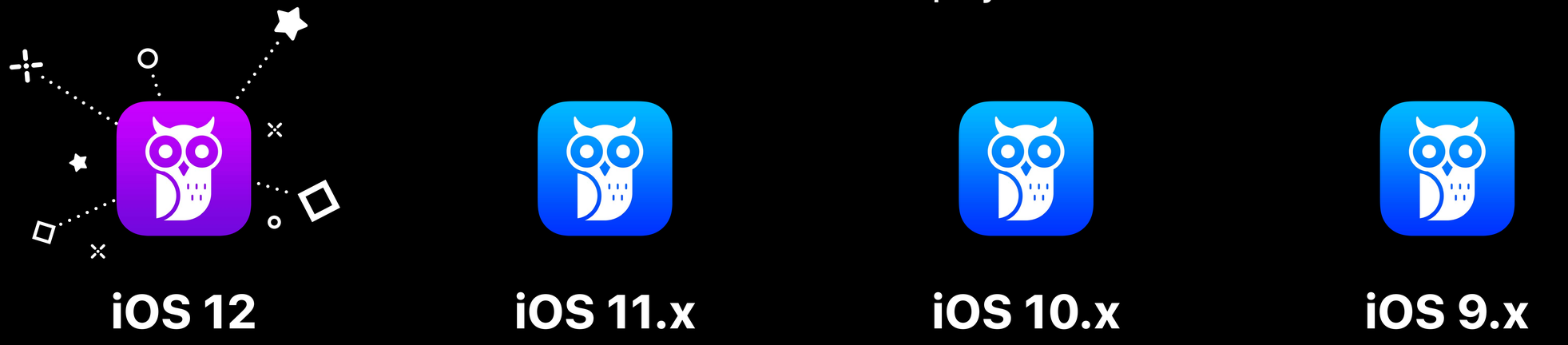

Traditional Backward Deployment




Backward Deployment with OS Variant Thinning

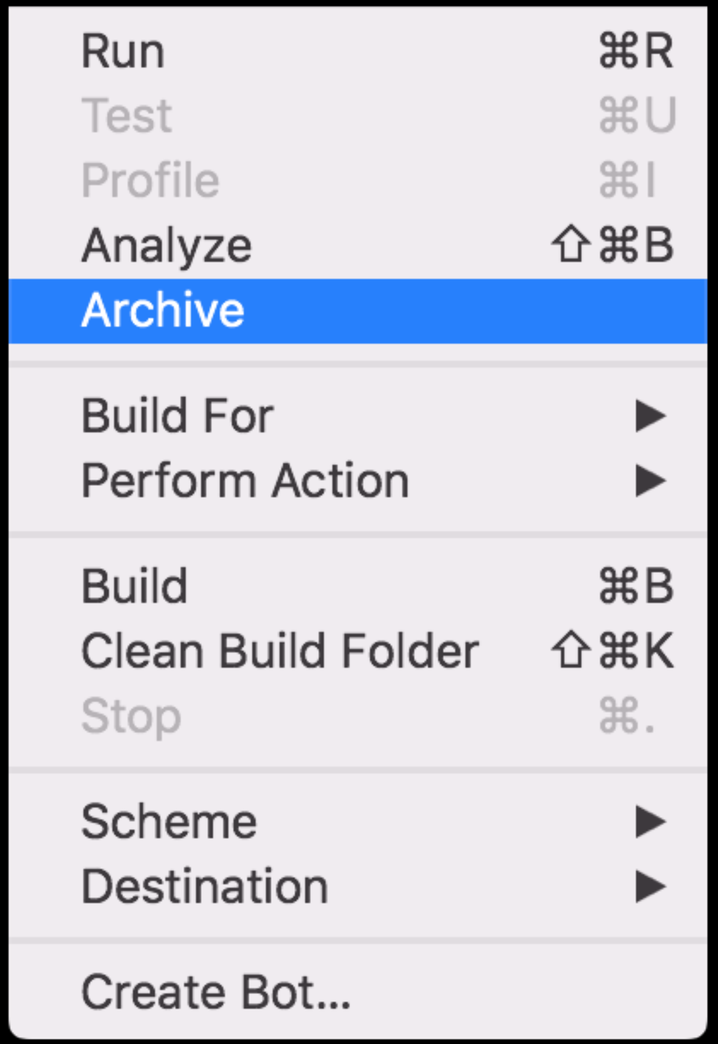
# Deployment Target and App Thinning

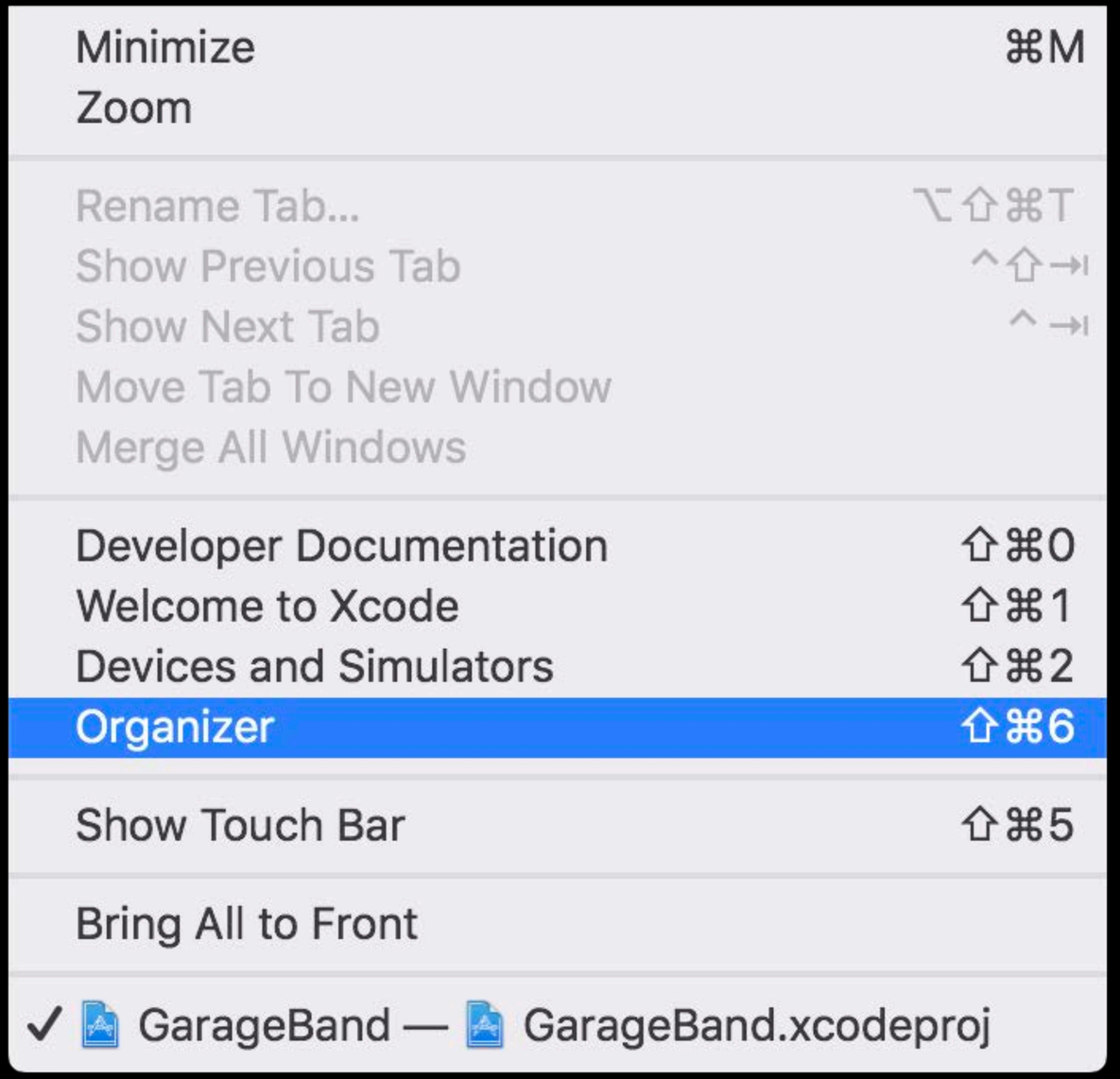

Traditional Backward Deployment

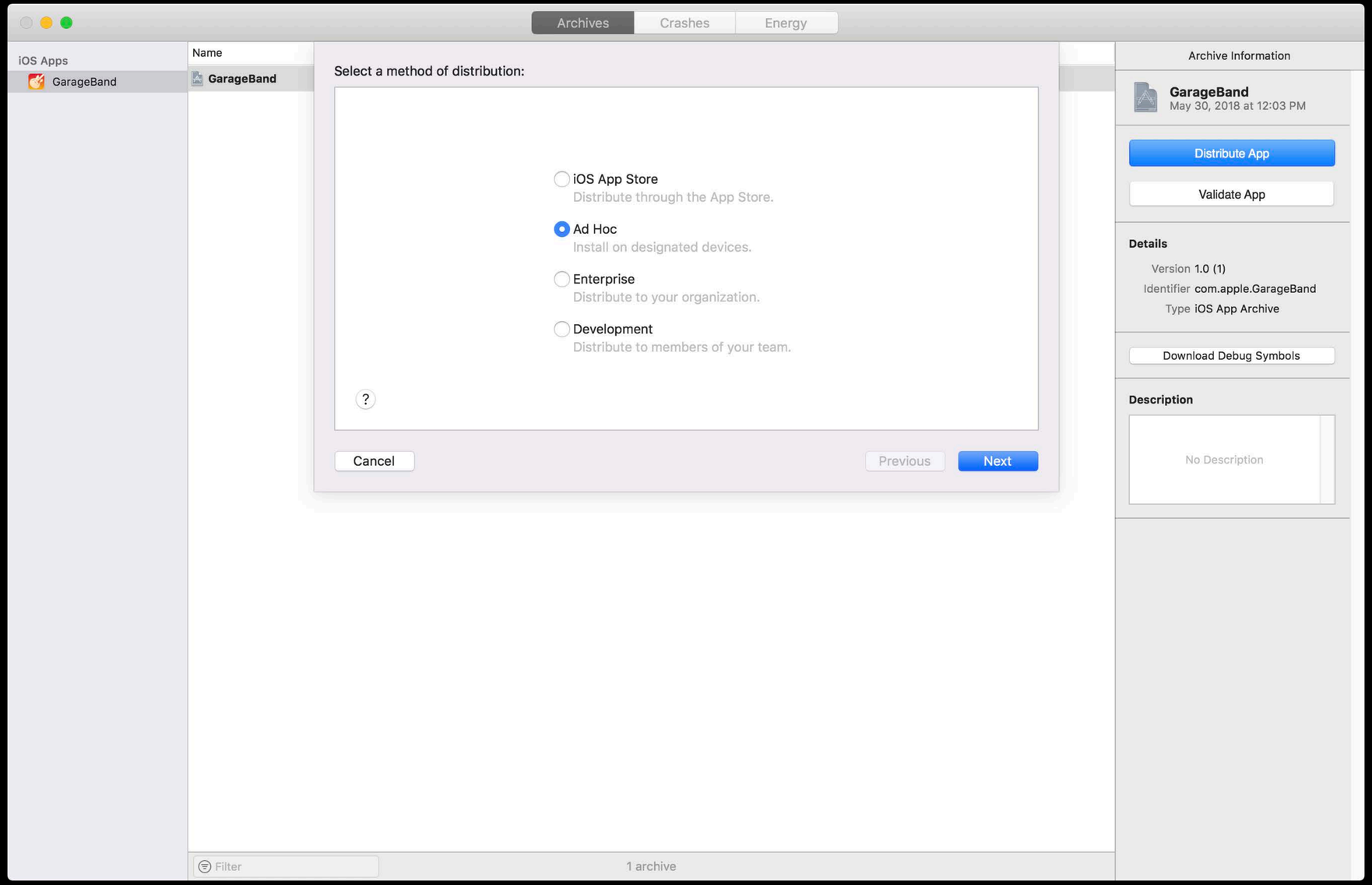


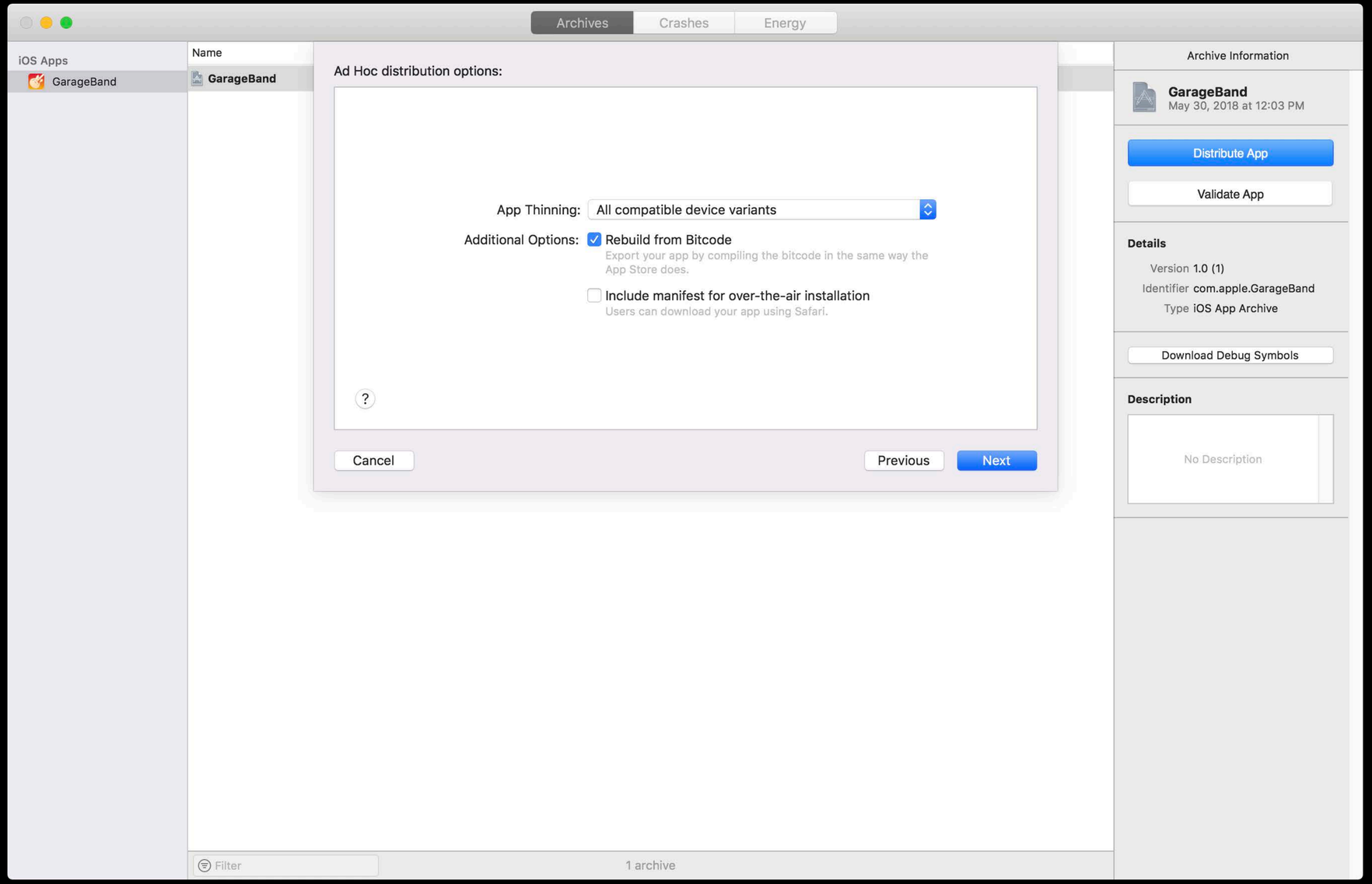

Backward Deployment with OS Variant Thinning

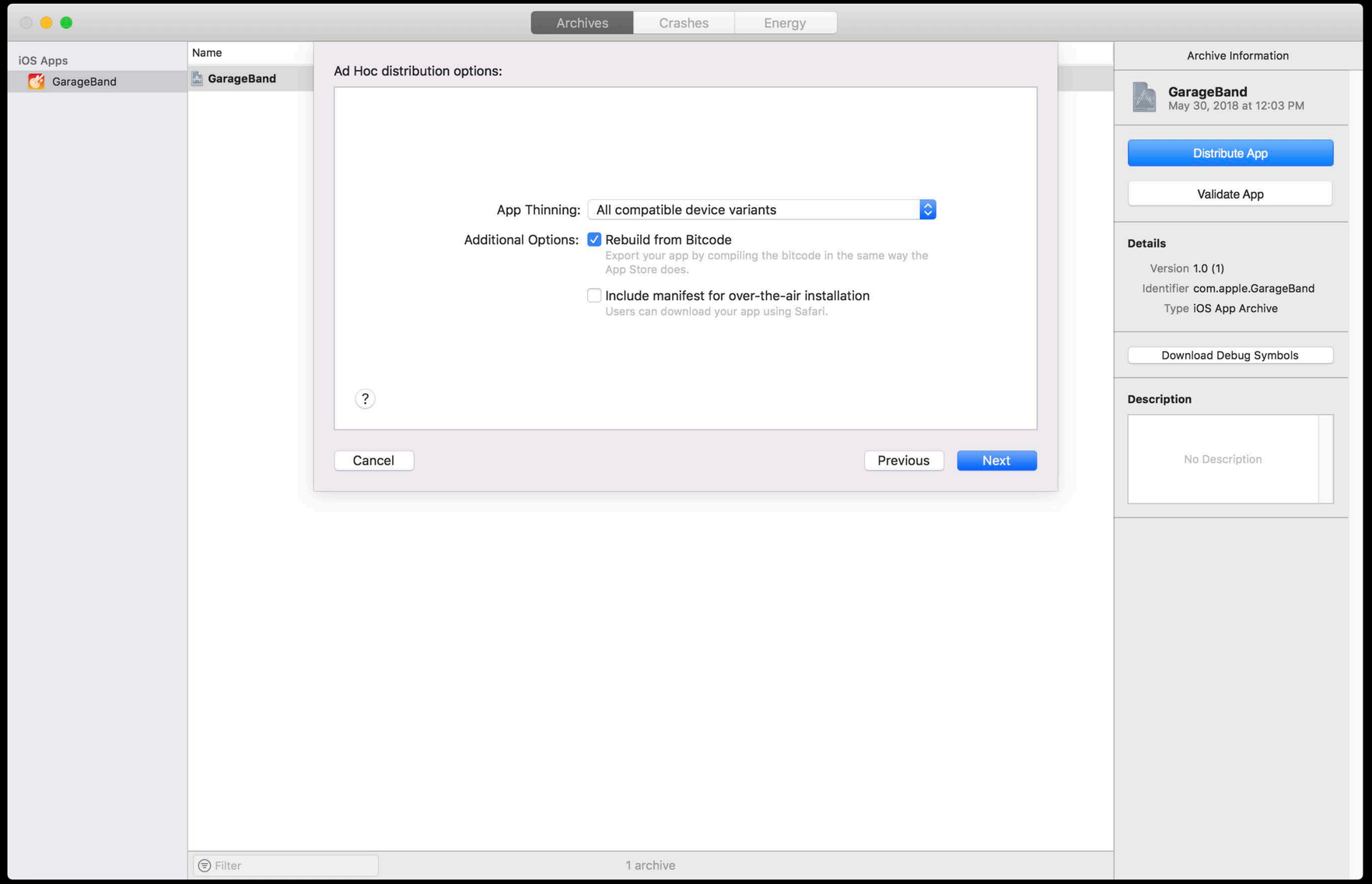
# App Thinning Export

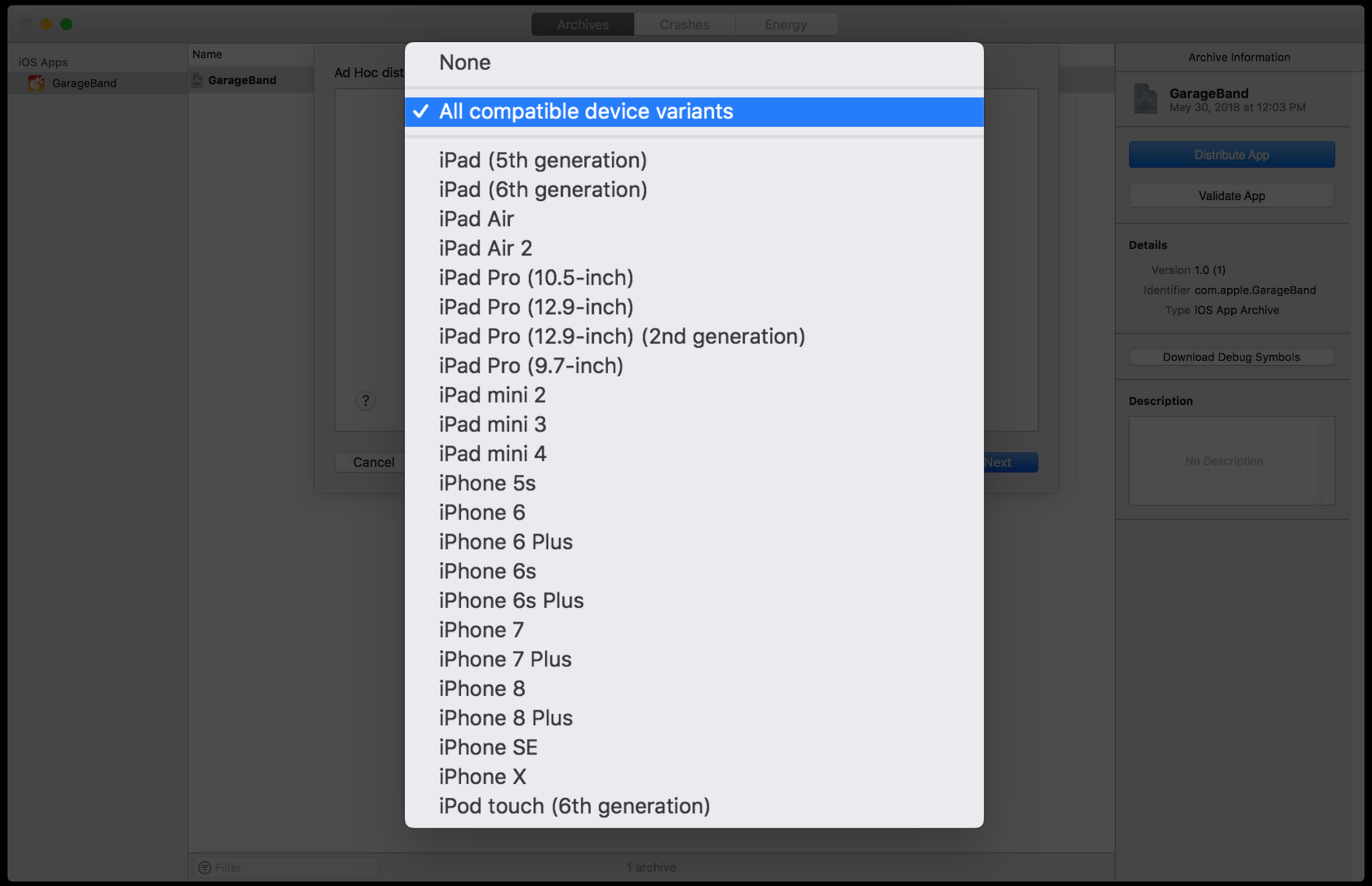

Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For ▶
Perform Action ▶
Build ⌘B
Clean Build Folder ⇧⌘K
Stop ⌘.
Scheme ▶
Destination ▶
Create Bot...


Minimize ⌘M
Zoom
Rename Tab... ⌥⇧⌘T
Show Previous Tab ^⇧⇥
Show Next Tab ^⇥
Move Tab To New Window
Merge All Windows
Developer Documentation ⇧⌘0
Welcome to Xcode ⇧⌘1
Devices and Simulators ⇧⌘2
Organizer ⇧⌘6
Show Touch Bar ⇧⌘5
Bring All to Front
GarageBand — GarageBand.xcodeproj


Archives
Crashes
Energy
iOS Apps
GarageBand
Name
GarageBand
Select a method of distribution:
iOS App Store
Distribute through the App Store.
Ad Hoc
Install on designated devices.
Enterprise
Distribute to your organization.
Development
Distribute to members of your team.
?
Cancel
Previous
Next
Filter
1 archive
Archive Information
GarageBand
May 30, 2018 at 12:03 PM
Distribute App
Validate App
Details
Version 1.0 (1)
Identifier com.apple.GarageBand
Type iOS App Archive
Download Debug Symbols
Description
No Description

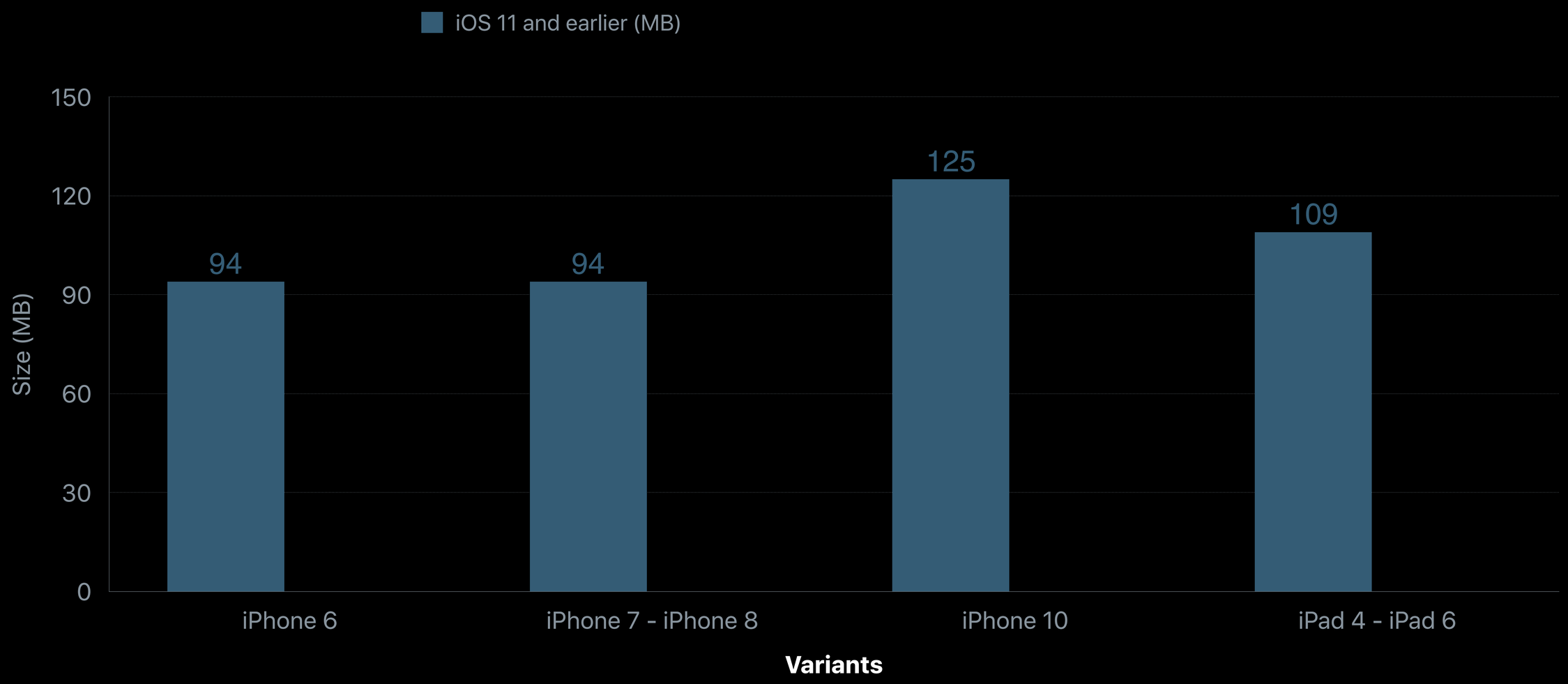

Archives
Crashes
Energy
iOS Apps
GarageBand
Name
GarageBand
Ad Hoc distribution options:
App Thinning:
All compatible device variants
Additional Options:
Rebuild from Bitcode
Export your app by compiling the bitcode in the same way the App Store does.
Include manifest for over-the-air installation
Users can download your app using Safari.
?
Cancel
Previous
Next
Filter
1 archive
Archive Information
GarageBand
May 30, 2018 at 12:03 PM
Distribute App
Validate App
Details
Version 1.0 (1)
Identifier com.apple.GarageBand
Type iOS App Archive
Download Debug Symbols
Description
No Description


Archives
Crashes
Energy
iOS Apps
GarageBand
Name
GarageBand
Ad Hoc distribution options:
App Thinning:
All compatible device variants
Additional Options:
Rebuild from Bitcode
Export your app by compiling the bitcode in the same way the App Store does.
Include manifest for over-the-air installation
Users can download your app using Safari.
?
Cancel
Previous
Next
Filter
1 archive
Archive Information
GarageBand
May 30, 2018 at 12:03 PM
Distribute App
Validate App
Details
Version 1.0 (1)
Identifier com.apple.GarageBand
Type iOS App Archive
Download Debug Symbols
Description
No Description

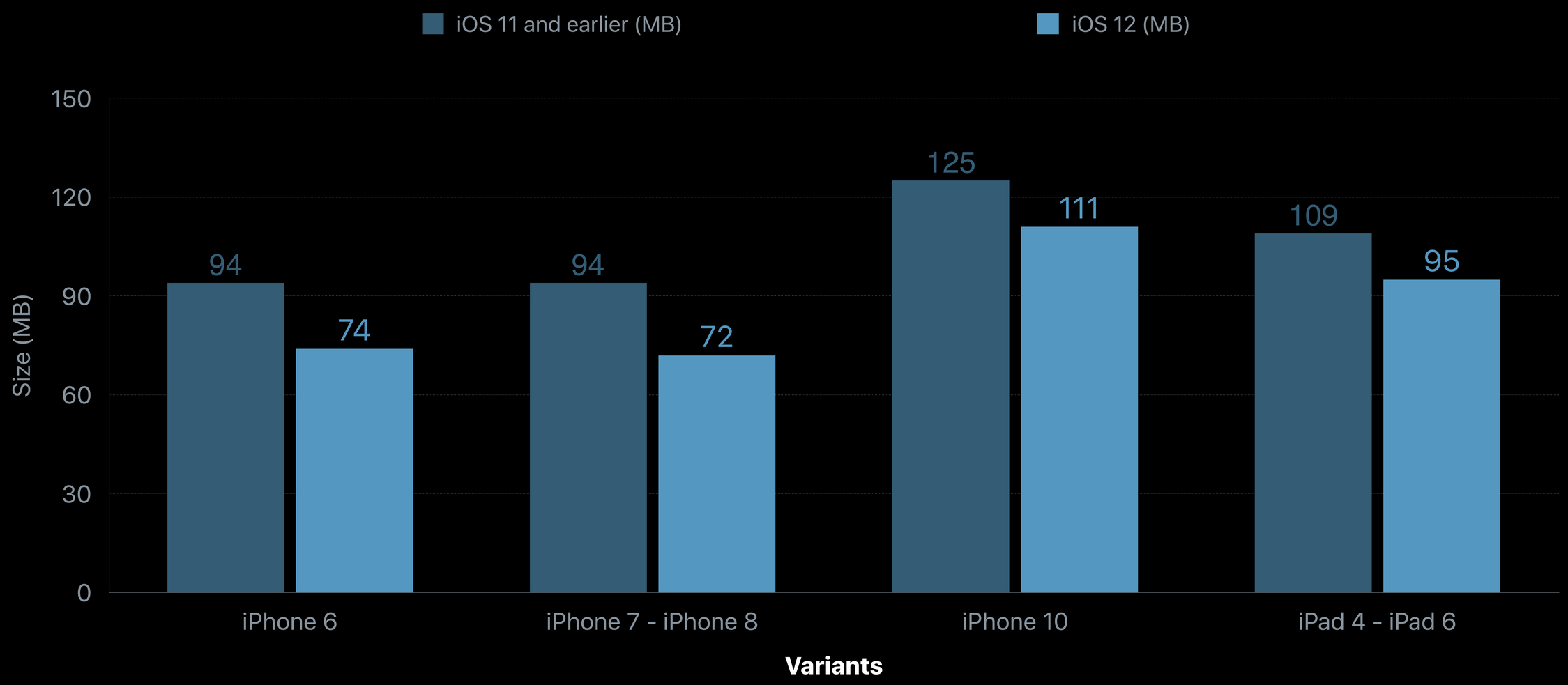

Archives
Crashes
Energy
iOS Apps
GarageBand
Name
GarageBand
Ad Hoc dist
Cancel
Next
Filter
1 archive
None
All compatible device variants
iPad (5th generation)
iPad (6th generation)
iPad Air
iPad Air 2
iPad Pro (10.5-inch)
iPad Pro (12.9-inch)
iPad Pro (12.9-inch) (2nd generation)
iPad Pro (9.7-inch)
iPad mini 2
iPad mini 3
iPad mini 4
iPhone 5s
iPhone 6
iPhone 6 Plus
iPhone 6s
iPhone 6s Plus
iPhone 7
iPhone 7 Plus
iPhone 8
iPhone 8 Plus
iPhone SE
iPhone X
iPod touch (6th generation)
Archive Information
GarageBand
May 30, 2018 at 12:03 PM
Distribute App
Validate App
Details
Version 1.0 (1)
Identifier com.apple.GarageBand
Type iOS App Archive
Download Debug Symbols
Description
No Description

# App Thinning Export

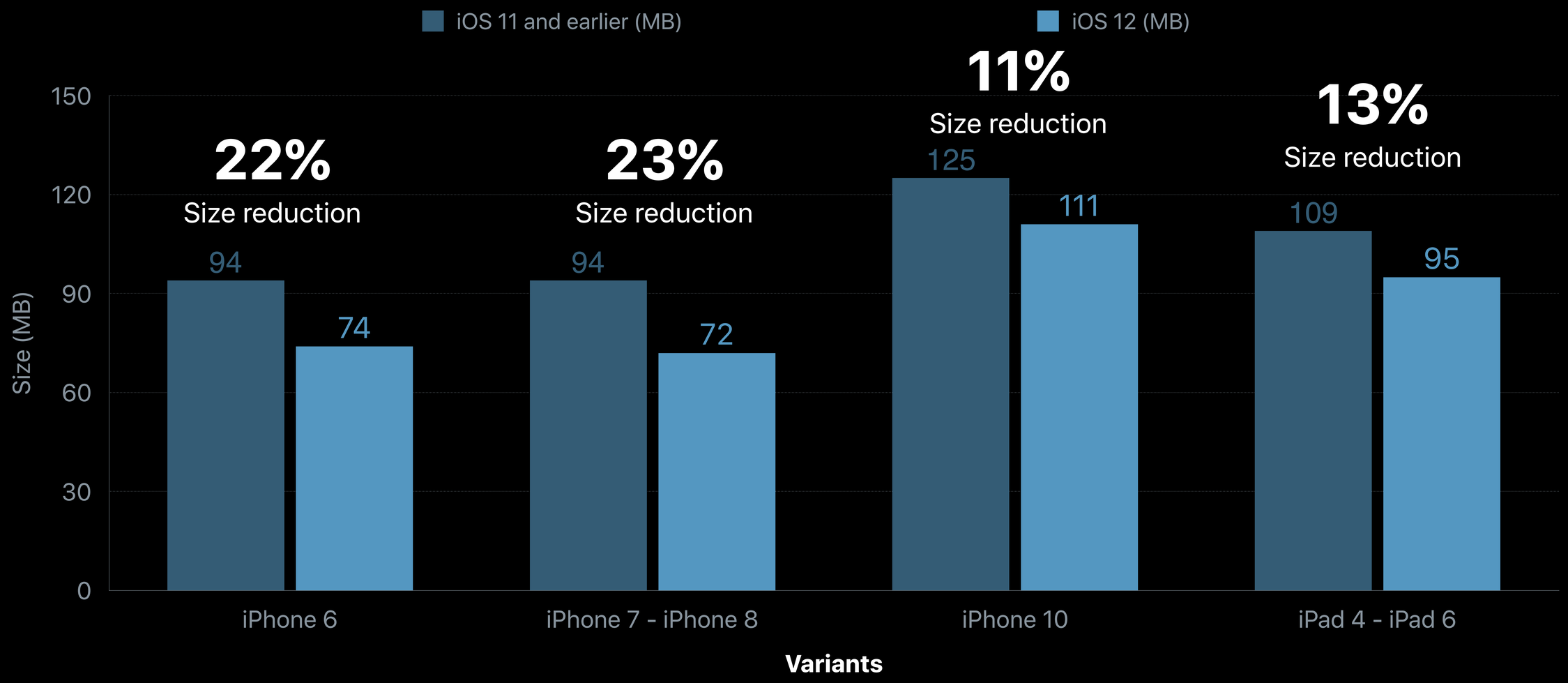
# App Thinning Export

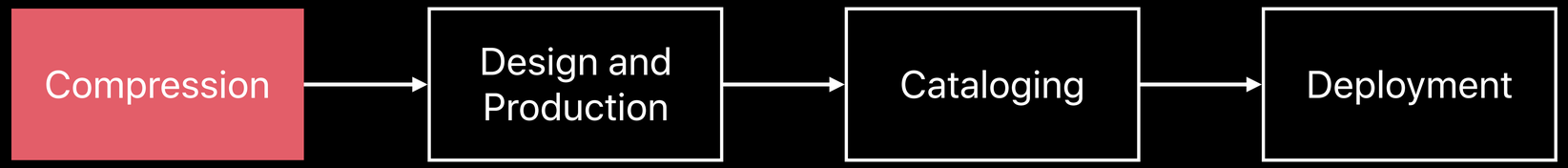


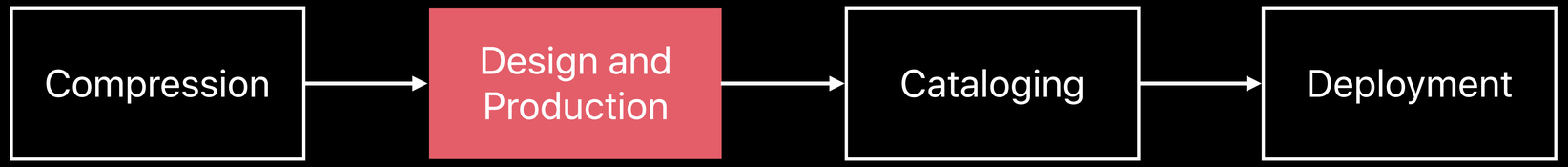
# App Thinning Export



# App Thinning Export




Compression
Design and Production
Cataloging
Deployment


Compression
Design and Production
Cataloging
Deployment

# Design and Production

# Artwork Assets

Assets come from many sources

They all come from humans!

Being organized pays big dividends

Color management

Working space

Stretchable images

Vector assets

Design for 2x

Color management

Working space

Stretchable images

Vector assets

Design for 2x

# Color Management

Pixels without color are just bytes!

Color profiles supply intended appearance

Maintain color profiles

Keep designer intent

# Color Management

Pixels without color are just bytes!

Color profiles supply intended appearance

Maintain color profiles

Keep designer intent

# Color Management

Apps run on broad range of displays

Color matching maps colors to device

# Color Management

Asset Catalogs perform color matching at build time

Artwork ready on device

Bonus: Color Profile eliminated

Color management

Working space

Stretchable images

Vector assets

Design for 2x

# Working Space

Use consistent color settings for all design files

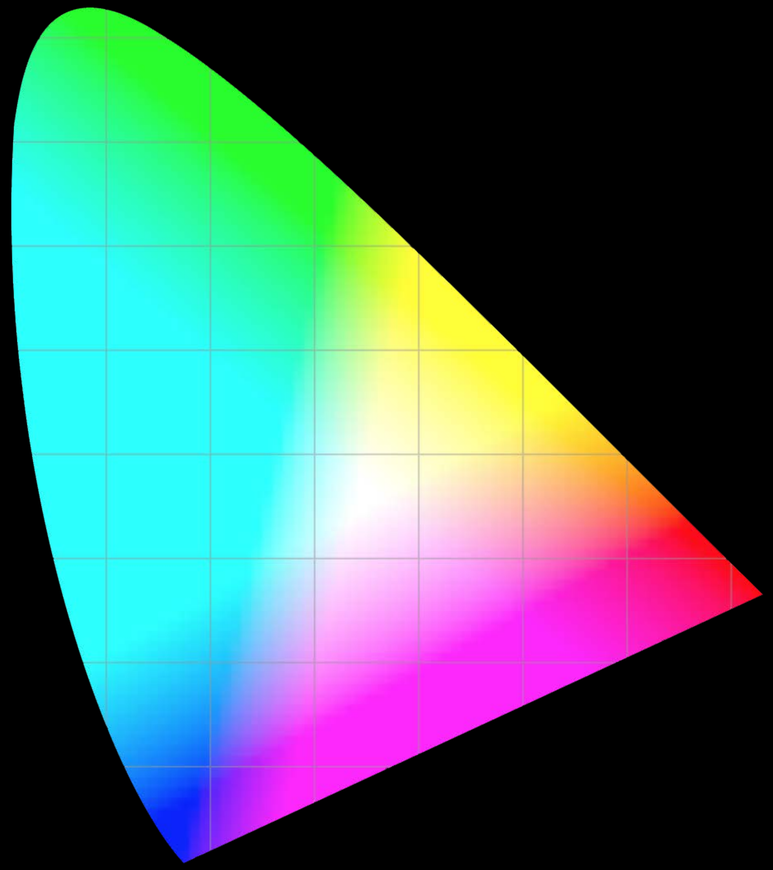
# Working Space

Use consistent color settings for all design files

sRGB / 8 bits for broad applicability

Display P3 / 16 bits for vibrant designs

Flexible Wide Color options



# Working Space

Use consistent color settings for all design files

sRGB / 8 bits for broad applicability

Display P3 / 16 bits for vibrant designs

Flexible Wide Color options

| | |
|---|---|
| Working with Wide Color | WWDC 2016 |
| Working with P3 | iOS Design Resources |

Color management

Working space

Stretchable images

Vector assets

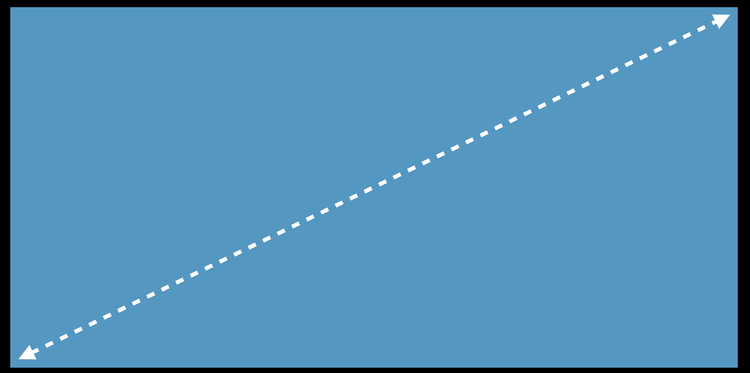
Design for 2x

# Stretchable Images

Adaptive UI uses stretching elements

# Stretchable Images

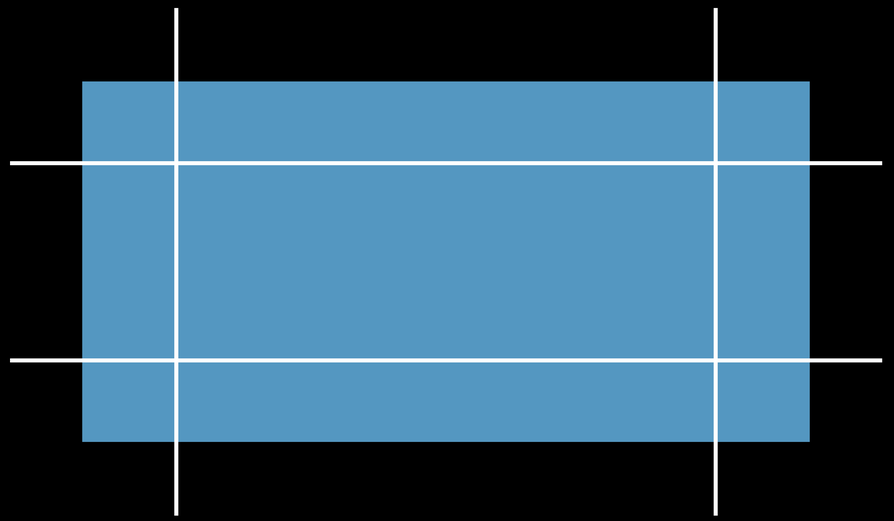
Adaptive UI uses stretching elements



# Stretchable Images

Design tools support slicing

Identify stretchable portion of image



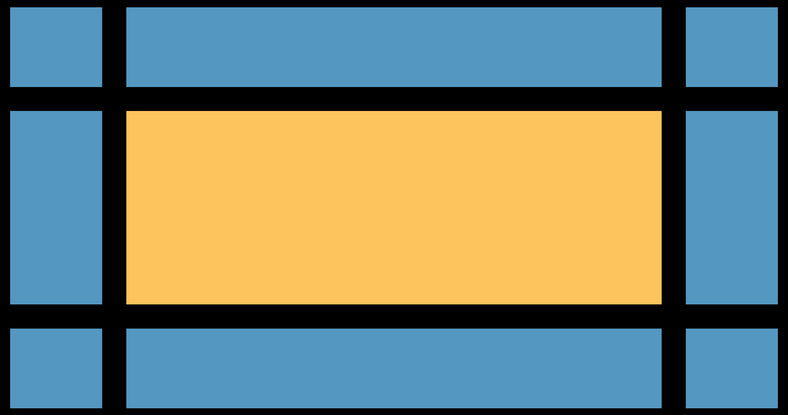
# Stretchable Images

Design tools support slicing

Identify stretchable portion of image

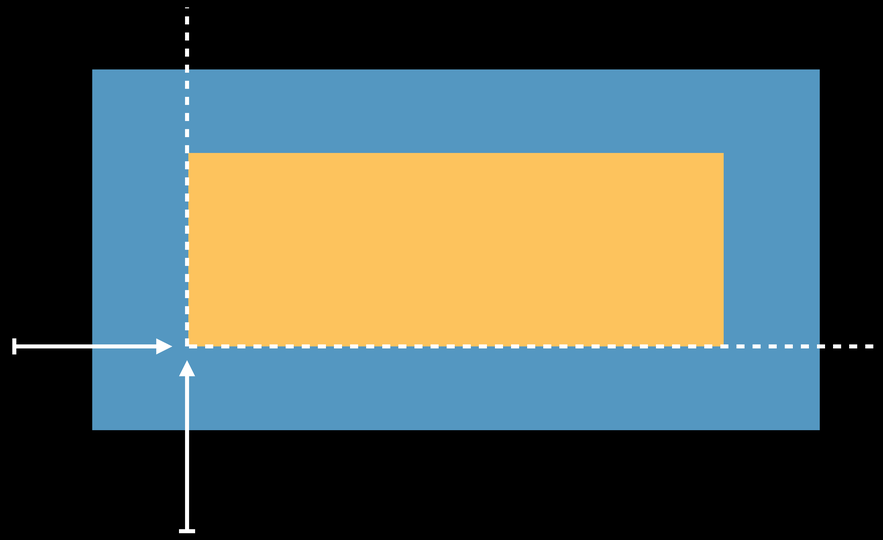
# Stretchable Images

Split image resources require complex drawing to reassemble



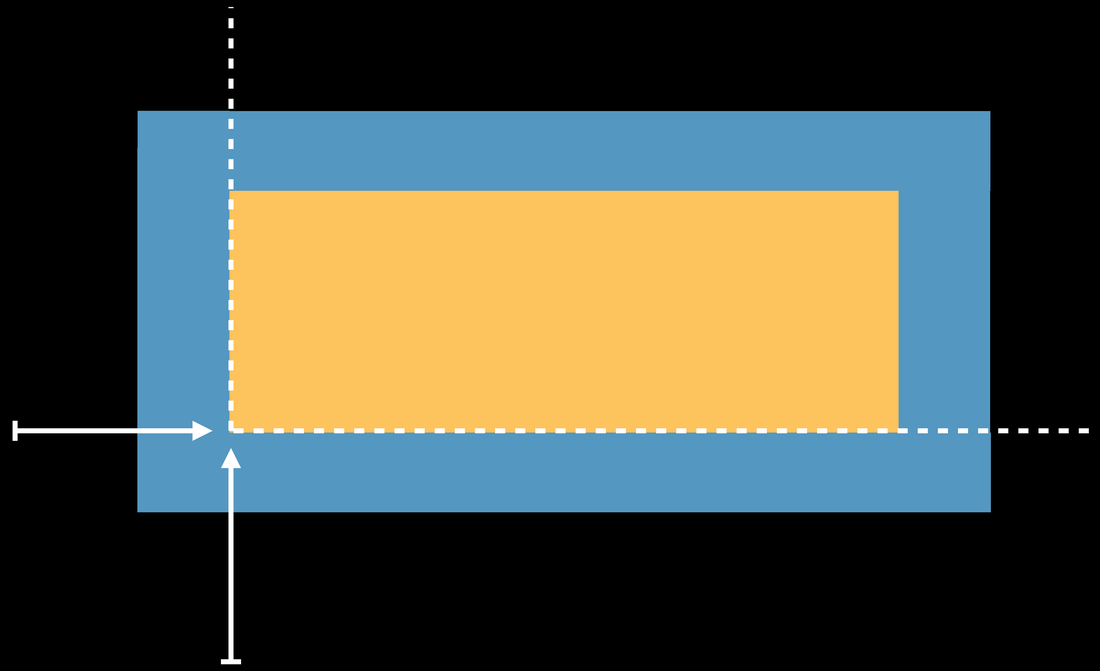
# Stretchable Images

Single image + metadata = smooth GPU animation

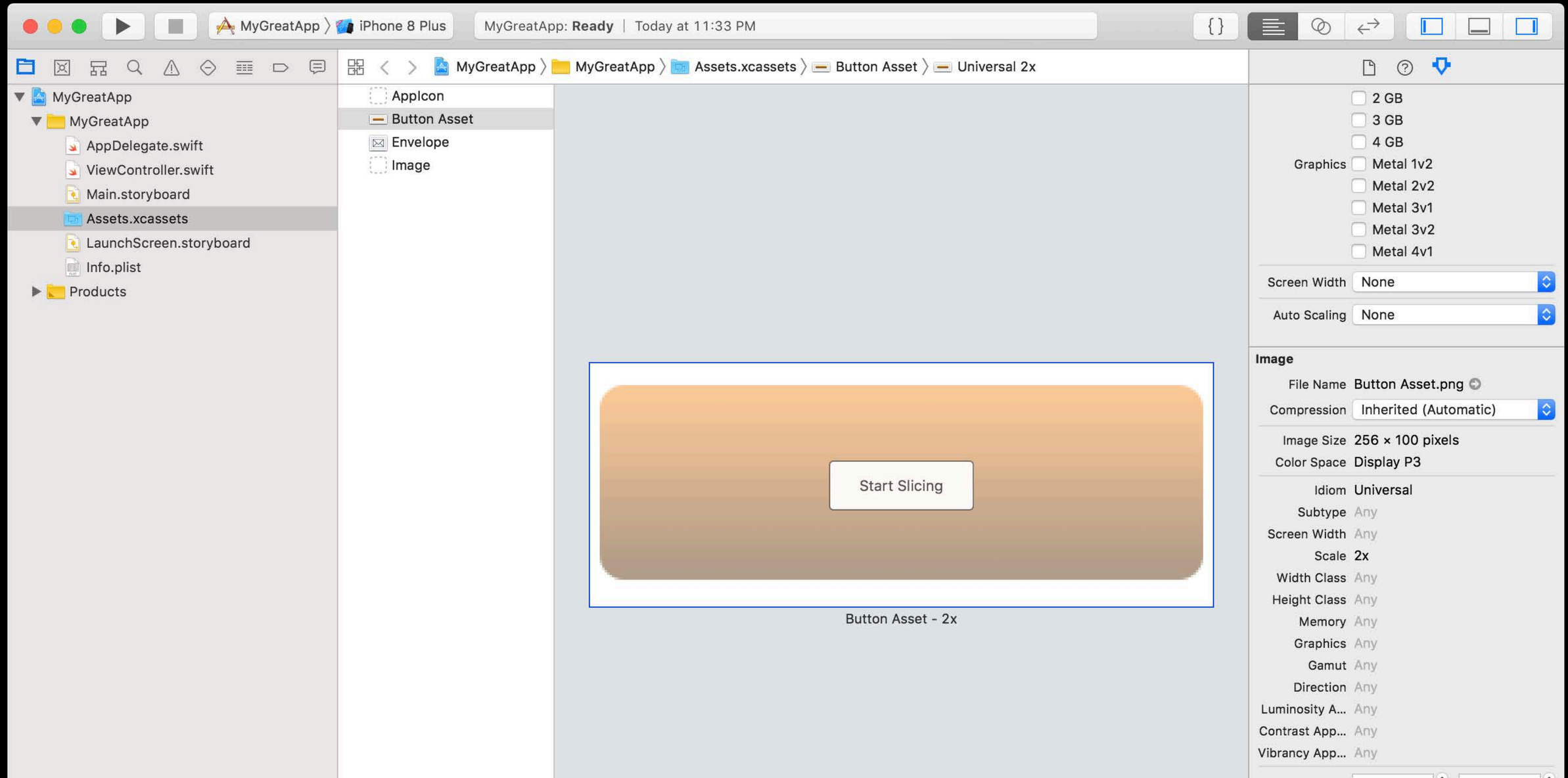


# Stretchable Images

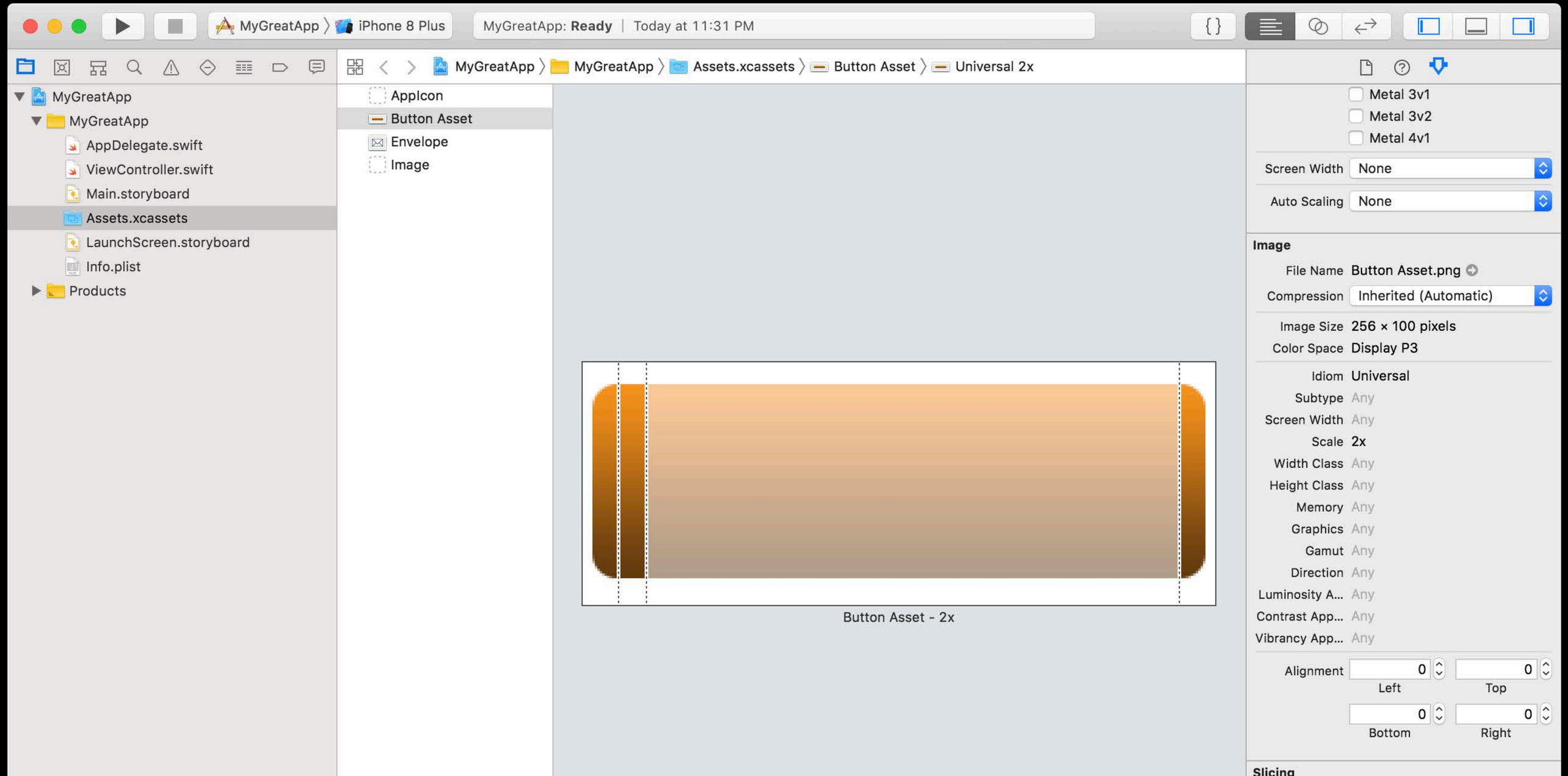
Single image + metadata = smooth GPU animation



# Asset Catalog “Show Slicing”

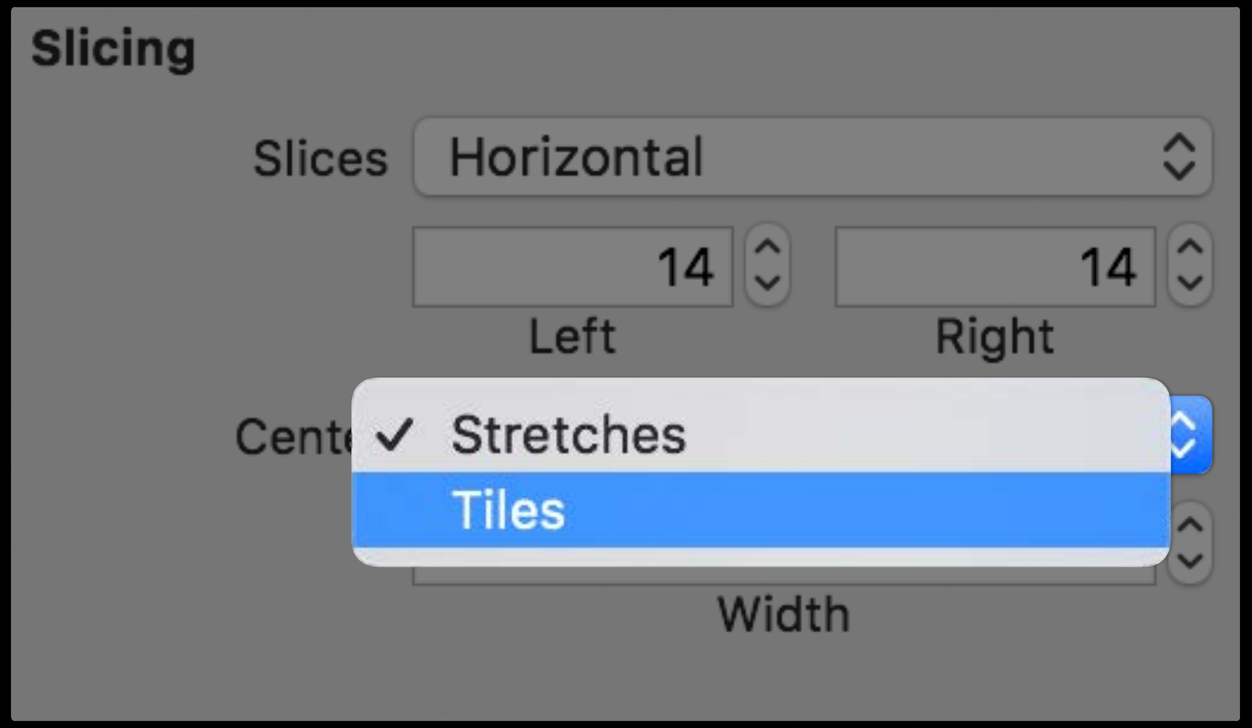


# Asset Catalog “Show Slicing”



# Asset Catalog “Show Slicing”

# Asset Catalog “Show Slicing”



# Asset Catalog Slicing

Keeps stretching metadata close to artwork

Better positioned for design updates

Color management

Working space

Stretchable images

Vector assets

Design for 2x

# Vector Assets

Distinct assets for different displays (1x, 2x, 3x)

# Vector Assets

Distinct assets for different displays (1x, 2x, 3x)

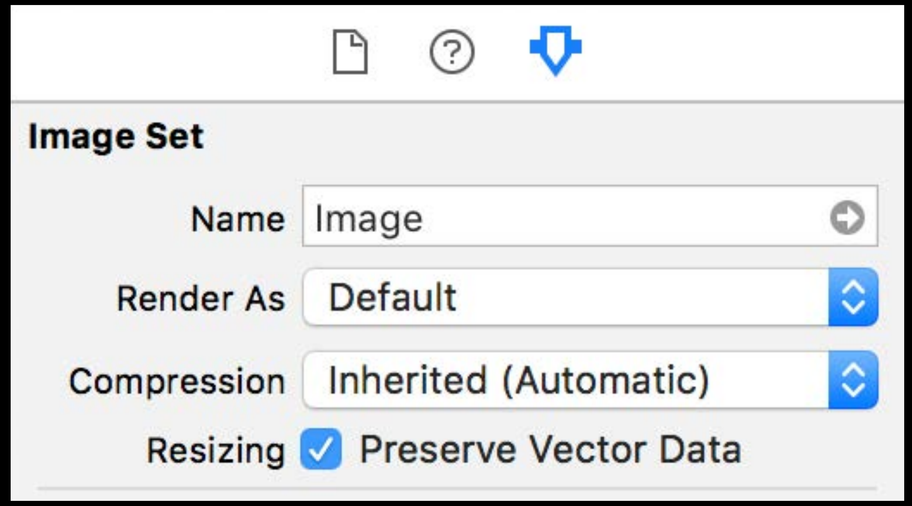
Address all needs with one vector asset in PDF format

# Vector Assets

Xcode generates optimized bitmaps per scale

Preserve Vector Data enables runtime resizing

Works better with Dynamic Type!



Color management

Working space

Stretchable images

Vector assets

Design for 2x

# Design for 2x

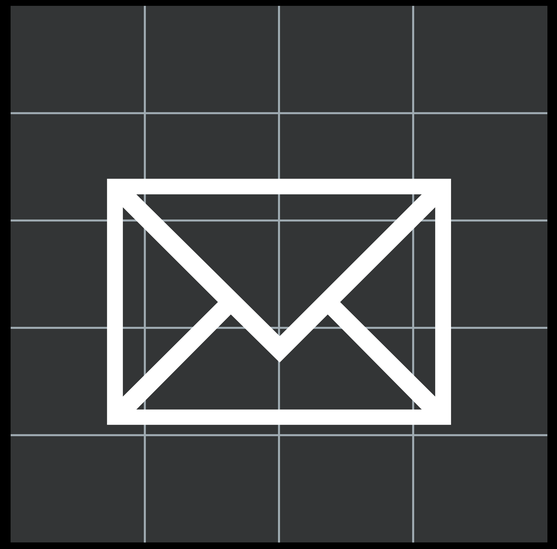
Retina 2x is the most common display density

Strokes landing between pixels still look fuzzy

# Design for 2x

Point boundary snapping ensures device pixel alignment

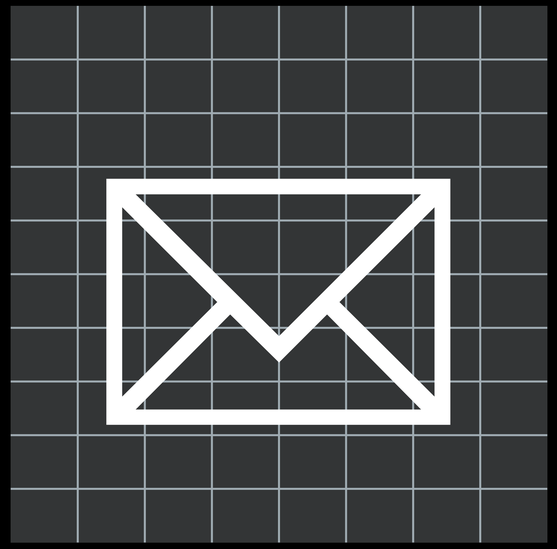
Vector assets can use a 2x grid for optimal stroke placement



# Design for 2x

Point boundary snapping ensures device pixel alignment

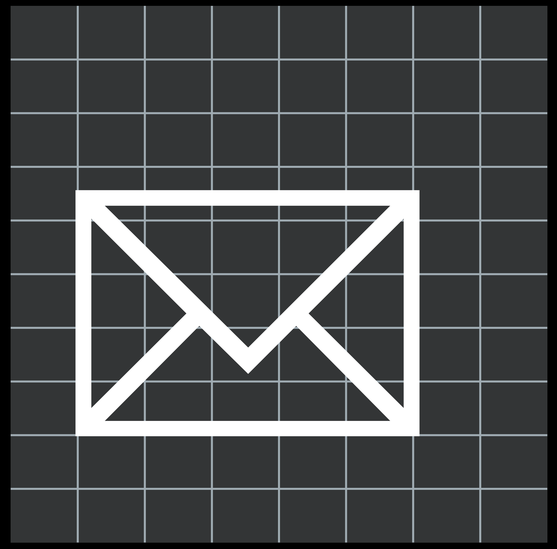
Vector assets can use a 2x grid for optimal stroke placement



# Design for 2x

Point boundary snapping ensures device pixel alignment

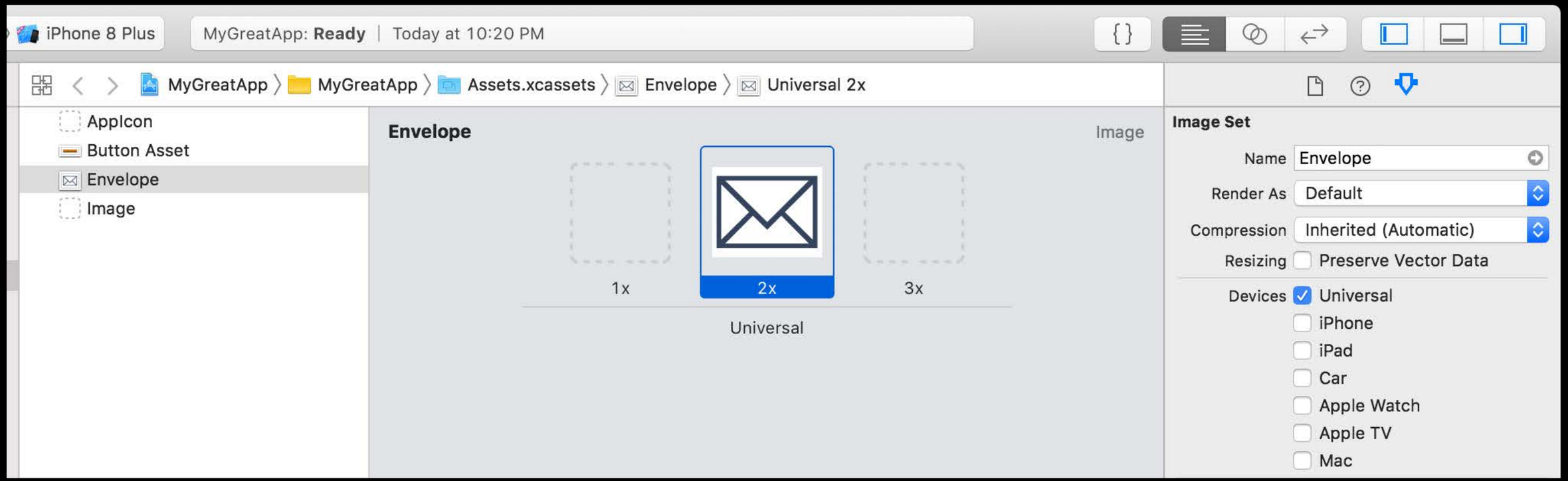
Vector assets can use a 2x grid for optimal stroke placement



# Design for 2x

Drop asset into ‘2x’ slot in Xcode

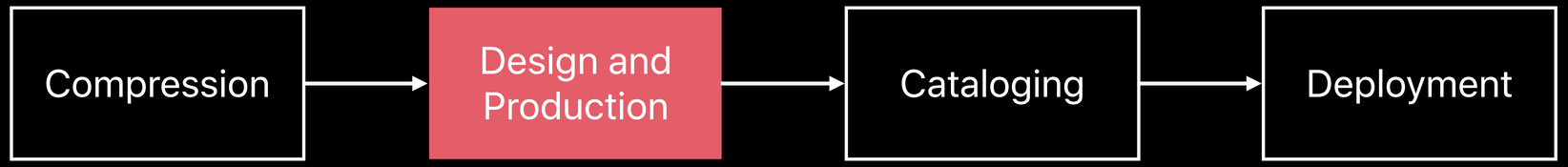
Remaining scales are prepared automatically

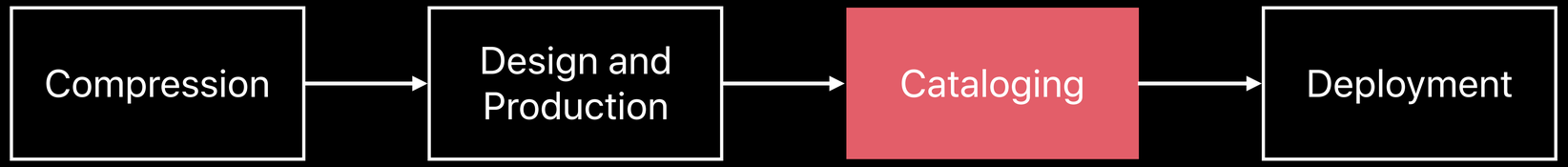


# Hinted Assets

More control of results

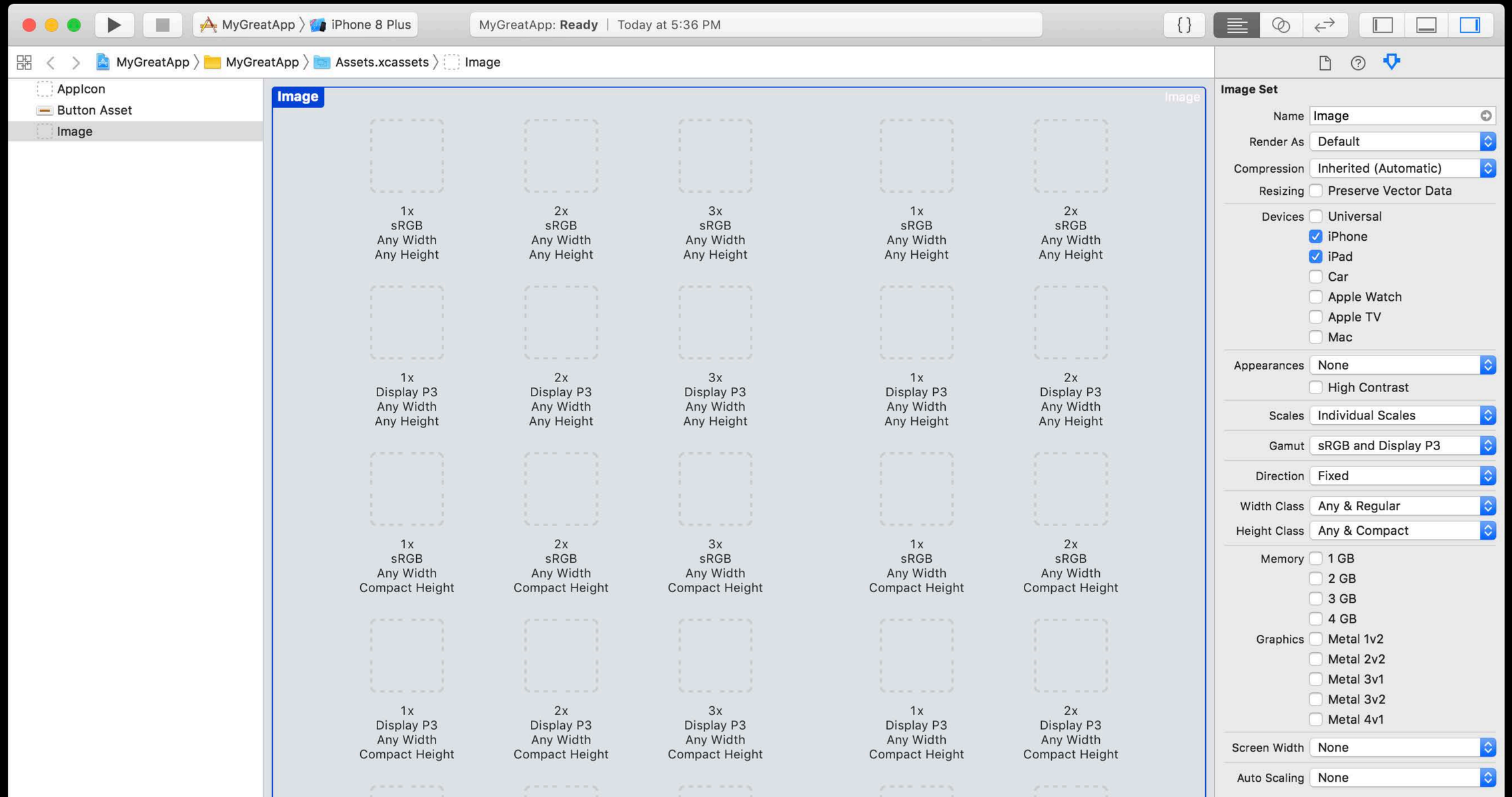
Drop into any slot


Compression
Design and Production
Cataloging
Deployment

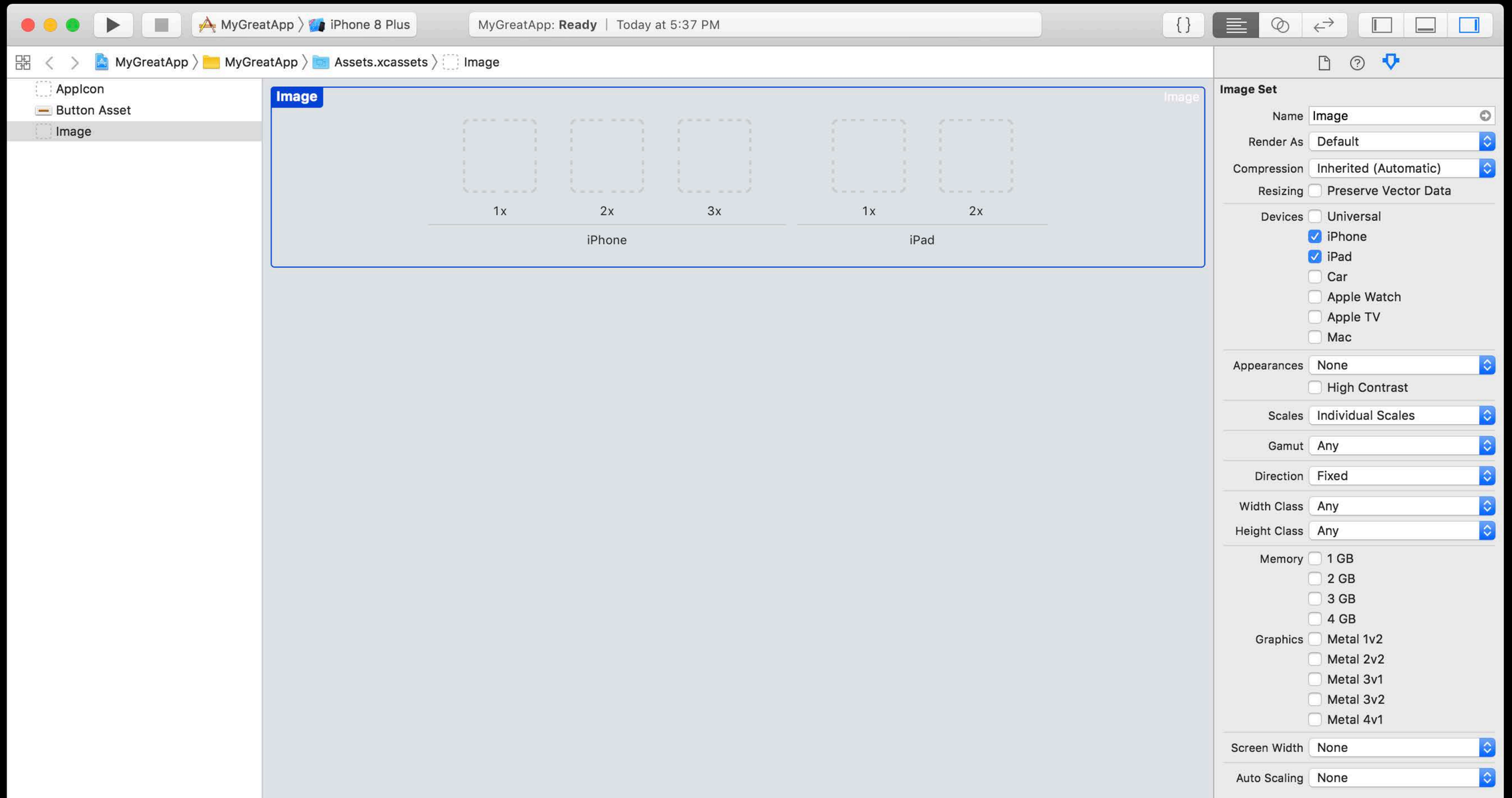

Compression
Design and Production
Cataloging
Deployment

# Cataloging

# Lots of options



# Use only what makes sense



Bundles

Namespaces

Bundles

Namespaces

# Bundles

Large projects present challenges

Solve it with multiple bundles

Effective reuse strategy!

# Bundles

Retrieve with:

```
UIImage(named: UIImage.Name, in: Bundle, compatibleWith: UITraitCollection)
Bundle.image(forResource: NSImage.Name) -> NSImage?
```

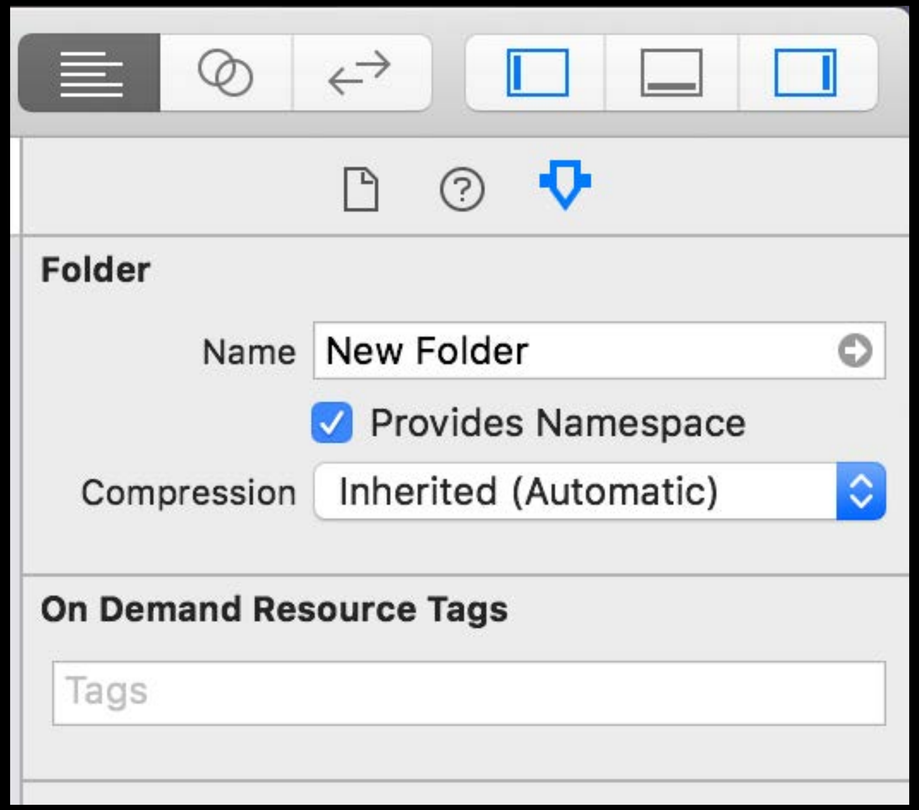
Uniqueness only within bundle scope

Bundles

Namespaces

# Namespaces

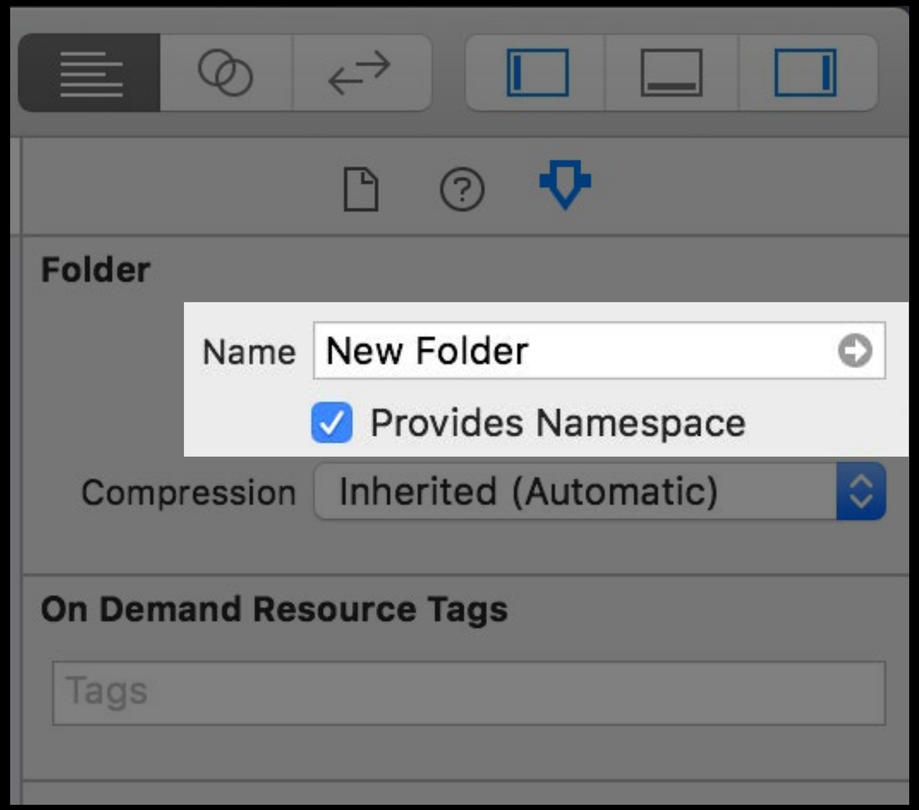
Large collections are a naming challenge

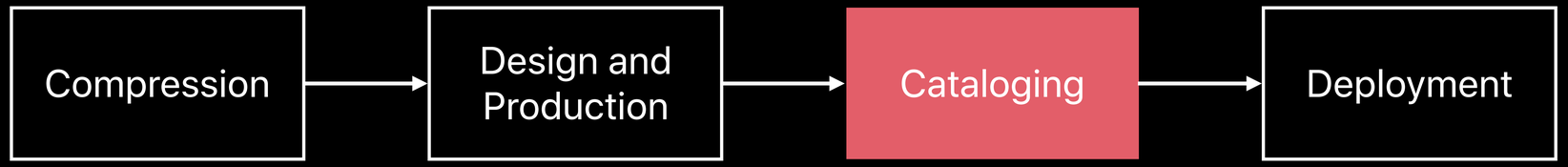


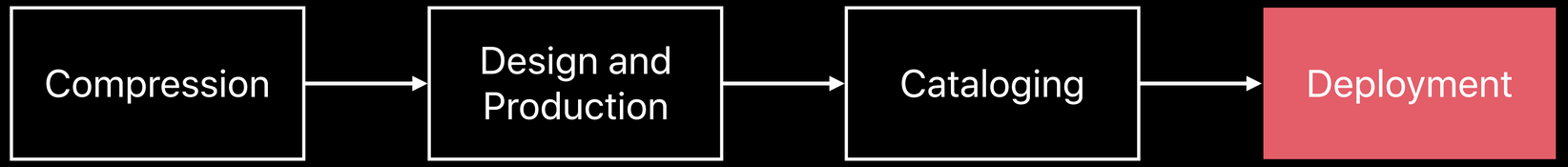
# Namespaces

Large collections are a naming challenge

Folders with namespace for grouping!




Compression
Design and Production
Cataloging
Deployment


Compression
Design and Production
Cataloging
Deployment

# Deployment

# App Thinning

You provide all the content variants

App Thinning picks the right subset per device

Performance classes

Sprite atlases

Performance classes

Sprite atlases

# Performance Classes

Hardware capabilities vary

Don’t constrain to least capable device!

Solve with adaptive resources

# Memory Classes

| 1GB | 2GB | 3GB | 4GB |
| --- | --- | --- | --- |

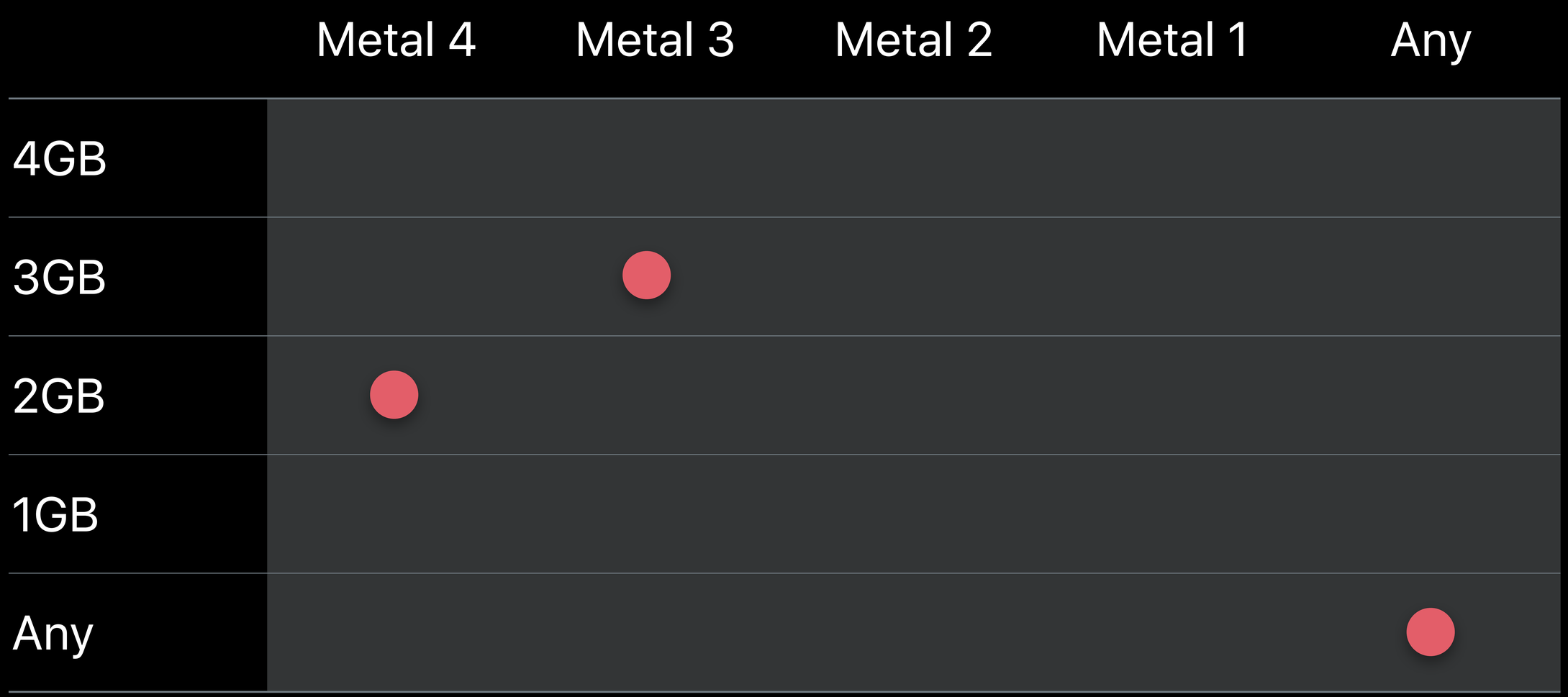
# Graphics Classes


Metal 1 Metal 2 Metal 3 Metal 4

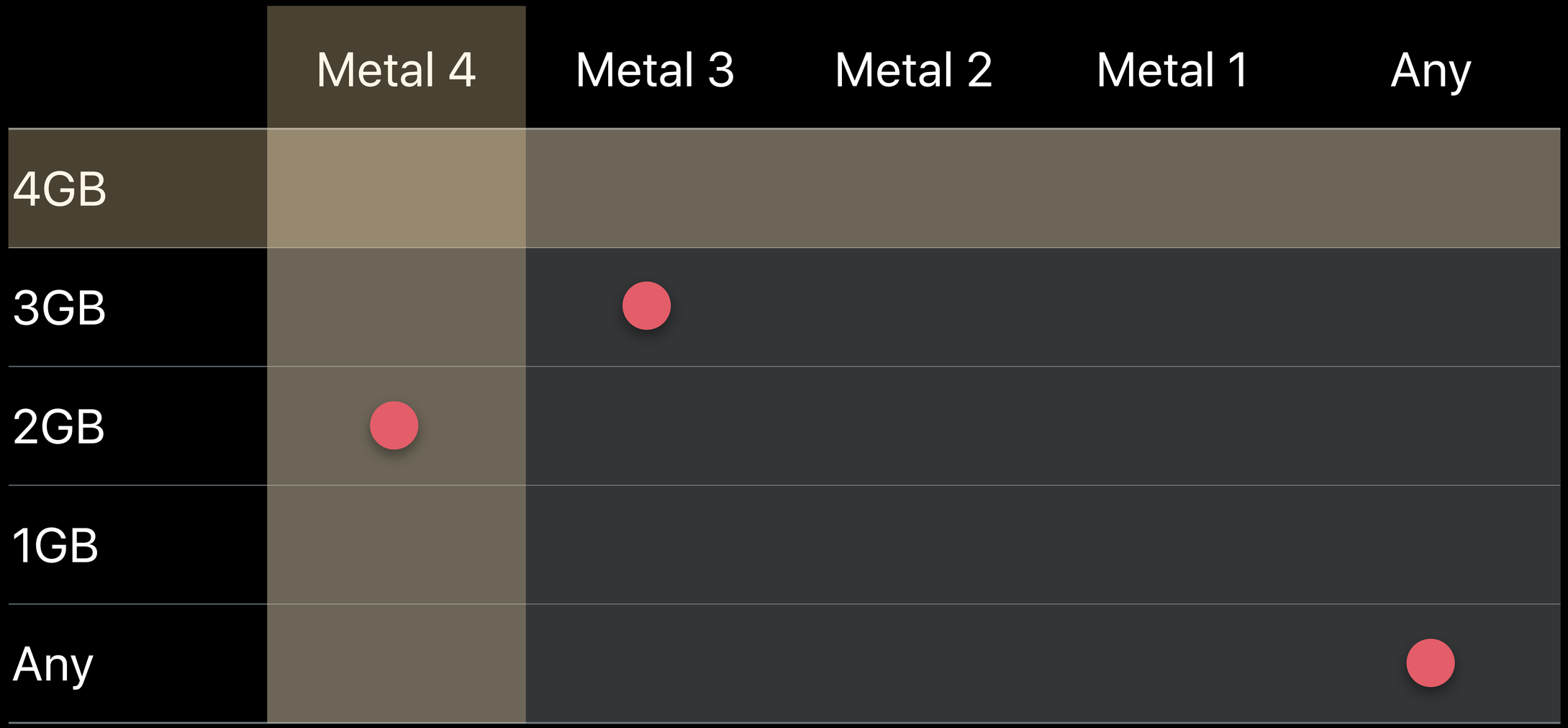
# Full Capability Matrix

| | Metal 4 | Metal 3 | Metal 2 | Metal 1 | Any |
|---|---|---|---|---|---|
| 4GB | | | | | |
| 3GB | | | | | |
| 2GB | | | | | |
| 1GB | | | | | |
| Any | | | | | |

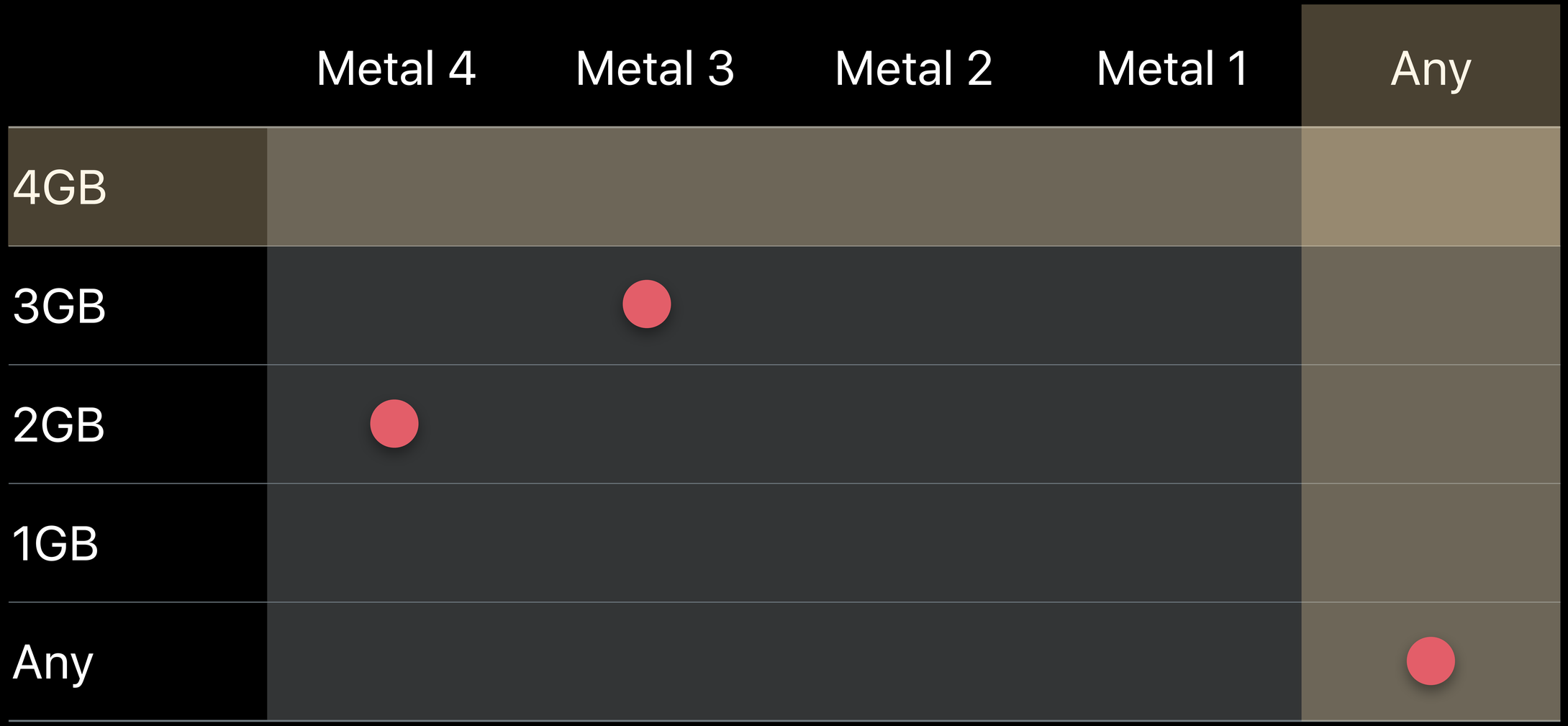
# Full Capability Matrix



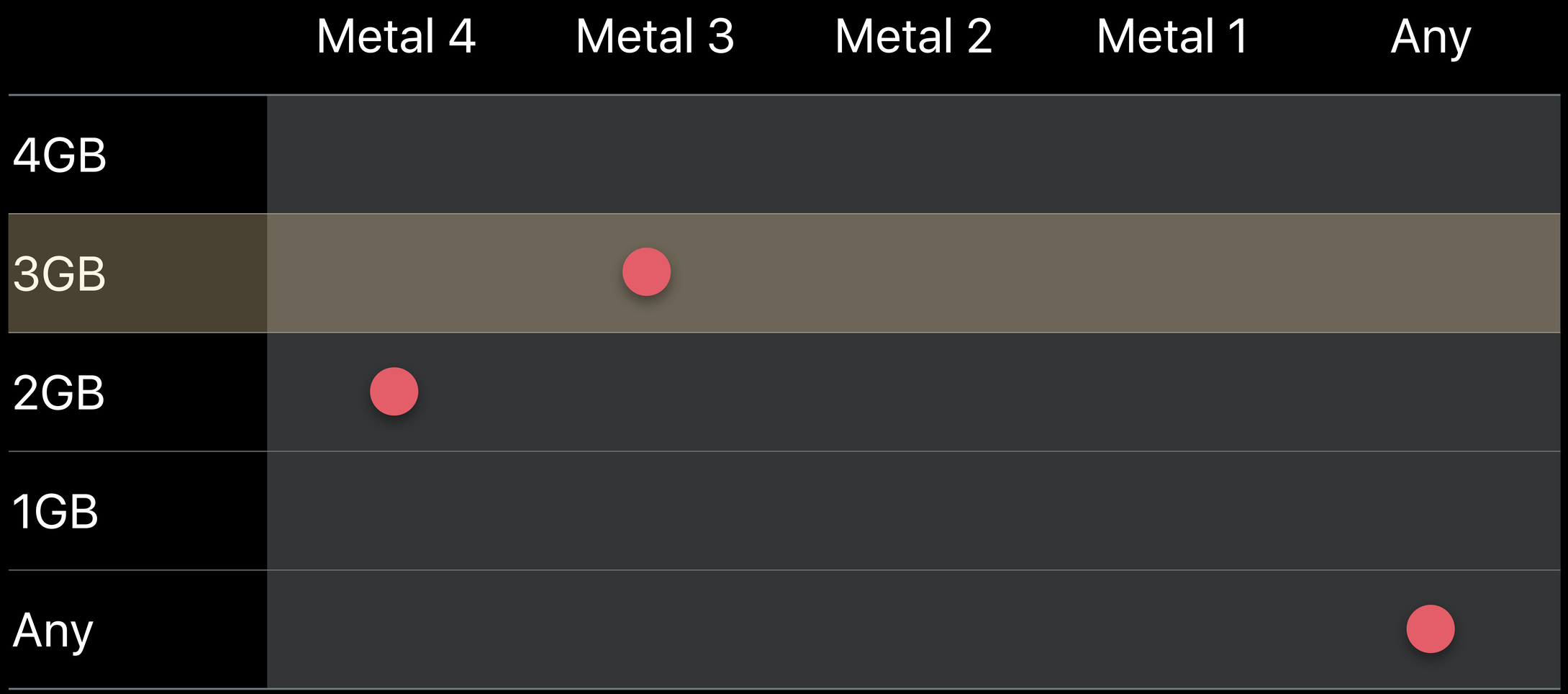
# Full Capability Matrix



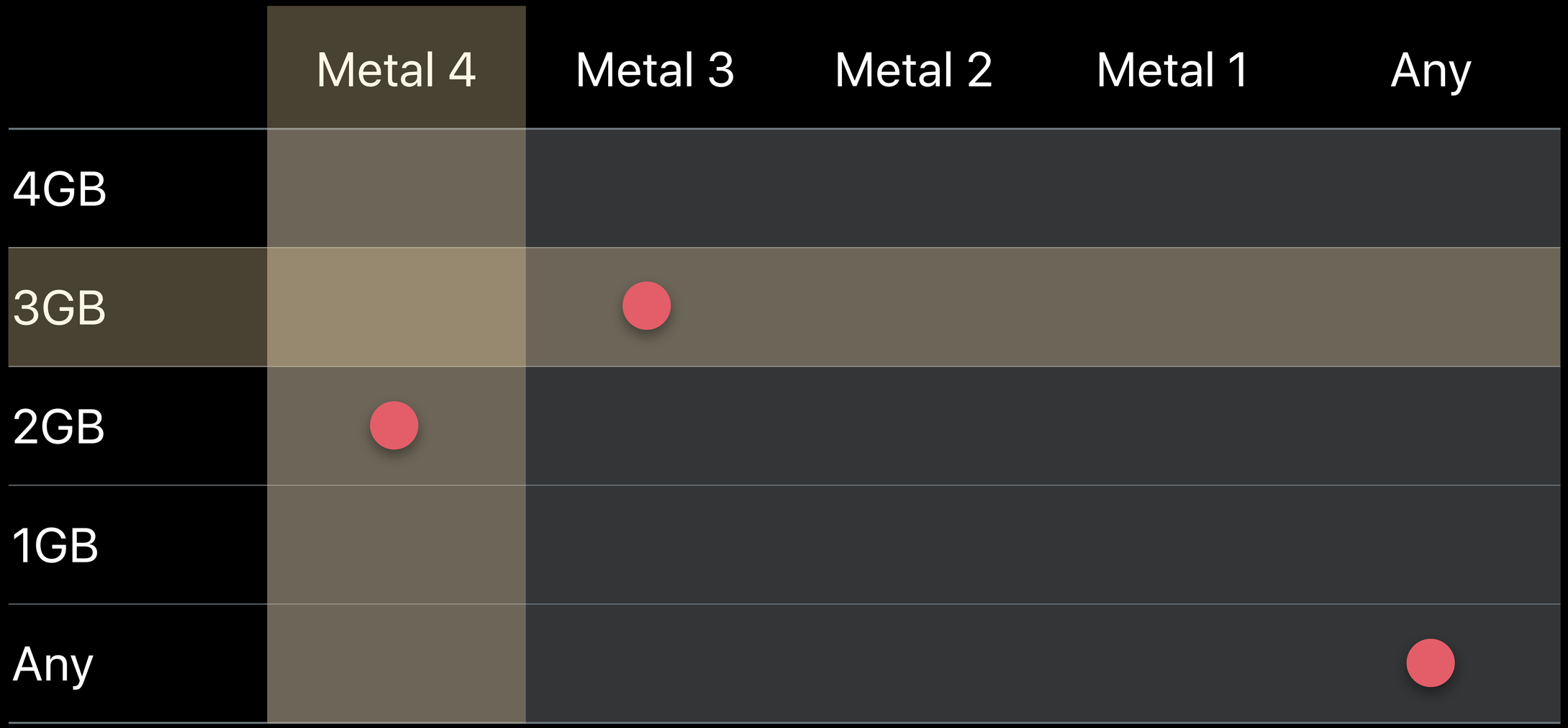
# Full Capability Matrix



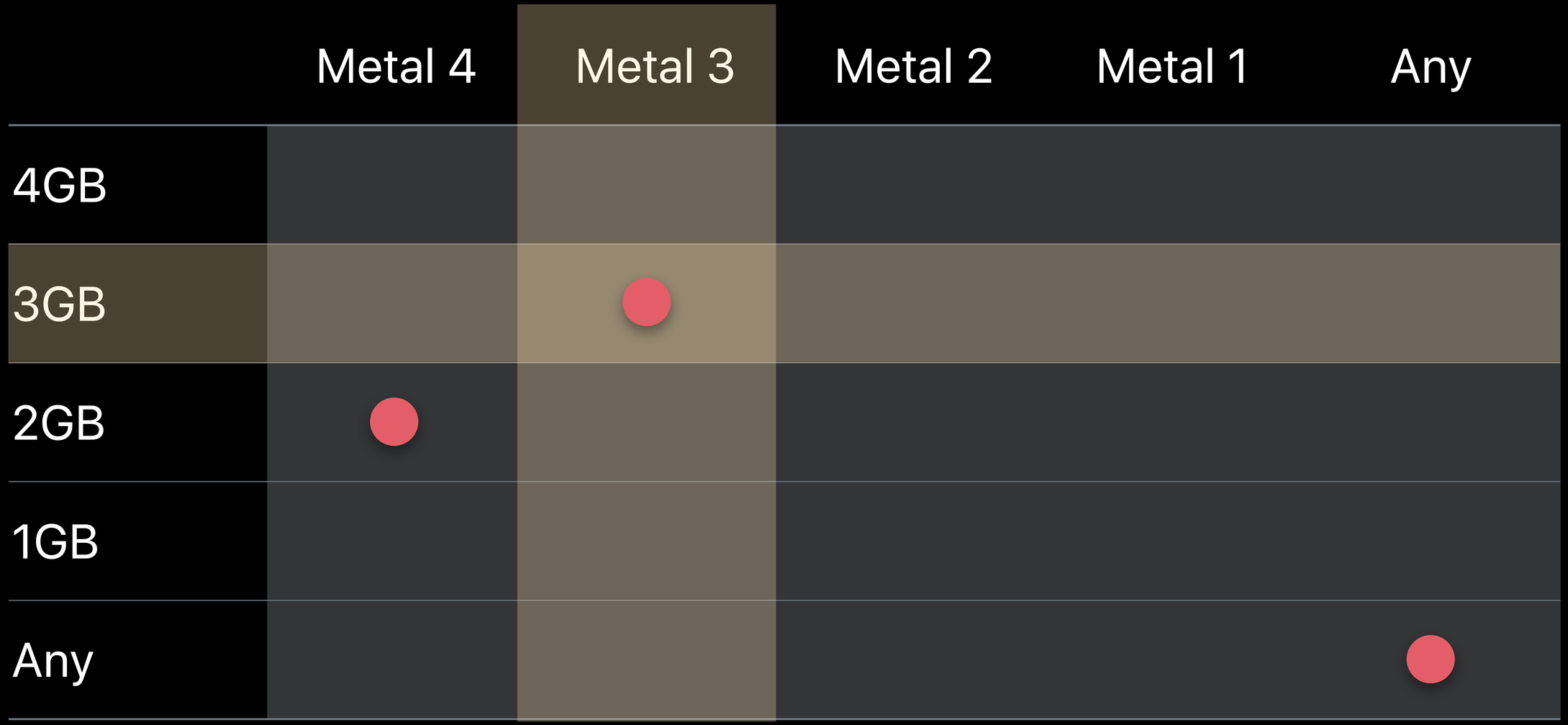
# Full Capability Matrix



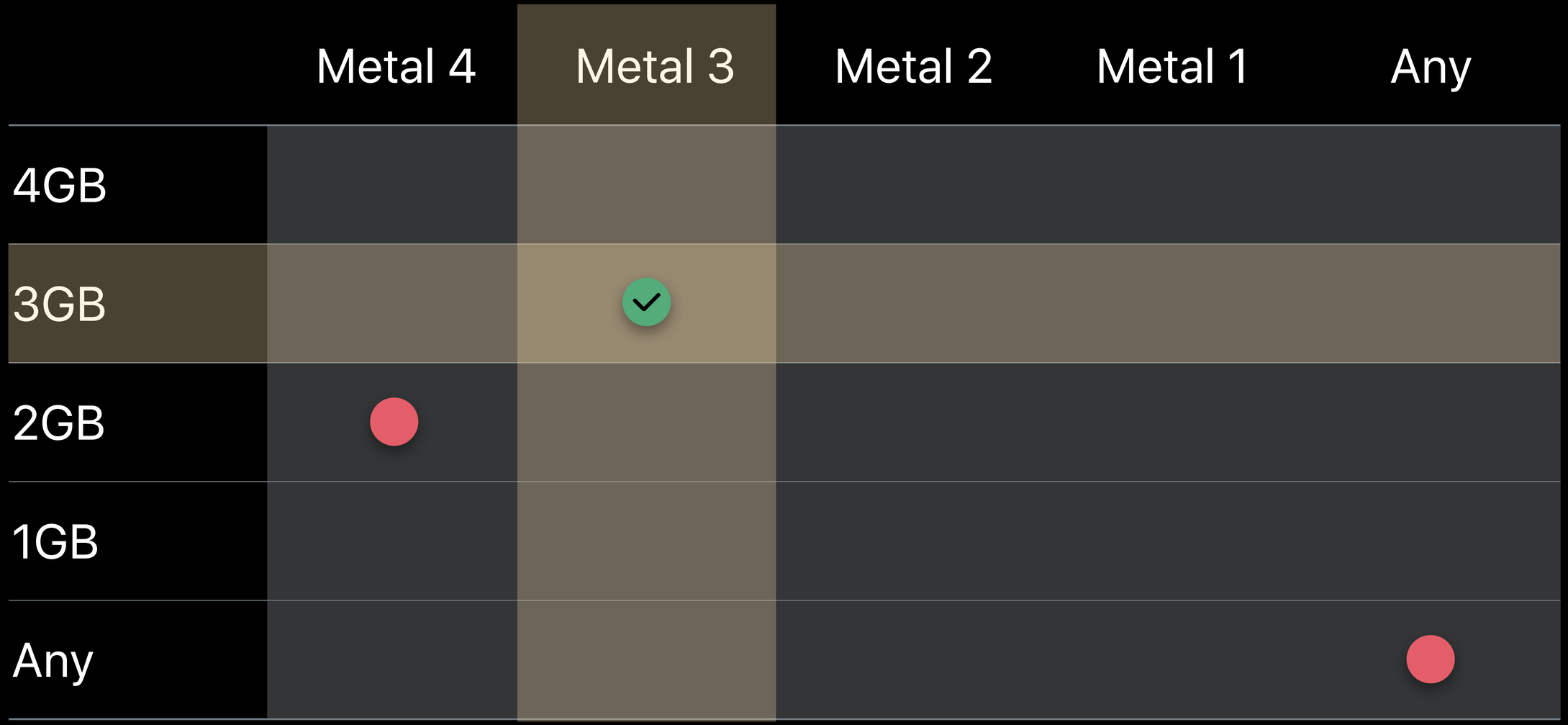
# Full Capability Matrix



# Full Capability Matrix



# Full Capability Matrix



# Using Performance Classes

Higher memory —> larger / richer assets

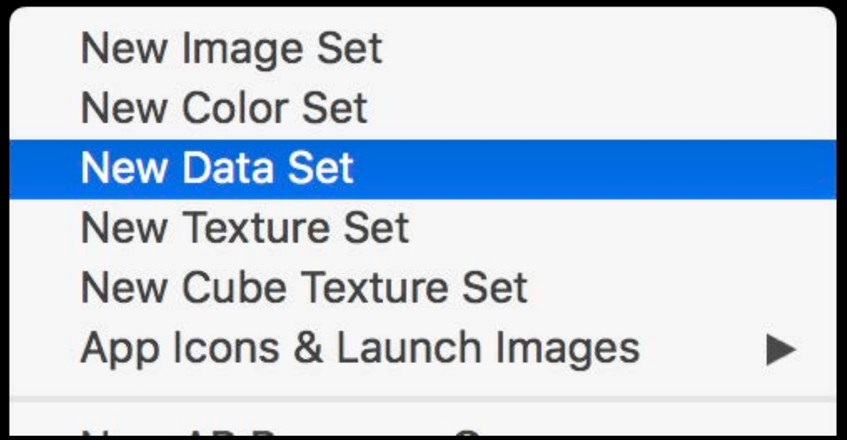
Higher graphics —> more complex assets

# Using Performance Classes

NSDataAsset provides flexible container

Example: Cut scene video

Example: Scene configuration via plist



# Using Performance Classes

NSDataAsset provides flexible container

Example: Cut scene video

Example: Scene configuration via plist

# Using Performance Classes

NSDataAsset provides flexible container

Example: Cut scene video

Example: Scene configuration via plist

Performance classes

Sprite atlases

# Sprite Atlases

Pack related images into a single unit

Entire atlas gets loaded at once

Images reference location within atlas

# Sprite Atlases

Access images using UIImage / NSImage

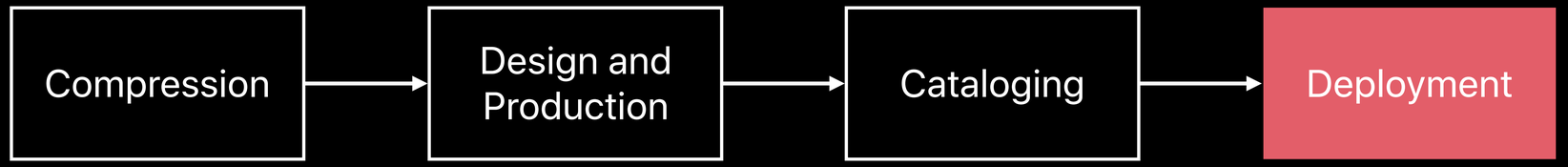
Use asynchronous loading for special cases

```
SKTextureAtlas.preloadTextureAtlasesNamed(_: [String],
                        withCompletionHandler: (Error?, [SKTextureAtlas]) -> Void)
```

# Sprite Atlases

Xcode automatically optimizes for App Thinning

Split by pixel format, device traits and compression type


Compression
Design and Production
Cataloging
Deployment

# Optimizing App Assets

Best choice  for image resources

10–20% less space for iOS 12 apps

App Thinning optimizes for latest OS

Use cataloging to adapt resources to your app

# More Information

https://developer.apple.com/wwdc18/227


WWDC18