Asset_less

asset catalog on a diet

Wolfgang Muhsal

- M.Sc. in Computer Science, TU Darmstadt, 2012
- Professionally developing for iOS since 2012
  - first contact with Obj-C/iOS-SDK in 2010
- Currently Freelancer in Berlin
  - projects with Audi, Telekom, something in Australia, …

The Story

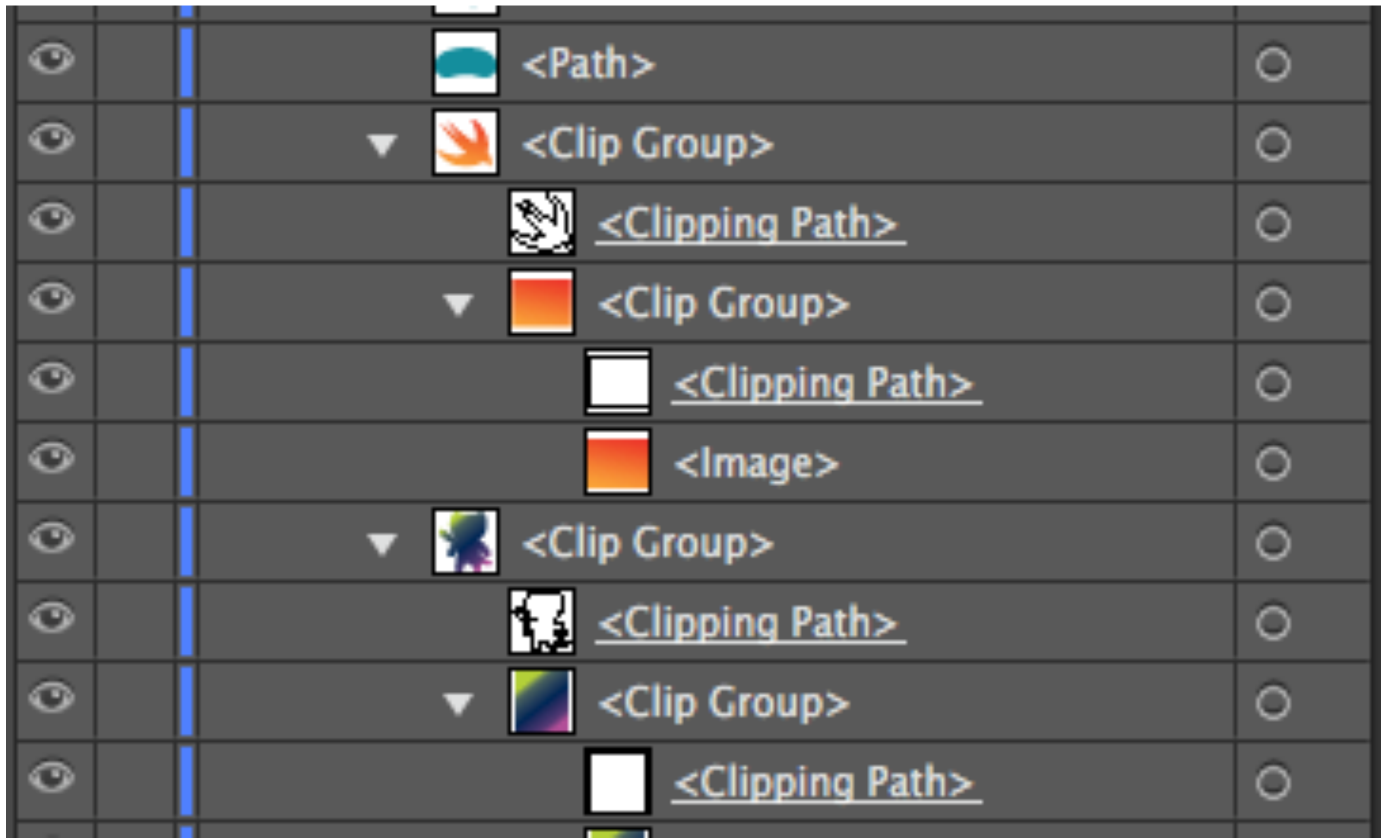
The Asset Catalog

- 110mb on disk, JPG and PNG and PDF and MP4
- App updates every 2 weeks, sums up to a lot
- Compile time >5min
    - and Xcode does randomly compile again

Content

- Similar images

- Complex-layered PDFs

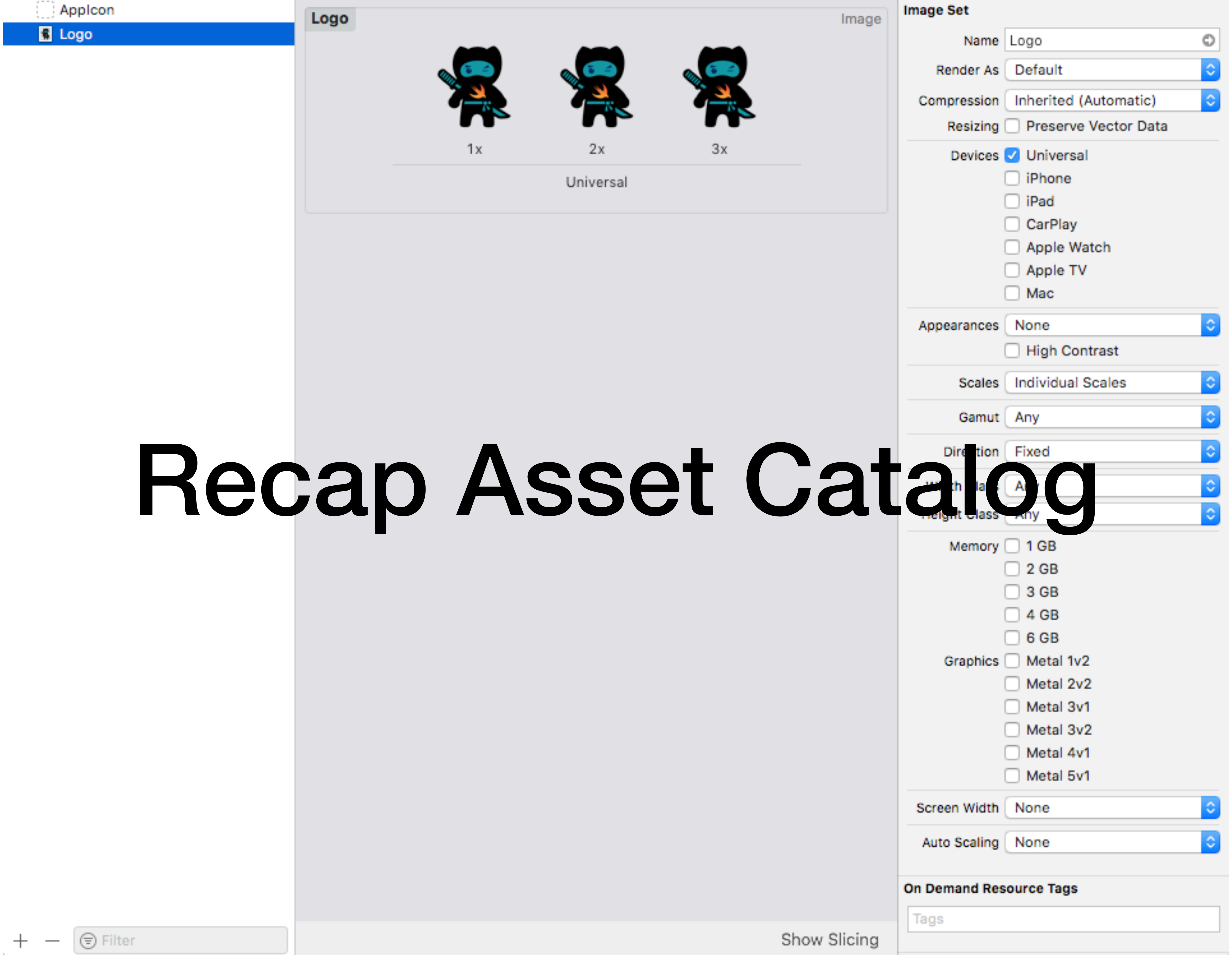


<Path>
<Clip Group>
<Clipping Path>
<Clip Group>
<Clipping Path>
<Image>
<Clip Group>
<Clipping Path>
<Clip Group>
<Clipping Path>

Content

- Tiny movies with short animations
- Other „Animations“ (onboarding sequences, …)

Recap Asset Catalog


AppIcon
Logo

Logo — Image
1x 2x 3x
Universal

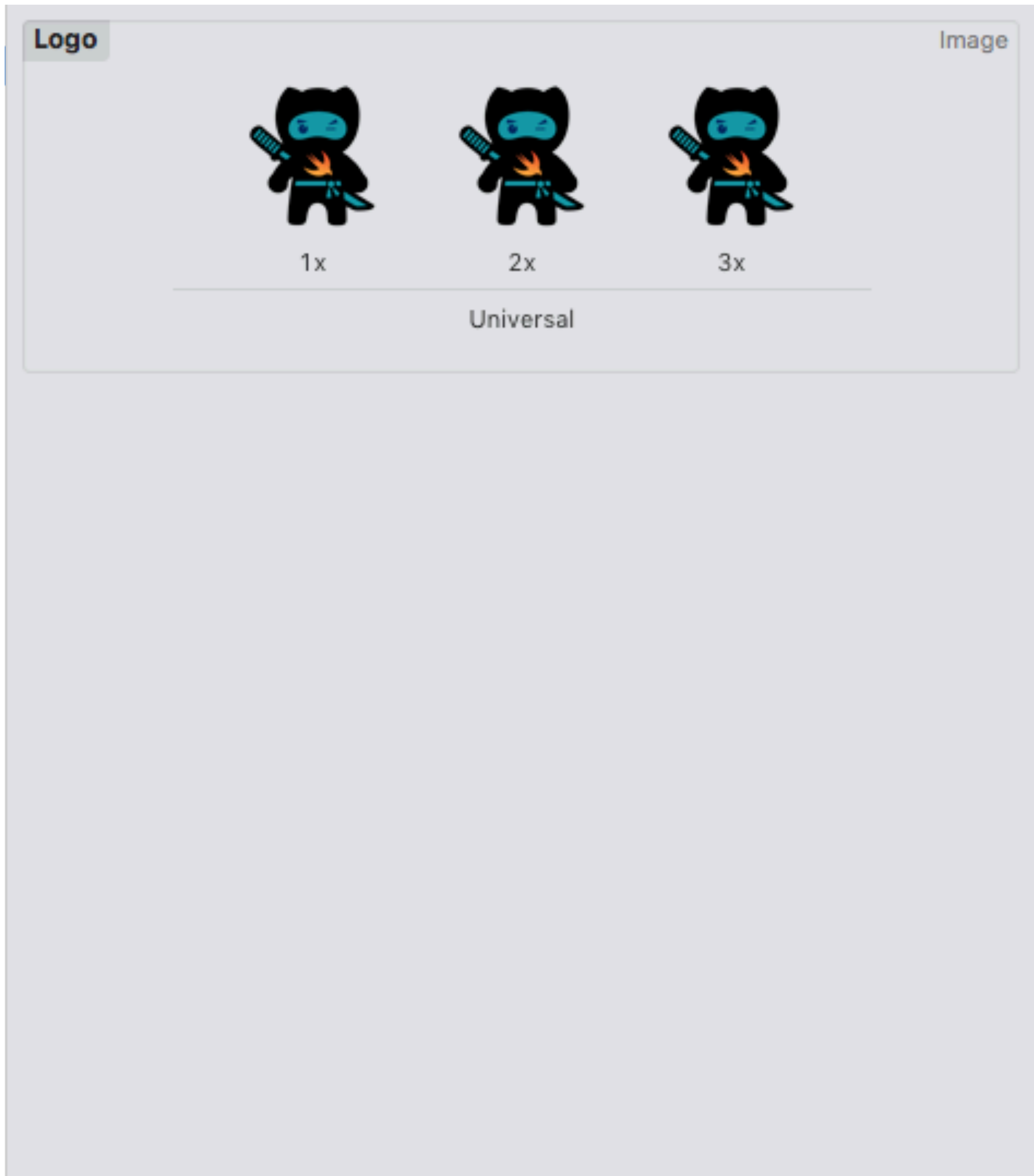
Image Set

Name: Logo
Render As: Default
Compression: Inherited (Automatic)
Resizing: [ ] Preserve Vector Data
Devices: [✓ Universal] [ ] iPhone [ ] iPad [ ] CarPlay [ ] Apple Watch [ ] Apple TV [ ] Mac
Appearances: None
[ ] High Contrast
Scales: Individual Scales
Gamut: Any
Direction: Fixed
Width Class: Any
Height Class: Any
Memory: [ ] 1 GB [ ] 2 GB [ ] 3 GB [ ] 4 GB [ ] 6 GB
Graphics: [ ] Metal 1v2 [ ] Metal 2v2 [ ] Metal 3v1 [ ] Metal 3v2 [ ] Metal 4v1 [ ] Metal 5v1
Screen Width: None
Auto Scaling: None

On Demand Resource Tags
Tags

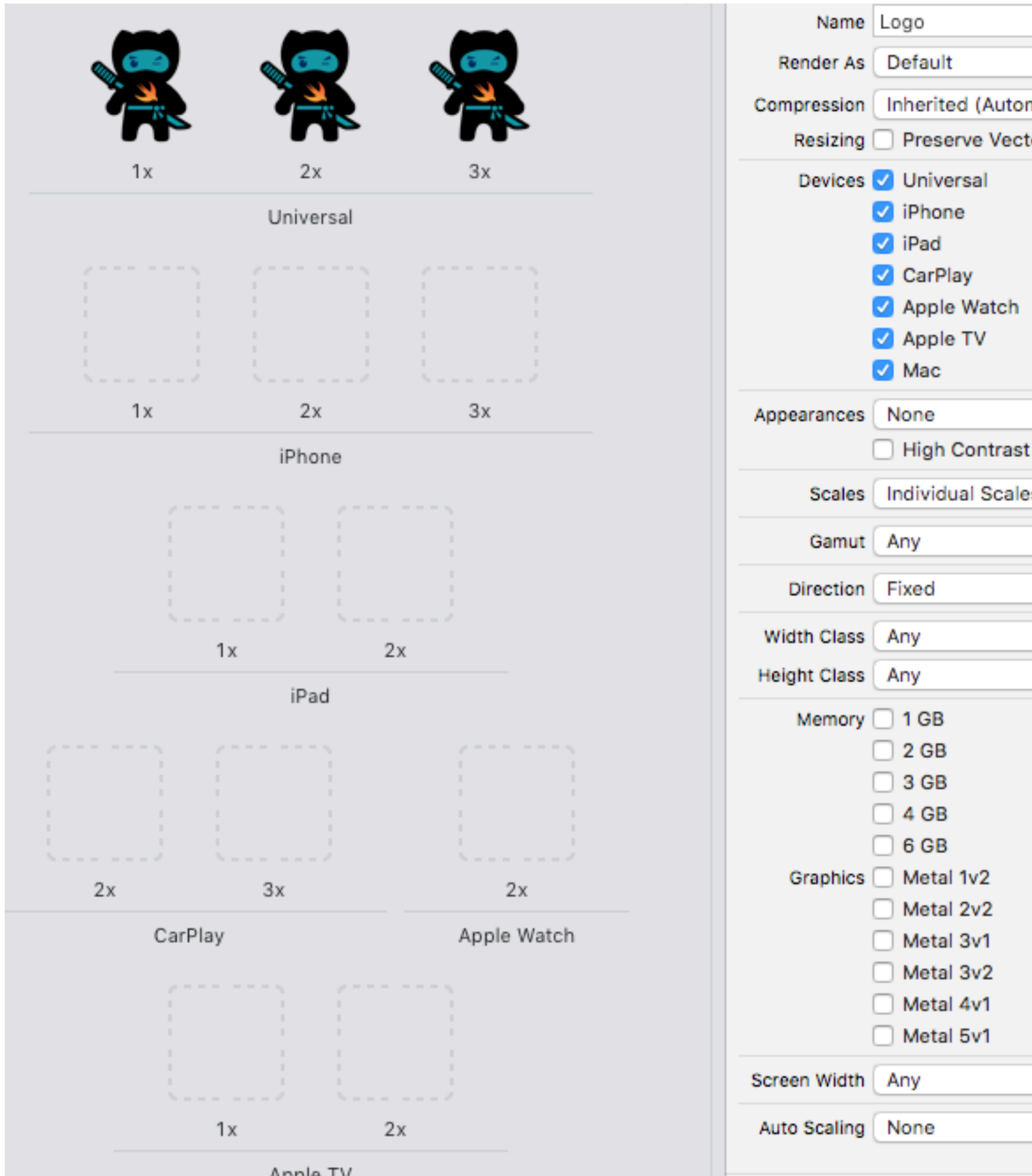
What the asset catalog can do

- Manage various sizes of an image



What the asset catalog can do

- Manage various sizes of an image
  - by device
  - memory
  - metal version
  - dark/light mode
  - direction
  - …



Name: Logo
Render As: Default
Compression: Inherited (Auton
Resizing: ☐ Preserve Vect
Devices: ✓ Universal, ✓ iPhone, ✓ iPad, ✓ CarPlay, ✓ Apple Watch, ✓ Apple TV, ✓ Mac
Appearances: None, ☐ High Contrast
Scales: Individual Scale
Gamut: Any
Direction: Fixed
Width Class: Any
Height Class: Any
Memory: ☐ 1 GB, ☐ 2 GB, ☐ 3 GB, ☐ 4 GB, ☐ 6 GB
Graphics: ☐ Metal 1v2, ☐ Metal 2v2, ☐ Metal 3v1, ☐ Metal 3v2, ☐ Metal 4v1, ☐ Metal 5v1
Screen Width: Any
Auto Scaling: None

What the asset catalog can do

- Take one PDF
  - converted on compile time to every size needed

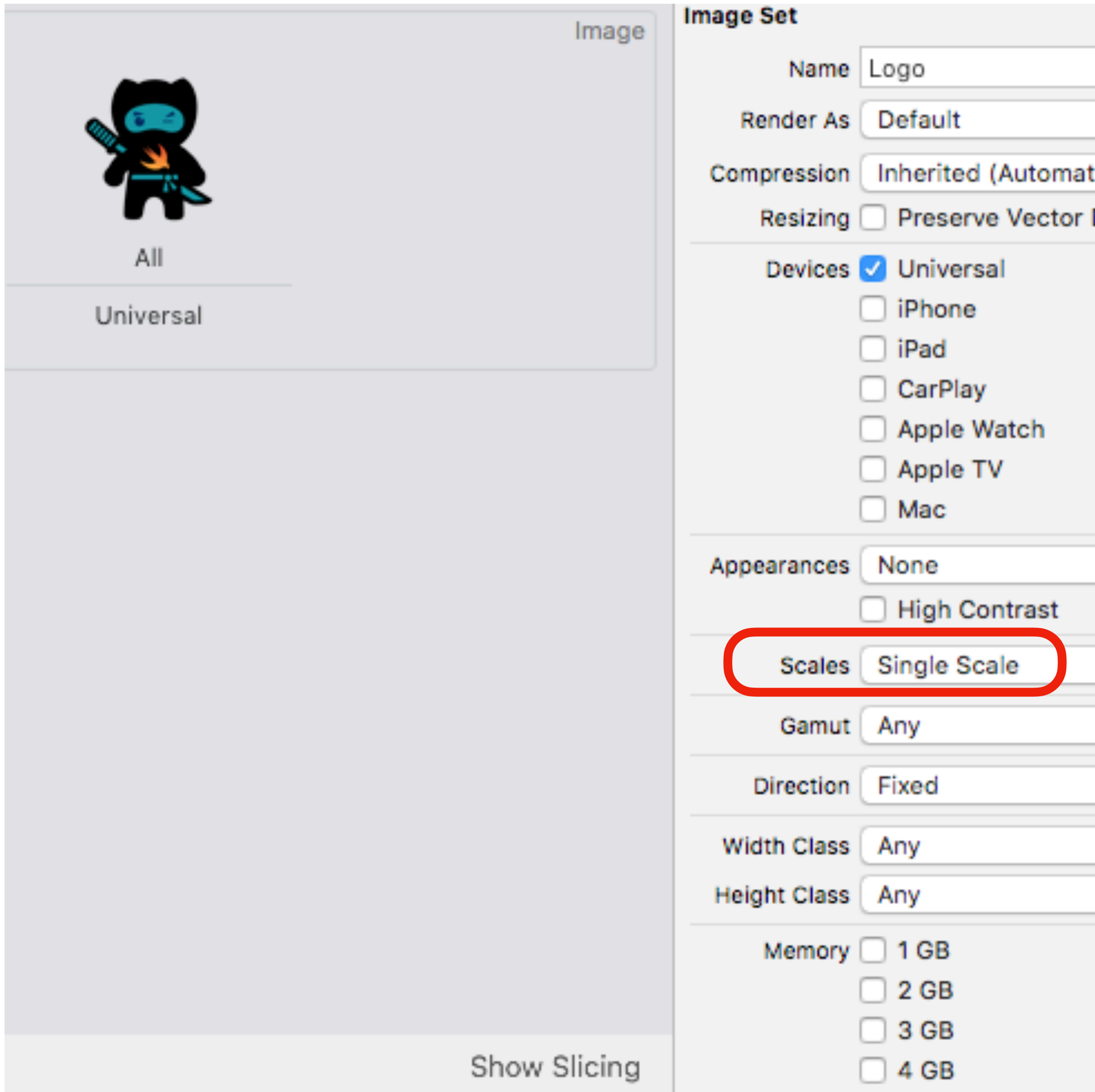
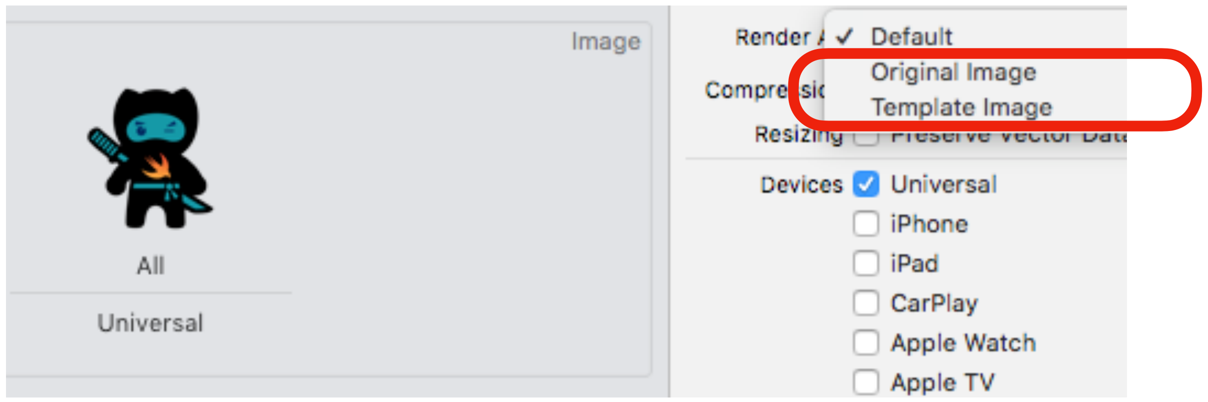


Image Set

Name	Logo
Render As	Default
Compression	Inherited (Automat
Resizing	☐ Preserve Vector
Devices	✓ Universal
	☐ iPhone
	☐ iPad
	☐ CarPlay
	☐ Apple Watch
	☐ Apple TV
	☐ Mac
Appearances	None
	☐ High Contrast
Scales	Single Scale
Gamut	Any
Direction	Fixed
Width Class	Any
Height Class	Any
Memory	☐ 1 GB
	☐ 2 GB
	☐ 3 GB
	☐ 4 GB

What the asset catalog can do



Image

All

Universal

Render A ✓ Default

Original Image

Template Image

Compre si

Resizing ☐ Preserve Vector Data

Devices ✓ Universal

☐ iPhone

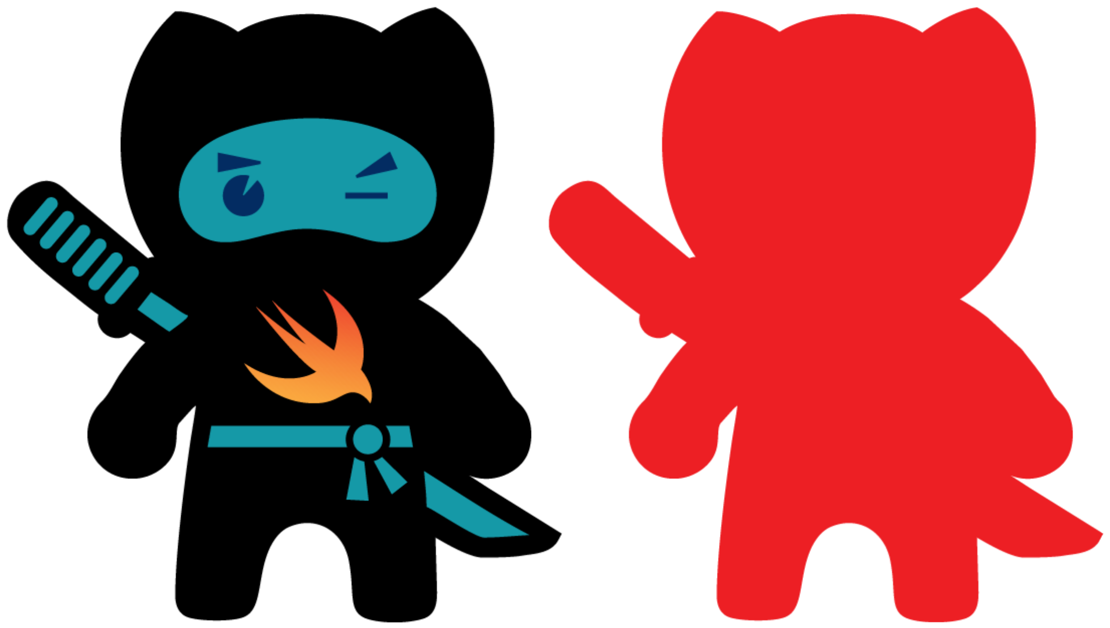
☐ iPad

☐ CarPlay

☐ Apple Watch

☐ Apple TV

- Tint an entire image



But

- All that was not enough or did not help at all…

We analyzed

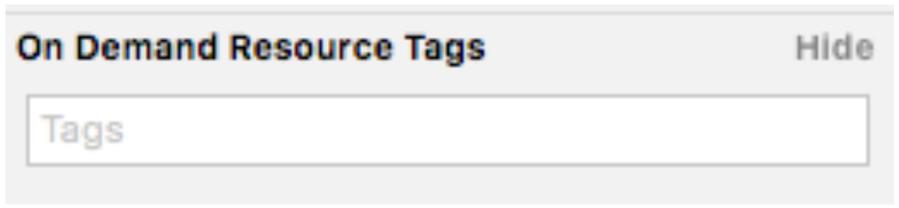
- Most large PDF files (and their generated images) were hardly every used

- Animations were simple vs large file size for a movie

- A lot of pseudo-redundancy

Solutions

- Large and randomly needed files as on demand resources?
  - App should work offline
- More image and video compression
  - Was already pretty good



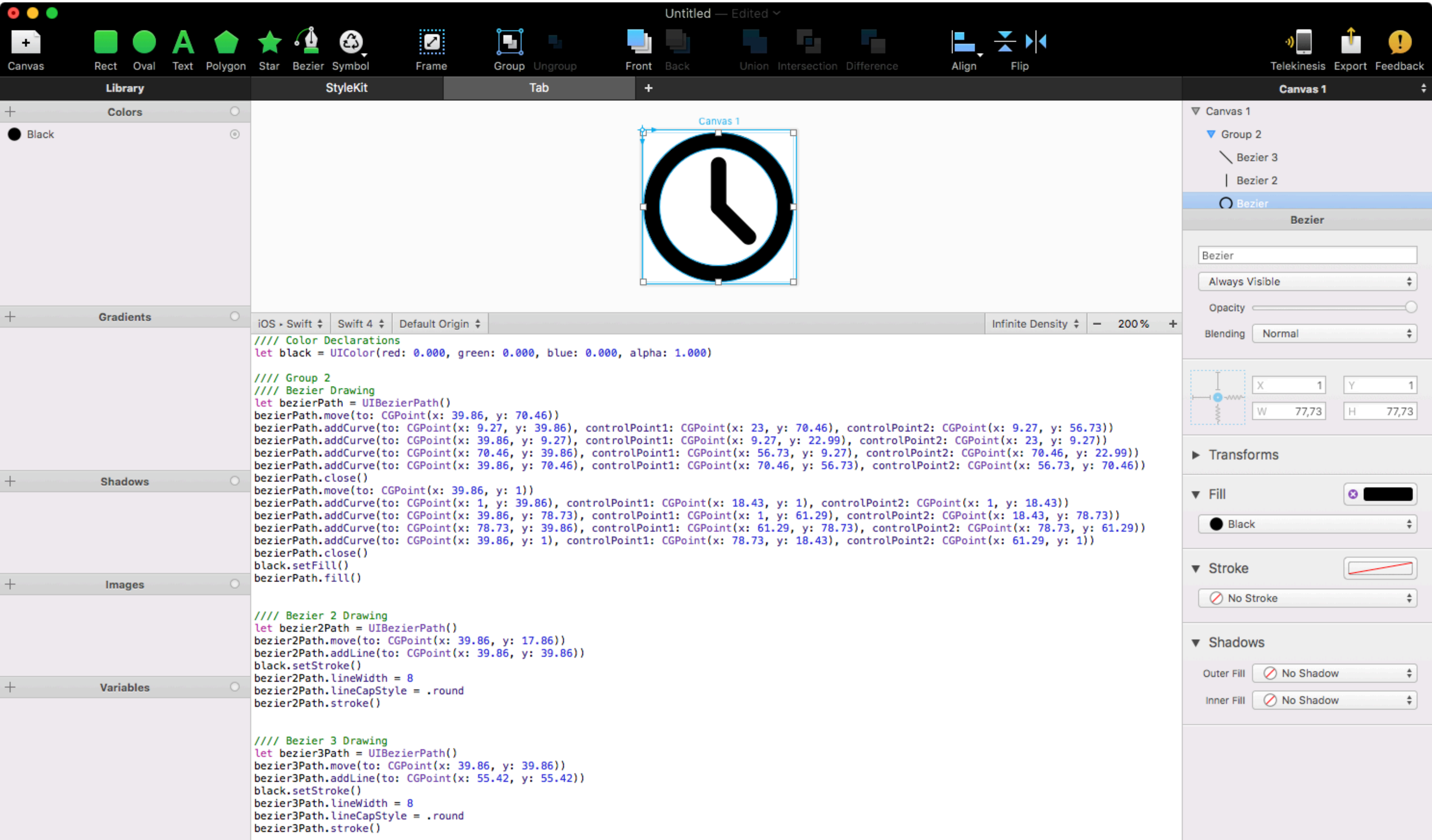
What if…

- We could draw images on runtime
    - instead of having ready to use images

What if…

- We could draw images on runtime and change tiny parts as needed
  - instead of having pseudo-redundancy

PaintCode



iOS ▸ Swift | Swift 4 | Default Origin | Infinite Density | 200 %

```
//// Color Declarations
let black = UIColor(red: 0.000, green: 0.000, blue: 0.000, alpha: 1.000)

//// Group 2
//// Bezier Drawing
let bezierPath = UIBezierPath()
bezierPath.move(to: CGPoint(x: 39.86, y: 70.46))
bezierPath.addCurve(to: CGPoint(x: 9.27, y: 39.86), controlPoint1: CGPoint(x: 23, y: 70.46), controlPoint2: CGPoint(x: 9.27, y: 56.73))
bezierPath.addCurve(to: CGPoint(x: 39.86, y: 9.27), controlPoint1: CGPoint(x: 9.27, y: 22.99), controlPoint2: CGPoint(x: 23, y: 9.27))
bezierPath.addCurve(to: CGPoint(x: 70.46, y: 39.86), controlPoint1: CGPoint(x: 56.73, y: 9.27), controlPoint2: CGPoint(x: 70.46, y: 22.99))
bezierPath.addCurve(to: CGPoint(x: 39.86, y: 70.46), controlPoint1: CGPoint(x: 70.46, y: 56.73), controlPoint2: CGPoint(x: 56.73, y: 70.46))
bezierPath.close()
bezierPath.move(to: CGPoint(x: 39.86, y: 1))
bezierPath.addCurve(to: CGPoint(x: 1, y: 39.86), controlPoint1: CGPoint(x: 18.43, y: 1), controlPoint2: CGPoint(x: 1, y: 18.43))
bezierPath.addCurve(to: CGPoint(x: 39.86, y: 78.73), controlPoint1: CGPoint(x: 1, y: 61.29), controlPoint2: CGPoint(x: 18.43, y: 78.73))
bezierPath.addCurve(to: CGPoint(x: 78.73, y: 39.86), controlPoint1: CGPoint(x: 61.29, y: 78.73), controlPoint2: CGPoint(x: 78.73, y: 61.29))
bezierPath.addCurve(to: CGPoint(x: 39.86, y: 1), controlPoint1: CGPoint(x: 78.73, y: 18.43), controlPoint2: CGPoint(x: 61.29, y: 1))
bezierPath.close()
black.setFill()
bezierPath.fill()


//// Bezier 2 Drawing
let bezier2Path = UIBezierPath()
bezier2Path.move(to: CGPoint(x: 39.86, y: 17.86))
bezier2Path.addLine(to: CGPoint(x: 39.86, y: 39.86))
black.setStroke()
bezier2Path.lineWidth = 8
bezier2Path.lineCapStyle = .round
bezier2Path.stroke()


//// Bezier 3 Drawing
let bezier3Path = UIBezierPath()
bezier3Path.move(to: CGPoint(x: 39.86, y: 39.86))
bezier3Path.addLine(to: CGPoint(x: 55.42, y: 55.42))
black.setStroke()
bezier3Path.lineWidth = 8
bezier3Path.lineCapStyle = .round
bezier3Path.stroke()
```

Bezier

Name: Bezier
Always Visible
Opacity
Blending: Normal
X: 1 | Y: 1
W: 77,73 | H: 77,73

Fill: Black
Stroke: No Stroke
Shadows
Outer Fill: No Shadow
Inner Fill: No Shadow

```
import UIKit

public class StyleKitName : NSObject {

    //// Drawing Methods

    @objc dynamic public class func drawCanvas1(frame targetFrame: CGRect = CGRect(x: 0, y: 0, width: 80, height: 80),
                                                 resizing: ResizingBehavior = .aspectFit) {
        //// General Declarations
        let context = UIGraphicsGetCurrentContext()!

        //// Resize to Target Frame
        context.saveGState()
        let resizedFrame: CGRect = resizing.apply(rect: CGRect(x: 0, y: 0, width: 80, height: 80), target: targetFrame)
        context.translateBy(x: resizedFrame.minX, y: resizedFrame.minY)
        context.scaleBy(x: resizedFrame.width / 80, y: resizedFrame.height / 80)


        //// Color Declarations
        let black = UIColor(red: 0.000, green: 0.000, blue: 0.000, alpha: 1.000)

        //// Group 2
        //// Bezier Drawing
        let bezierPath = UIBezierPath()
        bezierPath.move(to: CGPoint(x: 39.86, y: 70.46))
        [..]
        bezierPath.close()
        black.setFill()
        bezierPath.fill()


        //// Bezier 2 Drawing
        let bezier2Path = UIBezierPath()
        bezier2Path.move(to: CGPoint(x: 39.86, y: 17.86))
        bezier2Path.addLine(to: CGPoint(x: 39.86, y: 39.86))
        black.setStroke()
        bezier2Path.lineWidth = 8
        bezier2Path.lineCapStyle = .round
        bezier2Path.stroke()


        //// Bezier 3 Drawing
        let bezier3Path = UIBezierPath()
        bezier3Path.move(to: CGPoint(x: 39.86, y: 39.86))
        bezier3Path.addLine(to: CGPoint(x: 55.42, y: 55.42))
        black.setStroke()
        bezier3Path.lineWidth = 8
        bezier3Path.lineCapStyle = .round
        bezier3Path.stroke()

        context.restoreGState()

    }
}
```

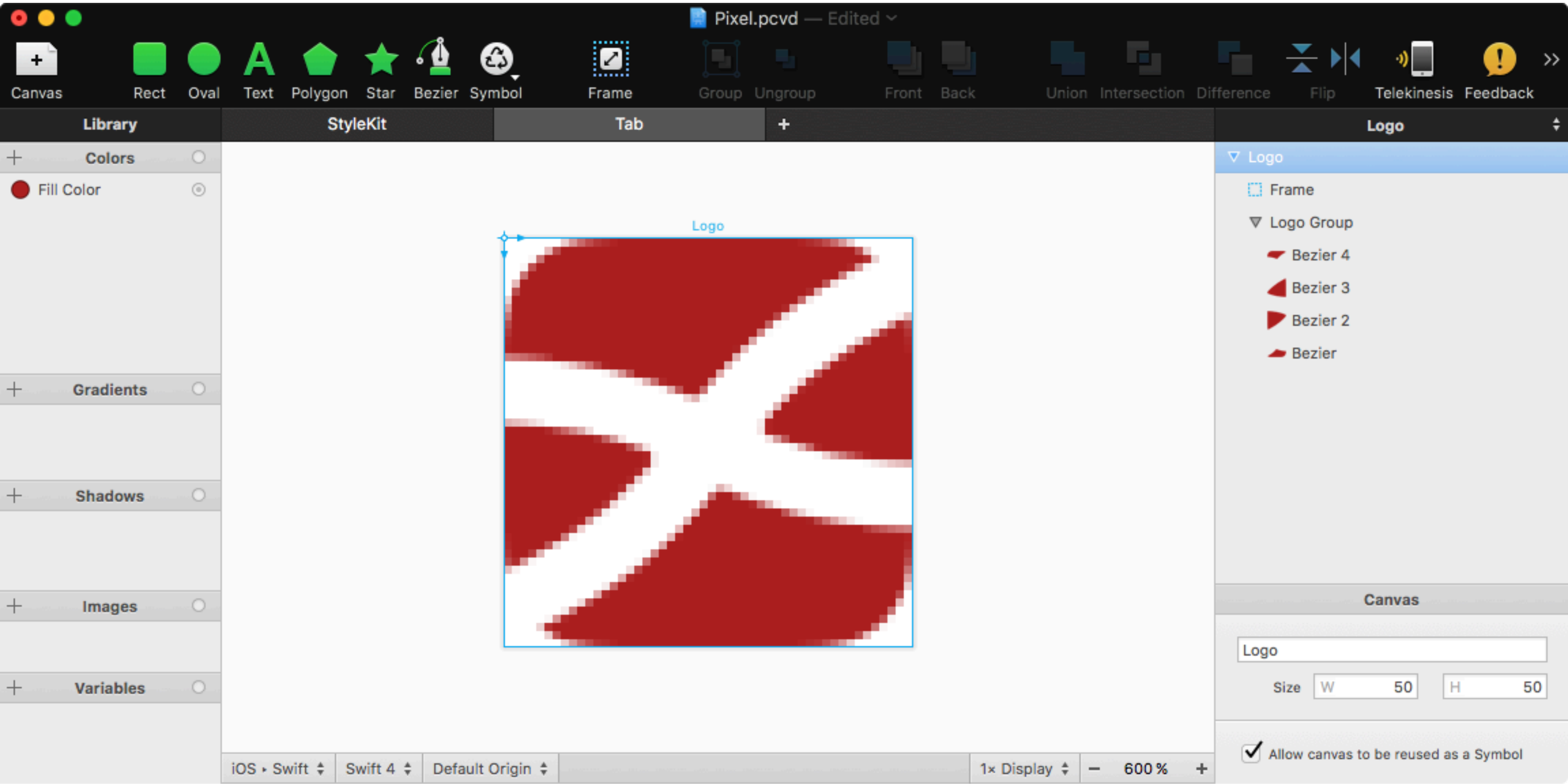
vs

saves just 1/14

PaintCode

- Import vector images from Illustrator or Sketch
- Generates drawing code
  - or code to generate images on the fly
  - smaller than image files
  - (or generate images/PDFs directly from PaintCode)
- And so much more!

PaintCode

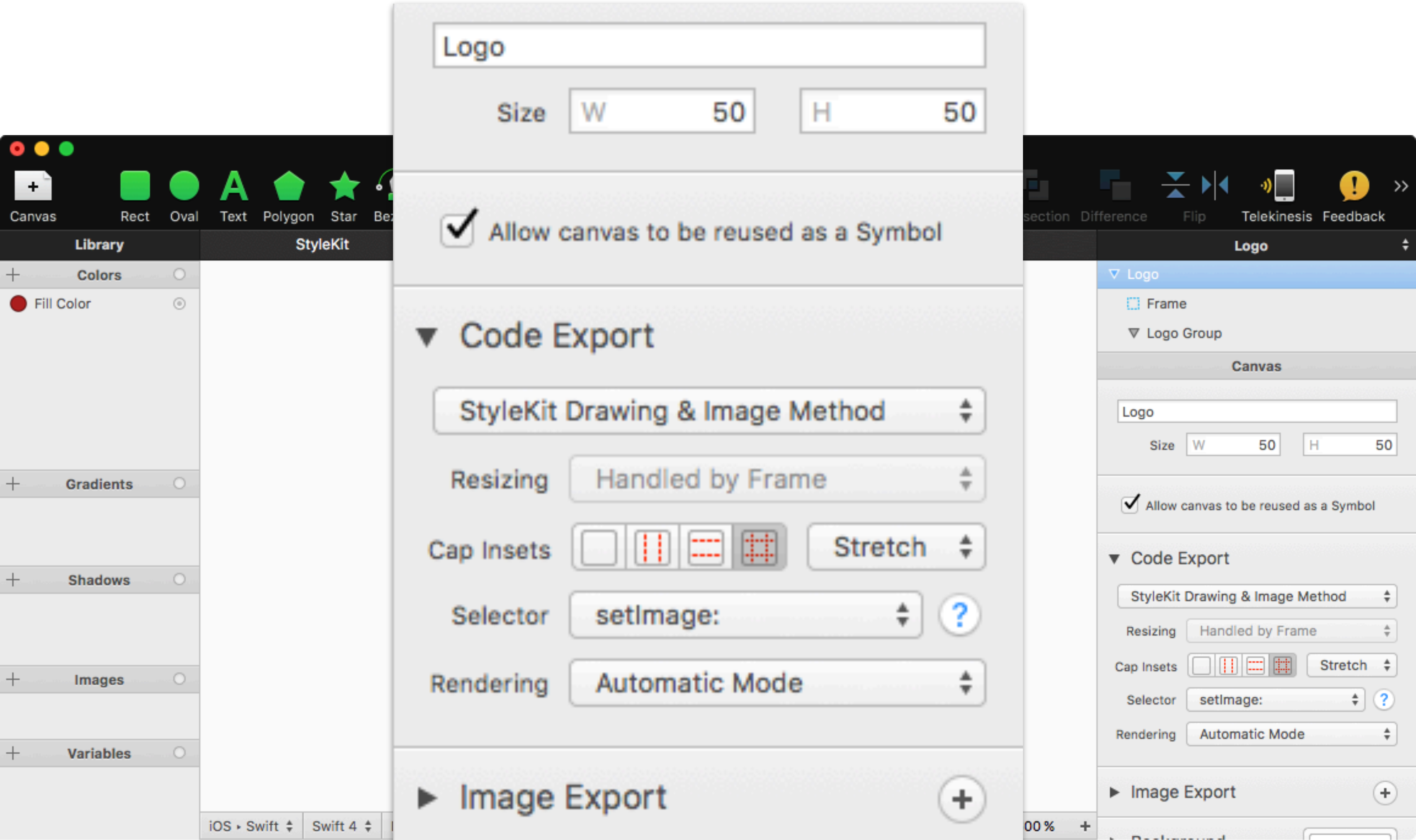


Logo

Size W 50 H 50

✓ Allow canvas to be reused as a Symbol

PaintCode



Logo

Size W 50 H 50

✓ Allow canvas to be reused as a Symbol

Code Export

StyleKit Drawing & Image Method

Resizing: Handled by Frame

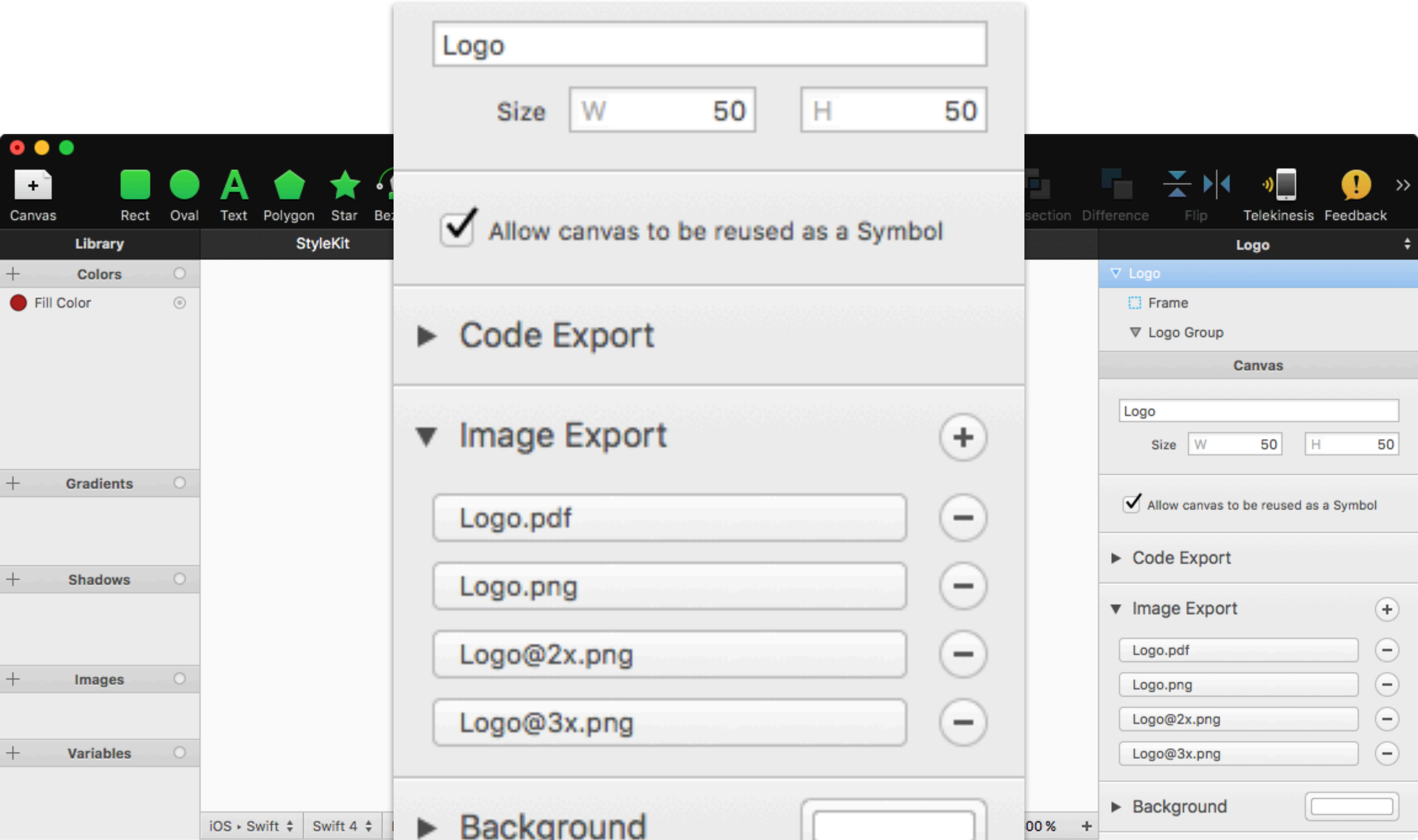
Cap Insets: Stretch

Selector: setImage:

Rendering: Automatic Mode

Image Export

PaintCode



Logo

Size W 50 H 50

✓ Allow canvas to be reused as a Symbol

▶ Code Export

▼ Image Export

Logo.pdf

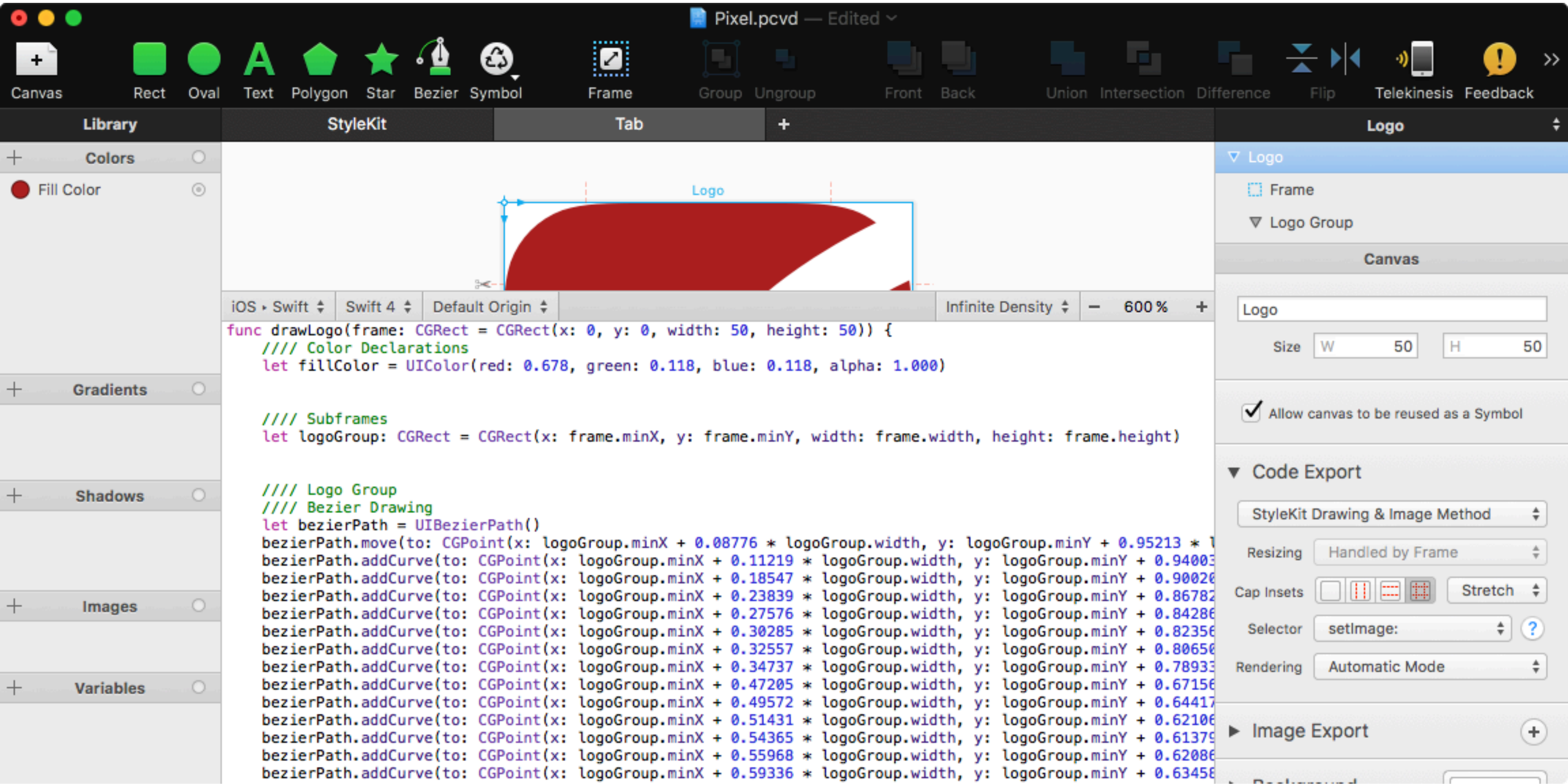
Logo.png

Logo@2x.png

Logo@3x.png

▶ Background

PaintCode



iOS ▸ Swift | Swift 4 | Default Origin | Infinite Density | 600 %

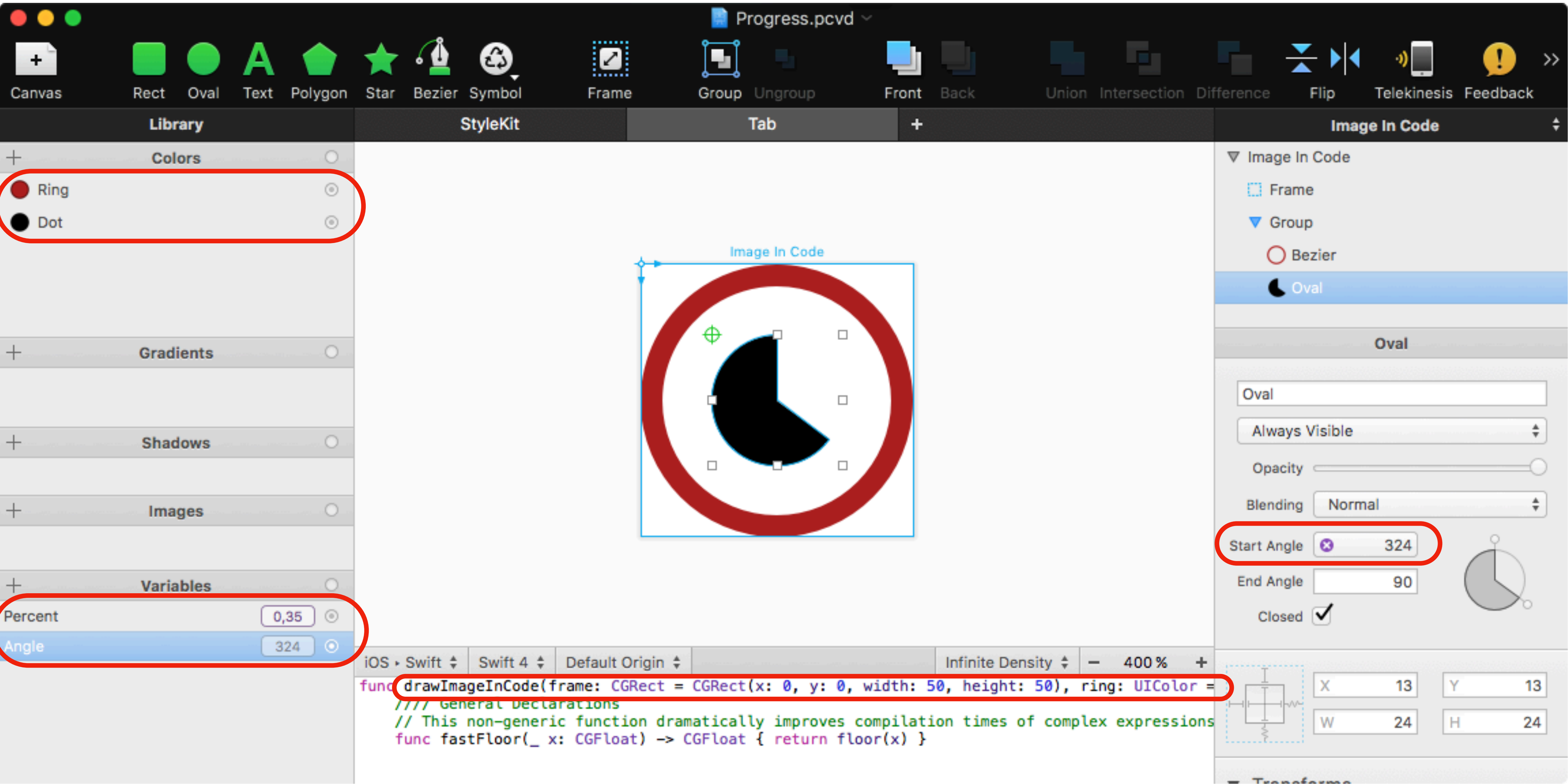
```
func drawLogo(frame: CGRect = CGRect(x: 0, y: 0, width: 50, height: 50)) {
    //// Color Declarations
    let fillColor = UIColor(red: 0.678, green: 0.118, blue: 0.118, alpha: 1.000)


    //// Subframes
    let logoGroup: CGRect = CGRect(x: frame.minX, y: frame.minY, width: frame.width, height: frame.height)


    //// Logo Group
    //// Bezier Drawing
    let bezierPath = UIBezierPath()
    bezierPath.move(to: CGPoint(x: logoGroup.minX + 0.08776 * logoGroup.width, y: logoGroup.minY + 0.95213 * l
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.11219 * logoGroup.width, y: logoGroup.minY + 0.94003
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.18547 * logoGroup.width, y: logoGroup.minY + 0.90020
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.23839 * logoGroup.width, y: logoGroup.minY + 0.86782
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.27576 * logoGroup.width, y: logoGroup.minY + 0.84286
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.30285 * logoGroup.width, y: logoGroup.minY + 0.82356
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.32557 * logoGroup.width, y: logoGroup.minY + 0.80650
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.34737 * logoGroup.width, y: logoGroup.minY + 0.78933
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.47205 * logoGroup.width, y: logoGroup.minY + 0.67156
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.49572 * logoGroup.width, y: logoGroup.minY + 0.64417
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.51431 * logoGroup.width, y: logoGroup.minY + 0.62106
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.54365 * logoGroup.width, y: logoGroup.minY + 0.61379
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.55968 * logoGroup.width, y: logoGroup.minY + 0.62086
    bezierPath.addCurve(to: CGPoint(x: logoGroup.minX + 0.59336 * logoGroup.width, y: logoGroup.minY + 0.63458
```

Logo
Size W 50 H 50
✓ Allow canvas to be reused as a Symbol
StyleKit Drawing & Image Method
Resizing: Handled by Frame
Cap Insets: Stretch
Selector: setImage:
Rendering: Automatic Mode

PaintCode



Colors
- Ring
- Dot

Variables

Percent	0,35
Angle	324

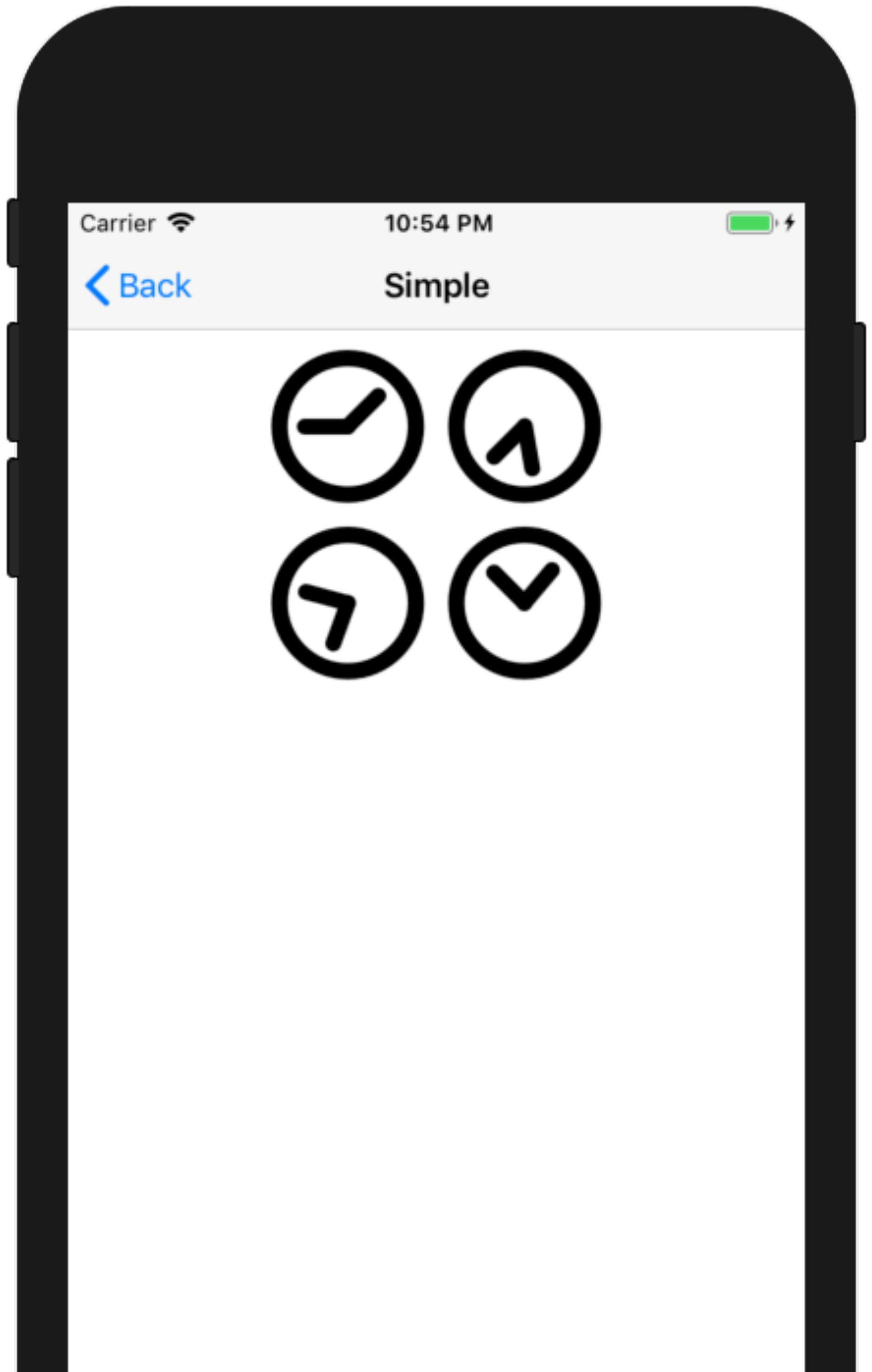
iOS ▸ Swift · Swift 4 · Default Origin · Infinite Density · 400 %

```
func drawImageInCode(frame: CGRect = CGRect(x: 0, y: 0, width: 50, height: 50), ring: UIColor =
    //// General Declarations
    // This non-generic function dramatically improves compilation times of complex expressions
    func fastFloor(_ x: CGFloat) -> CGFloat { return floor(x) }
```

Name	Oval
Visibility	Always Visible
Opacity	
Blending	Normal
Start Angle	324
End Angle	90
Closed	✓
X	13
Y	13
W	24
H	24

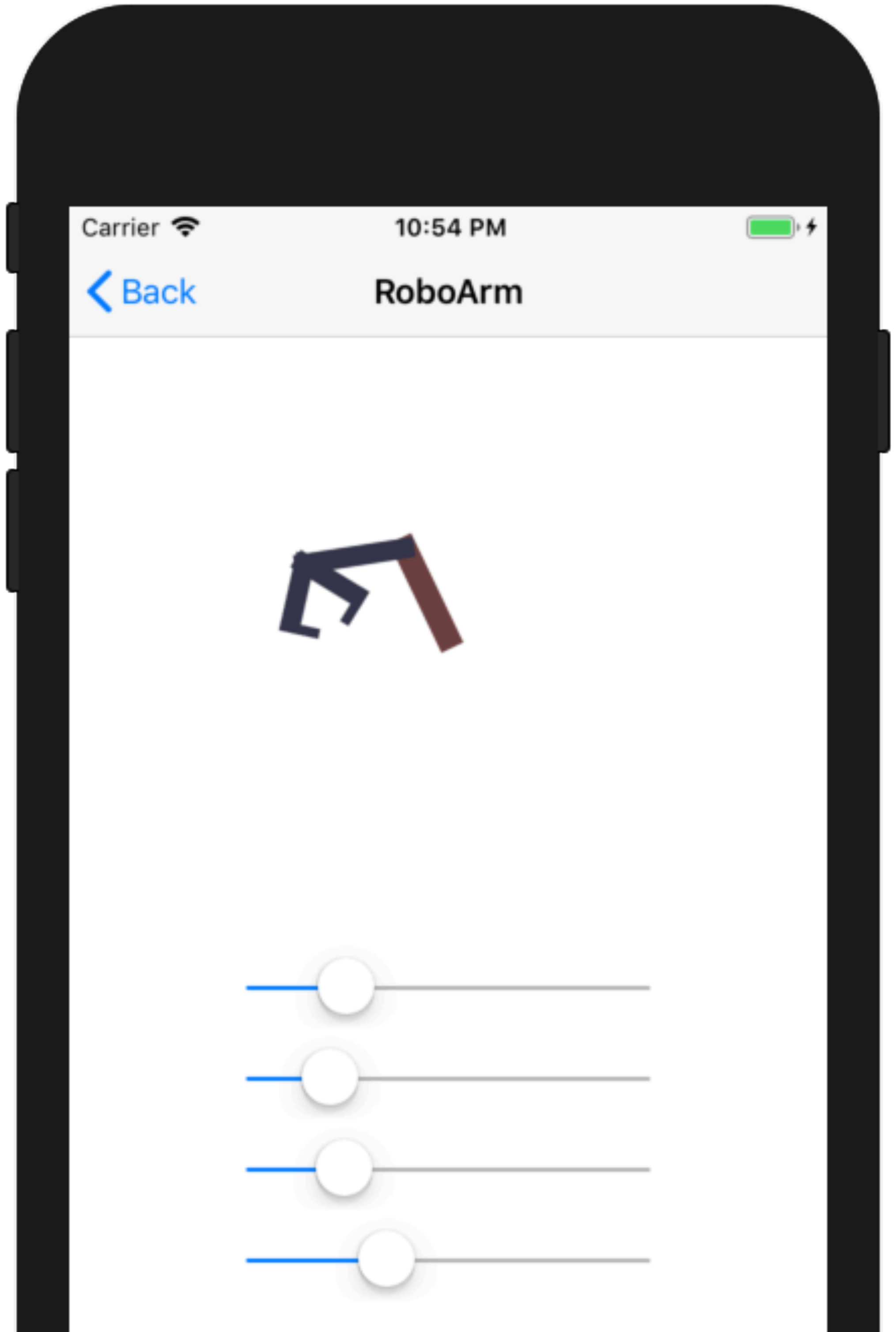
Solving our problems

- We can have one Canvas in PaintCode for our clock
  - and modify it using parameters to be any version we want
  - „one asset“ -> endless options
  - far better than gazillion images



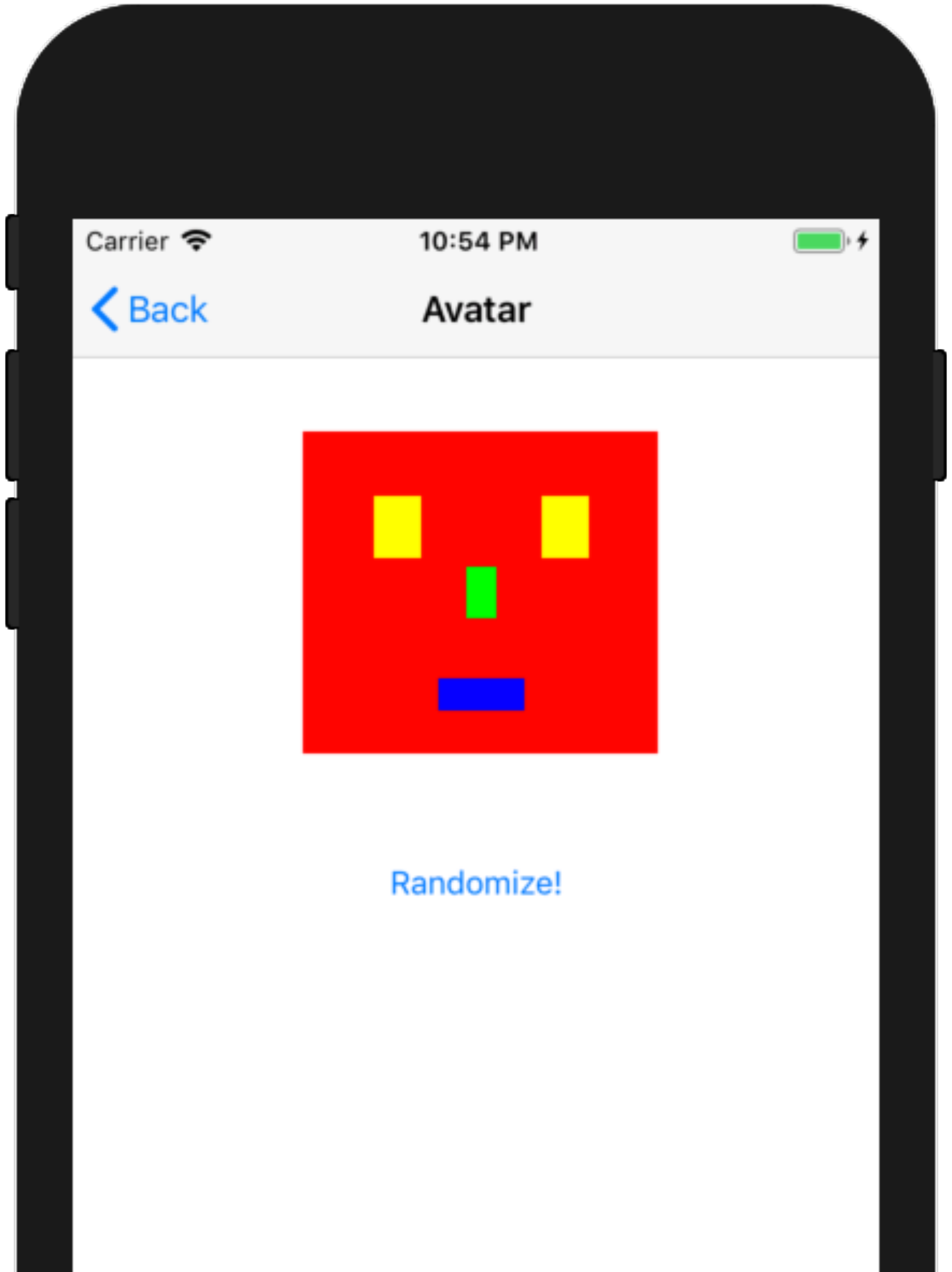
Solving our problems

- Small animations can be solved with parameterized drawing code
  - (rapidly redrawn)
  - no need for a large video file



The sky is not limit

- PaintCode can do a lot
  - without writing one line of CoreGraphics code yourself



DEMO
TIME!

Issues with PaintCode

- One binary file on disk
  - tricky for teams, rule: only edit it on one branch!
- $100/$200 per user
  - but RoI is fast

Issues with PaintCode

- Image generation or drawing happens every time
  - thats bad for e.g. images in UITableViewCells
  - may need a caching strategy
- Only suitable for constructed images/animations
  - actual photos and videos need to remain as JPG or MP4

On the plus side

- Easy to learn, with some affinity to vector graphics
    - even for designers
    - developers and designers can work on the same file
- Usually the source file is part of the repo and therefor safe

On the plus side

- Dynamic graphics
- One „asset“ for every screen resolution
- Reduction of app size
- Reduction of compile time
- Easy to generate small animations
- It’s been around for at about 6 years (or more)

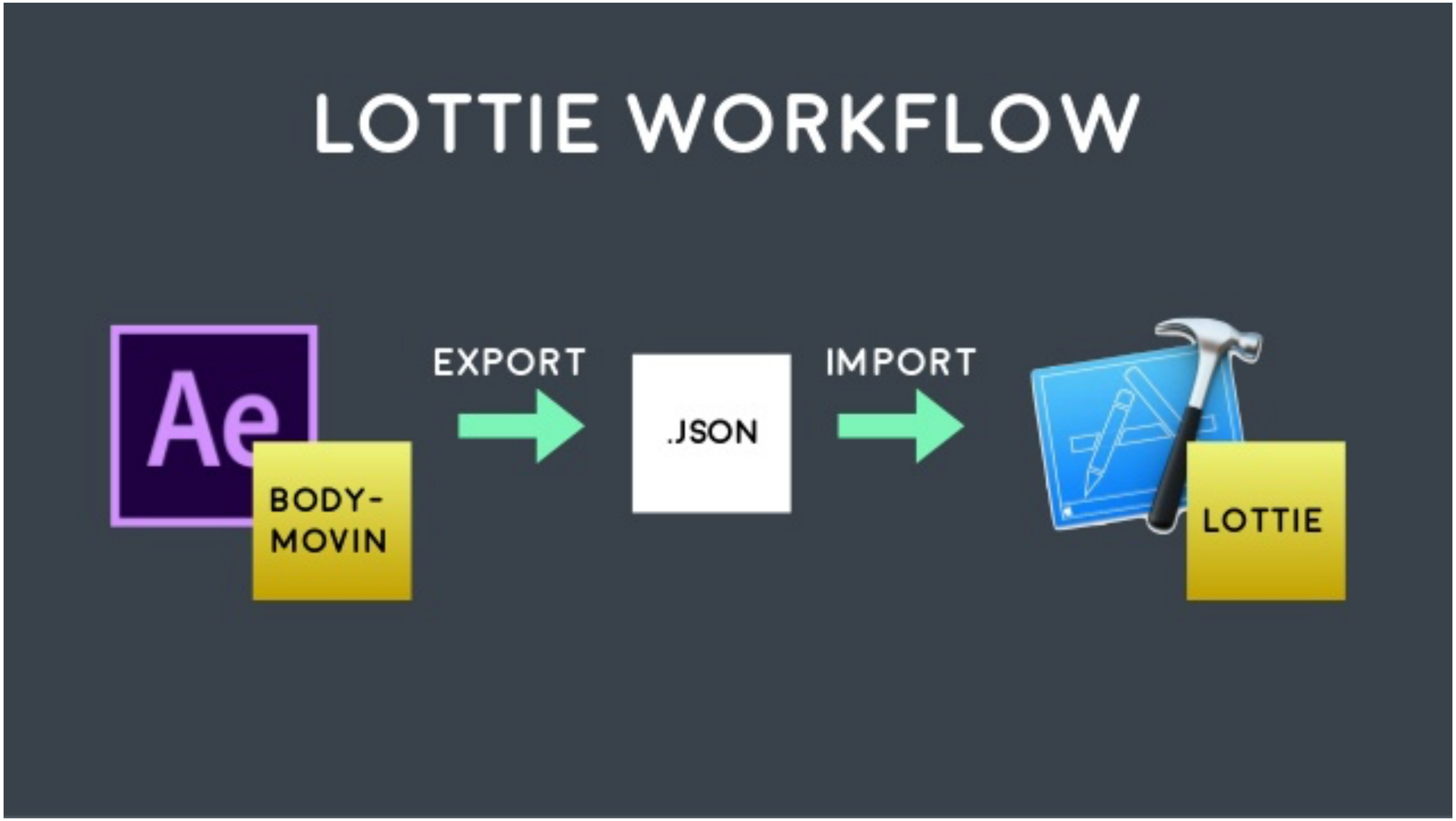
Outcome of PaintCode

- AssetCatalog reduced to 7mb on disk
- Compile time <20sec

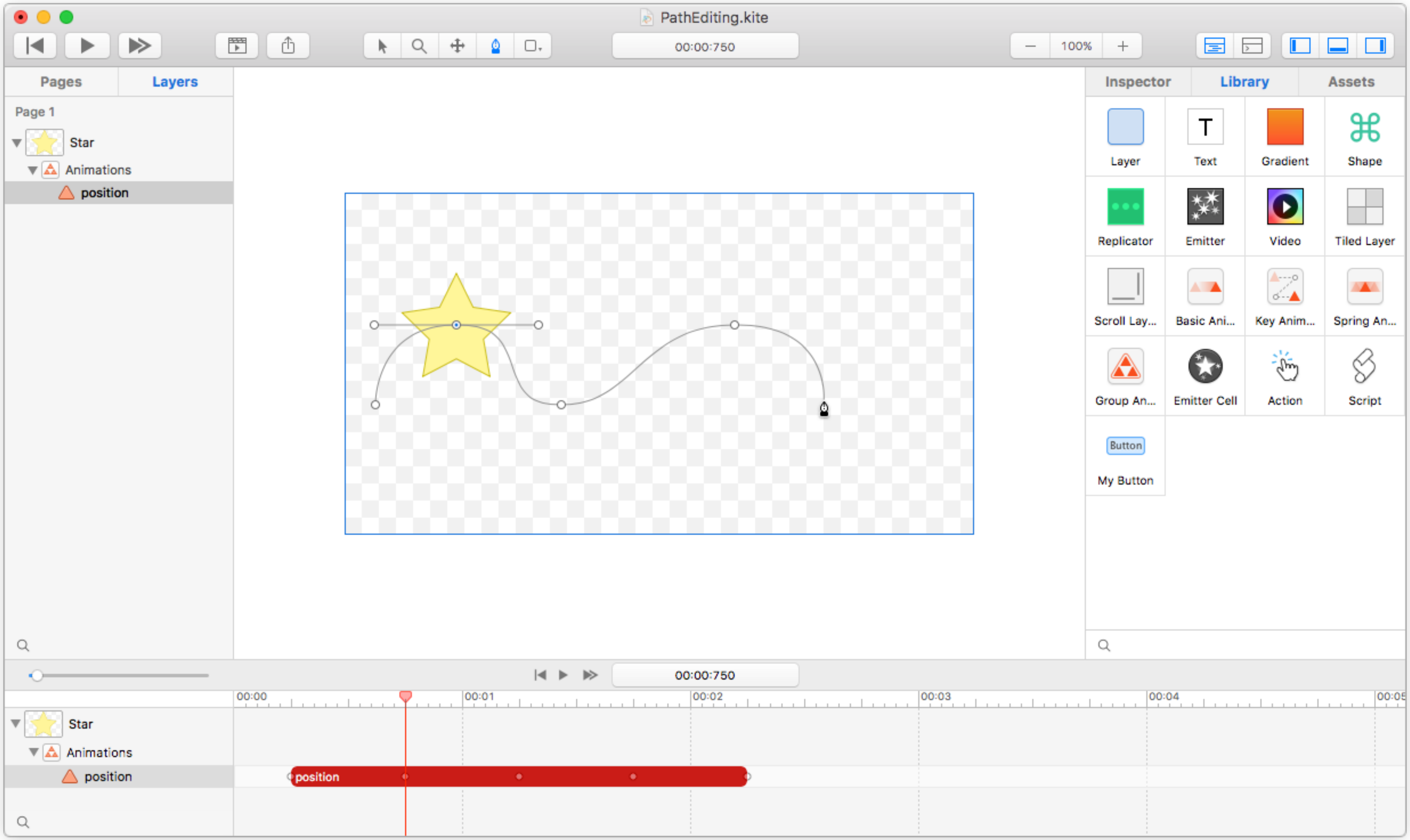
Word of Warning

Don’t replace everything
in your AssetCatalog with drawing code
just because you can!

Lottie by AirBnb



Kite Compositor



That’s it. Thanks!

Questions?