Bringing Apps to Life with Drawing, Animation, and Gestures

Andrew Bancroft

@andrewcbancroft www.andrewcbancroft.com

11:21

Home

Front Desk
Due on Aug 1, 2021
5 items
60% complete
On budget

Main Entrance
Due on May 30, 2021
2 items
0% complete
On budget

1st Floor Hallway
Due on Jul 31, 2021
4 items
25% complete
On budget

2nd Floor Hallway
Due on Jul 31, 2021
4 items
0% complete
On budget

1st Floor Guest Rooms
Due on Nov 1, 2021
7 items
29% complete
Over budget

2nd Floor Guest Rooms
Due on Nov 1, 2021
7 items
0% complete
On budget

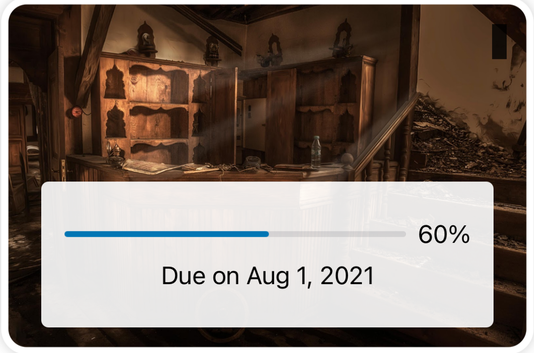
Restaurant

11:22

Home

Edit

Front Desk



Work Quality

★ ★ ★ ☆

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [*] Paint walls
- [ ] Hang new artwork

Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

11:22

Cancel

Done

PROJECT INFO

Front Desk	
Due Date	Aug 1, 2021
Work Quality Rating	Good
Flagged for review	

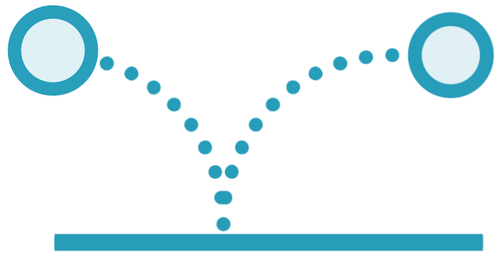
PUNCH LIST

Remodel front desk	Complete
Retile entry	Complete
Replace light fixtures	Complete
Paint walls	In Progress
Hang new artwork	Not Started

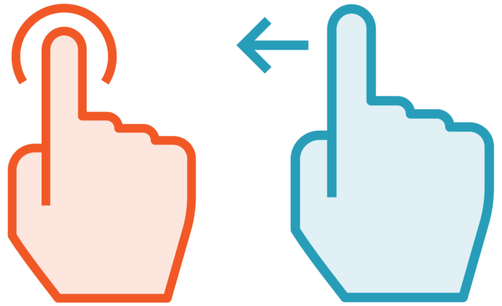
BUDGET

Spent To-Date	18350

Drawing



Animation



Gestures

Drawing

Animation

Gestures

ProgressView

11:46

Home Edit

Front Desk

Work Quality

★ ★ ★ ☆ View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [*] Paint walls
- [ ] Hang new artwork

Budget

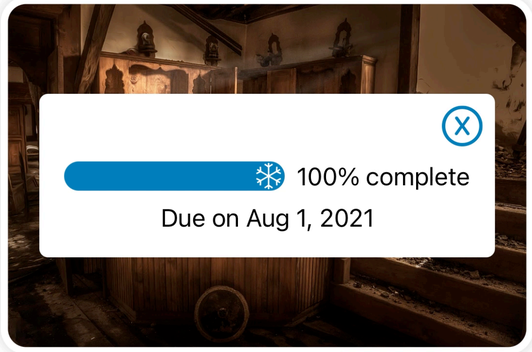
On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

11:51

Home Edit

Front Desk



100% complete

Due on Aug 1, 2021

Work Quality

★ ★ ★ ☆ View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [x] Paint walls
- [x] Hang new artwork

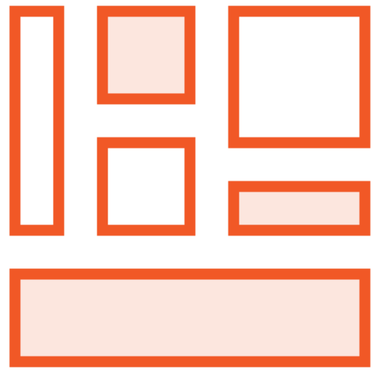
Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

Think about drawing like SwiftUI “thinks” about drawing.

Drawing System



Layout System

iOS App Development Learning Path Resources

iOS Getting Started — Building Single View Applications

SwiftUI Layout Process

SwiftUI Layout Process

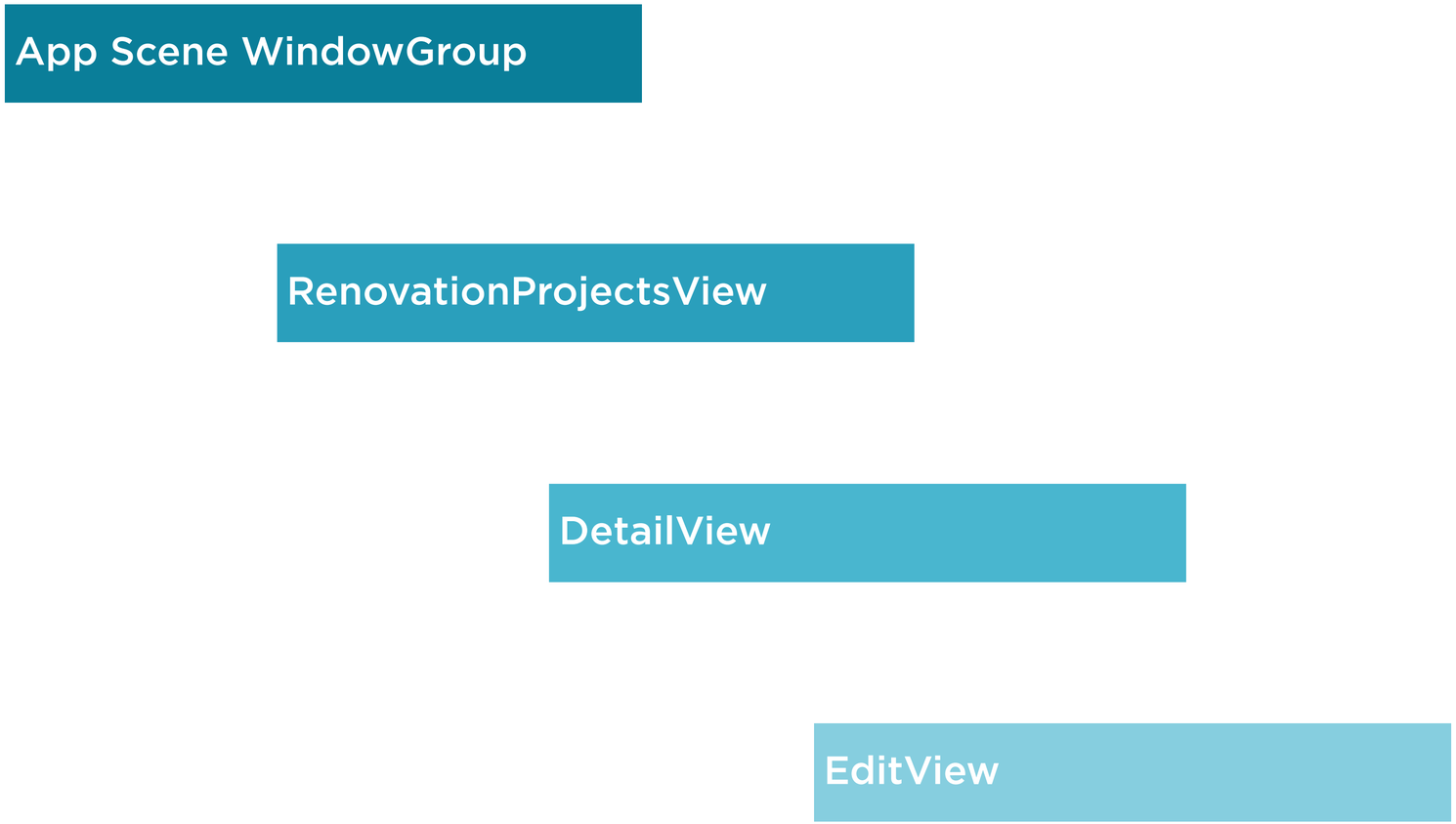
1

How much space a particular View takes up

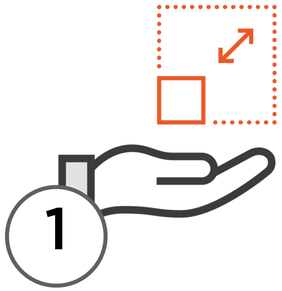
2

Where a particular View gets positioned onto an iOS screen

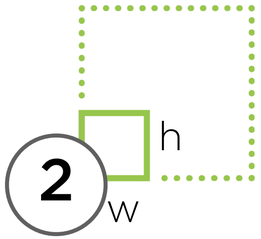
3


App Scene WindowGroup
RenovationProjectsView
DetailView
EditView

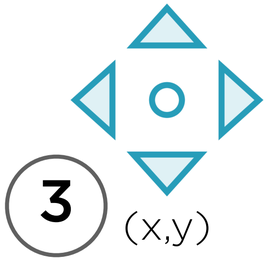
SwiftUI Layout Process



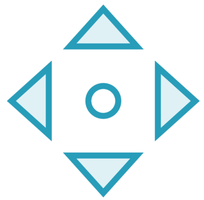
A **parent** View offers a child View some space



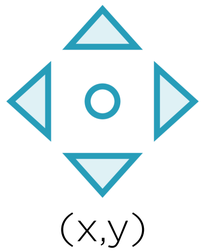
The **child** considers the offer and chooses a size for itself that is up to and including the total amount offered to it, then reports back to the layout system



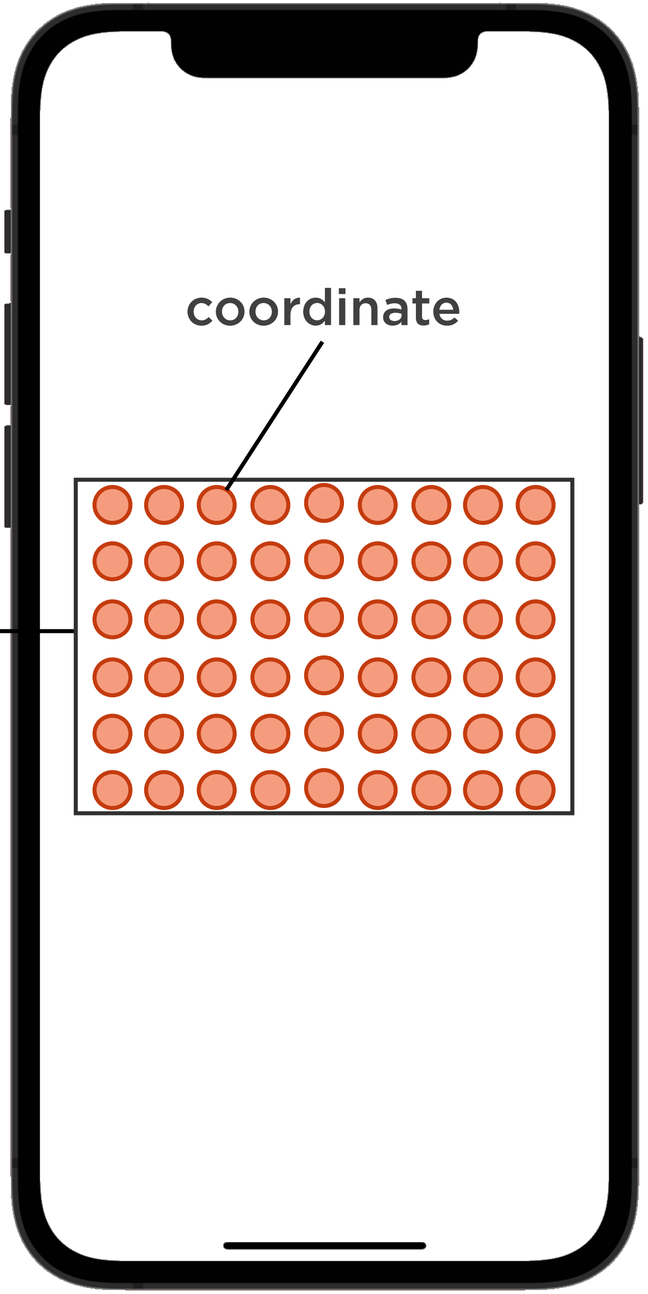
The **parent** View uses the height and width that its child chose, and positions that child View directly in the center of its own available coordinate space



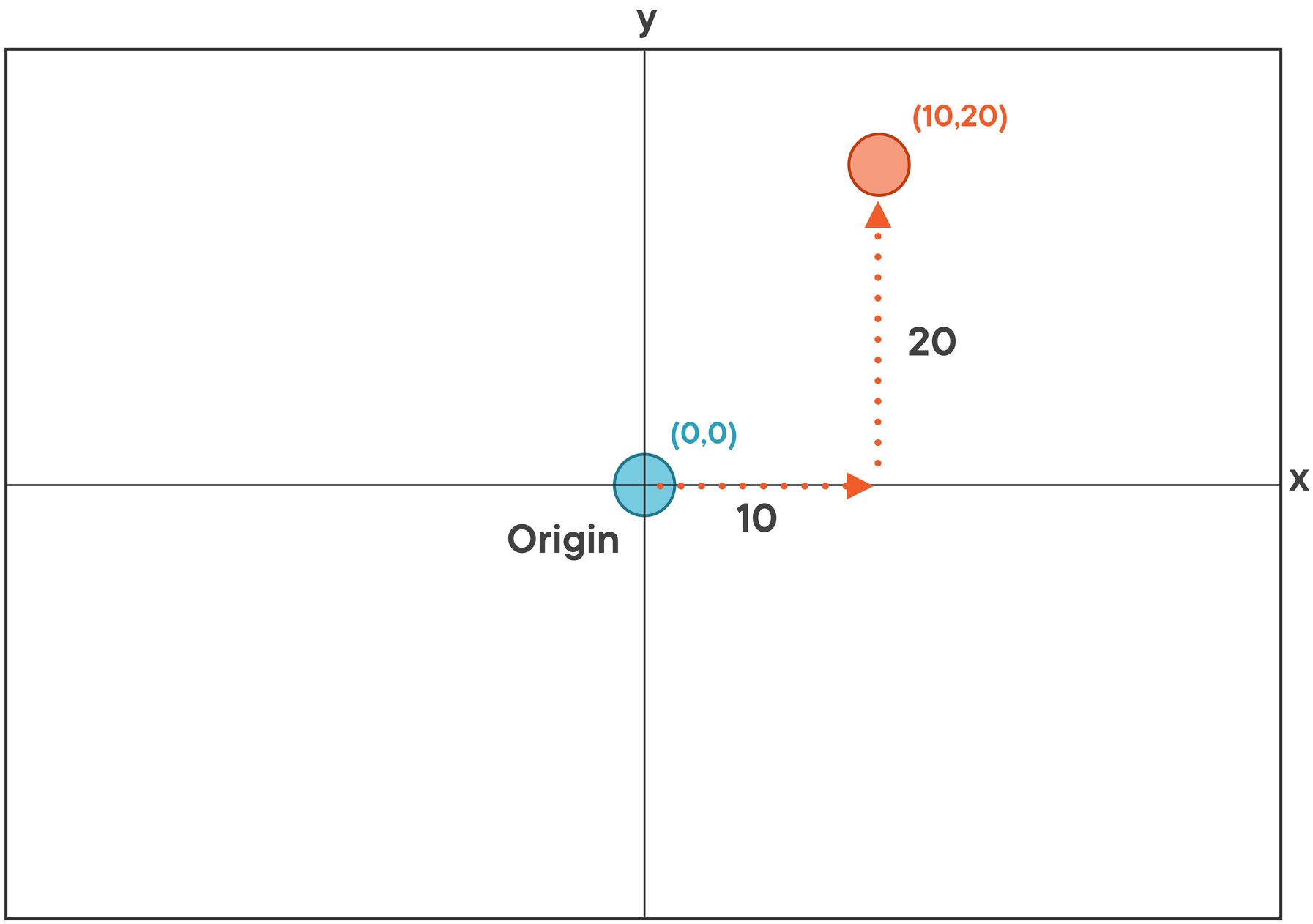
coordinate space



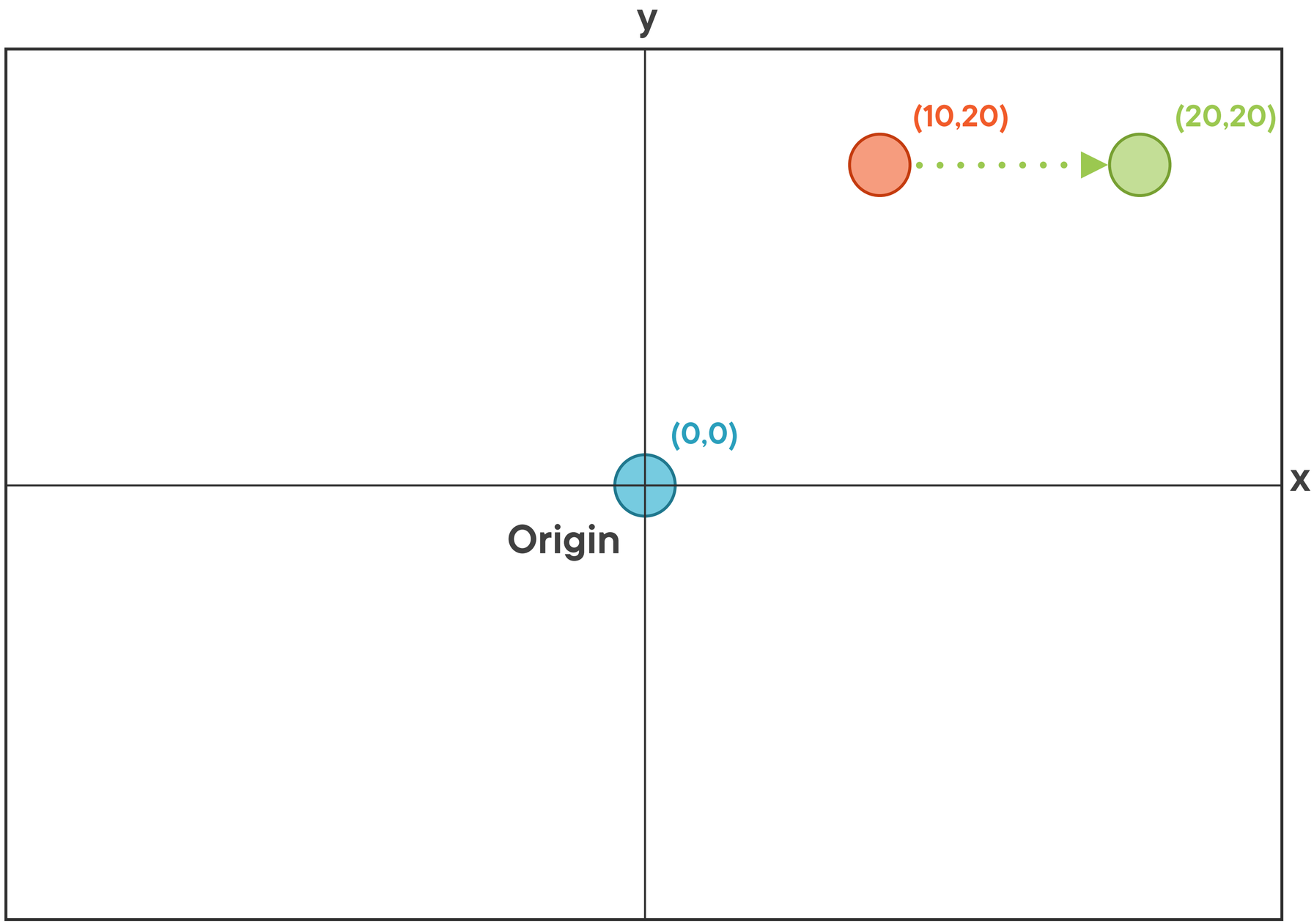
coordinate space



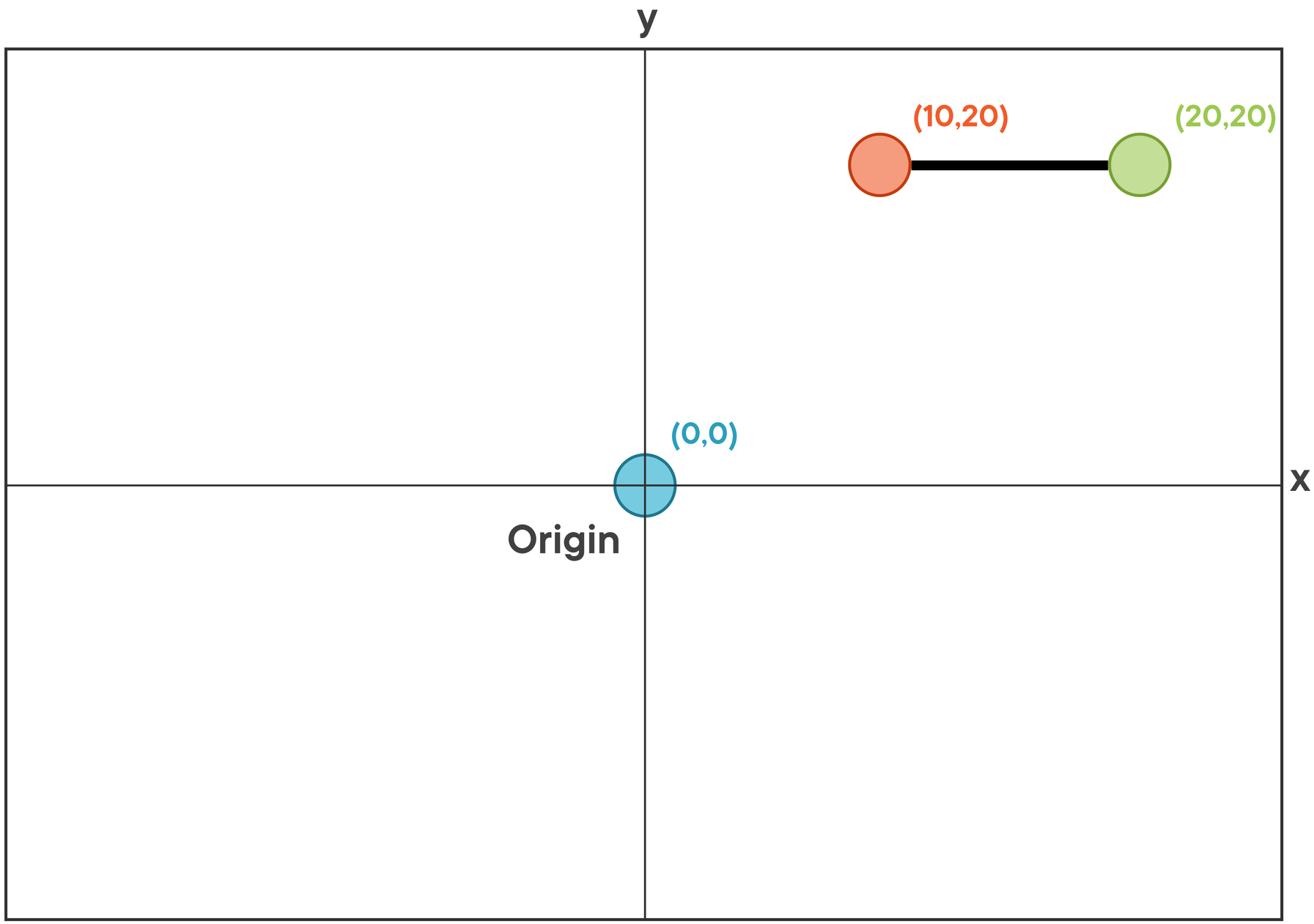
Cartesian Coordinate System

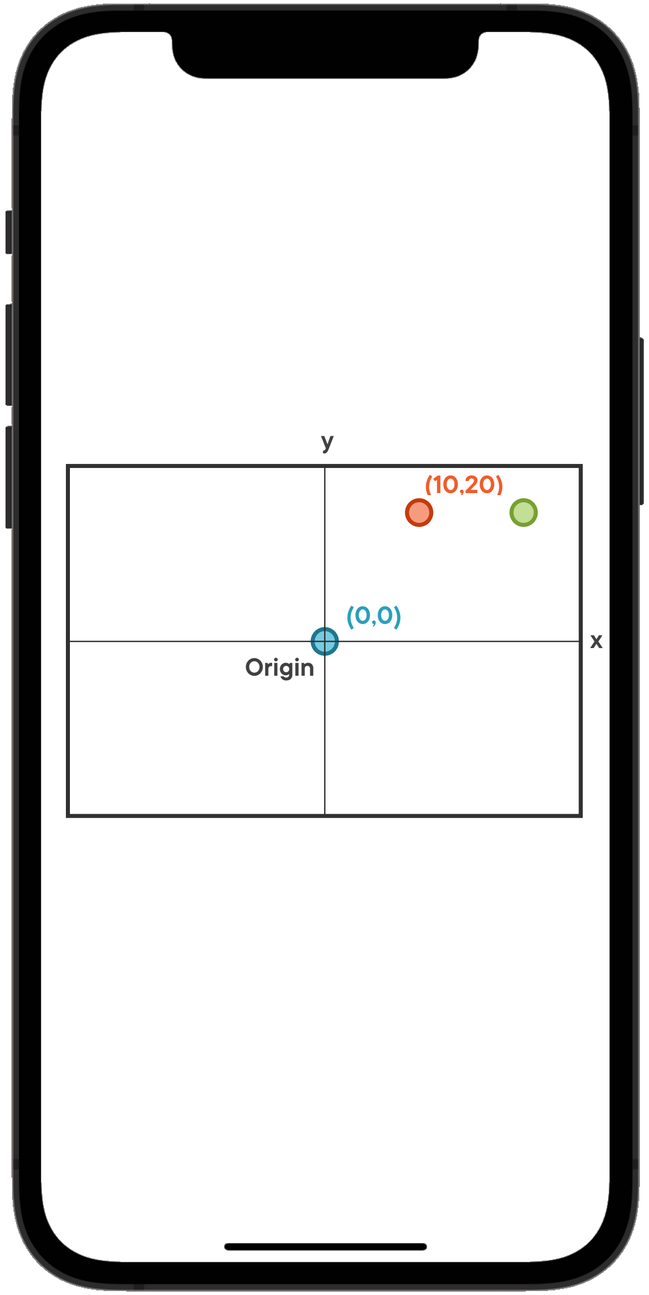


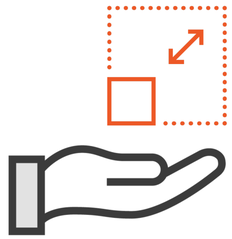
Cartesian Coordinate System



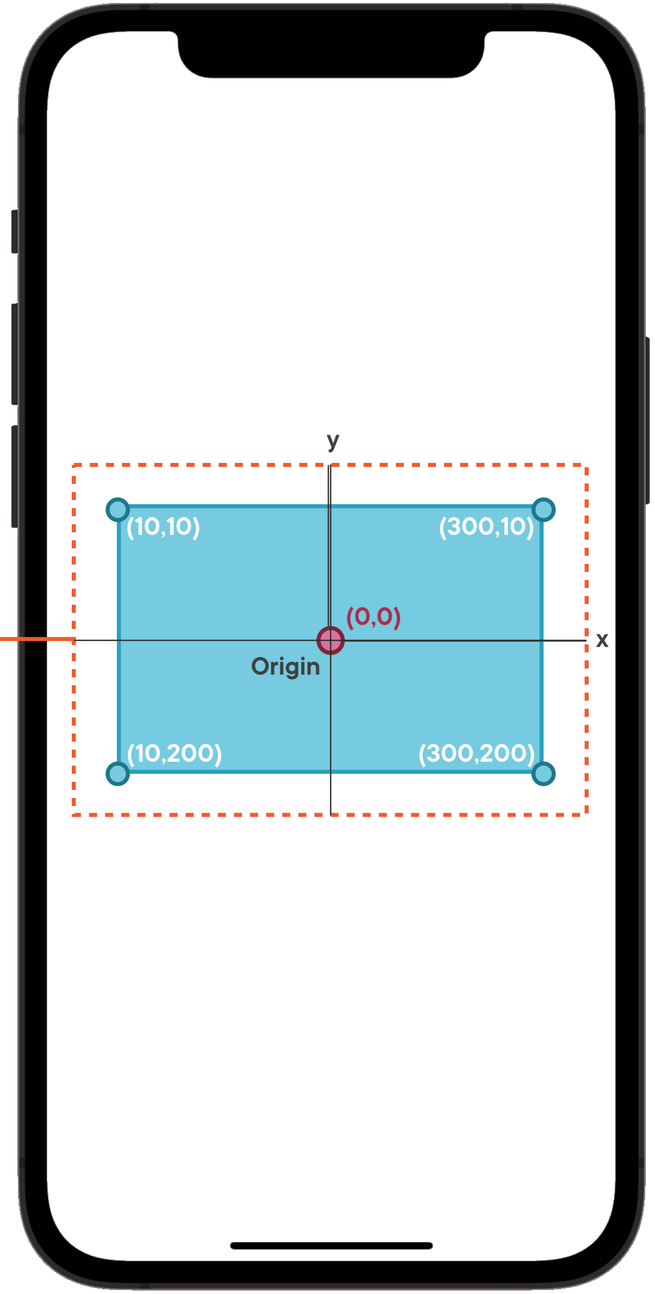
Cartesian Coordinate System

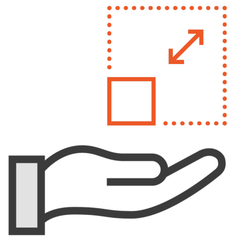



y
(10,20)
(0,0)
Origin
x

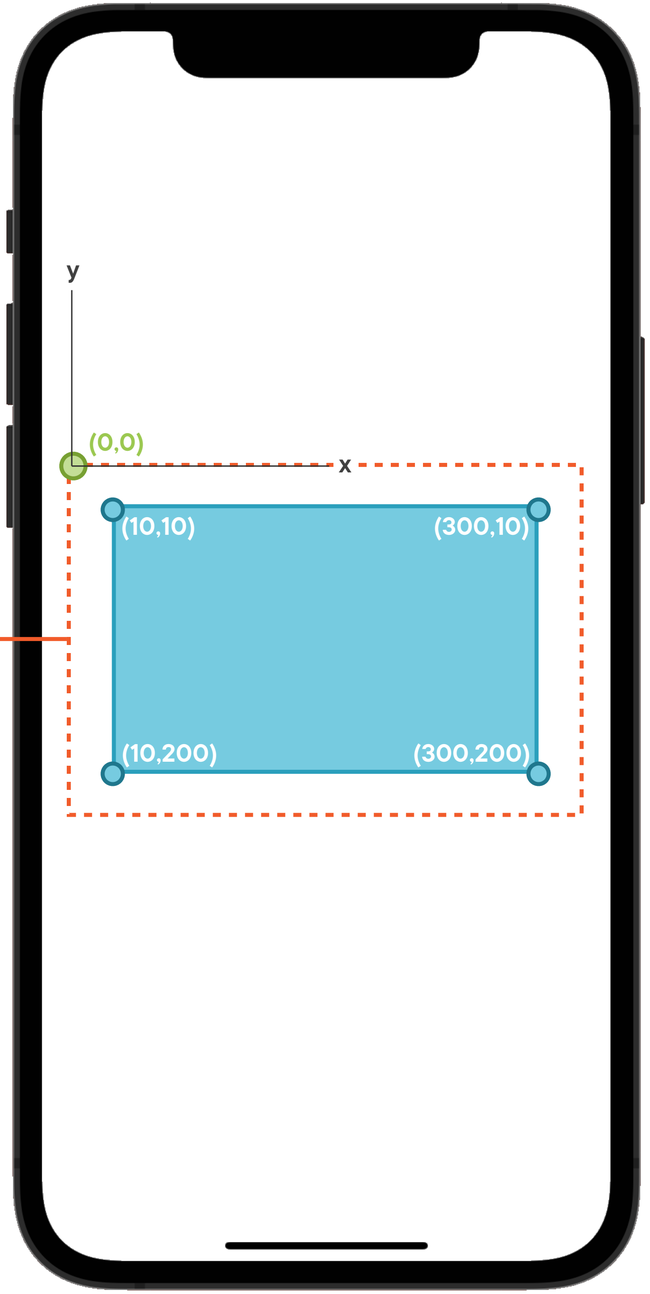


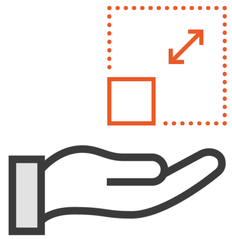
Offered space



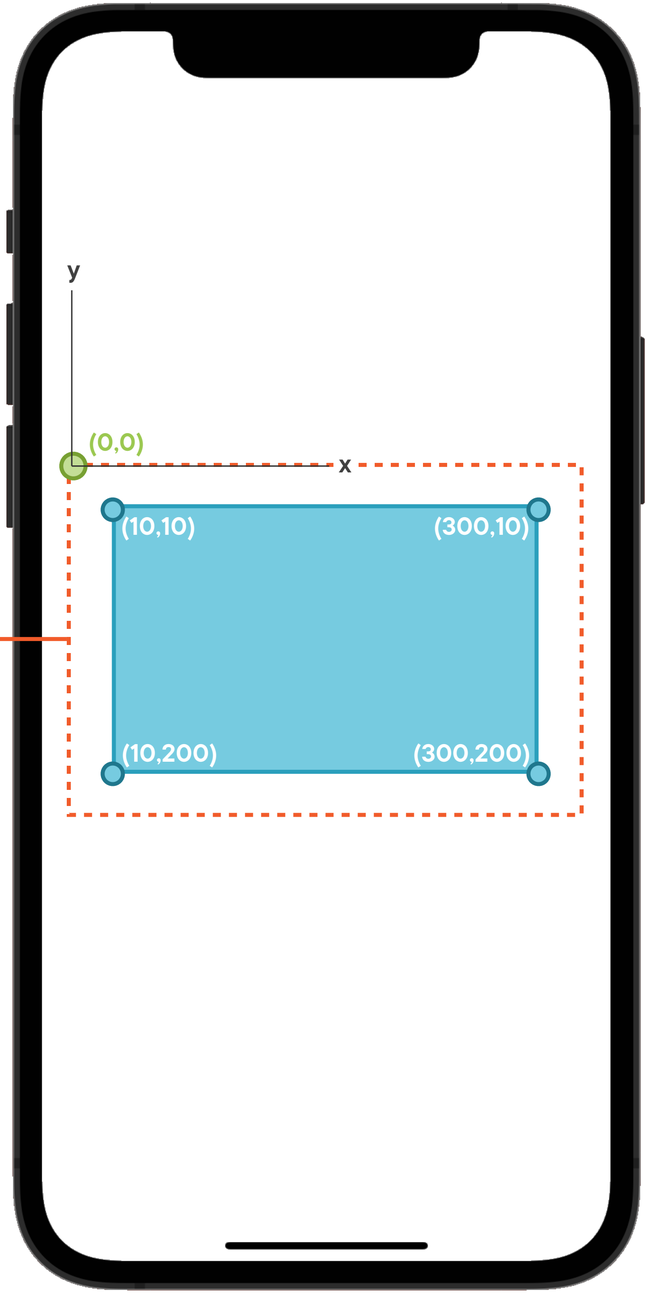


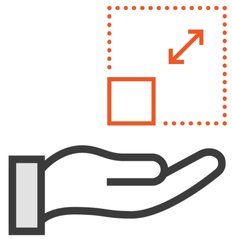
Offered space



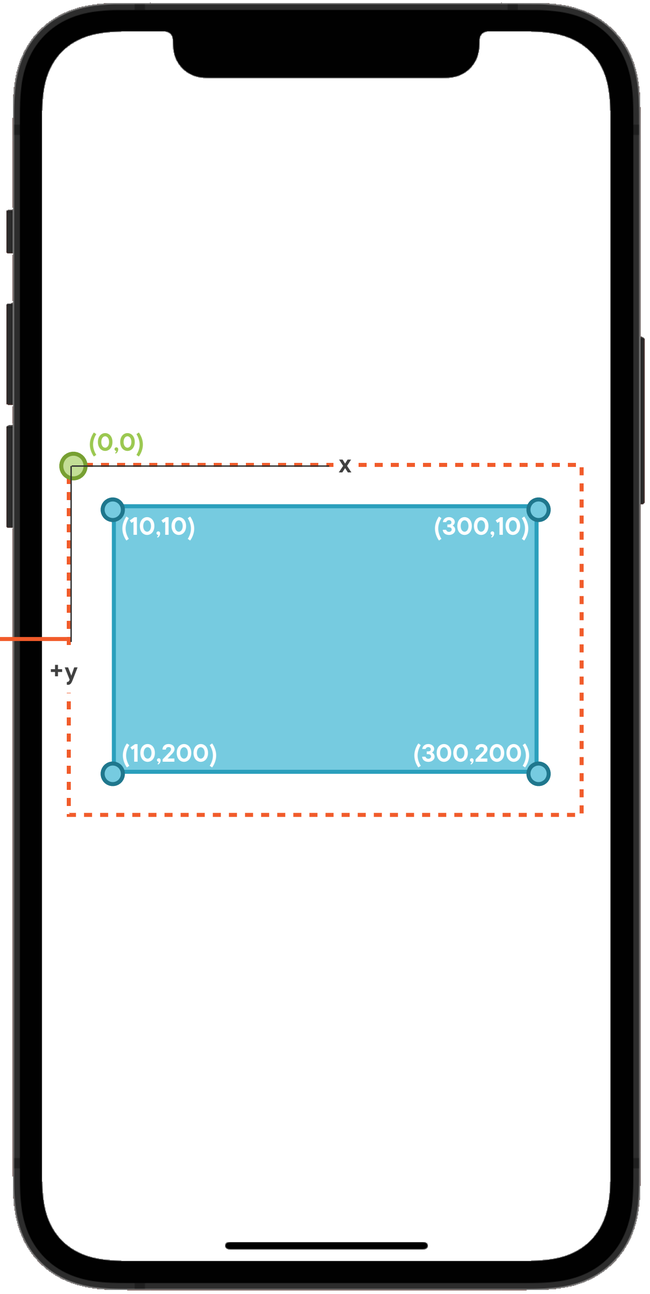


Offered space





Offered space



Move to the coordinate (10,10)

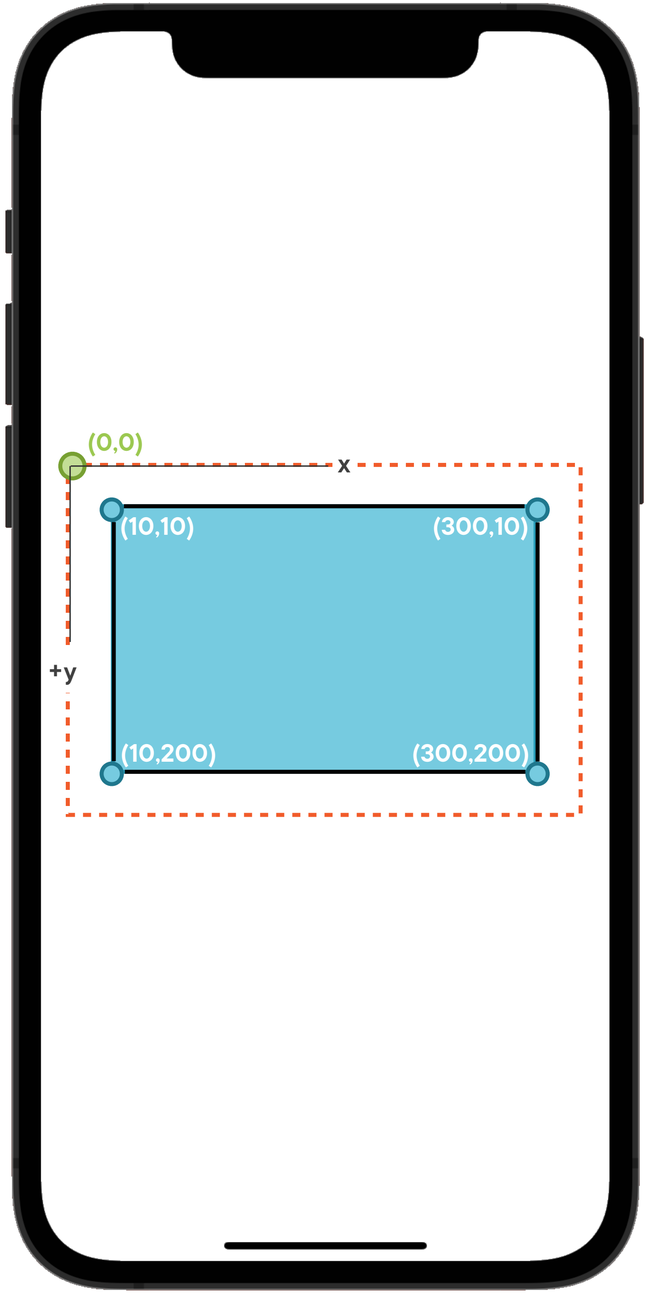
Draw a line from (10,10) to (300,10)

Draw a line from (300,10) to (300,200)

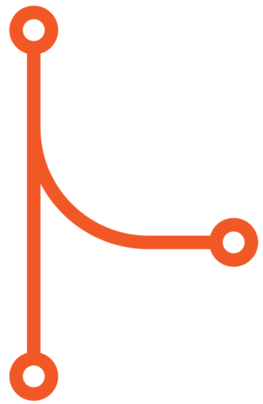
Draw a line from (300,200) to (10,200)

Draw a line from (10,200) to (10,10)

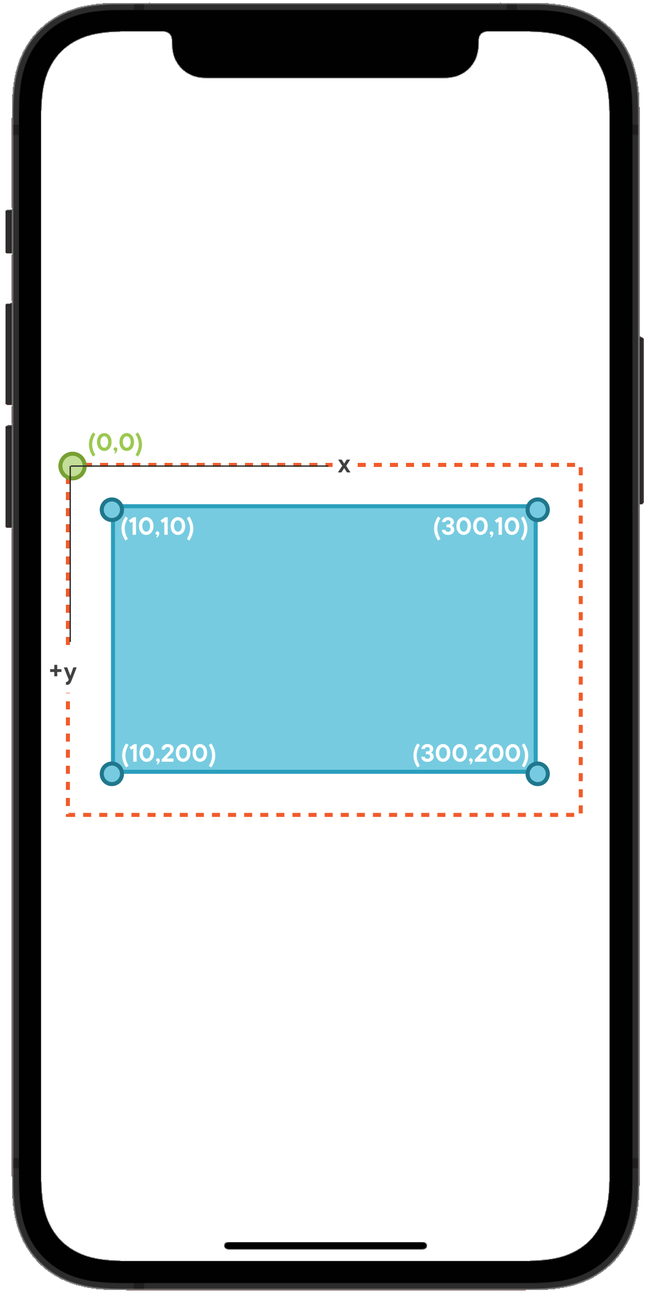
Fill the whole area with the color blue



Drawing with Paths



Path


(0,0)
x
+y
(10,10)
(300,10)
(10,200)
(300,200)

Move to the coordinate (10,10)

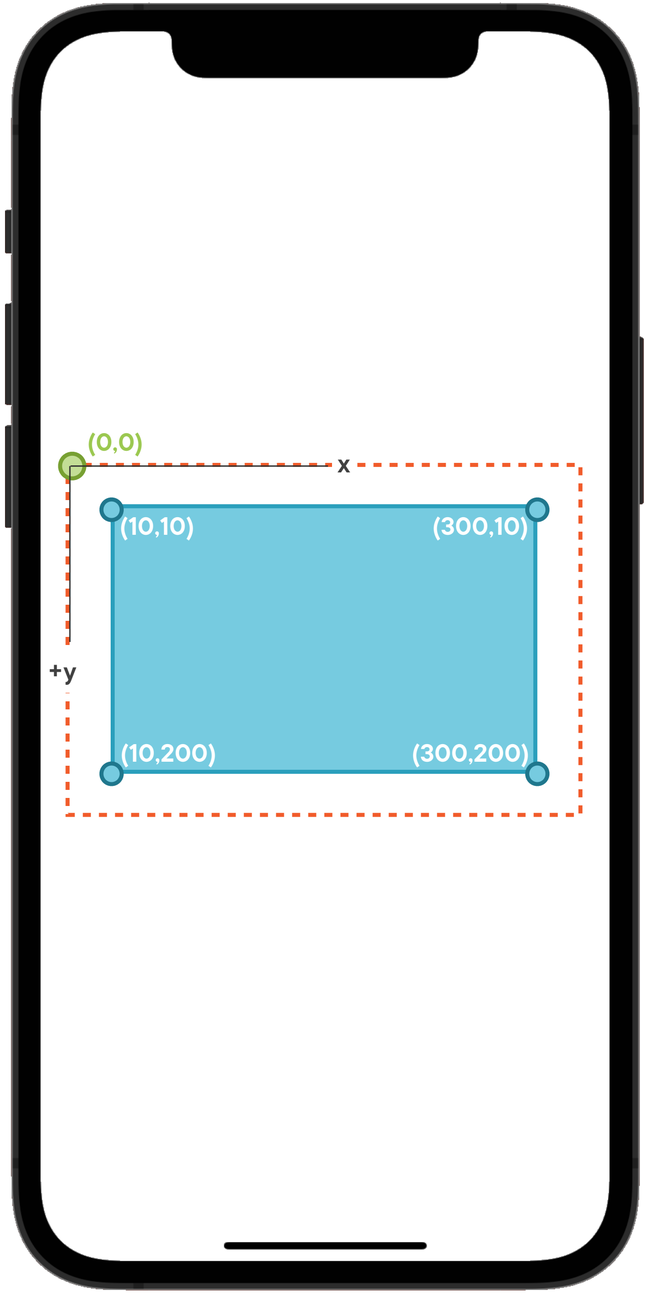
Draw a line from (10,10) to (300,10)

Draw a line from (300,10) to (300,200)

Draw a line from (300,200) to (10,200)

Draw a line from (10,200) to (10,10)

Fill the whole area with the color blue



RenoTracker 〉 iPhone 12

RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 6:06 PM

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

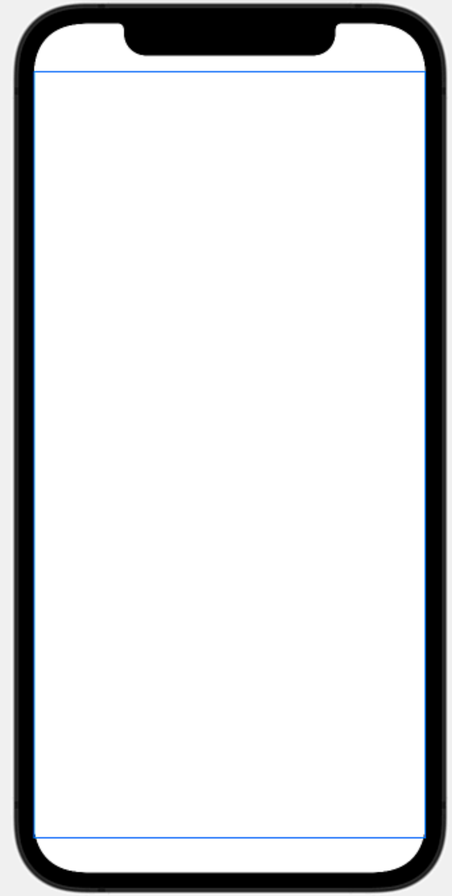
- RenoTracker M
  - RenoTracker
    - RenoTrackerApp.swift
    - RenovationProjectsView.swift
    - RenovationProjectRow.swift
    - DetailView.swift
    - EditView.swift
    - InspectionLogView.swift
    - Snowflake.swift A
    - Models
    - Assets.xcassets
    - Info.plist
    - Preview Content
  - Products

```swift
import SwiftUI

struct Snowflake: View {
    var body: some View {
        Path { mainPath in
            mainPath.move(to: CGPoint(x: 10, y: 10))
        }
    }
}

struct Snowflake_Previews: PreviewProvider {
    static var previews: some View {
        Snowflake()
    }
}
```

Preview



Path 390×763

75%


CGPoint

Core Graphics

Core Graphics

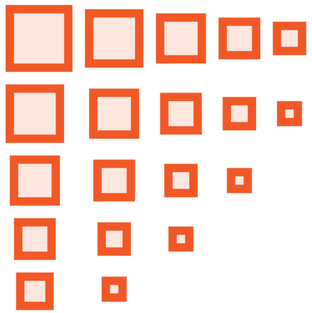
Contains types and functions that enable you and I to perform 2-dimensional graphics rendering.

Can we use Path to draw a *snowflake*?

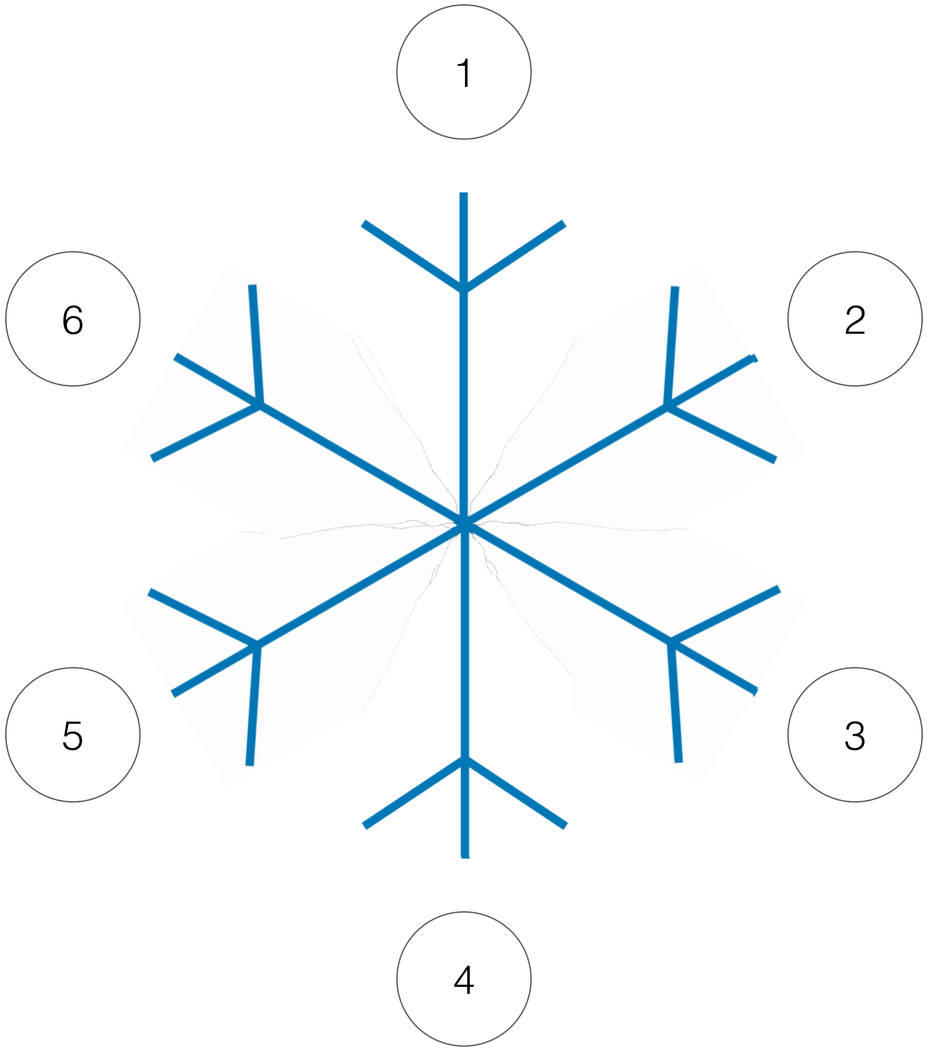
Paths can be used to draw any two-dimensional shape that your imagination can conjure.

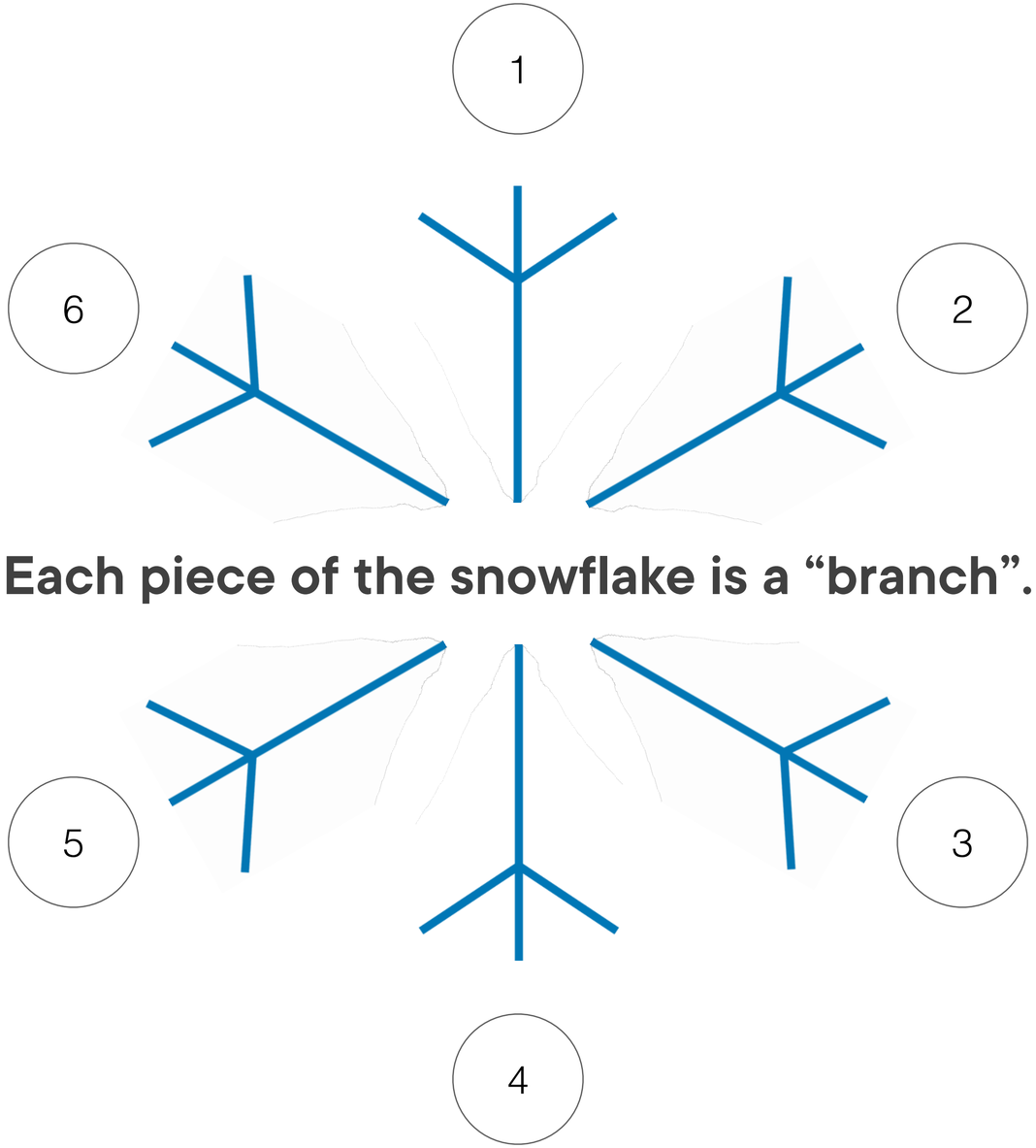
Which characteristics of a snowflake do you want to include in a drawing?

Which characteristics of a snowflake do you want to include in a drawing?

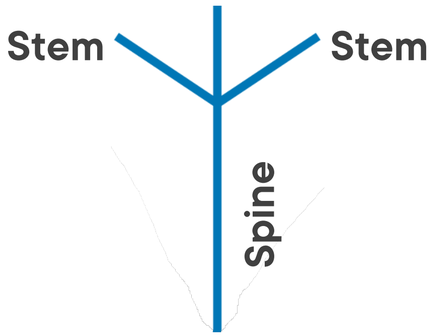


Break the drawing down into its component parts.

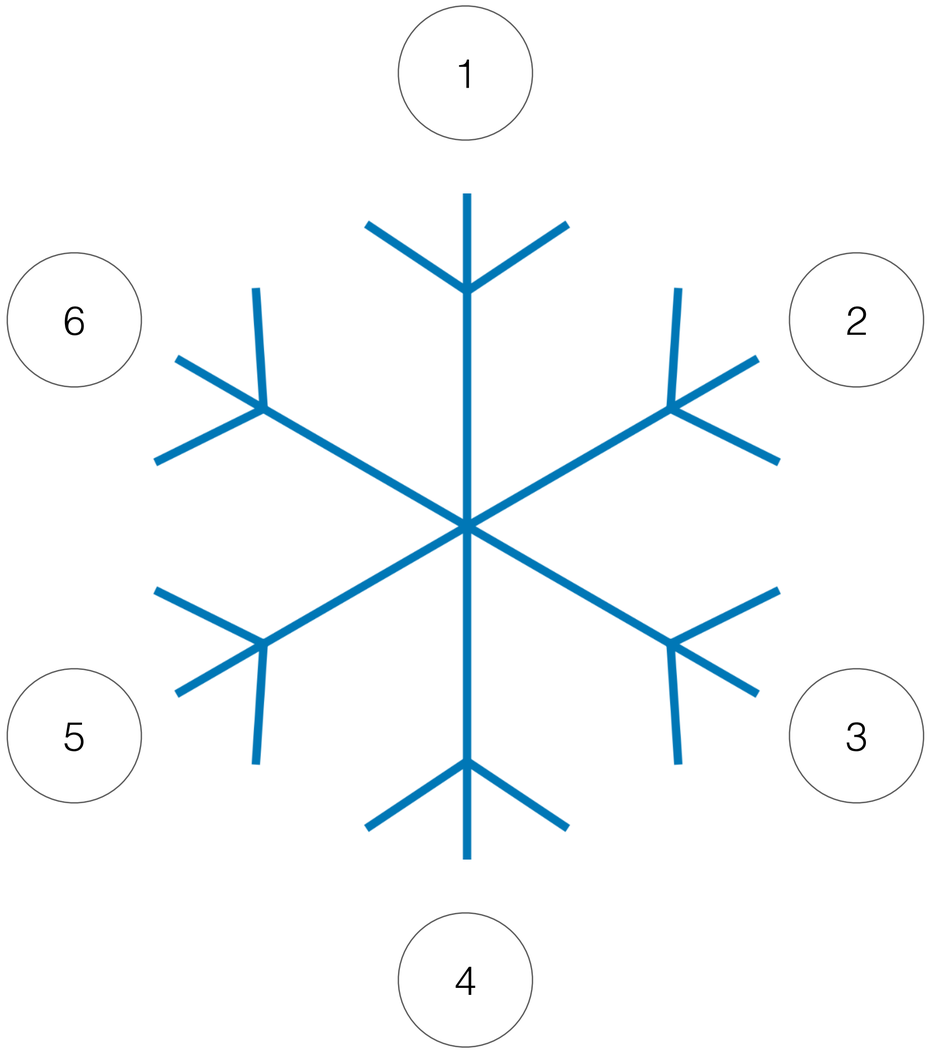

1
2
3
4
5
6

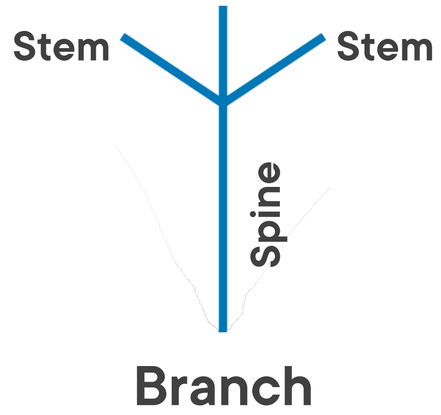


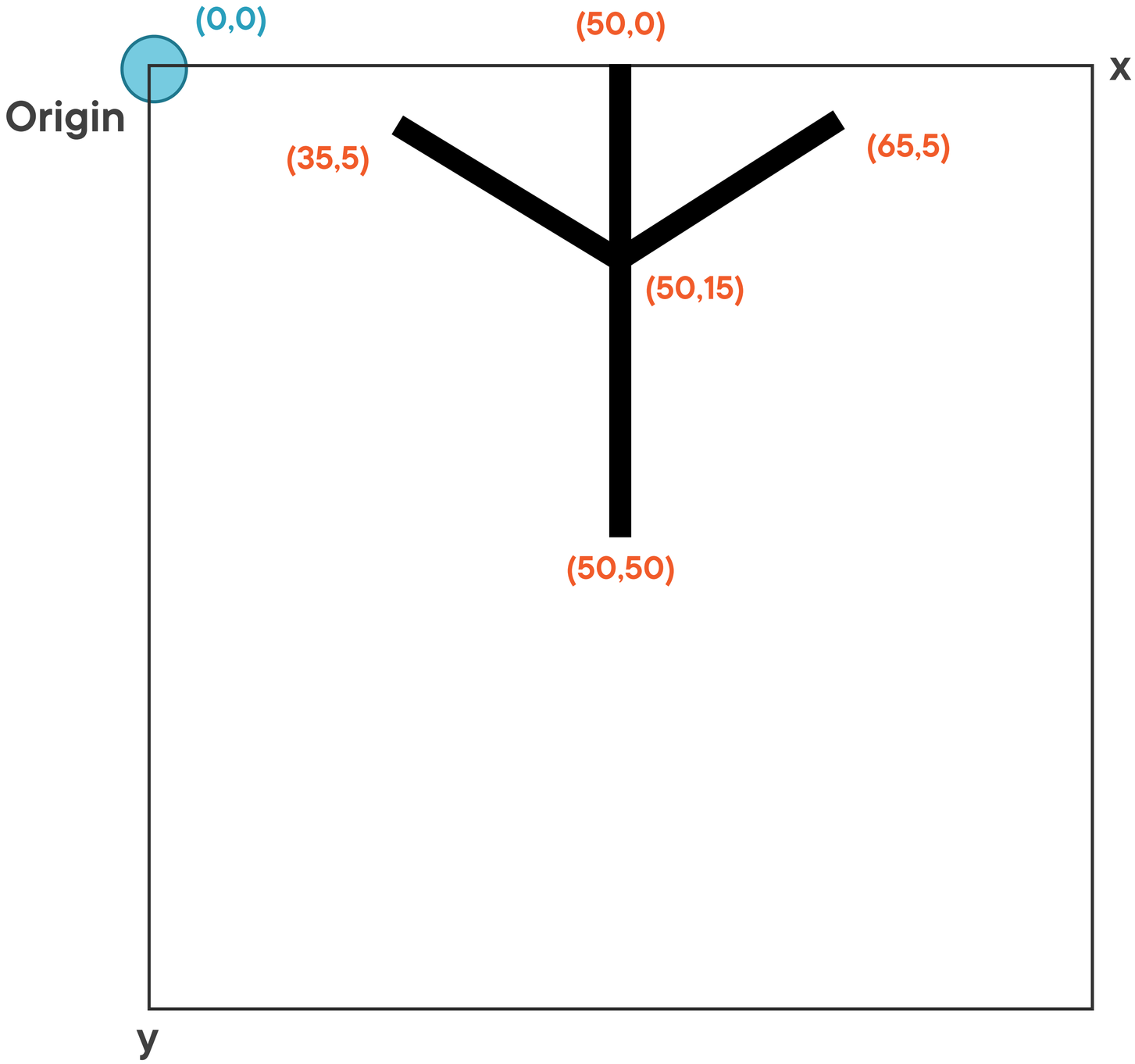
Each piece of the snowflake is a “branch”.



Branch


1
2
3
4
5
6


Stem
Stem
Spine
Branch


(0,0)
(50,0)
x
Origin
(35,5)
(65,5)
(50,15)
(50,50)
y

What if we do want a Path to be drawn in such a way that the points and line lengths are relative to the amount of space that was offered by the layout system?

RenoTracker 〉 iPhone 12

RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 6:06 PM

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

- RenoTracker M
  - RenoTracker
    - RenoTrackerApp.swift
    - RenovationProjectsView.swift
    - RenovationProjectRow.swift
    - DetailView.swift
    - EditView.swift
    - InspectionLogView.swift
    - Snowflake.swift A
    - Models
    - Assets.xcassets
    - Info.plist
    - Preview Content
  - Products

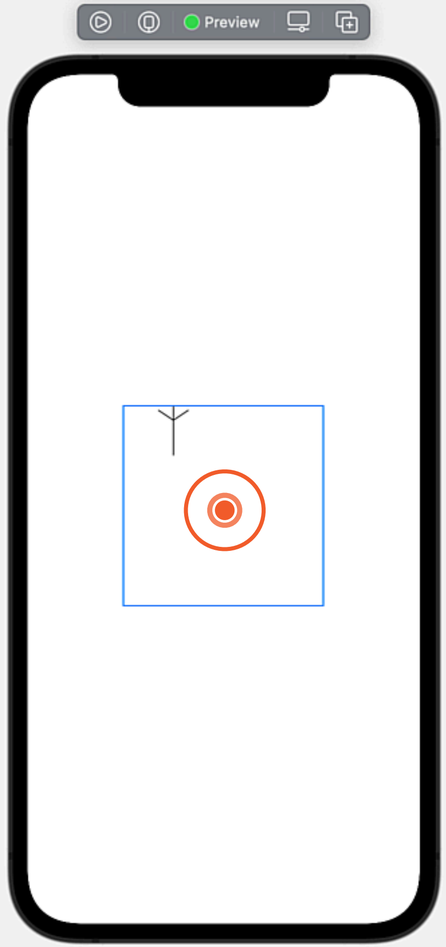
```swift

import SwiftUI

struct Snowflake: View {
    var body: some View {
        Path { mainPath in
            mainPath.move(to: CGPoint(x: 50, y: 0))
            mainPath.addLine(to: CGPoint(x: 50, y: 50))
            mainPath.move(to: CGPoint(x: 50, y: 15))
            mainPath.addLine(to: CGPoint(x: 35, y: 5))
            mainPath.move(to: CGPoint(x: 50, y: 15))
            mainPath.addLine(to: CGPoint(x: 65, y: 5))
        }
        .stroke(Color.black)
    }
}

struct Snowflake_Previews: PreviewProvider {
    static var previews: some View {
        Snowflake()
            .frame(width: 200, height: 200)
    }
}

```

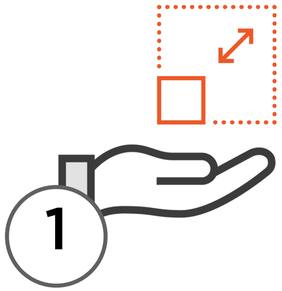


Path 200×200

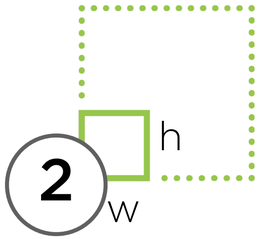
75%

Using GeometryReader

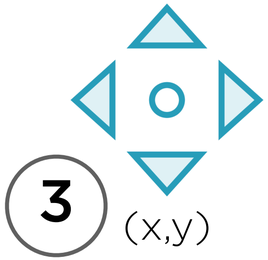
SwiftUI Layout Process



A **parent** View offers a child View some space



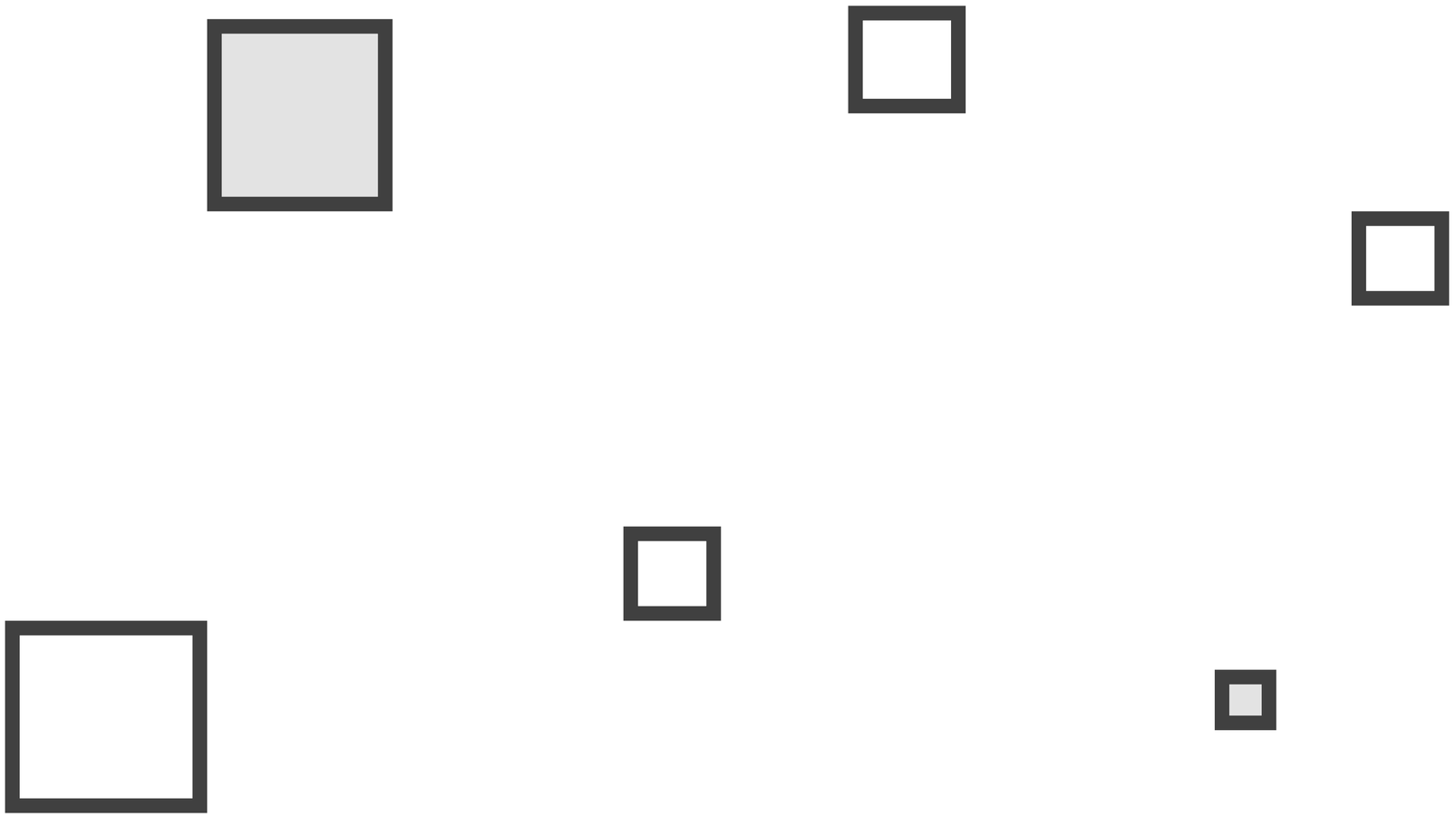
The **child** considers the offer and chooses a size for itself that is up to and including the total amount offered to it, then reports back to the layout system

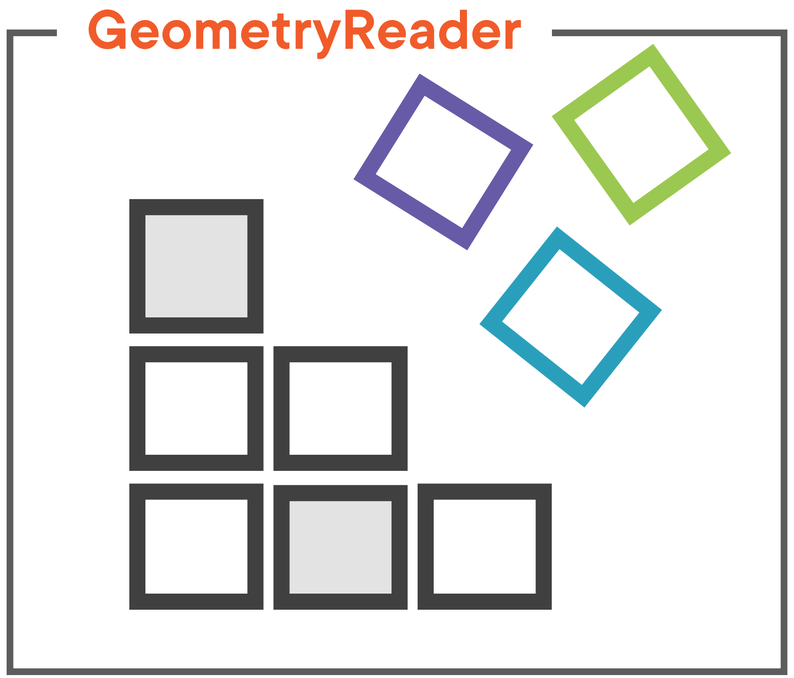


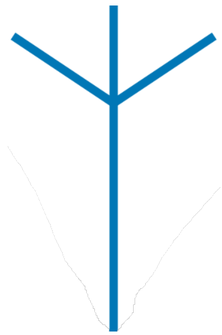
The **parent** View uses the height and width that its child chose, and positions that child View directly in the center of its own available coordinate space

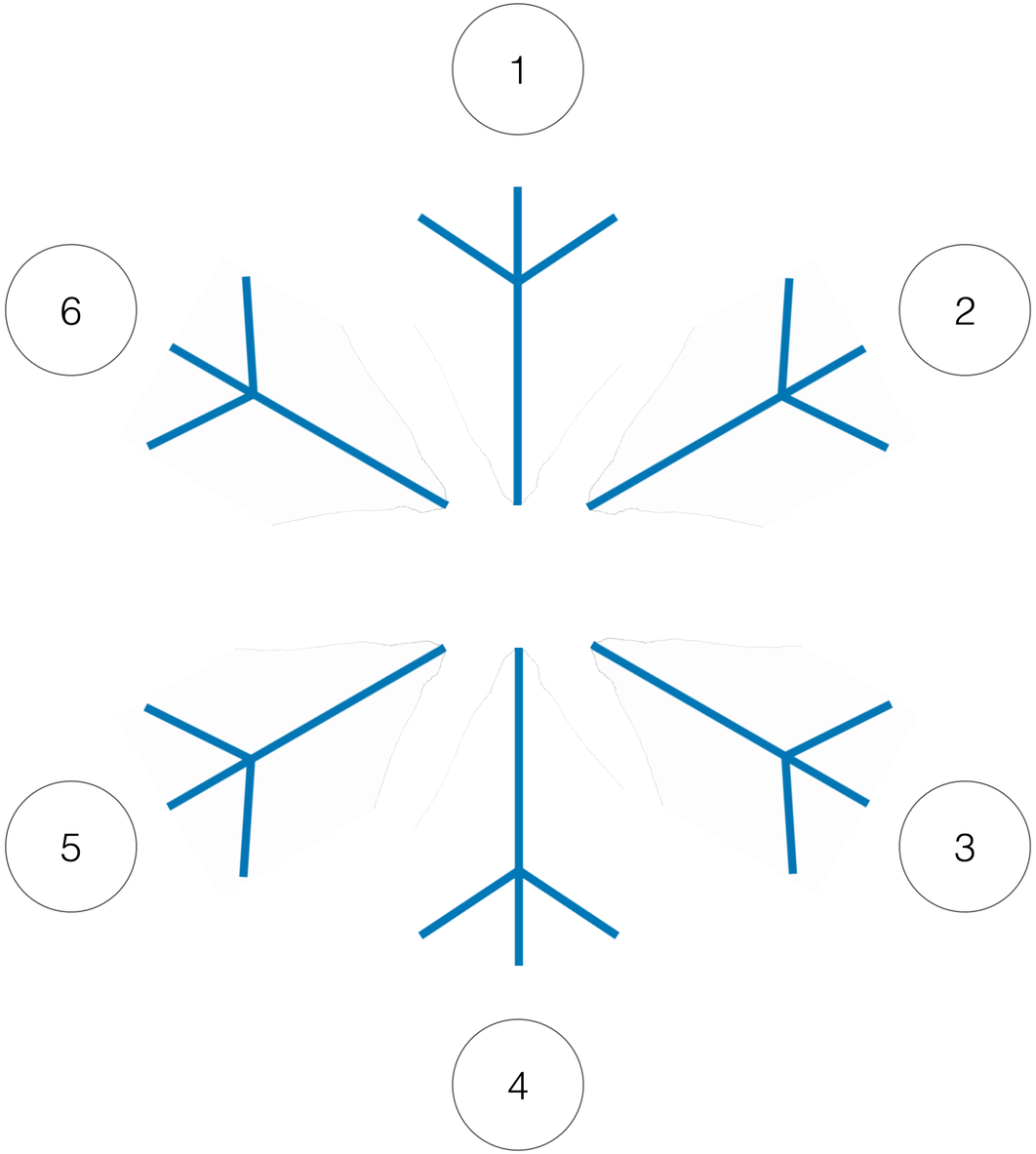
Inspect the dimensions of the space that's being offered at runtime...

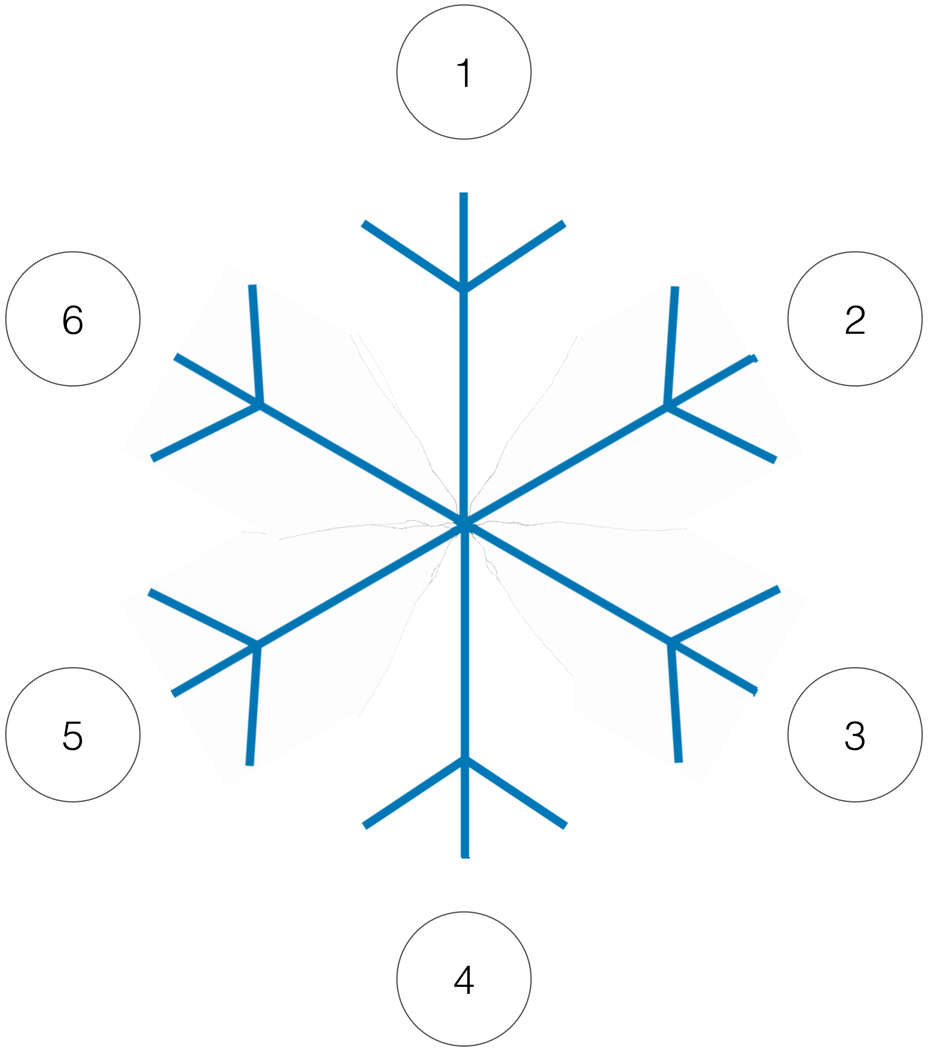
The View can dynamically determine its width, height, and position in a way that's directly related to the amount of space that's been offered.




GeometryReader

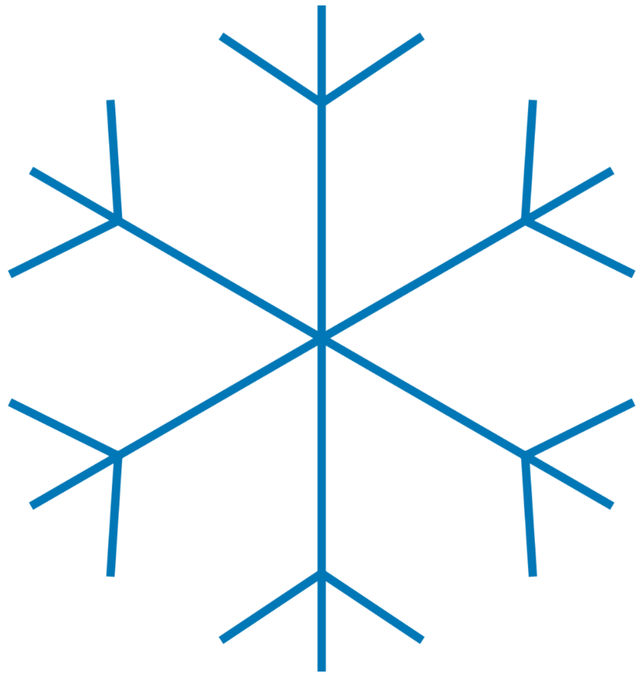


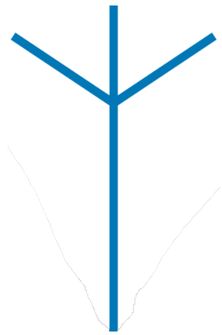

1
2
3
4
5
6

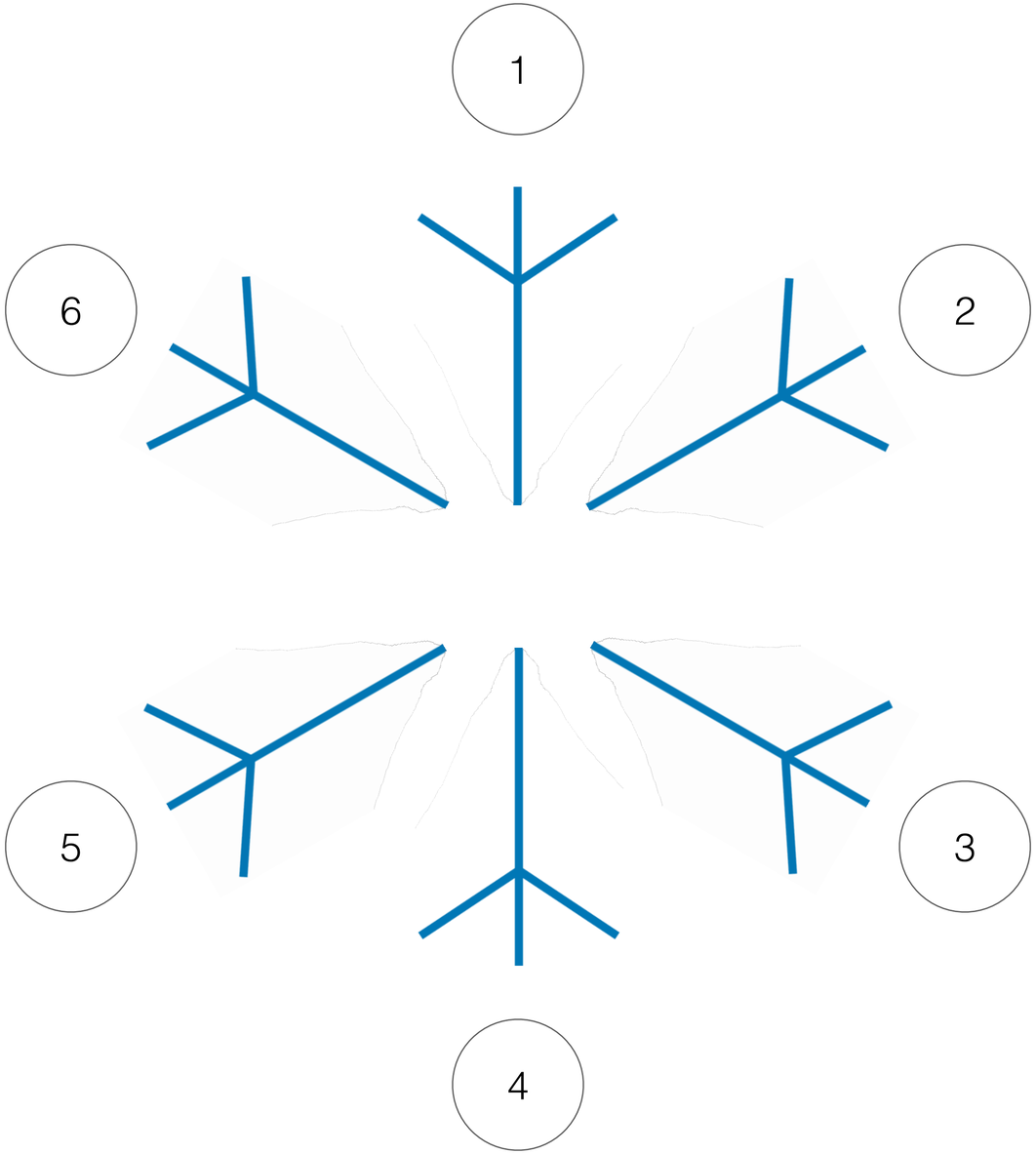

1
2
3
4
5
6

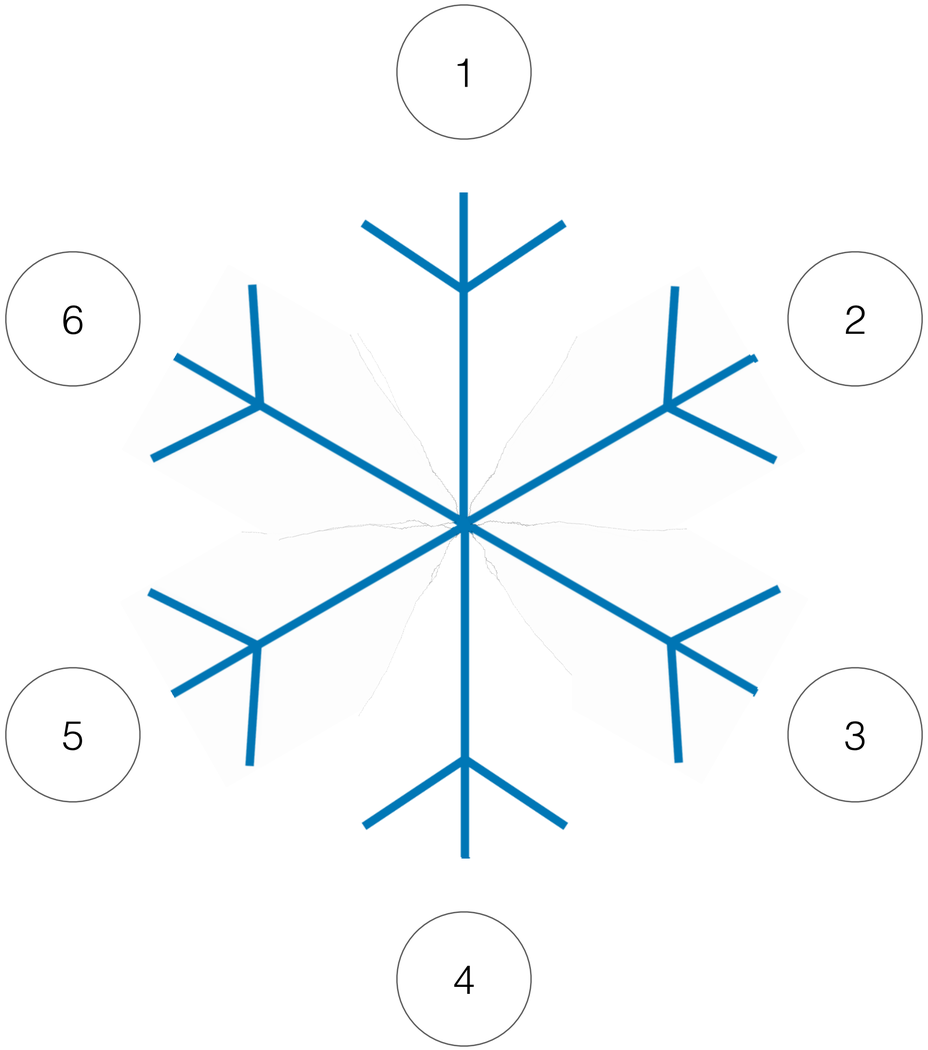
How do we apply transformations to Paths?

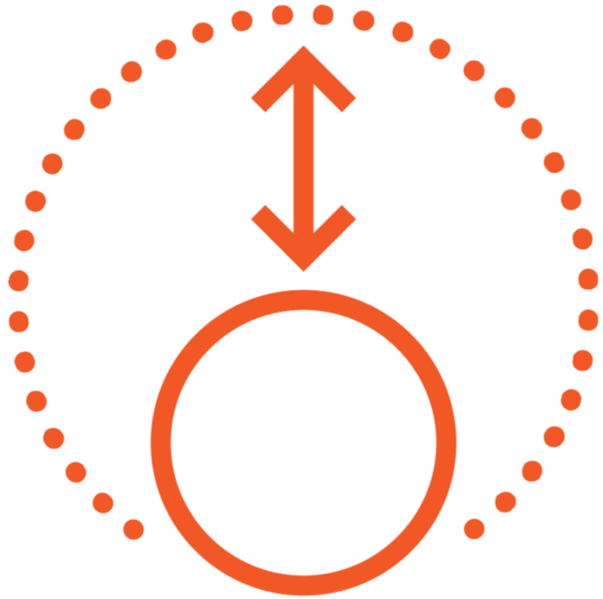
Applying Transformations to Paths






1
2
3
4
5
6


1
2
3
4
5
6

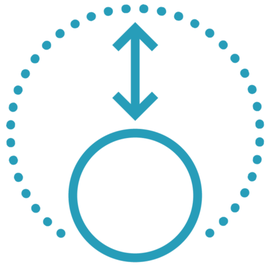


Applying Transformations to Paths

Transformations are represented by a type called CGAffineTransform.

CGAffineTransform

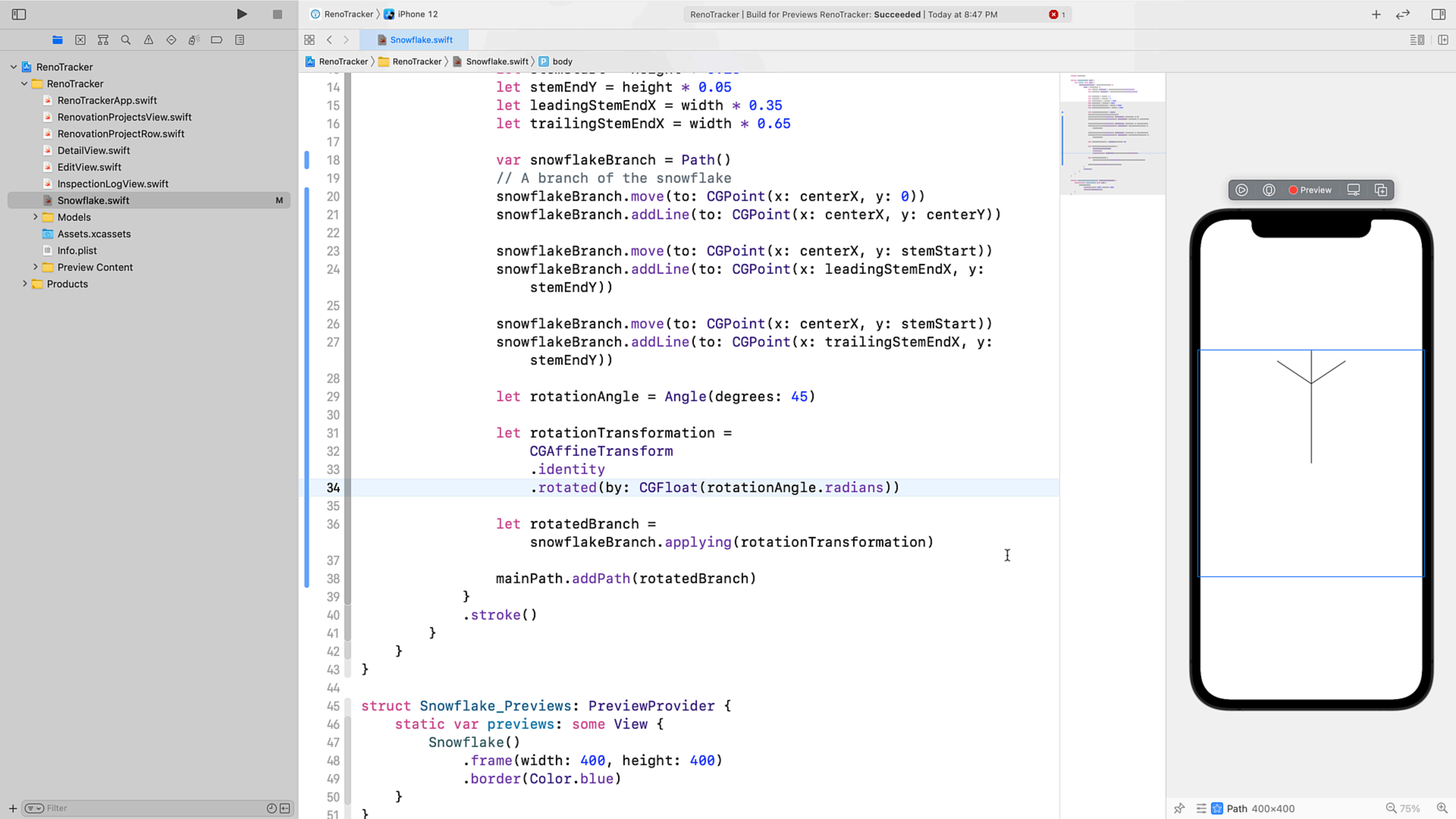
Rotate

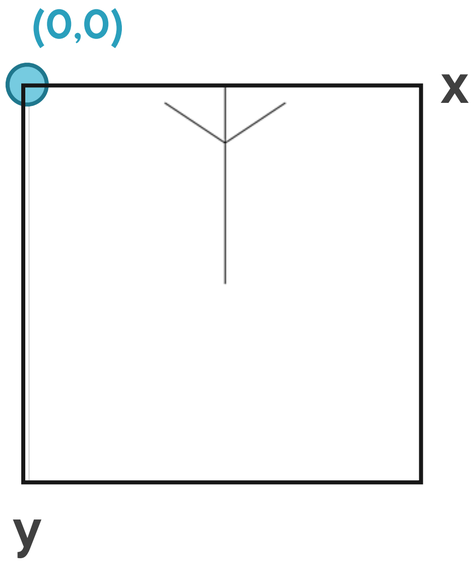


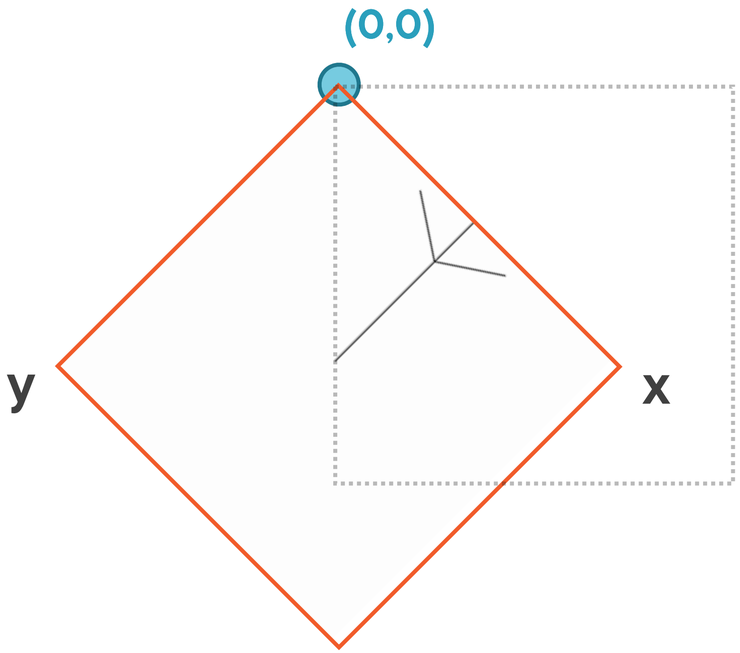
Scale

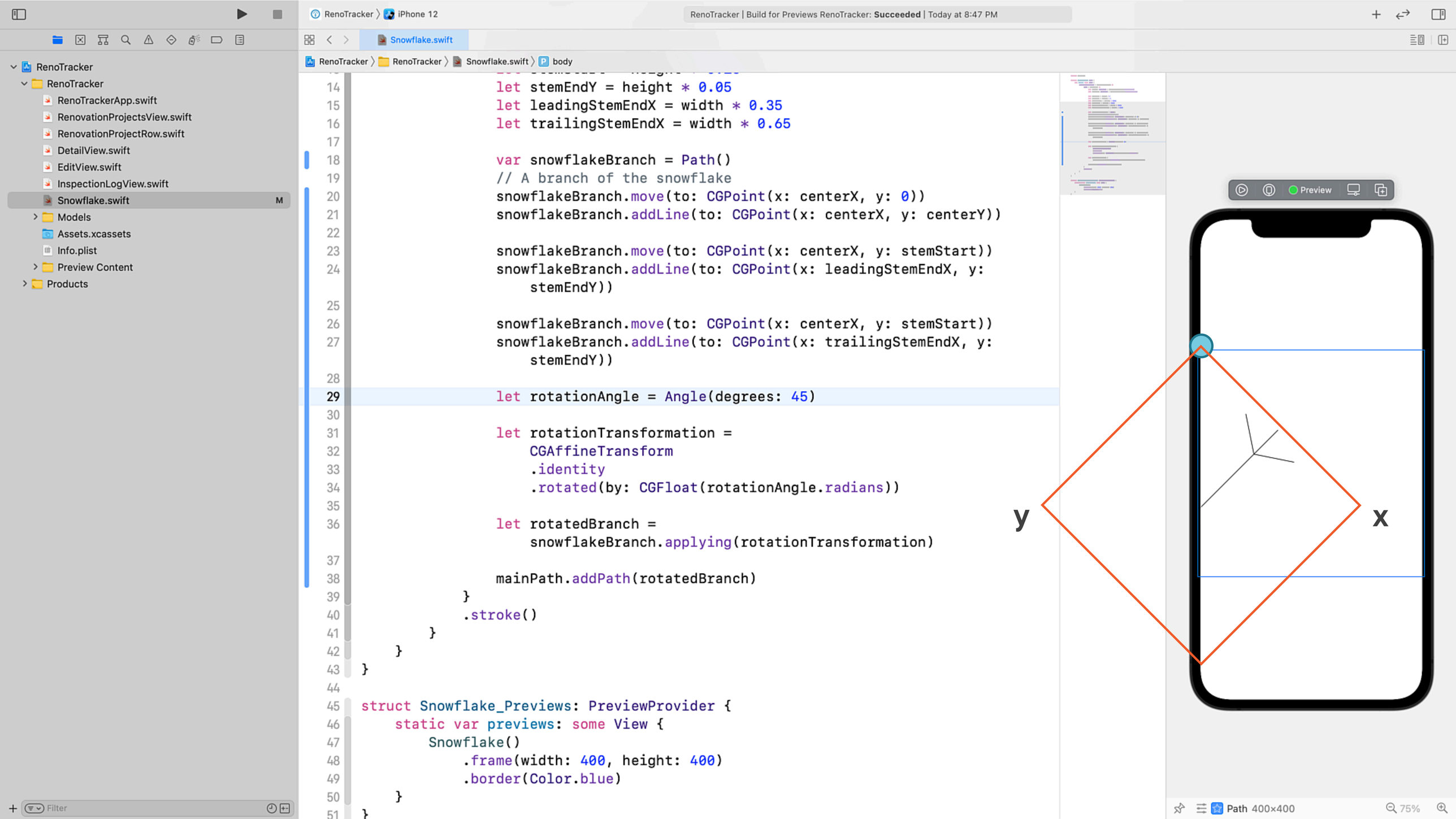
Translate

Skew


RenoTracker
iPhone 12
RenoTracker | Build for Previews RenoTracker: Succeeded | Today at 8:47 PM
Snowflake.swift
RenoTracker
RenoTracker
Snowflake.swift
body
RenoTracker
RenoTracker
RenoTrackerApp.swift
RenovationProjectsView.swift
RenovationProjectRow.swift
DetailView.swift
EditView.swift
InspectionLogView.swift
Snowflake.swift
M
Models
Assets.xcassets
Info.plist
Preview Content
Products
let stemEndY = height * 0.05
let leadingStemEndX = width * 0.35
let trailingStemEndX = width * 0.65
var snowflakeBranch = Path()
// A branch of the snowflake
snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
stemEndY))
snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
stemEndY))
let rotationAngle = Angle(degrees: 45)
let rotationTransformation =
CGAffineTransform
.identity
.rotated(by: CGFloat(rotationAngle.radians))
let rotatedBranch =
snowflakeBranch.applying(rotationTransformation)
mainPath.addPath(rotatedBranch)
}
.stroke()
}
}
}
struct Snowflake_Previews: PreviewProvider {
static var previews: some View {
Snowflake()
.frame(width: 400, height: 400)
.border(Color.blue)
}
}
Preview
Path 400×400
75%


(0,0)
x
y


(0,0)
y
x

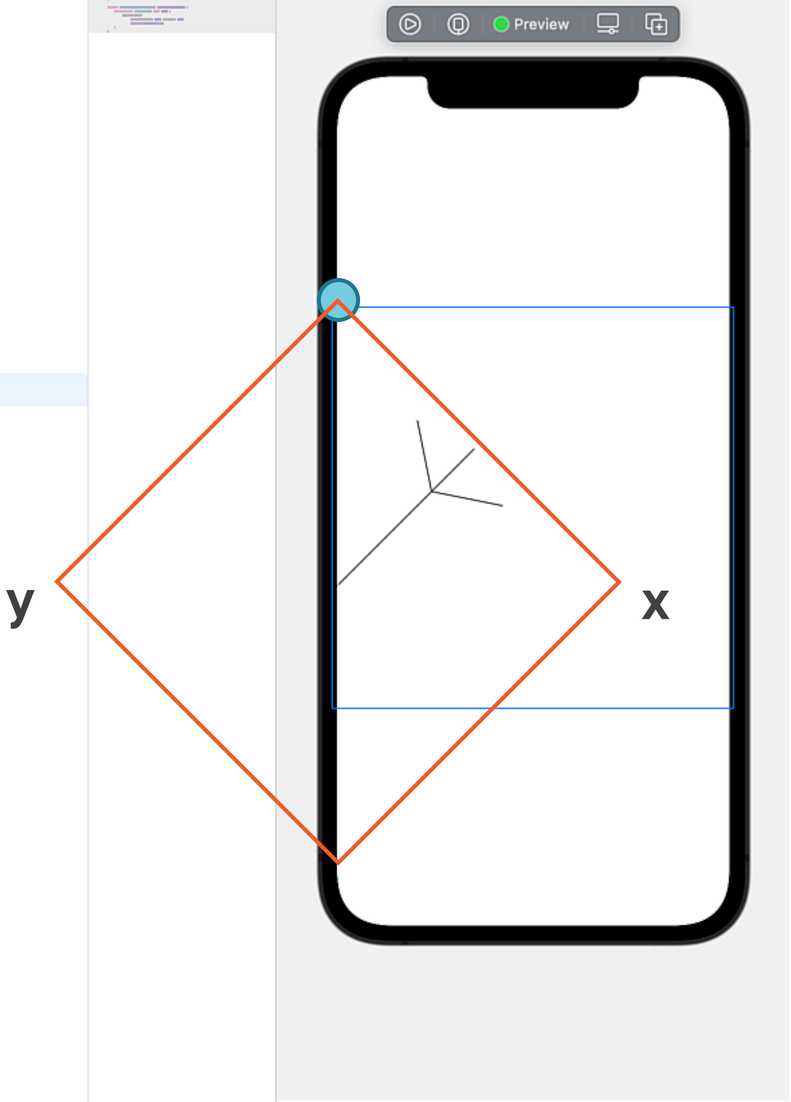

RenoTracker
iPhone 12
RenoTracker | Build for Previews RenoTracker: Succeeded | Today at 8:47 PM
Snowflake.swift
RenoTracker
RenoTracker
Snowflake.swift
body
RenoTracker
RenoTracker
RenoTrackerApp.swift
RenovationProjectsView.swift
RenovationProjectRow.swift
DetailView.swift
EditView.swift
InspectionLogView.swift
Snowflake.swift
M
Models
Assets.xcassets
Info.plist
Preview Content
Products
Filter
let stemEndY = height * 0.05
let leadingStemEndX = width * 0.35
let trailingStemEndX = width * 0.65
var snowflakeBranch = Path()
// A branch of the snowflake
snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
stemEndY))
snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
stemEndY))
let rotationAngle = Angle(degrees: 45)
let rotationTransformation =
CGAffineTransform
.identity
.rotated(by: CGFloat(rotationAngle.radians))
let rotatedBranch =
snowflakeBranch.applying(rotationTransformation)
mainPath.addPath(rotatedBranch)
}
.stroke()
}
}
}
struct Snowflake_Previews: PreviewProvider {
static var previews: some View {
Snowflake()
.frame(width: 400, height: 400)
.border(Color.blue)
}
}
Preview
y
x
Path 400×400
75%

RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```

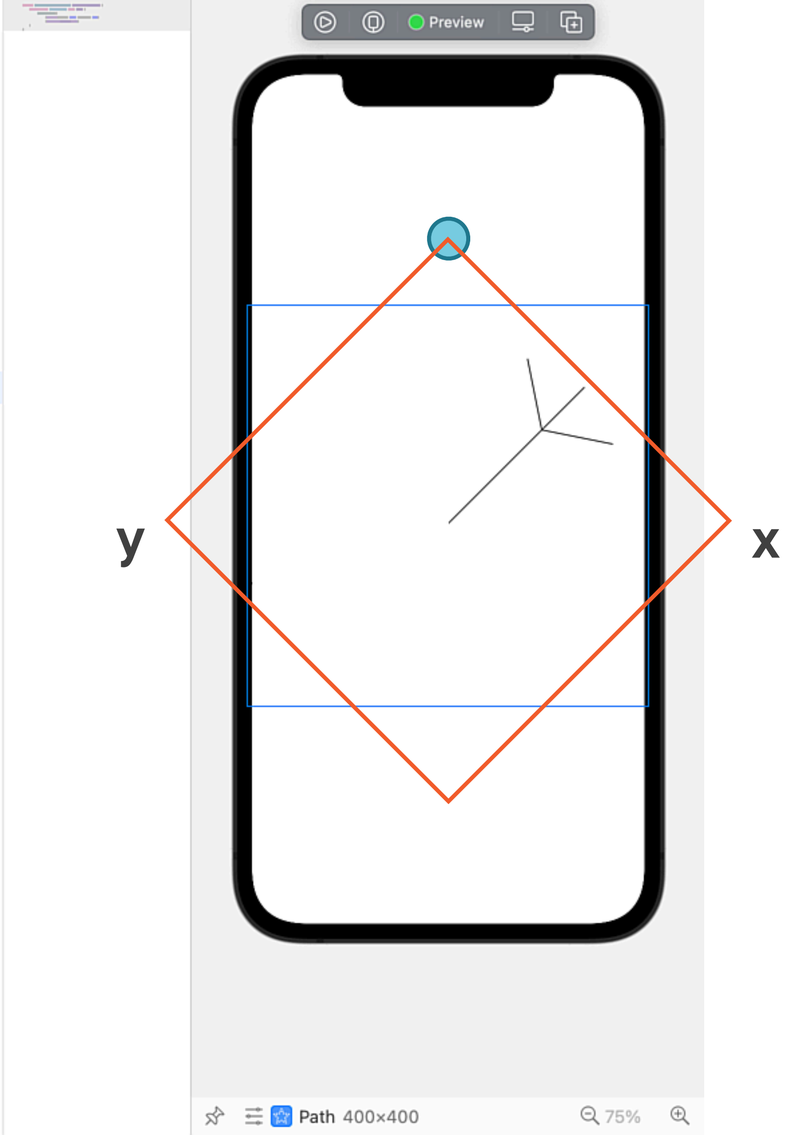


RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```

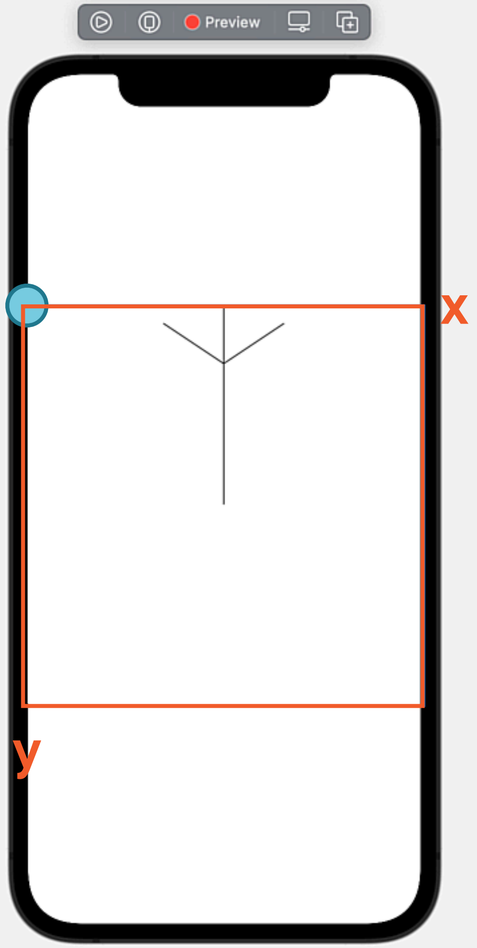


RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM | 1

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```



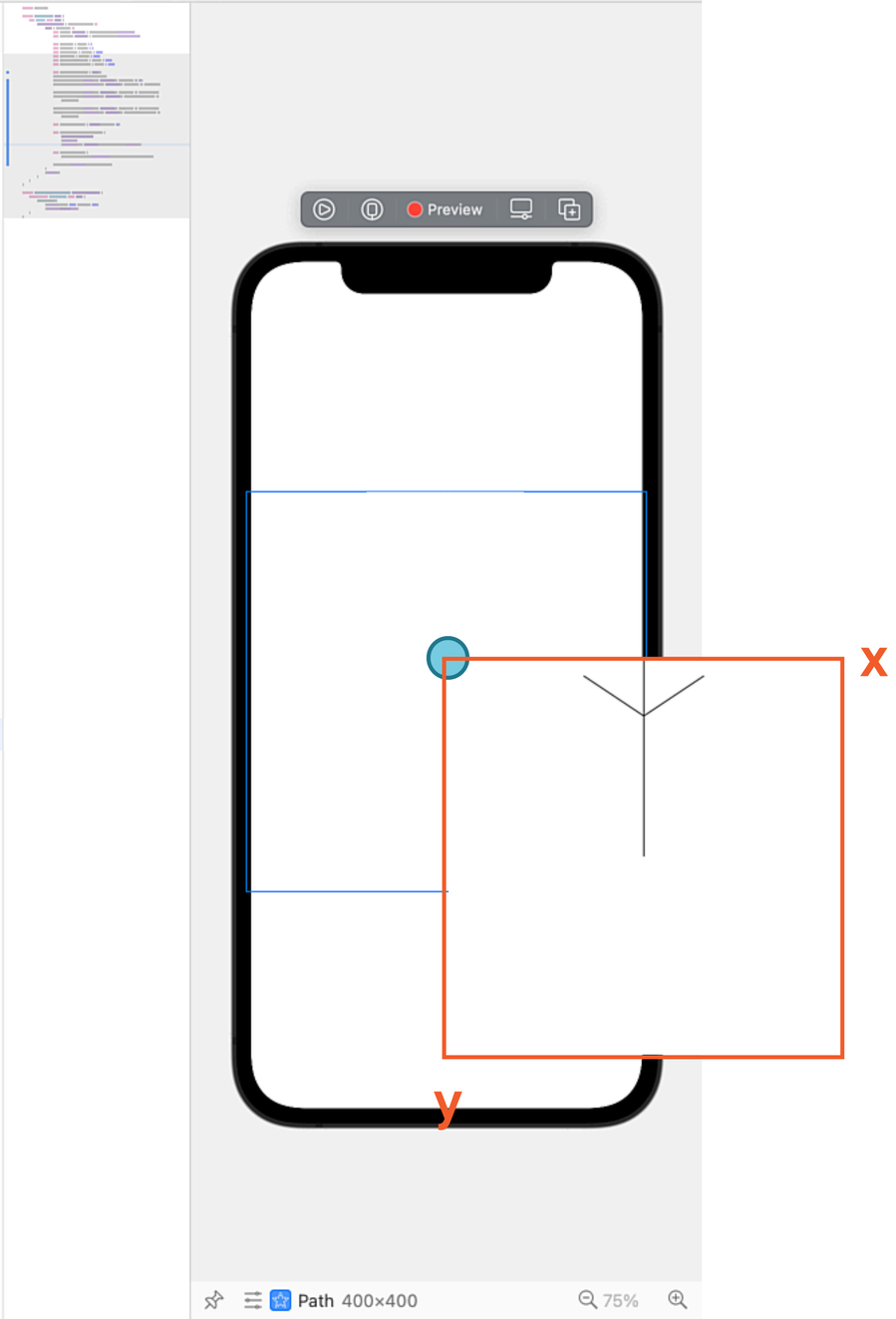
Path 400×400 75%

RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM | 1

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```

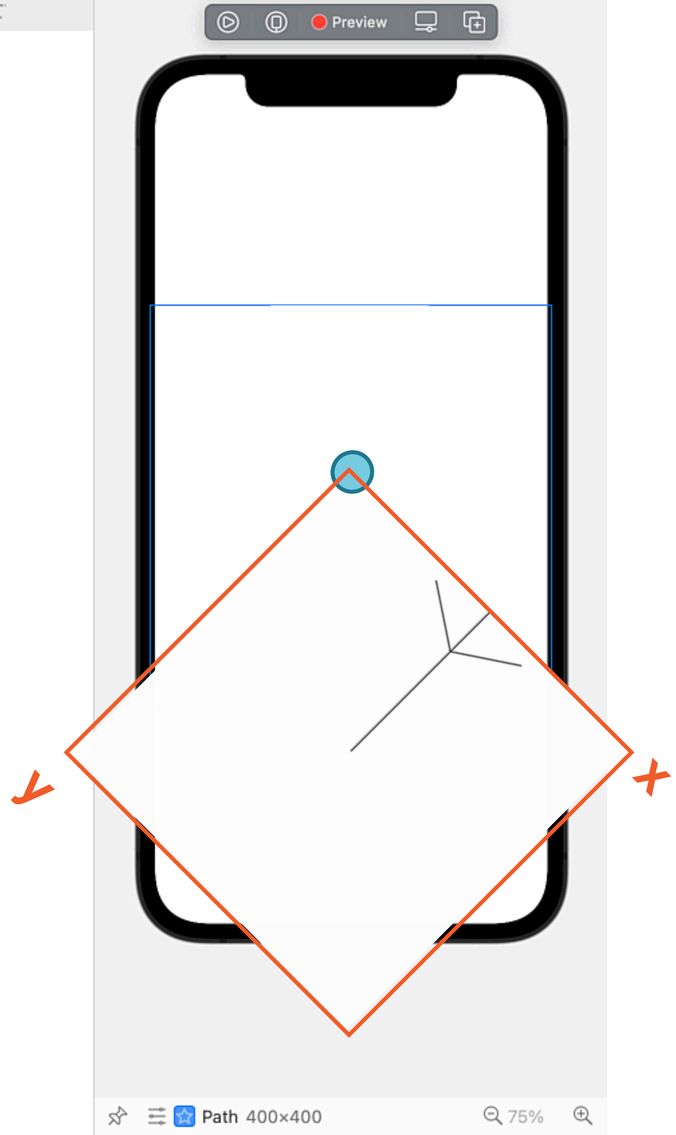


RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM | 1

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```

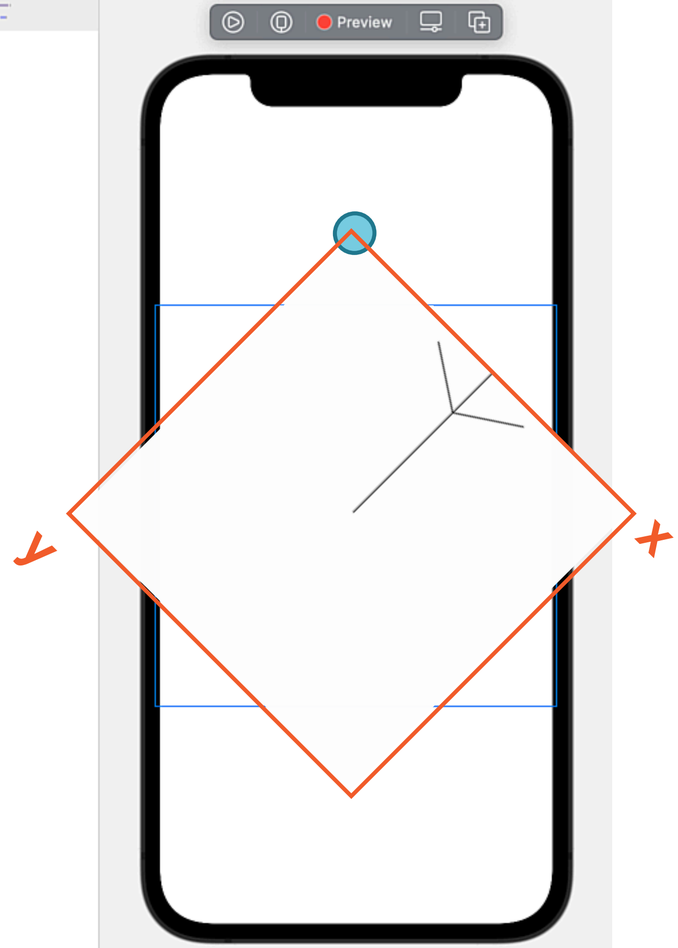


RenoTracker 〉 iPhone 12 | RenoTracker | Build for Previews RenoTracker: **Succeeded** | Today at 8:47 PM | 1

Snowflake.swift

RenoTracker 〉 RenoTracker 〉 Snowflake.swift 〉 body

```
14                  let stemEndY = height * 0.05
15                  let leadingStemEndX = width * 0.35
16                  let trailingStemEndX = width * 0.65
17
18                  var snowflakeBranch = Path()
19                  // A branch of the snowflake
20                  snowflakeBranch.move(to: CGPoint(x: centerX, y: 0))
21                  snowflakeBranch.addLine(to: CGPoint(x: centerX, y: centerY))
22
23                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
24                  snowflakeBranch.addLine(to: CGPoint(x: leadingStemEndX, y:
                        stemEndY))
25
26                  snowflakeBranch.move(to: CGPoint(x: centerX, y: stemStart))
27                  snowflakeBranch.addLine(to: CGPoint(x: trailingStemEndX, y:
                        stemEndY))
28
29                  let rotationAngle = Angle(degrees: 45)
30
31                  let rotationTransformation =
32                      CGAffineTransform
33                      .identity
34                      .rotated(by: CGFloat(rotationAngle.radians))
35
36                  let rotatedBranch =
                        snowflakeBranch.applying(rotationTransformation)
37
38                  mainPath.addPath(rotatedBranch)
39              }
40              .stroke()
41          }
42      }
43  }
44
45  struct Snowflake_Previews: PreviewProvider {
46      static var previews: some View {
47          Snowflake()
48              .frame(width: 400, height: 400)
49              .border(Color.blue)
50      }
51  }
```



Path 400×400 75%

How can we make our drawings more reusable?

Building Reusable Shapes

Swift

Featured
App Frameworks
Accessibility
App Clips
AppKit
Bundle Resources
Foundation
Swift
SwiftUI
Essentials
Introducing SwiftUI
App Structure and Behavior
User Interface
Views and Controls
View Layout and Presentati...
Drawing and Animation
Essentials
Drawing Paths and Shap...
Building Custom Views i...
Shape
Defining a Shape's Path
func path(in: CGRect)...
func trim(from: CGFlo...
Transforming a Shape
func transform(CGAffi...
func size(CGSize) -> s...
func size(width: CGFl...
func scale(CGFloat, a...
func scale(x: CGFloat,...
func rotation(Angle, a...
func offset(CGSize) -...
func offset(CGPoint) -...
func offset(x: CGFloat...
Setting the Stroke Char...
func stroke<S>(S, line...
func stroke<S>(S, styl...
func stroke(lineWidth:...
func stroke(style: Stro...
Filling a Shape
func fill<S>(S, style: Fi...
func fill(style: FillStyle...
Animation
Animating Views and Tra...
Animation

Filter

Protocol

Shape

A 2D shape that you can use when drawing a view.

Declaration

```
protocol Shape : Animatable, View
```

Overview

Shapes without an explicit fill or stroke get a default fill based on the foreground color.

You can define shapes in relation to an implicit frame of reference, such as the natural size of the view that contains it. Alternatively, you can define shapes in terms of absolute coordinates.

Availability

iOS 13.0+

macOS 10.15+

Mac Catalyst 13.0+

tvOS 13.0+

watchOS 6.0+

Framework

SwiftUI

On This Page

Declaration
Overview
Topics
Relationships

Topics

Defining a Shape's Path

```
func path(in: CGRect) -> Path
```

Describes this shape as a path within a rectangular frame of reference.

Required.

```
func trim(from: CGFloat, to: CGFloat) -> some Shape
```

Trims this shape by a fractional amount based on its representation as a path.

How do we use animation in our SwiftUI apps?

Understanding Animations

Animation signals that important changes have occurred in the application.

Animation

Describe the way to change a View's visual appearance from some starting state to some ending state over time.

Start

End

Start

End

Start

End

I want this car to drive at a constant speed and arrive at the destination in six seconds.

Start

End

I want this car to gradually accelerate to its peak driving speed and then gradually decelerate at the end.

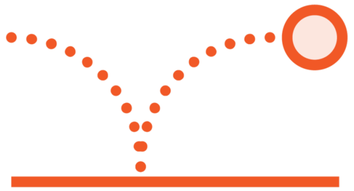

Start
End

How do we take the concept of change over time and apply it to a user interface?

Adding Basic Animation

Animation

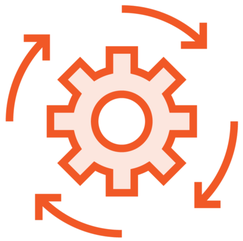
Describe the way to change a View's visual appearance from some starting state to some ending state over time.



Some animatable aspect of a View depends on a variable that changes.

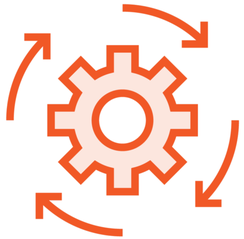
A View at rest will stay at rest
unless something about the state
it depends on changes.

Context



SwiftUI recomputes a View’s body

Context

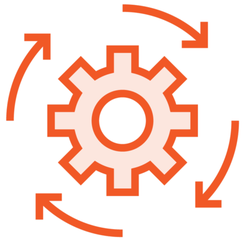


SwiftUI recomputes a View's body

Animatable Aspect

An aspect of a View that can be represented as a decimal number

Context



SwiftUI recomputes a View's body

Animatable Aspect

An aspect of a View that can be represented as a decimal number

---

Arguments to View initializers or View modifiers that require one of Swift's decimal data types (like Float, Double, or CGFloat) serve as candidates for animation.

Swift
Accessibility
App Clips
AppKit
Bundle Resources
Foundation
Swift
SwiftUI
Essentials
Introducing SwiftUI
App Structure and Behavior
User Interface
Views and Controls
View Layout and Presentati...
Drawing and Animation
Essentials
Drawing Paths and Shap...
Building Custom Views i...
Shape
Animation
Animating Views and Tra...
Animation
Animatable
AnimatableModifier
func withAnimation<Res...
AnimatablePair
EmptyAnimatableData
AnyTransition
Shapes
Rectangle
Edge
RoundedRectangle
Circle
Ellipse
Capsule
Path
Transformed Shapes
InsettableShape
ScaledShape
RotatedShape
OffsetShape
TransformedShape
ContainerRelativeShape
Paints, Styles, and Gradie...
Color
Filter

Protocol

VectorArithmetic

A type that can serve as the animatable data of an animatable type.

Declaration

```
protocol VectorArithmetic : AdditiveArithmetic
```

Overview

`VectorArithmetic` extends the `AdditiveArithmetic` protocol with scalar multiplication and a way to query the vector magnitude of the value. Use this type as the `animatableData` associated type of a type that conforms to the `Animatable` protocol.

Availability

iOS 13.0+

macOS 10.15+

Mac Catalyst 13.0+

tvOS 13.0+

watchOS 6.0+

Framework

SwiftUI

On This Page

Declaration
Overview
Topics
Relationships

Topics

Instance Properties

`var magnitudeSquared: Double`

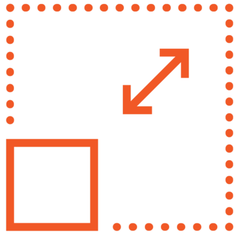
Returns the dot-product of this vector arithmetic instance with itself.

Required.

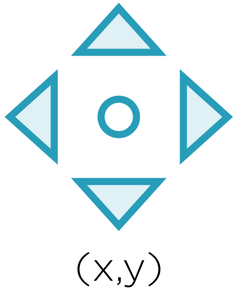
Instance Methods

`func scale(by: Double)`

Multiplies each component of this value by the given value.



Size



Position

Angle of Rotation

Opacity

Color

Swift
App Frameworks
Accessibility
App Clips
AppKit
Bundle Resources
Foundation
Swift
SwiftUI
Essentials
Introducing SwiftUI
App Structure and Behavior
User Interface
Views and Controls
View Layout and Presentati...
Drawing and Animation
Essentials
Drawing Paths and Shap...
Building Custom Views i...
Shape
Animation
Animating Views and Tra...
Animation
Animatable
AnimatableModifier
func withAnimation<Res...
AnimatablePair
EmptyAnimatableData
AnyTransition
Shapes
Rectangle
Edge
RoundedRectangle
Circle
Ellipse
Capsule
Path
Transformed Shapes
InsettableShape
ScaledShape
RotatedShape
OffsetShape
TransformedShape
ContainerRelativeShape
Paints, Styles, and Gradie...
Color
Filter

Structure

Color

An environment-dependent color.

Declaration

```
@frozen struct Color
```

Overview

A `Color` is a late-binding token: SwiftUI only resolves it to a concrete value just before using it in a given environment.

Availability

iOS 13.0+

macOS 10.15+

Mac Catalyst 13.0+

tvOS 13.0+

watchOS 6.0+

Framework

SwiftUI

On This Page

Declaration
Overview
Topics
Relationships

Topics

Creating a Color

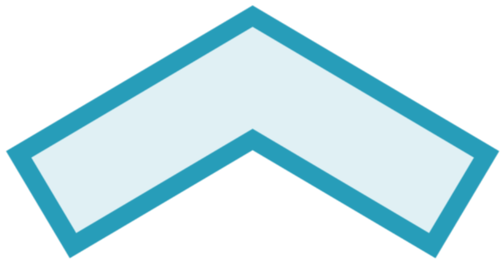
`init(String, bundle: Bundle?)`

Creates a named color.

`init(Color.RGBColorSpace, red: Double, green: Double, blue: Double, opacity: Double)`

`init(Color.RGBColorSpace, white: Double, opacity: Double)`

`init(hue: Double, saturation: Double, brightness: Double, opacity:`



Rotate 180 degrees

Rotate 180 degrees

Conditions for Animation

Data that's changeable

The appearance of your view is connected to that changeable (decimal value) data

Turn on animations with the animation modifier or wrap the trigger point in a withAnimation function

How do we animate changes to the View hierarchy?

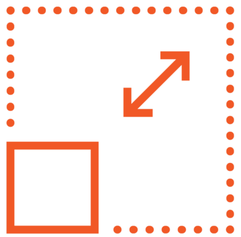
Animating View Hierarchy Changes with Transitions

Add a View to the View Hierarchy

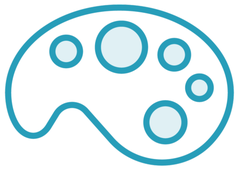
Add a View to the View Hierarchy

Remove a View from the View Hierarchy

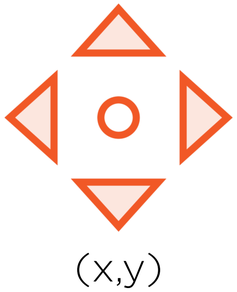
Remove a View from the View Hierarchy



Size



Color



Position

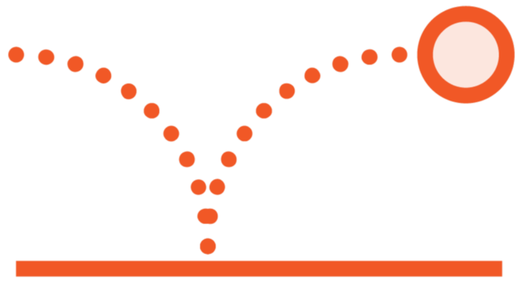
Angle of Rotation

showProgressInfoCard = true

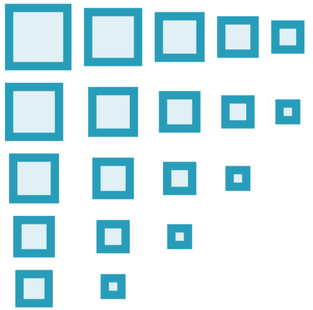
showProgressInfoCard = false

Transition

Instruct SwiftUI to animate changes in the View hierarchy smoothly over time.



Animation



Transition

How do we combine multiple animation effects and multiple transitions to create even more dynamic visual experiences?

Combining Multiple Animations and Transitions

9:38

Home Edit

Front Desk

Work Quality

★ ★ ★ ☆ View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [*] Paint walls
- [ ] Hang new artwork

Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

9:44

Home Edit

Front Desk

Work Quality

★ ★ ★ ☆ View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [*] Paint walls
- [ ] Hang new artwork

Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

Home

Edit

Front Desk

Work Quality

★ ★ ★ ☆

View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [ ] Paint walls
- [ ] Hang new artwork

Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

Home

Edit

Front Desk

Work Quality

★ ★ ★ ☆

View Inspection Log

Punch List

- [x] Remodel front desk
- [x] Retile entry
- [x] Replace light fixtures
- [ ] Paint walls
- [ ] Hang new artwork

Budget

On Budget

Amount Allocated:	$25,000.00
Spent to-date:	$18,350.00
Amount remaining:	**$6,650.00**

How do we respond to user gestures in SwiftUI?

Responding to Gestures

How does SwiftUI let us tap into the gesture recognition system of iOS?

How does SwiftUI let us tap into the gesture recognition system of iOS?

A

By applying one of SwiftUI's gesture modifiers to the View.

Home

Edit

Ski Lift 1

Work Quality

★ ★ ★ ★

View Inspection Log

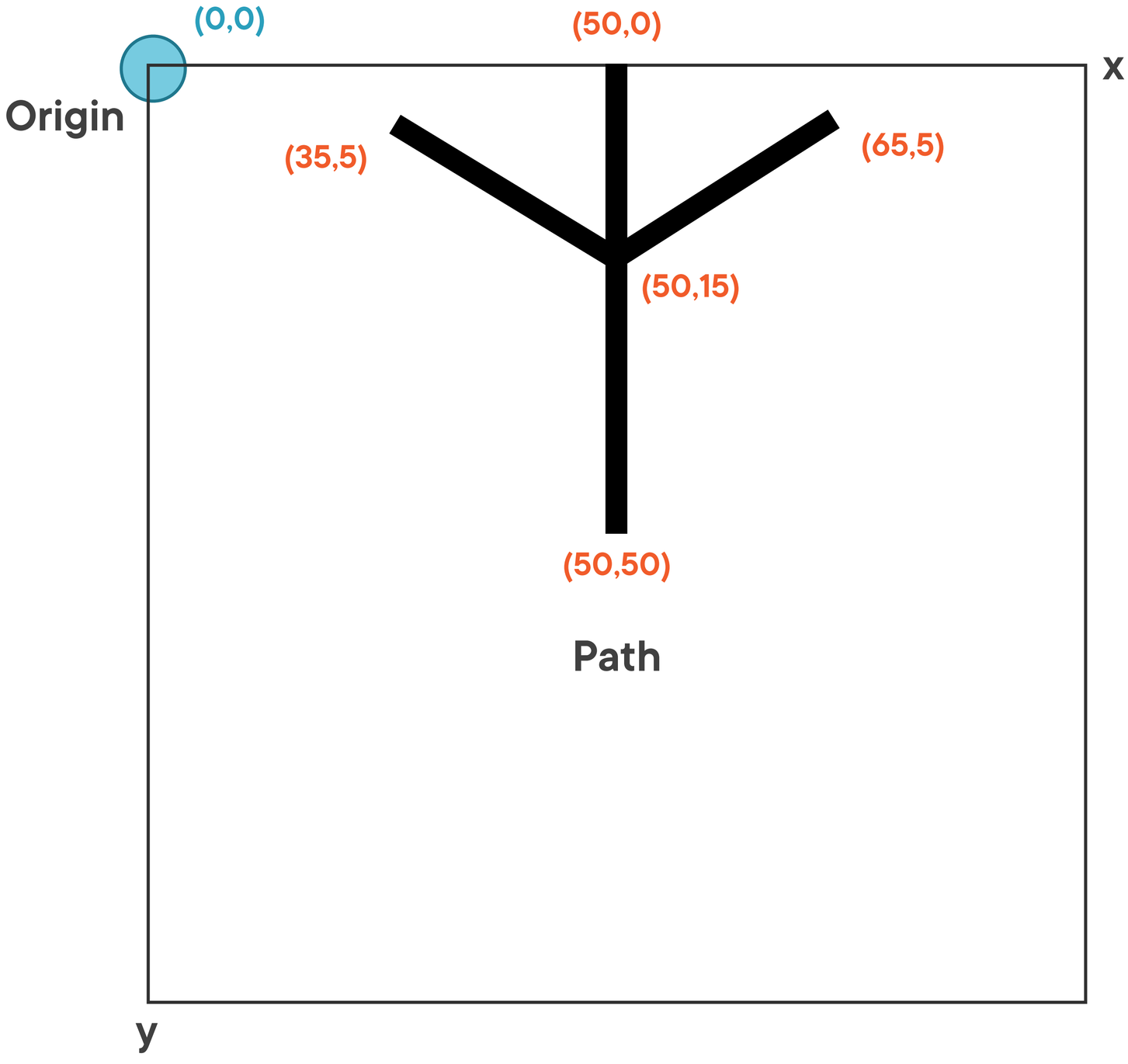
Punch List

- [x] Repair lift motor
- [*] Install new cables
- [*] Paint lift towers
- [ ] Install new chairs

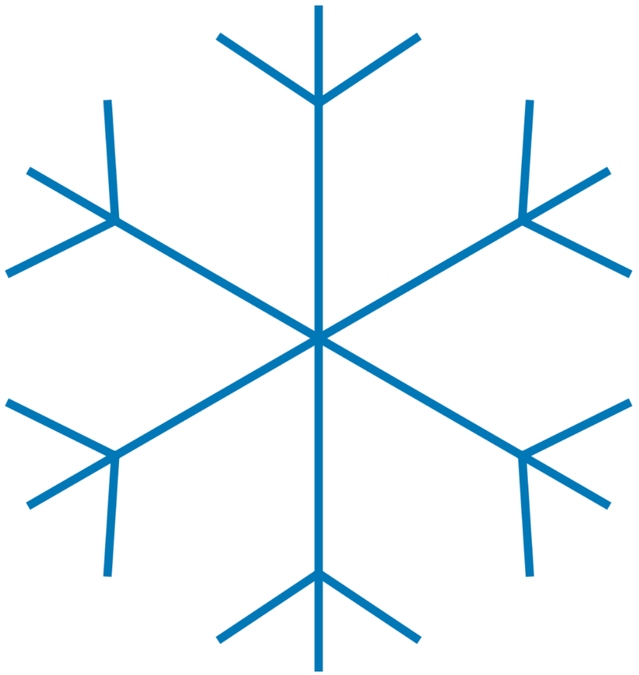
Budget

On Budget

Amount Allocated:	$24,500.00
Spent to-date:	$21,600.00
Amount remaining:	**$2,900.00**


(0,0)
(50,0)
x
Origin
(35,5)
(65,5)
(50,15)
(50,50)
Path
y

With GeometryReader, a View can dynamically determine its width, height, and position in a way that's directly related to the amount of space that's been offered.



Shape

Animation

I want this car to gradually accelerate to its peak driving speed and then gradually decelerate at the end.

Add a View to the View Hierarchy

Add a View to the View Hierarchy

Remove a View from the View Hierarchy

Remove a View from the View Hierarchy

Gestures

SwiftUI isn’t the only Apple UI framework being used in today’s Apple Developer world.

UIKit

How do we mix SwiftUI and UIKit to leverage the power of both frameworks?