

# Let's Scroll

**ScrollView from Zero**

**Ratnesh Jain**

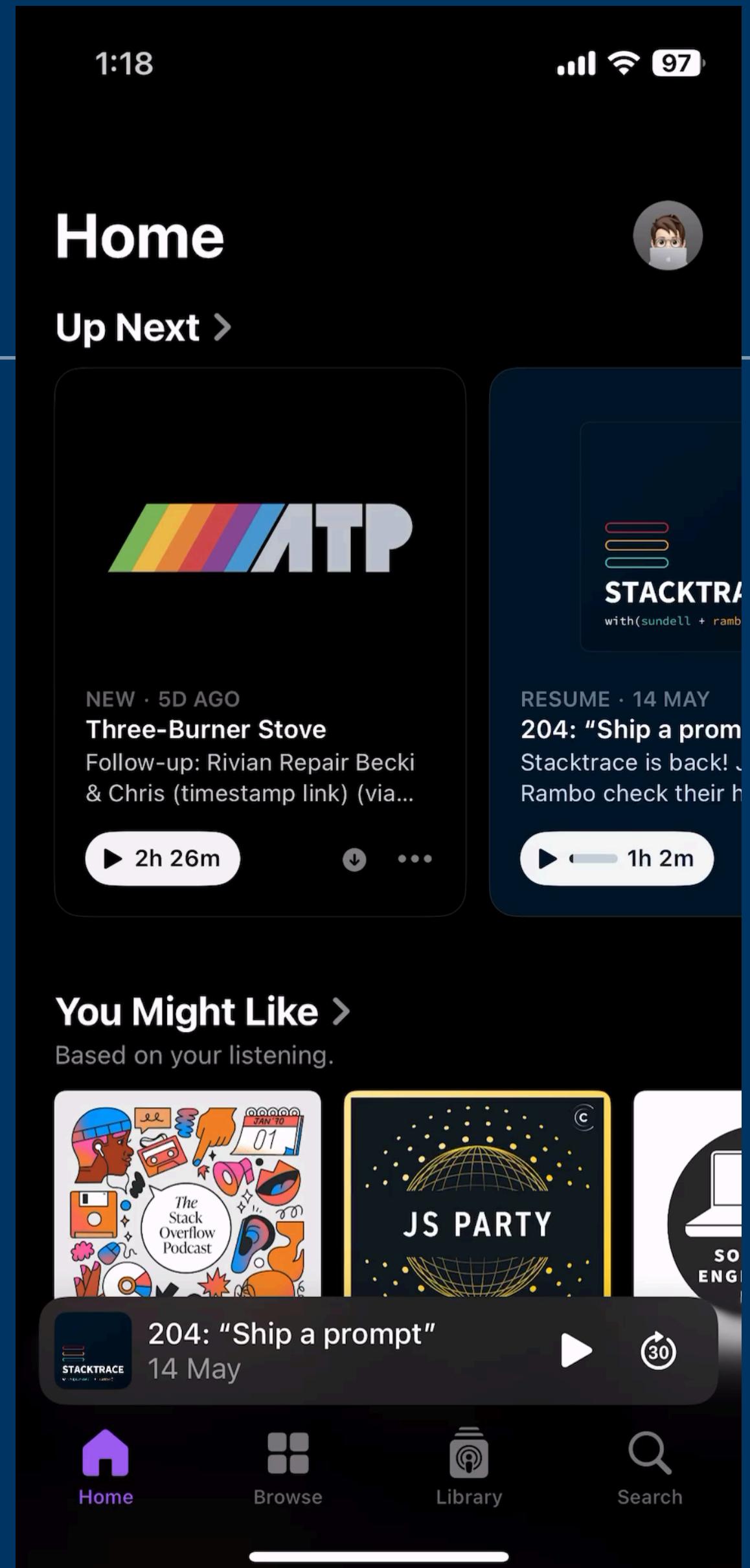
# Today

- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# Necessity of ScrollView

## Why Study ScrollView

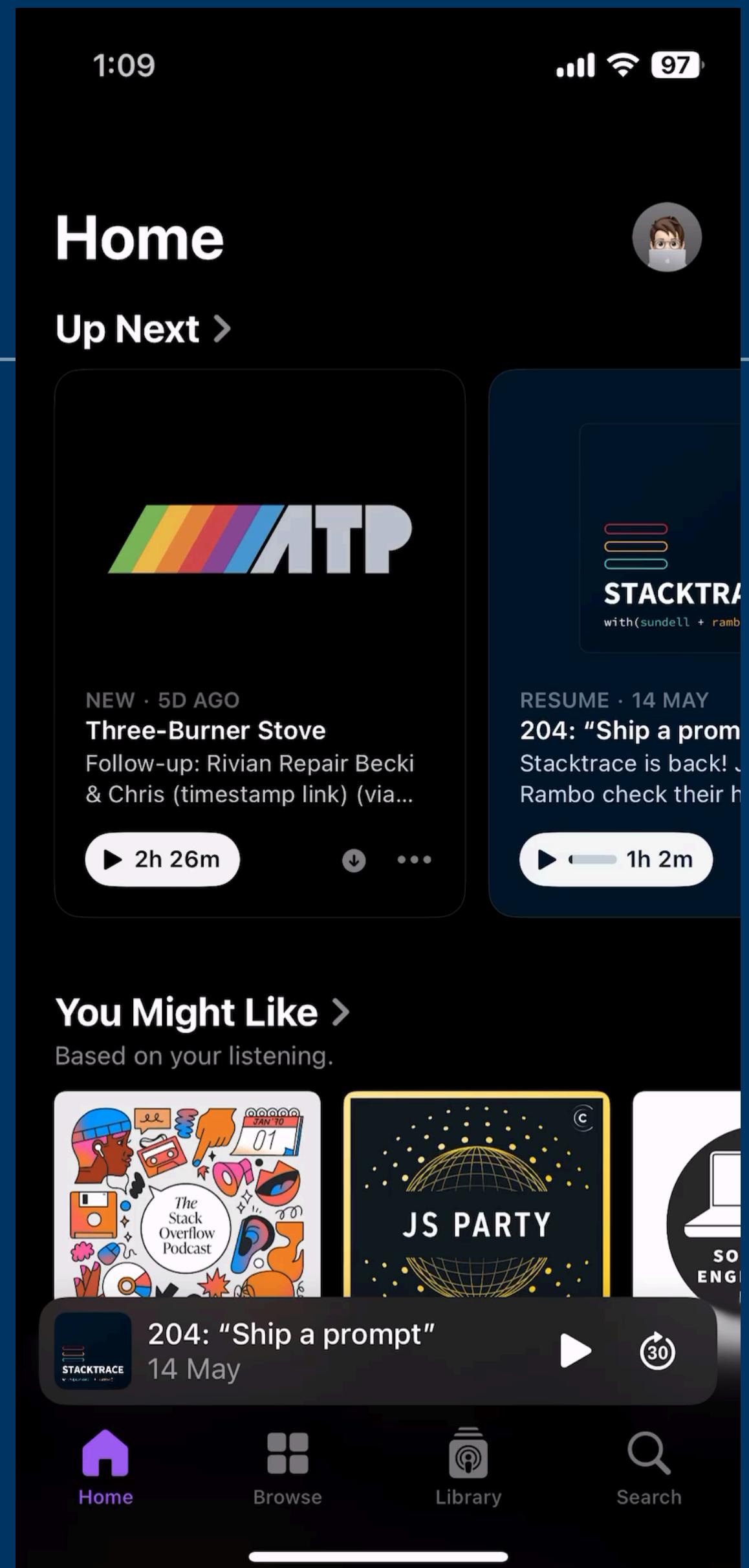
- Very Large content to show



# Necessity of ScrollView

## Why Study ScrollView

- Very Large content to show
- Unified smooth scrolling



# Necessity of ScrollView

## Why Study ScrollView

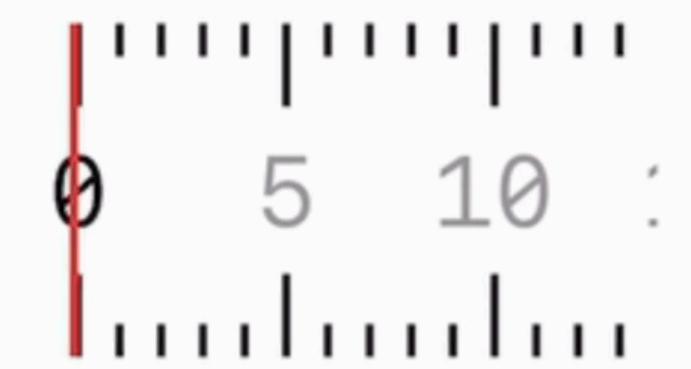
- Very Large content to show
- Unified smooth scrolling
- Better keyboard obstruction handling



# Necessity of ScrollView

## Why Study ScrollView

- Very Large content to show
- Unified smooth scrolling
- Better keyboard obstruction handling
- Unique style of controls/components



Ref: <https://uvolchyk.medium.com/scrolling-pickers-in-swiftui-de4a9c653fb6>

# Today

- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?

Begin with **SwiftUI**

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?

**SwiftUI** uses other(**existing**)  
**technologies** for **rendering**  
for some **parts**

When SwiftUI sees “Never” type in the View Tree

Metal, Core Animation, Core Graphics, Core Text, UIKit\*, AppKit\* ...

Why ScrollView Now?  
In the Era of SwiftUI we want be

- What should I learn?
- SwiftUI or UIKit?

CREATIVE in  
those parts,

We should LEARN  
it

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?

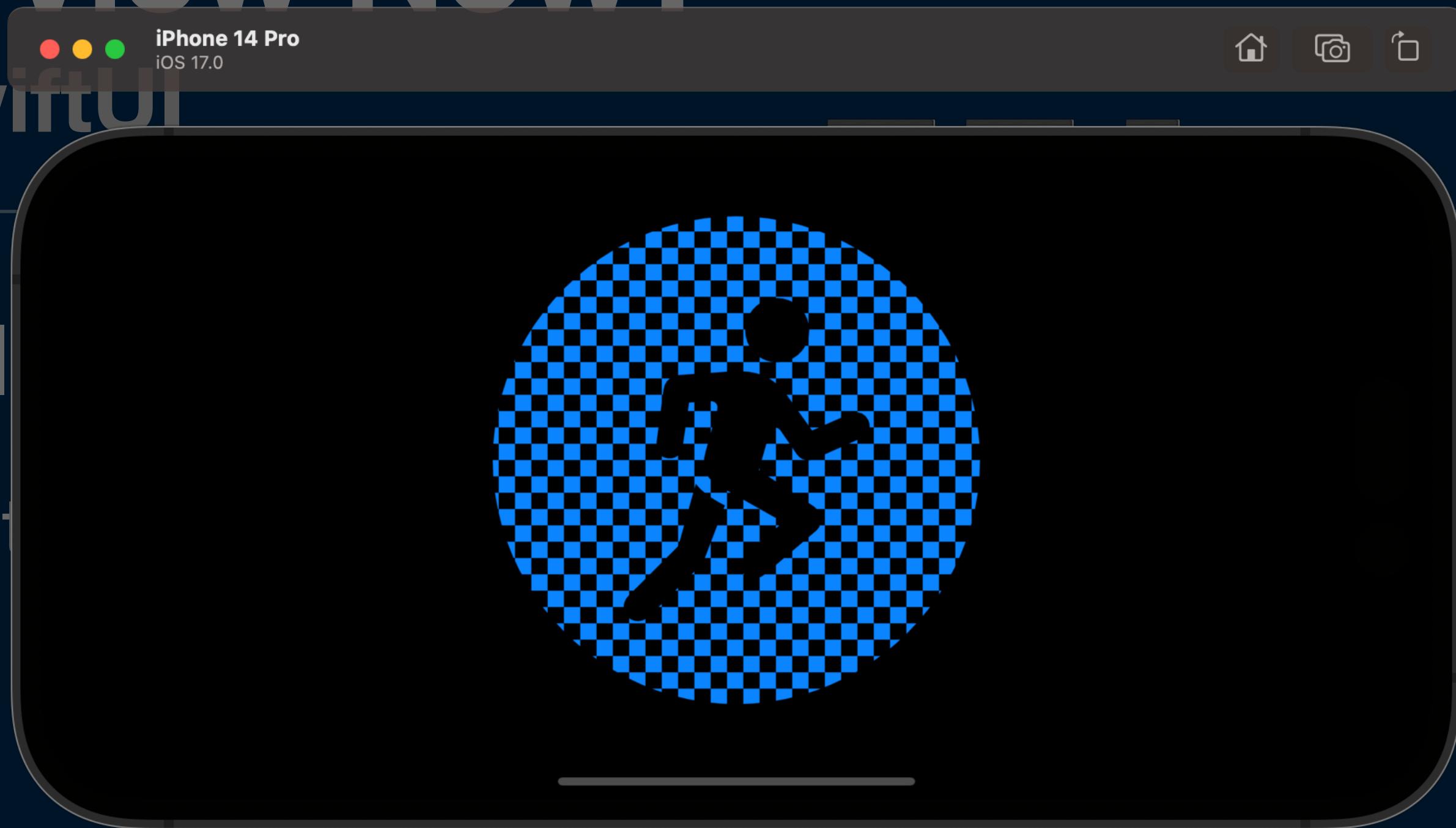


```
Image(systemName: "figure.run.circle.fill")
    .font(.system(size: 300))
    .colorEffect(ShaderLibrary.checkerboard(.float(10), .color(.blue)))
```

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?

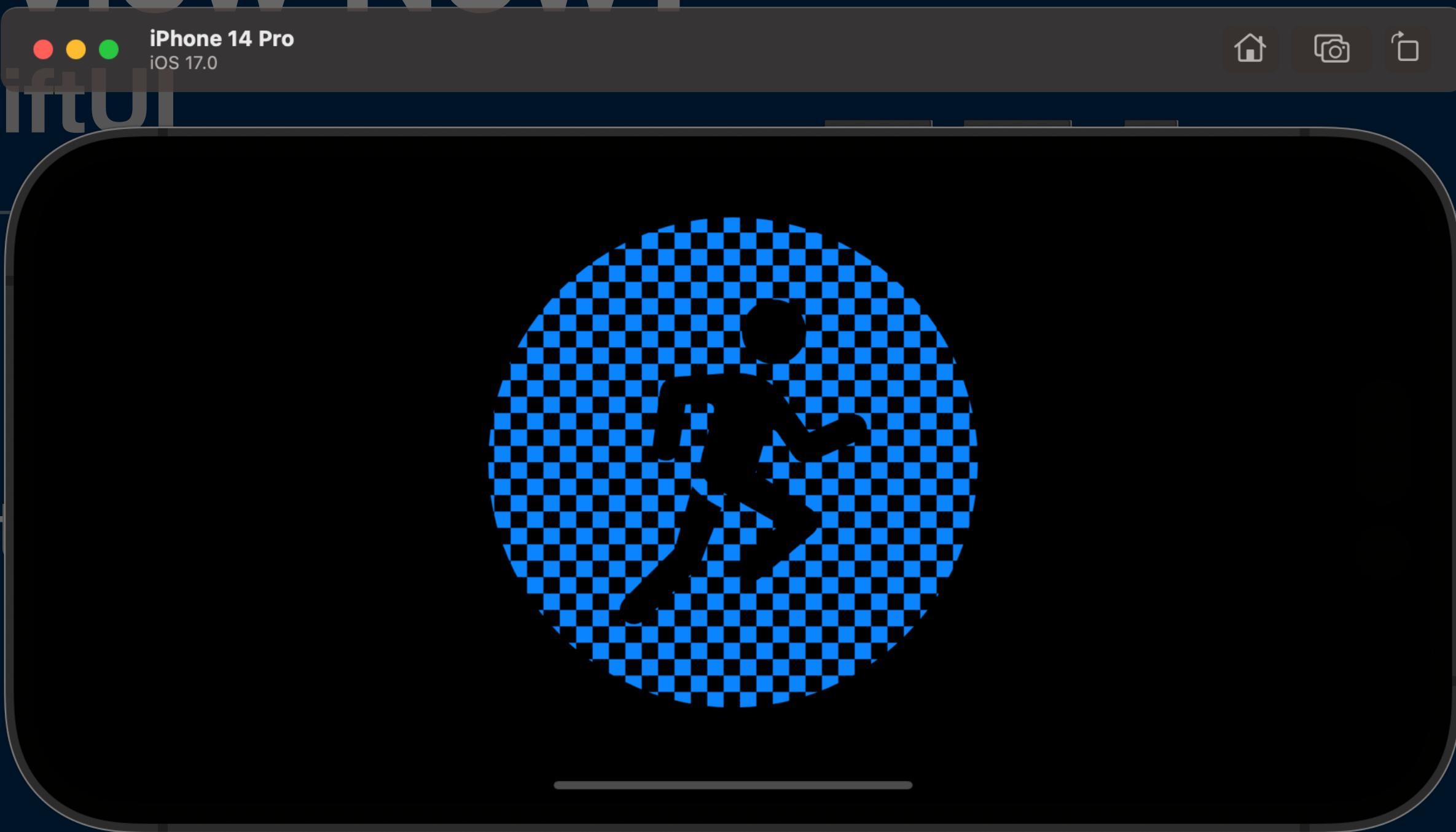


```
Image(systemName: "figure.run.circle.fill")
    .font(.system(size: 300))
    .colorEffect(ShaderLibrary.checkerboard(.float(10), .color(.blue)))
```

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?

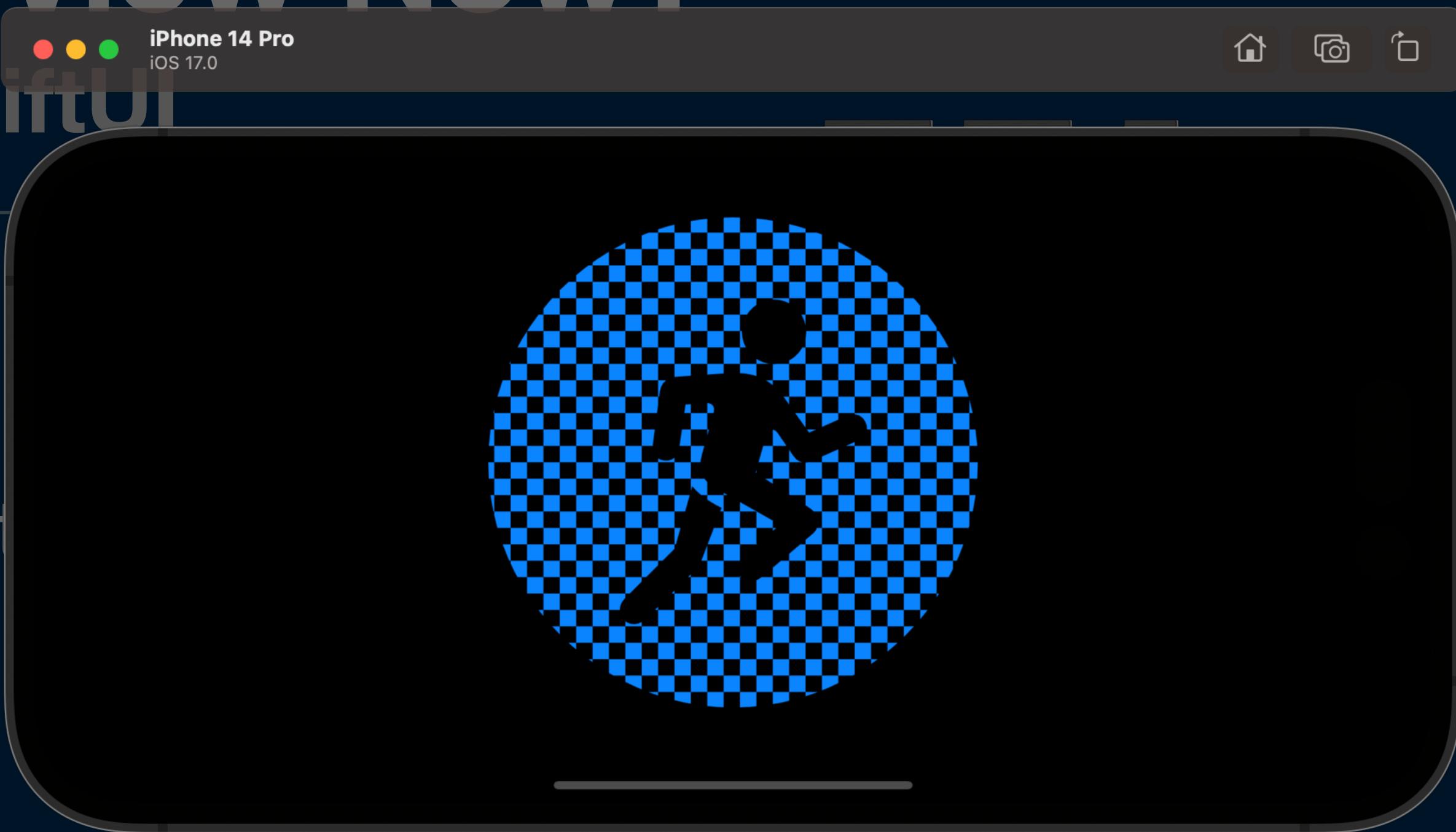


```
Image(systemName: "figure.run.circle.fill")
    .font(.system(size: 300))
    .colorEffect(ShaderLibrary.checkerboard(.float(10), .color(.blue)))
```

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?

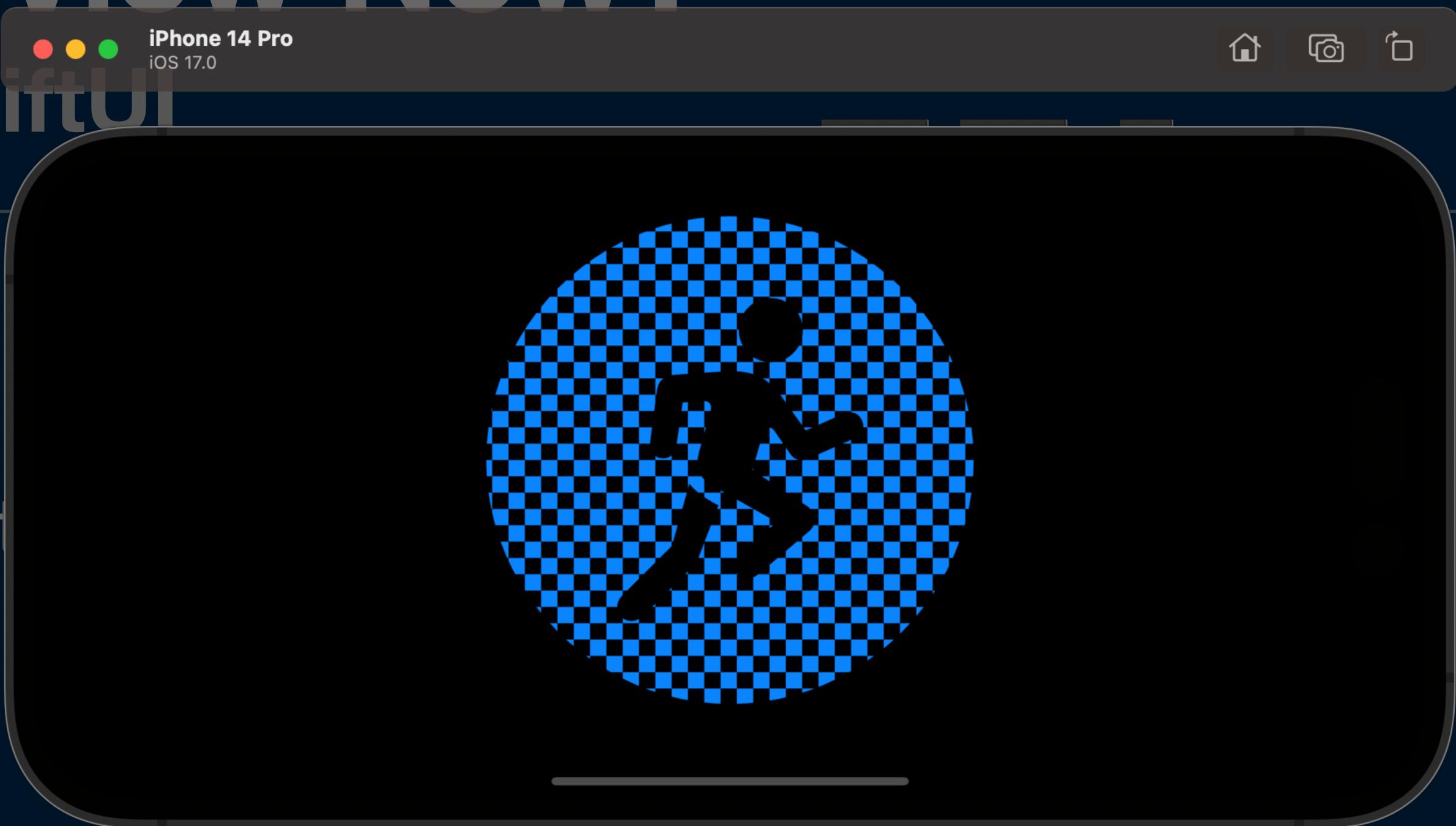


```
Image(systemName: "figure.run.circle.fill")
    .font(.system(size: 300))
    .colorEffect(ShaderLibrary.checkerboard(.float(10), .color(.blue)))
```

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn?
- SwiftUI or UIKit?



```
Image(systemName: "figure.run.circle.fill")
    .font(.system(size: 300))
    .colorEffect(ShaderLibrary.checkerboard(.float(10), .color(.blue)))
```

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
- SwiftUI or UIKit?



Metal

Available since 2014

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?
- How do I Learn it?

- Whenever I had trouble understanding a concept.

**Adam Savage (from Myth Busters)**



- Whenever I had trouble understanding a concept.
- I go back and I research people who discovered the concept.

**Adam Savage (from Myth Busters)**



- Whenever I had trouble understanding a concept.
- I go back and I research people who discovered the concept.
- I look at the story how they come to understand it.

**Adam Savage (from Myth Busters)**  
How simple ideas lead to scientific discoveries (Ted talk)



“I am a big fan of **Mike Ash’s**  
**Let’s Build ...** series of articles”

Quote from “Understanding UIScrollView”

by **Ole Begemann**  
Co-author of “Advanced Swift”



# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build

Let's Build **dispatch\_queue**  
at (2015-09-04 13:22)

Let's Build **Swift.Array**  
at (2015-04-17 13:42)

Let's Build **@synchronized**  
at (2015-02-20 14:26)

Let's Build **Swift Notifications**  
at (2015-01-23 14:52)

Let's Build **Dispatch Groups**  
at (2013-08-17 02:19)

Let's Build **stringWithFormat:**  
at (2013-05-17 14:40)

Let's Build **NSInvocation, Part II**  
at (2013-03-22 14:57)

Let's Build **NSInvocation, Part I**  
at (2013-03-08 14:33)

Let's Build **UITableView**  
at (2013-02-22 15:35)

Let's Build **Key-Value Coding**  
at (2013-02-08 14:17)

Let's Build **NSObject**  
at (2013-01-25 15:32)

Let's Build A **Mach-O Executable**  
at (2012-11-30 17:59)

# Mike Ash's Let's Build



<https://mikeash.com/pyblog/?tag=letsbuild>



# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?
- How do I Learn it?

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?
- How do I Learn it?
- What are the benefits?

# Why ScrollView Now?

In the Era of SwiftUI

- What should I learn?  
    • SwiftUI or UIKit?
- How do I learn?  
    • What are the benefits?



## Machinery



# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?

- SwiftUI, UIKit, etc.

**Better Understanding**

- How do I Learn it?

**of the NEW APIs**

- What are the benefits?

# Why ScrollView Now?

## In the Era of SwiftUI

- What should I learn ?
  - SwiftUI or UIKit?
- Help us take **Better** decisions.
- What are the benefits?

# Why ScrollView Now?

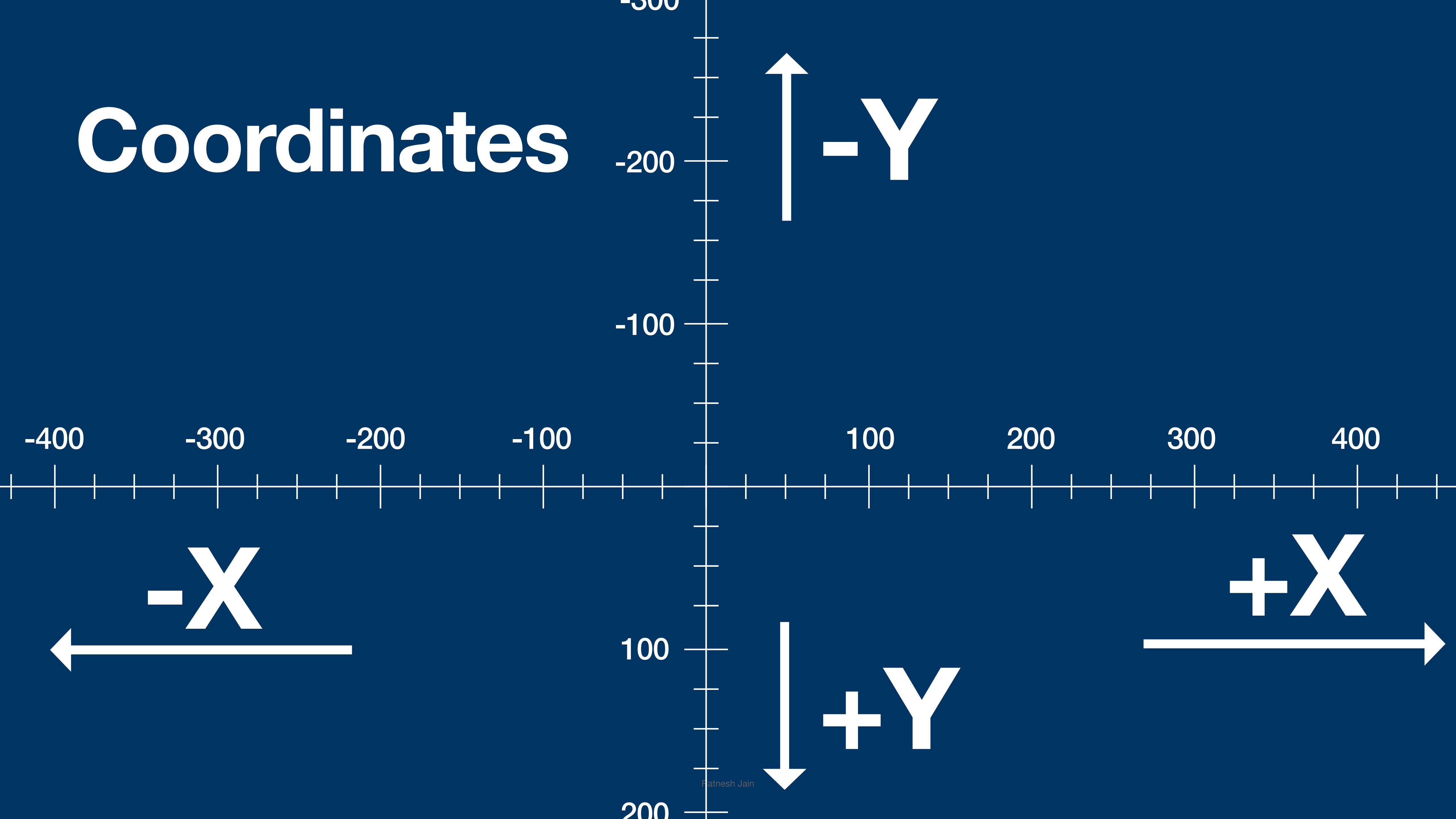
## In the Era of SwiftUI

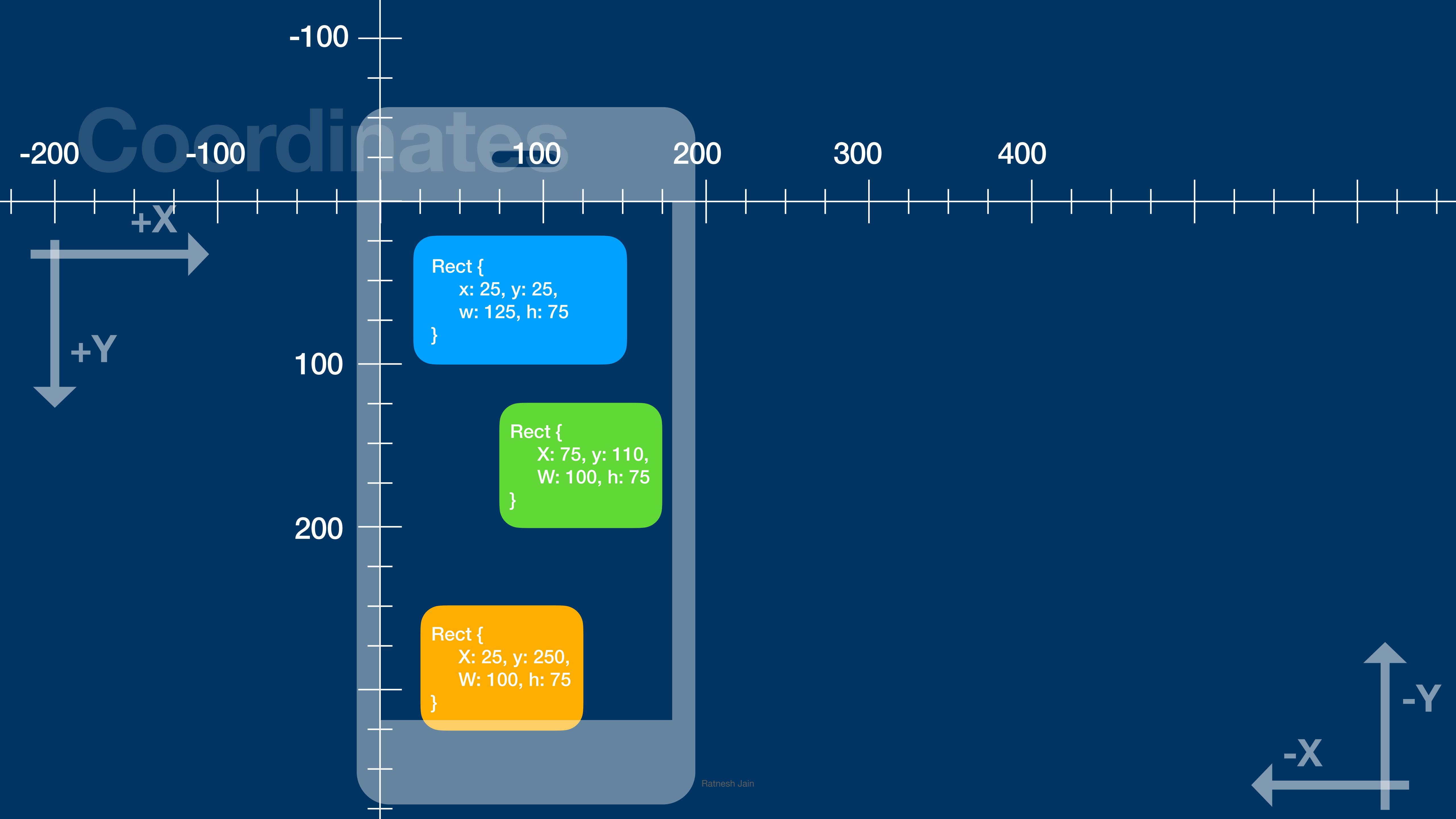
- What should I learn ?
  - SwiftUI or UIKit?
- How do I Learn it?
- What are the benefits?
  - Help us understand behind the scene machinery
  - Better understanding of the new APIs
  - Help us take better decisions.

# Today

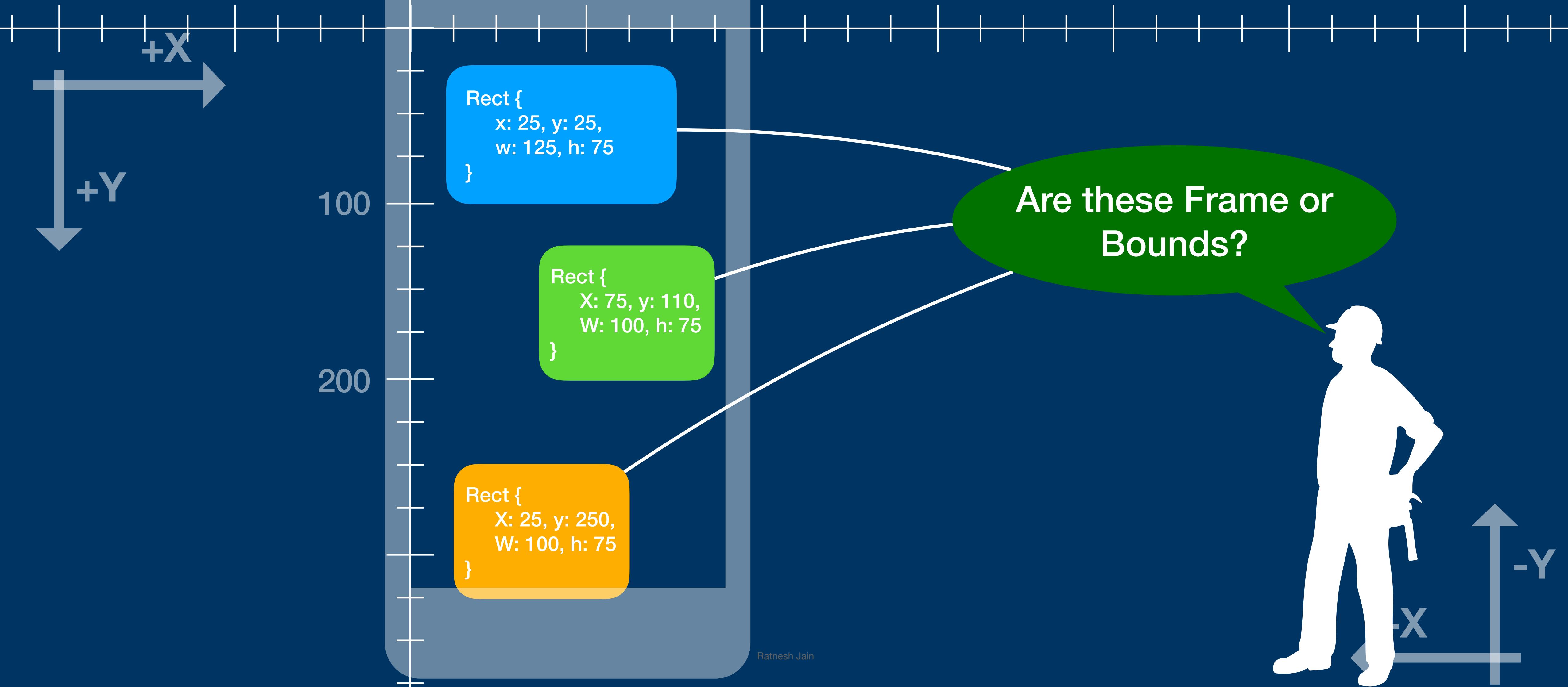
- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# Coordinates





# Coordinates



# Coordinates

# Bounds are rect from it's Own Coordinate System

```
Rect {  
    X: 25, y: 250,  
    W: 100, h: 75  
}
```

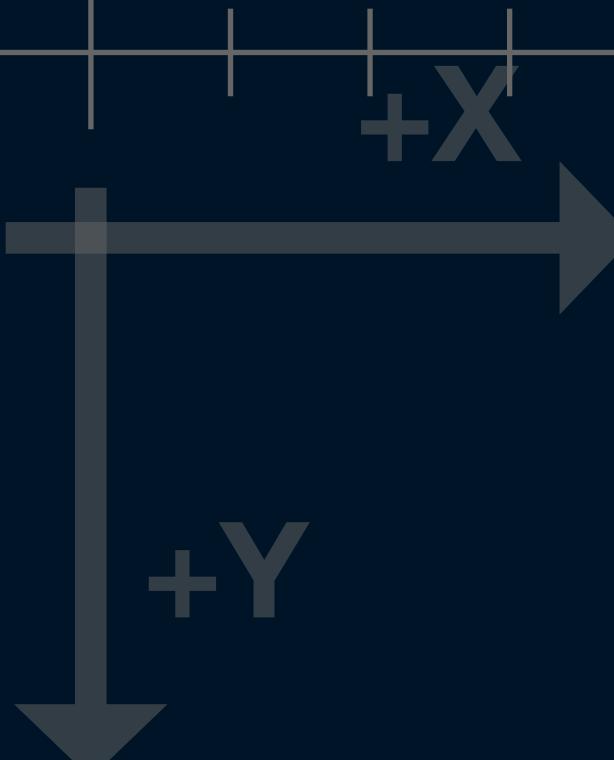
```
Rect {  
    x: 75, y: 110,  
    w: 100, h: 75  
}
```

```
Rect {  
    x: 25, y: 25,  
    w: 125, h: 75  
}
```

Are these Frame or  
Bounds?

# Parent's Coordinate System

Coordinates



Ratnesh Jain

Rect {  
x: 25, y: 25,  
w: 125, h: 75  
}

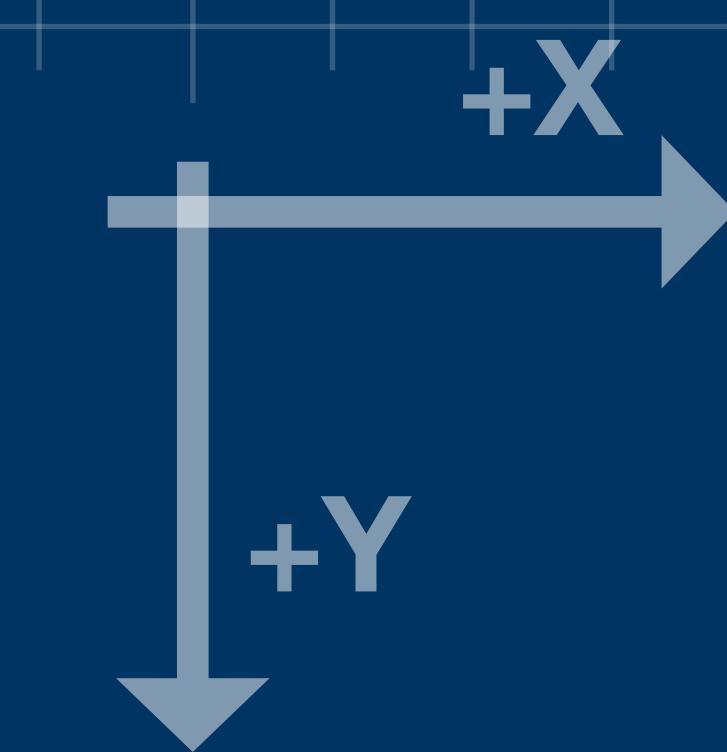
Rect {  
x: 75, y: 125,  
w: 100, h: 75  
}

Rect {  
X: 25, y: 250,  
W: 100, h: 75  
}

Are these Frame or  
Bounds?



# Coordinates



200

300

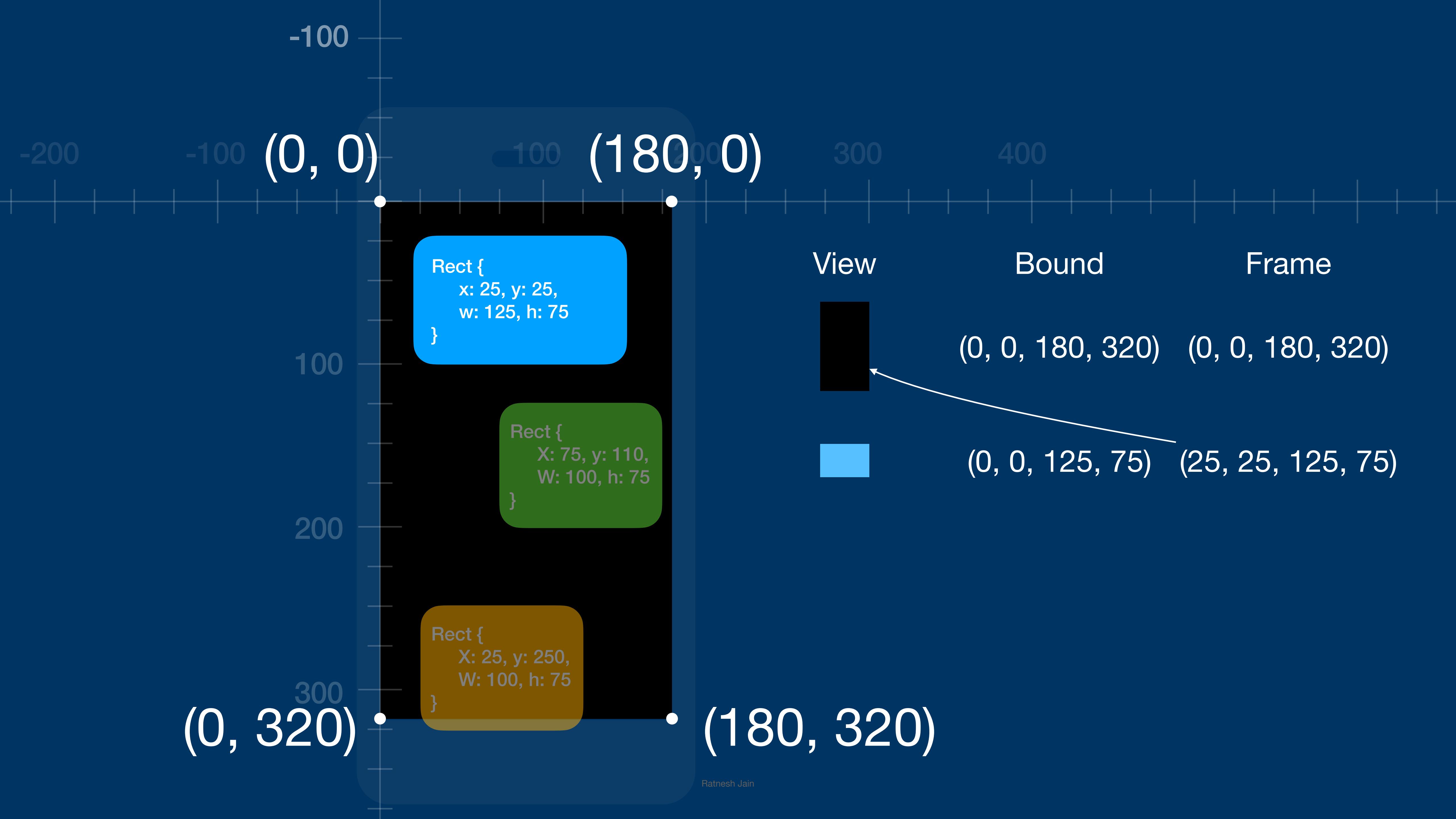
400

Rect {  
x: 25, y: 25,  
w: 125, h: 75  
}

Rect {  
X: 75, y: 110,  
W: 100, h: 75  
}

Rect {  
X: 25, y: 250,  
W: 100, h: 75  
}





(0, 0)

(125, 0)

(0, 75)

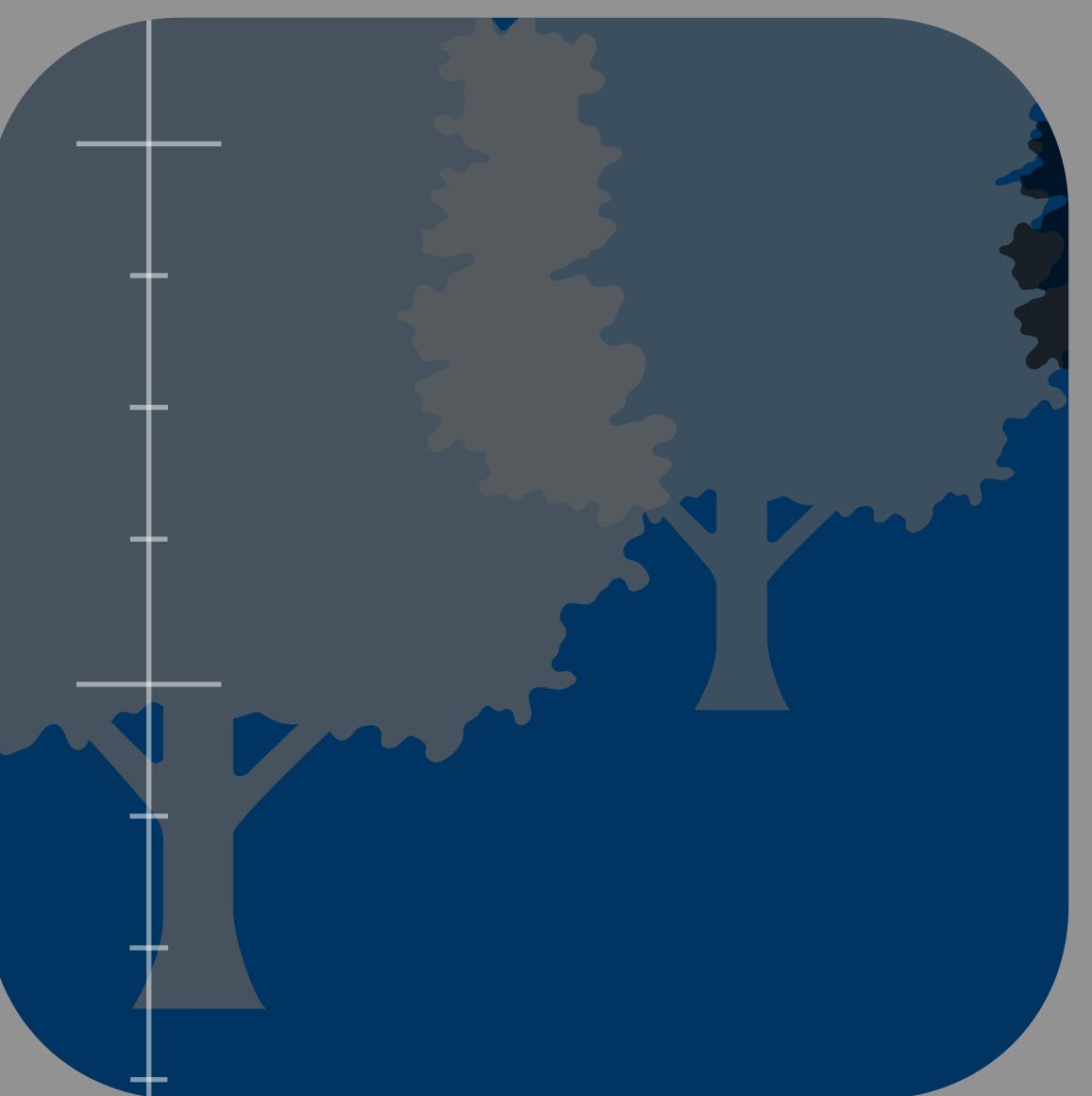
(125, 75)

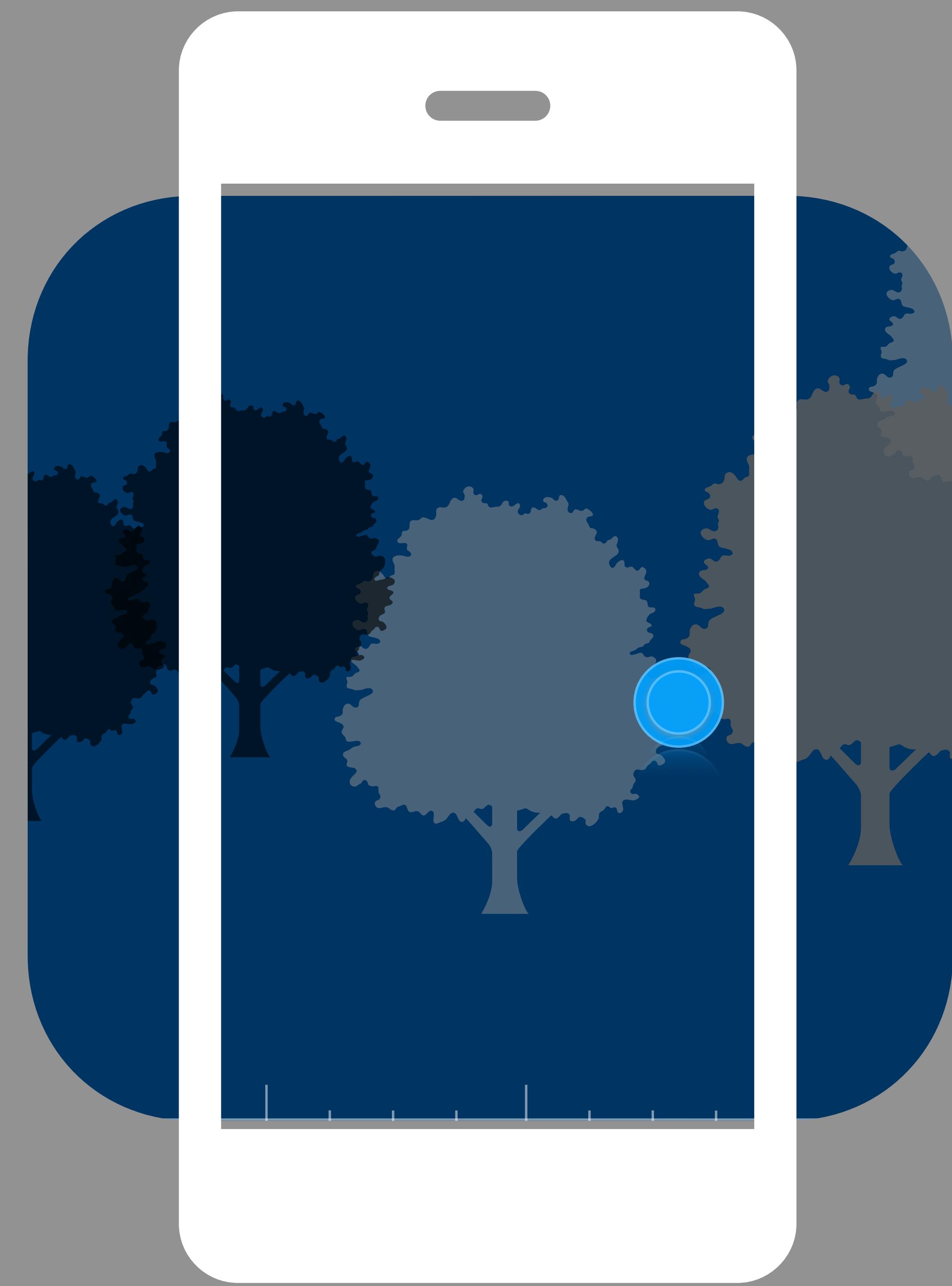
Rect {  
x: 25, y: 25,  
w: 125, h: 75  
}

# Today

- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# Physics

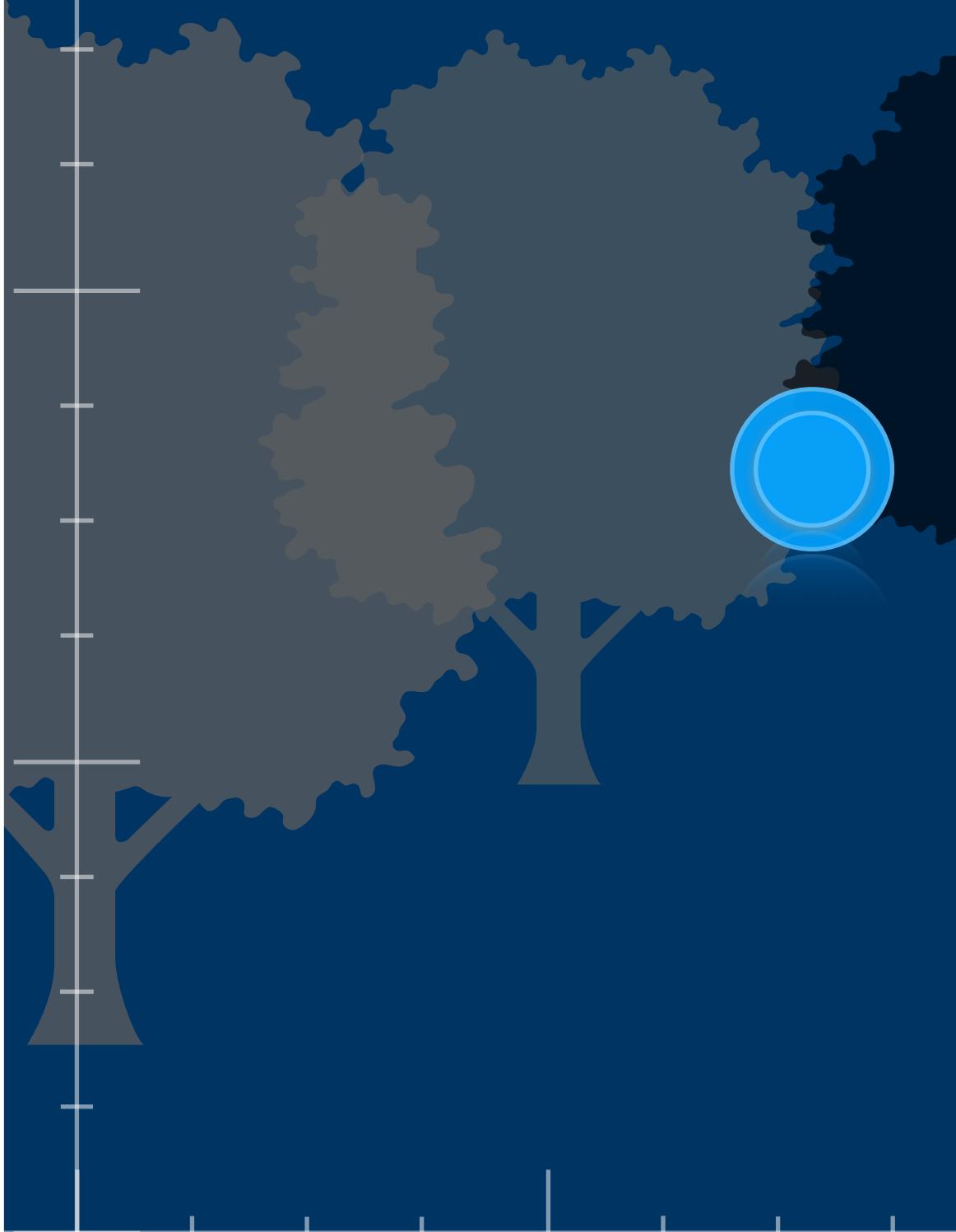




# Physics



# Physics



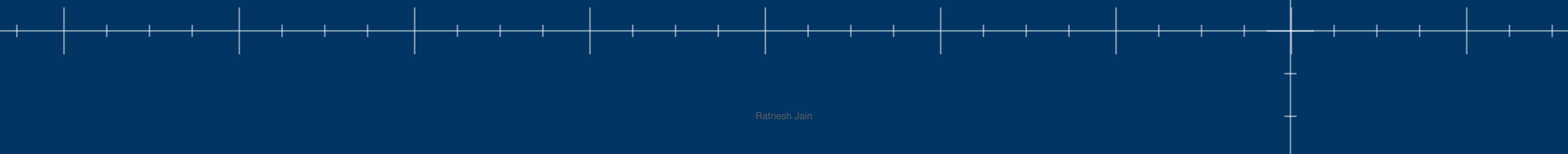
# Physics

## Frame of Reference

# Today

- **Necessity of ScrollView**
- **Why UIScrollView now?**
  - In the time of SwiftUI
- **Coordinates**
  - Cartesian
- **Physics**
  - FoR, Time, Distance, Velocity, Acceleration
- **Some Code**
  - ScrollView

# Time





# Time



# Today

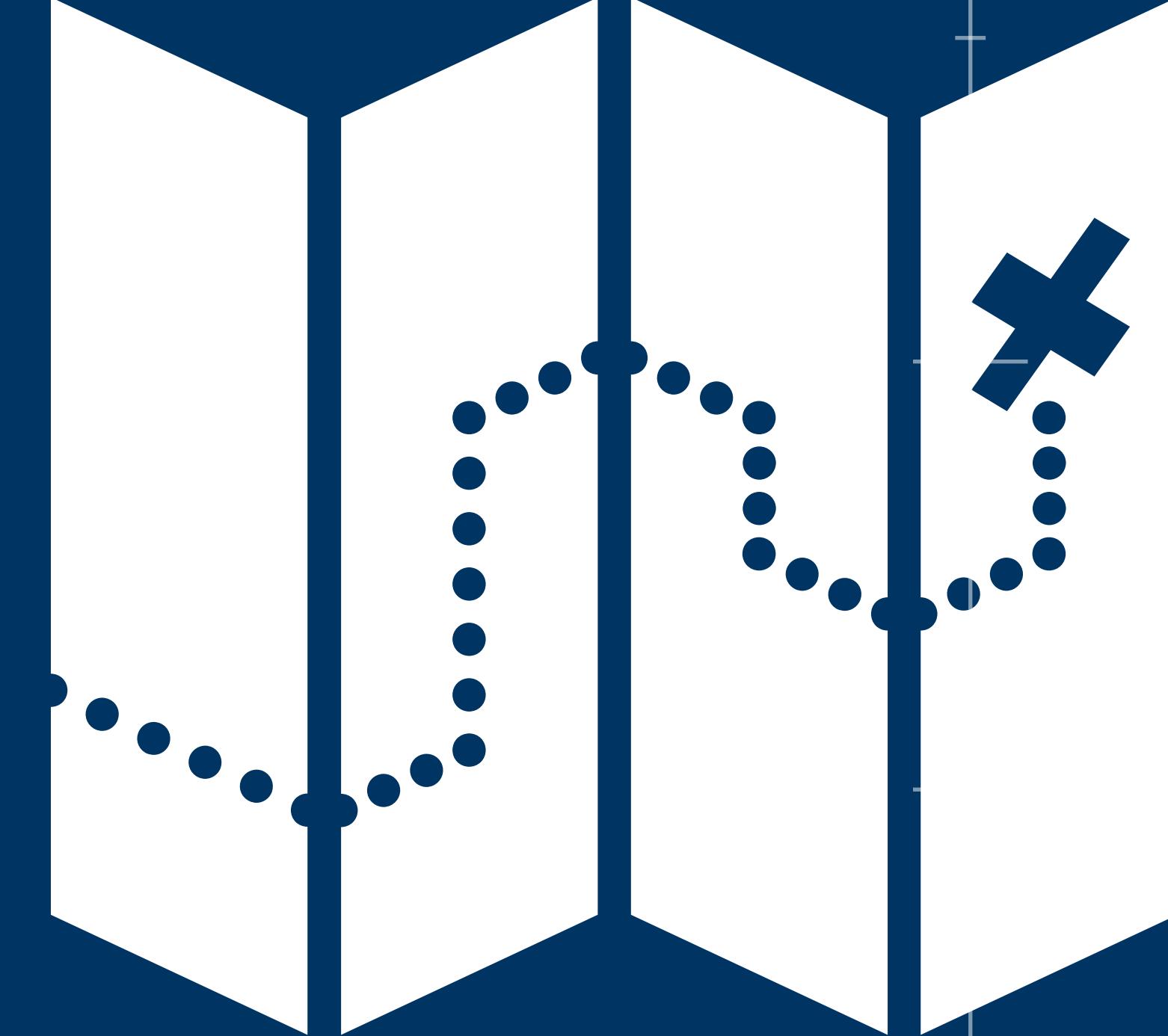
- **Necessity of ScrollView**
- **Why UIScrollView now?**
  - In the time of SwiftUI
- **Coordinates**
  - Cartesian
- **Physics**
  - FoR, Time, Distance, Velocity, Acceleration
- **Some Code**
  - ScrollView

# Distance

$$d = P_2 - P_1$$

$p_2$  = New location

$p_1$  = Old location





# Today

- **Necessity of ScrollView**
- **Why UIScrollView now?**
  - In the time of SwiftUI
- **Coordinates**
  - Cartesian
- **Physics**
  - FoR, Time, Distance, Velocity, Acceleration
- **Some Code**
  - ScrollView

# Velocity

$$v = \frac{\text{distance}}{\text{time}}$$

$$v = \frac{\Delta d}{\Delta t}$$

# Velocity

$$v = \frac{distance}{time}$$
$$v = \frac{\Delta d}{\Delta t} =$$

$$\frac{670px}{5s}$$

# Velocity

*distance*

$$v = \frac{\text{distance}}{\text{time}}$$

$$v = \frac{\Delta d}{\Delta t}$$

$$v \cdot \Delta t = \Delta d$$

# Velocity

*distance*

$$v = \frac{\text{distance}}{\text{time}}$$

$$v = \frac{\Delta d}{\Delta t}$$

$$\Delta d = v \cdot \Delta t$$

# Today

- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# Acceleration

$$a = \frac{\text{velocity}}{\text{time}}$$

$$a = \frac{\Delta v}{\Delta t}$$

# Acceleration

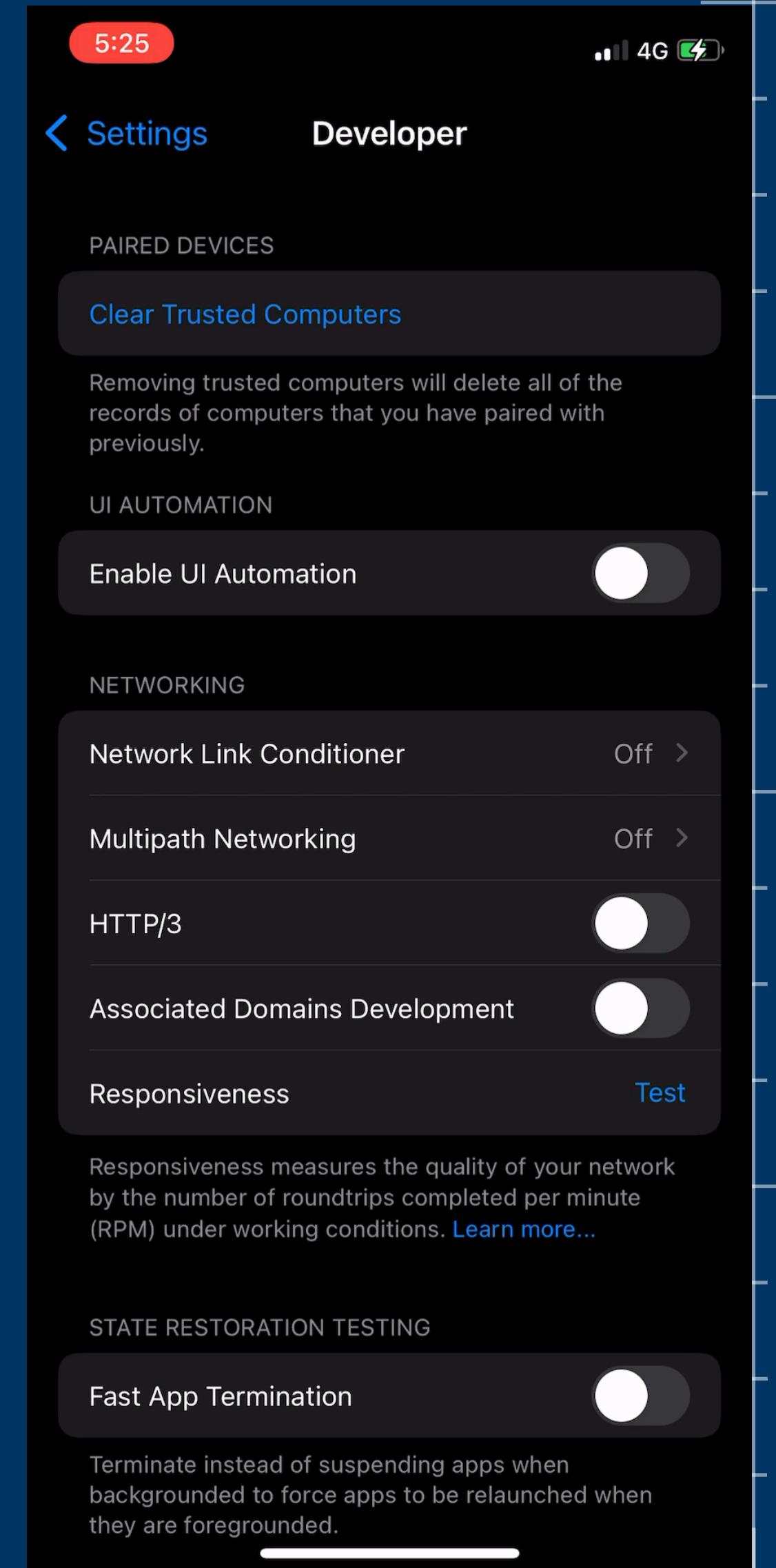
$$a = \frac{velocity}{time}$$
$$a = \frac{\Delta v}{\Delta t}$$



# Acceleration

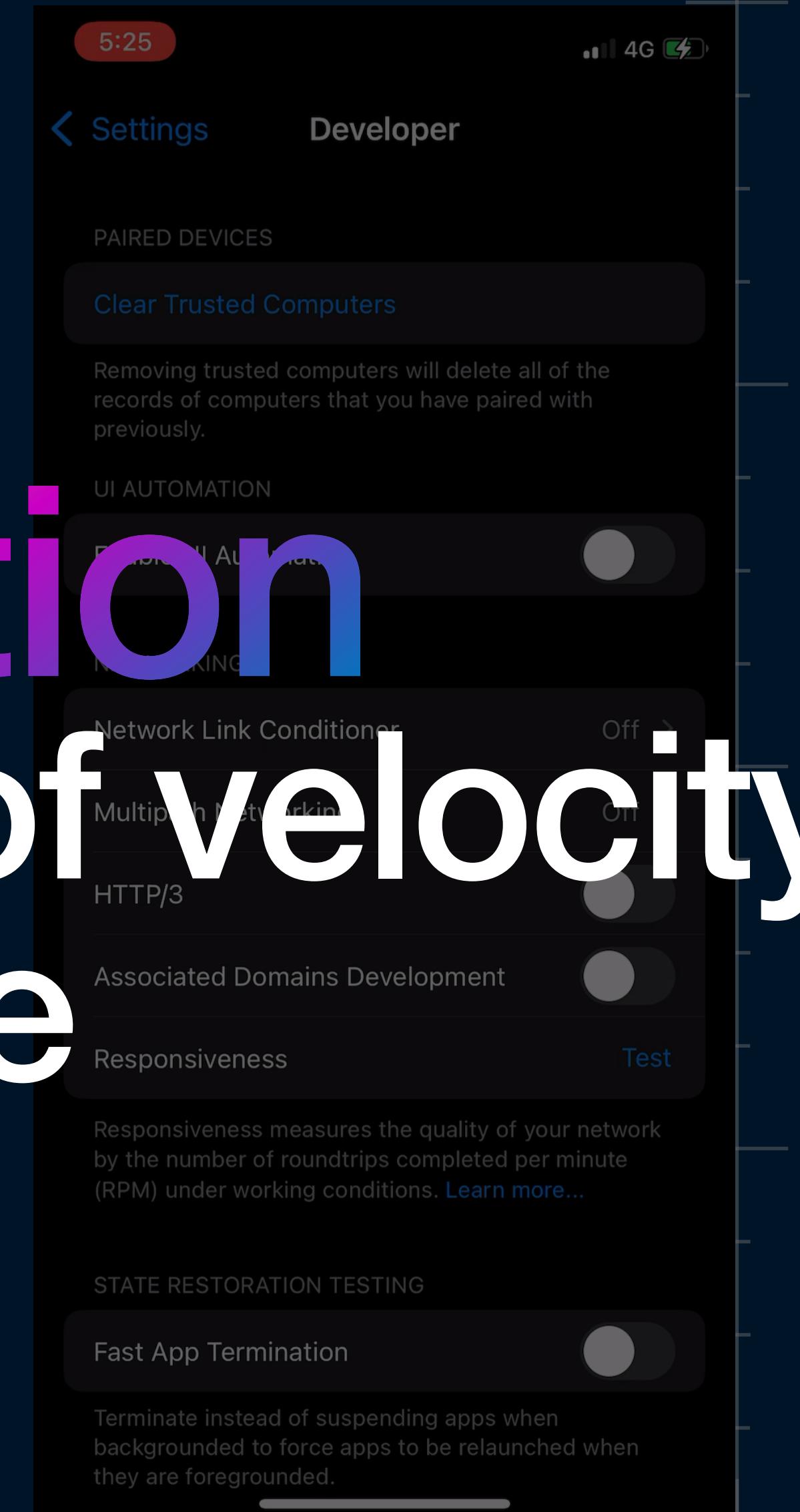
$$acceleration = + \frac{\Delta v}{\Delta t}$$

$$deceleration = - \frac{\Delta v}{\Delta t}$$



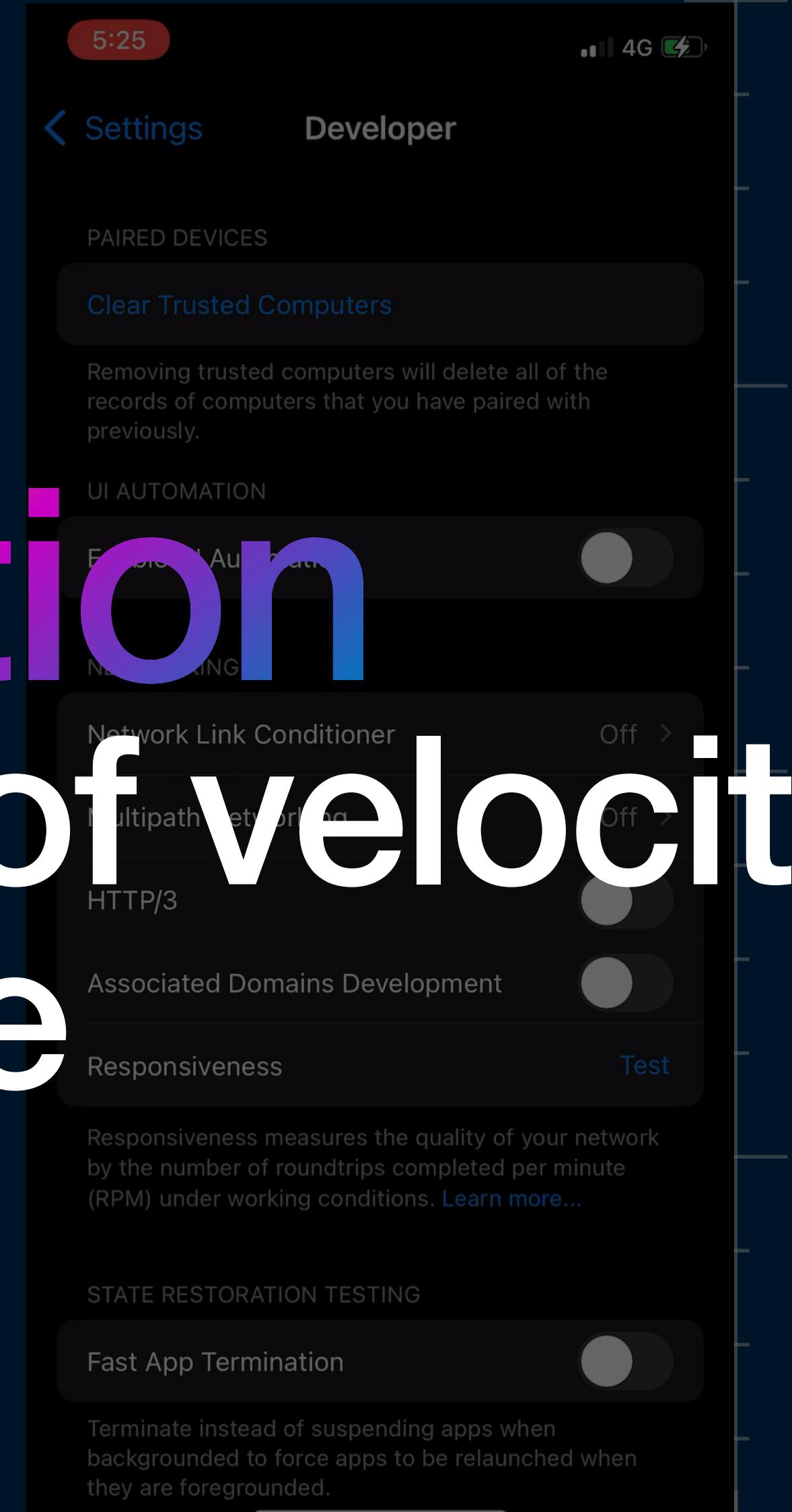
# Acceleration

*acceleration* = **Acceleration**  
Positive change of velocity  
*deceleration* = - **Over time**

$$\text{acceleration} = \frac{\Delta v}{\Delta t}$$
$$\text{deceleration} = - \frac{\Delta v}{\Delta t}$$


# Acceleration

*acceleration* = **Deceleration**  
Negative change of velocity  
*deceleration* = 
$$\frac{\Delta v}{\Delta t}$$
 Over time



# Acceleration

*acceleration* = **Deceleration**  
Velocity is decreasing  
$$deceleration = -\frac{\Delta v}{\Delta t}$$



# Acceleration

$$\text{acceleration} = + \frac{\Delta v}{\Delta t}$$

$$\text{deceleration} = - \frac{\Delta v}{\Delta t}$$

# Acceleration

$$deceleration = - \frac{\Delta v}{\Delta t}$$

# Acceleration

$$decelerationRate = \frac{\Delta v}{\Delta t}$$

$$\Delta t = 1 \text{ Frame}$$

# Acceleration

$$decelerationRate = \frac{\Delta v}{\Delta t}$$

$$\Delta t = 1 \text{ Frame}$$

$$decelerationRate = \Delta v$$

# Acceleration

*decelerationRate =  $\Delta v$*

*decelerationRate = 0.95*

*v2 = v1 × decelerationRate*

# Acceleration

*decelerationRate = 0.95*

$v2 = v1 \times decelerationRate$

Start velocity	Calculation	New velocity
100	$100 \times 0.95$	95
95	$95 \times 0.95$	90.25
90.25	$90 \times 0.95$	85.7375
85.7375	$85.7375 \times 0.95$	81.45

# Acceleration

*decelerationRate = 0.95*

$v_f = v_i \times \text{decelerationRate}$

# Deceleration

Velocity is decreasing  
over time by 5%

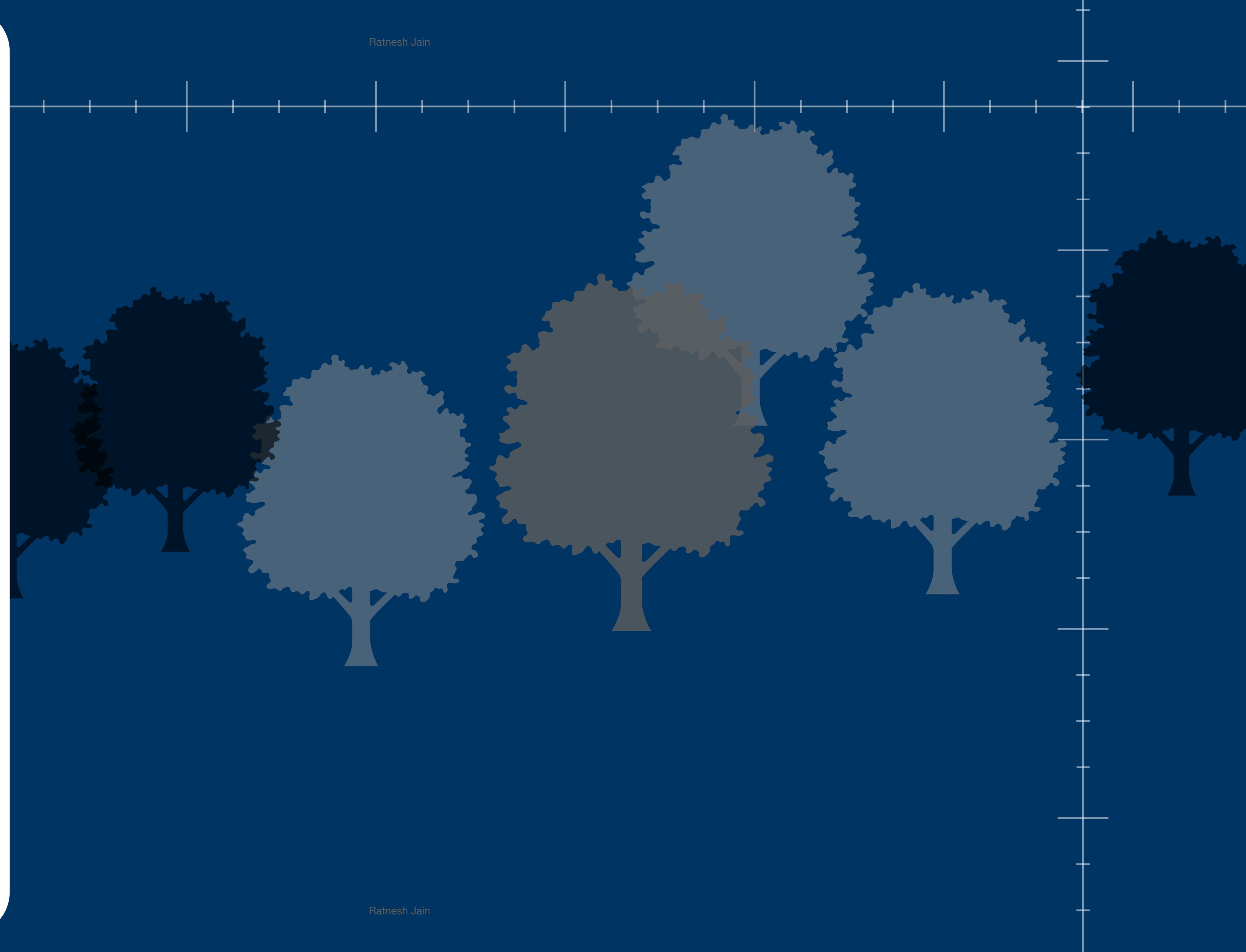
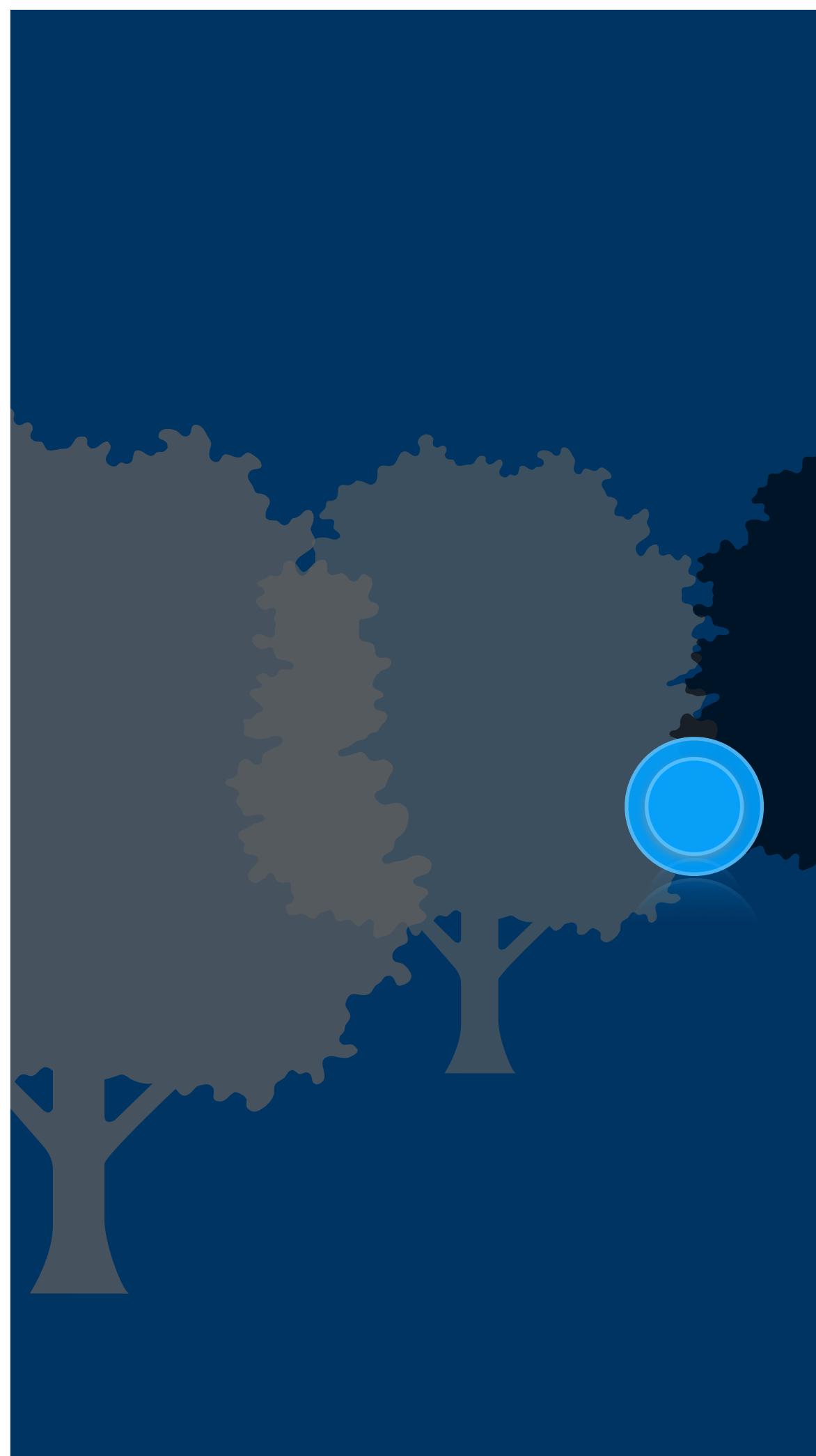
Start Velocity	Calculation	New Velocity
100	$100 \times 0.95$	95
95	$95 \times 0.95$	90.25
90.25	$90.25 \times 0.95$	85.7375
85.7375	$85.7375 \times 0.95$	81.45

# Today

- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# ScrollView Vocabulary

Ratnesh Jain

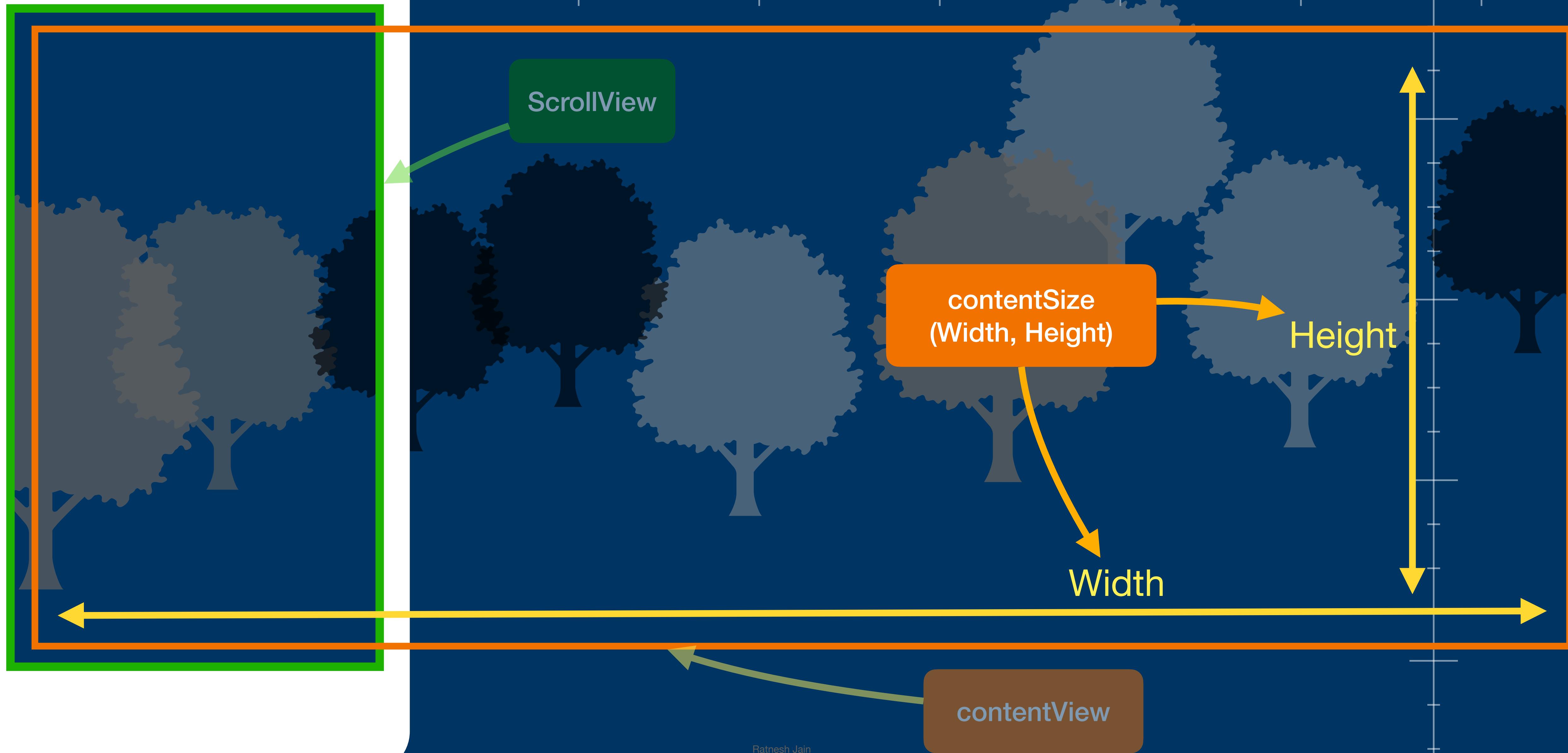


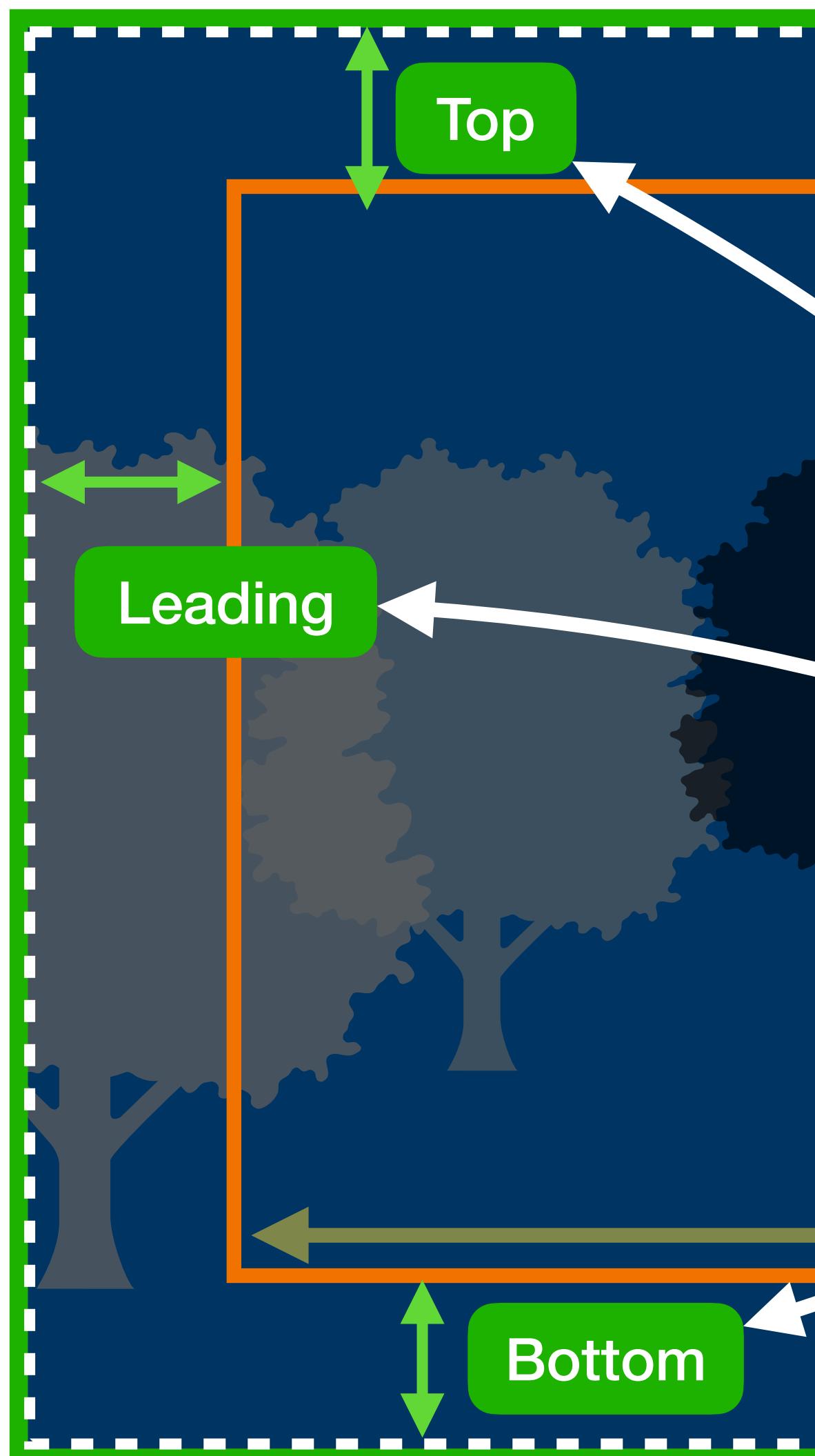
Ratnesh Jain

ScrollView

ScrollView

contentView





ContentInset  
(Top, Leading, Bottom, Trailing)

contentView

ScrollView

contentSize  
(Width, Height)

Trailing

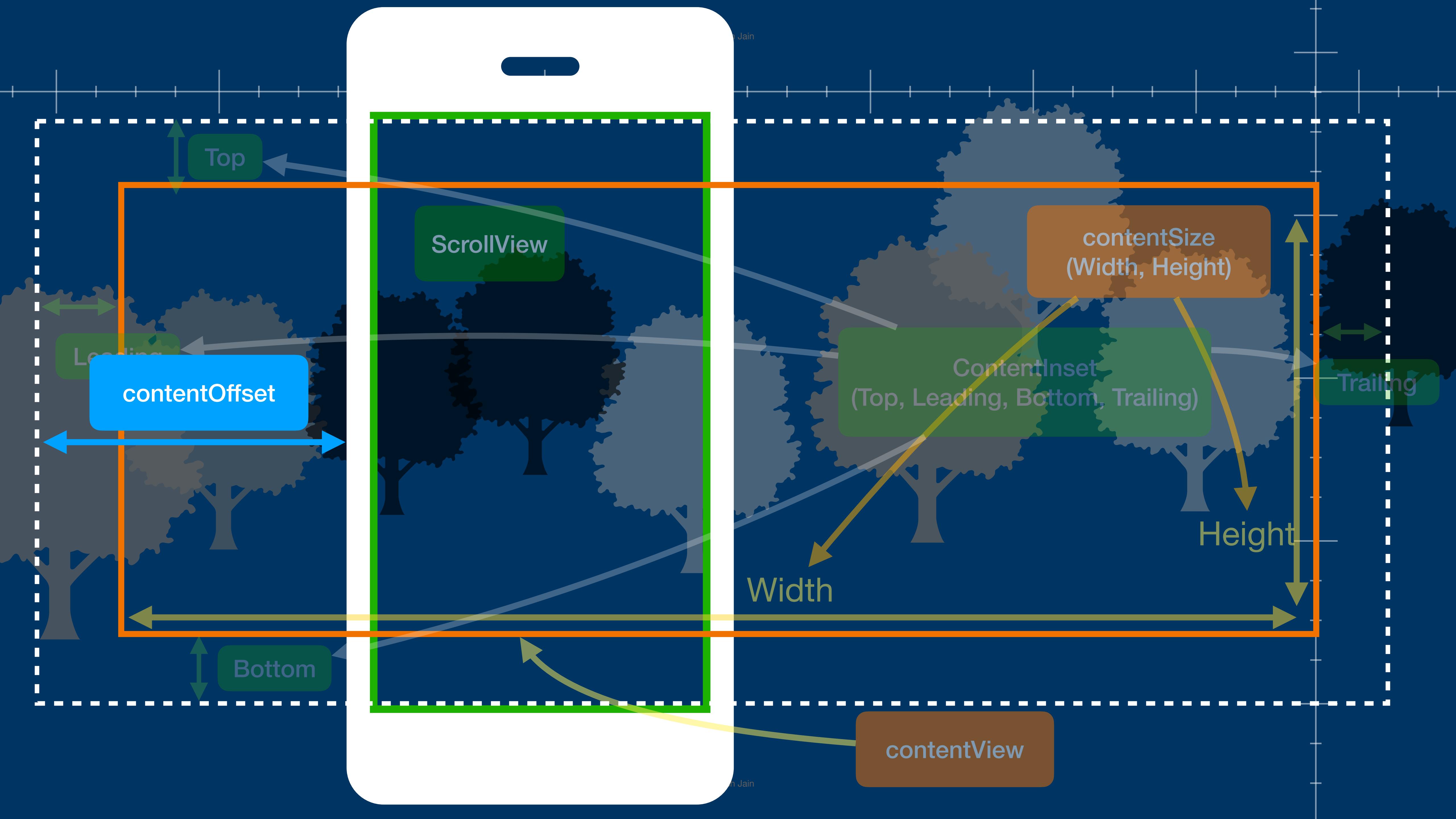
Width

Height

Bottom

Top

Leading



bounds.origin.x = 280

Top

Leading

contentOffset

Bottom

ScrollView

contentSize  
(Width, Height)

ContentInset  
(Top, Leading, Bottom, Trailing)

Height

Width

contentView

# ScrollView Vocabulary

- ScrollView (ViewPort)
- contentView
- contentSize
- contentInset
- contentOffset

# Some Code

- Create a ScrollView from Zero
- Deceleration.



# Today

- Necessity of ScrollView
- Why UIScrollView now?
  - In the time of SwiftUI
- Coordinates
  - Cartesian
- Physics
  - FoR, Time, Distance, Velocity, Acceleration
- Some Code
  - ScrollView

# References

- **Understanding ScrollView from Objc.io**
  - <https://www.objc.io/issues/3-views/scroll-view/>
- **Understanding ScrollView by Ole Begemann**
  - <https://oleb.net/blog/2014/04/understanding-uiscrollview/>

# Thank you

