App Store Interactive Transition Revealed

Wirawit Rueopas

(What I've learnt after trying to implement)

App Store Interactive Transition

Wirawit Rueopas

Demo

Why custom interaction?

Why custom interaction?

Delightful

Why custom interaction?

Delightful

Suit with context

Why custom interaction?

Delightful

Guide user

Suit with context

Why write one yourself?

Why write one yourself?

Less code

Why write one yourself?

Less code

More control

Overview

Overview

Basics

Overview

Basics

5 Phases of Interaction

Overview

Basics

5 Phases of Interaction

Deep dive

Basics

```
let vc = MyViewController()

present(vc)
```

Basics

```
let vc = MyViewController()

present(vc)
```

Basics

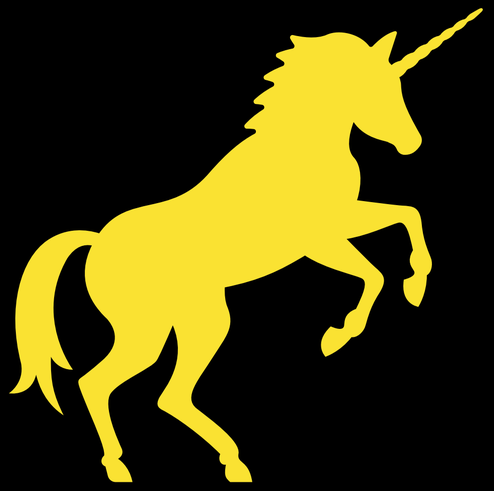
```
let vc = MyViewController()

present(vc)
```

Basics

```
let vc = MyViewController()

present(vc)
```

Basics

```
let vc = MyViewController()

present(vc)
```



MAGIC!

Custom Presentation and Transition

```
let vc = MyViewController()

present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()
```

```
present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()
```

```
present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()
```

?

```
present(vc)
```

Custom Presentation and Transition

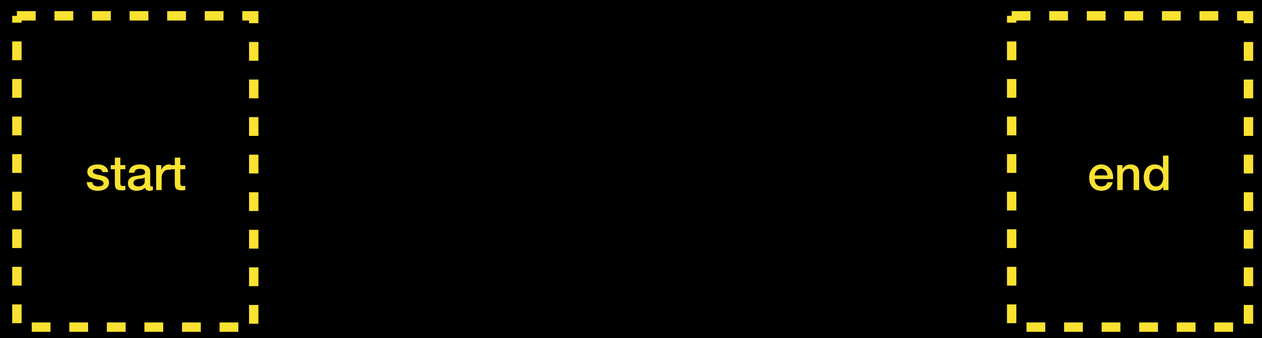
```
let vc = MyViewController()


vc.modalPresentationStyle = …
vc.modalTransitionStyle = …



present(vc)
```

Custom Presentation and Transition

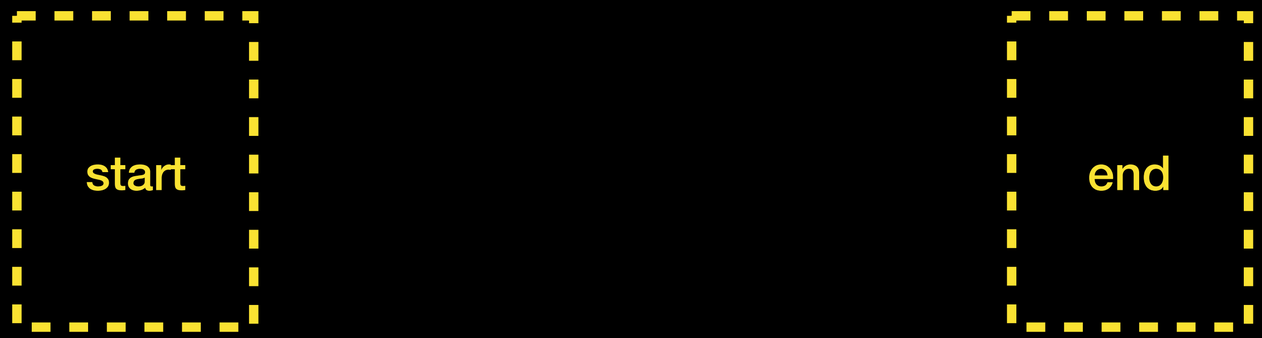
```
let vc = MyViewController()


vc.modalPresentationStyle = …
vc.modalTransitionStyle = …



present(vc)
```

Presentation vs Transition

Presentation vs Transition

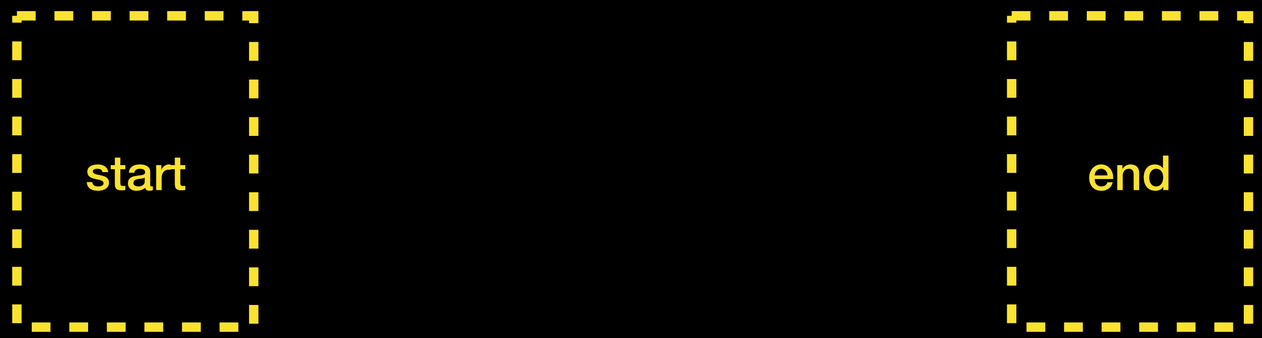


Presentation vs Transition



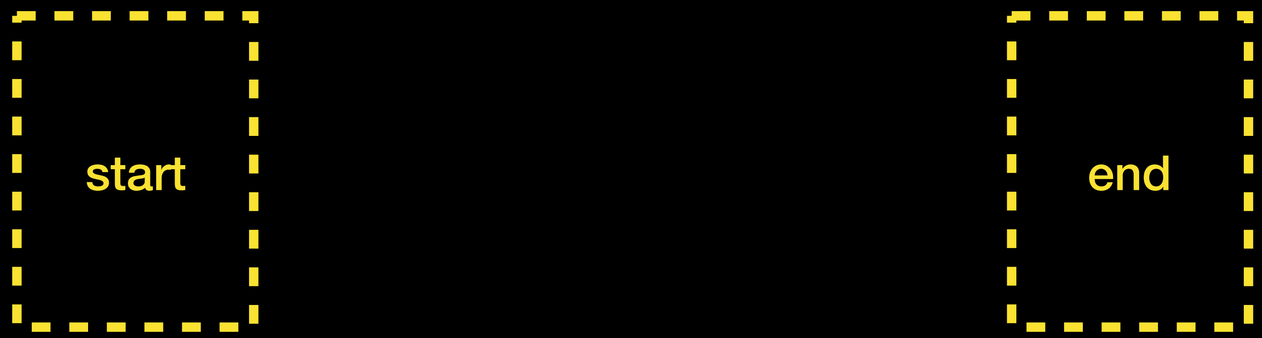
- Layout

Presentation vs Transition



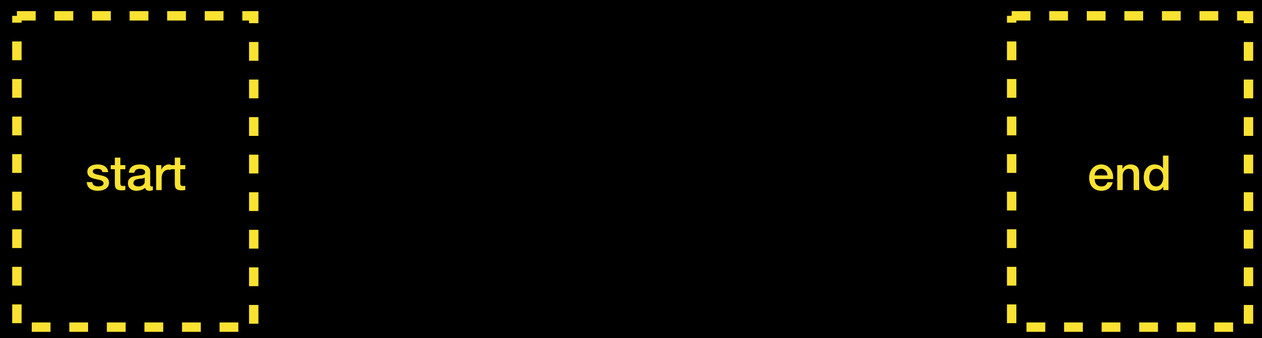
- Layout
- Backdrop view (Chrome)

Presentation vs Transition

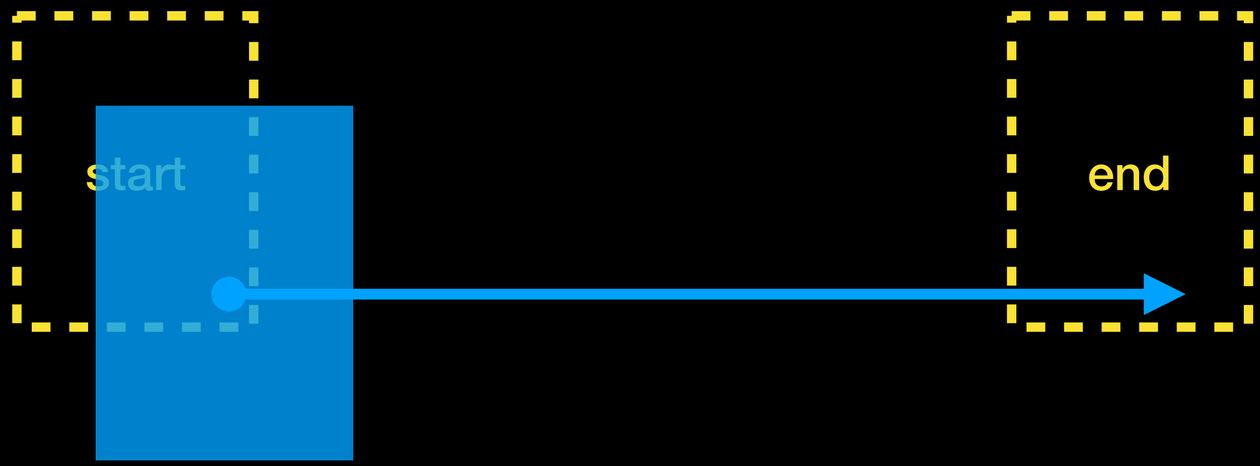


- Layout
- Backdrop view (Chrome)
- etc.

Presentation vs Transition



Presentation vs Transition



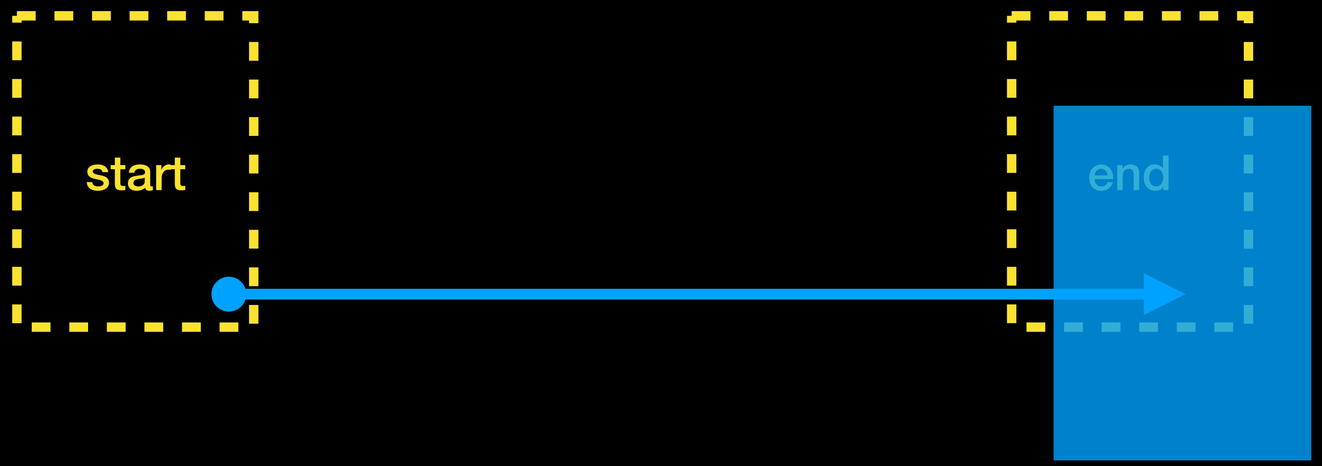
Presentation vs Transition

Animation

Presentation vs Transition

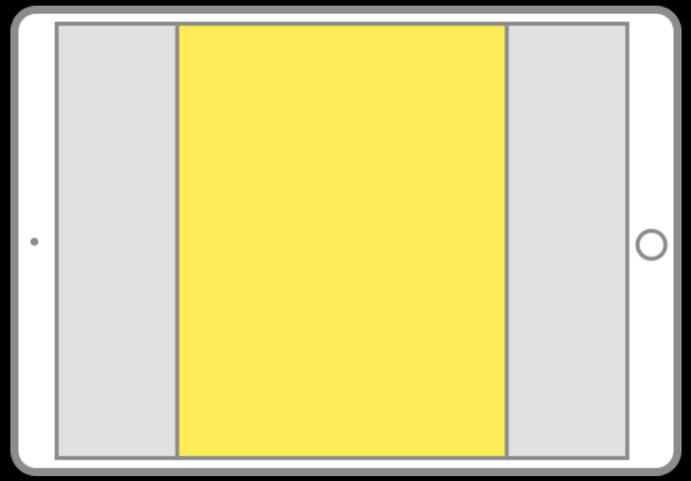
Animation

Presentation vs Transition

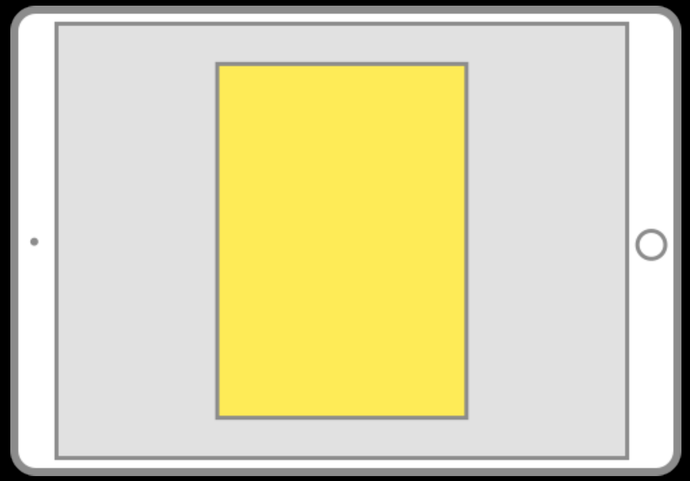


UIModalPresentationStyle

.fullScreen

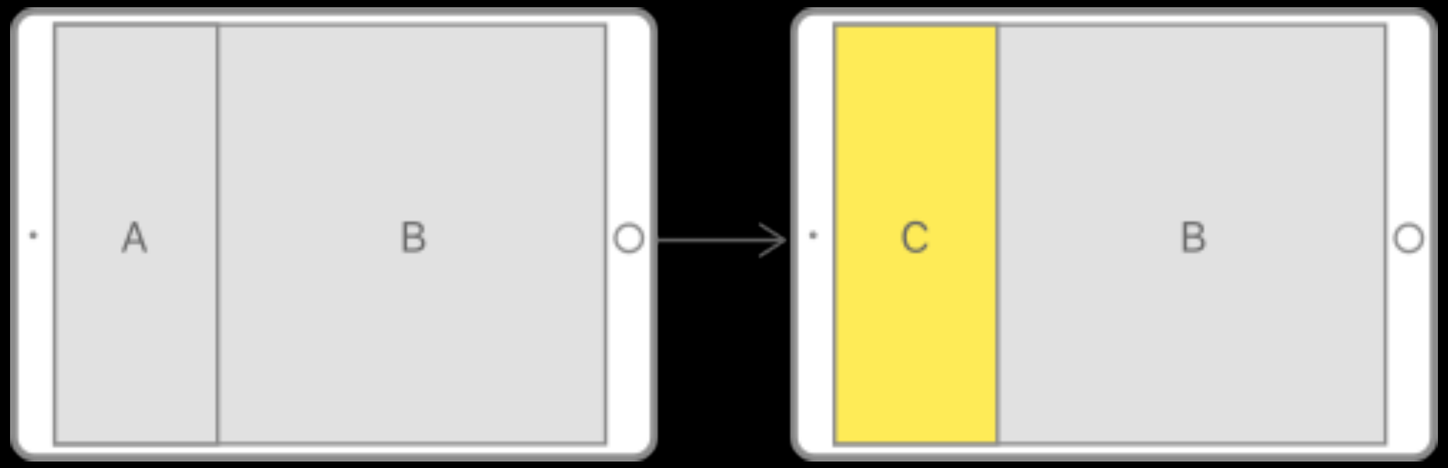


.pageSheet



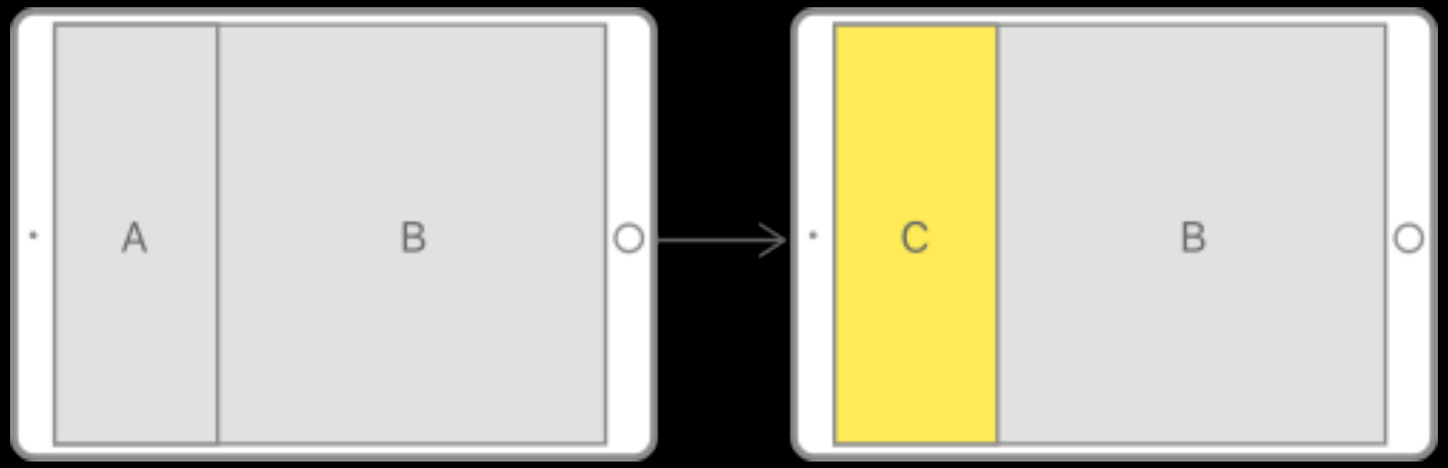
.formSheet

Presentation Styles



Current context

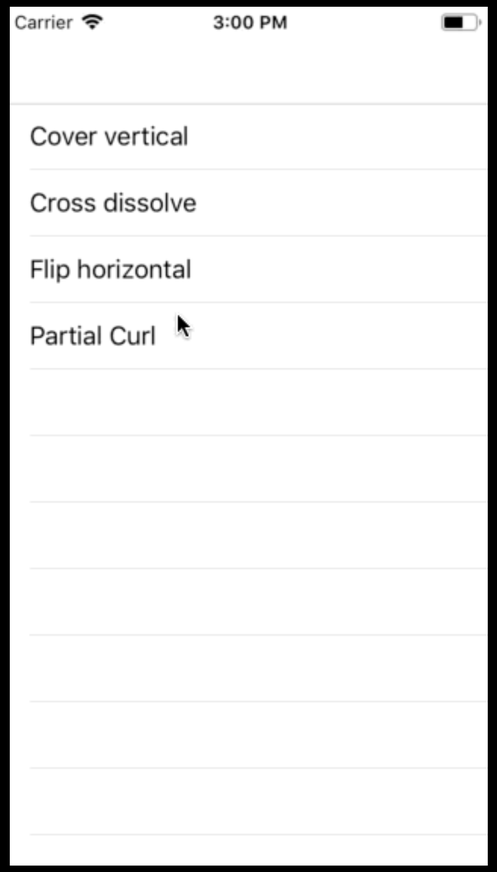
Presentation Styles



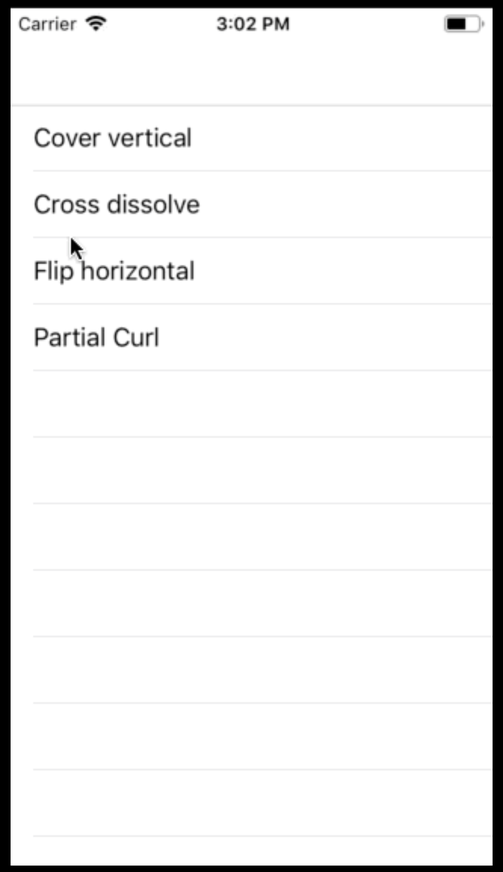
Current context

```
A.definesPresentationContext = true
```

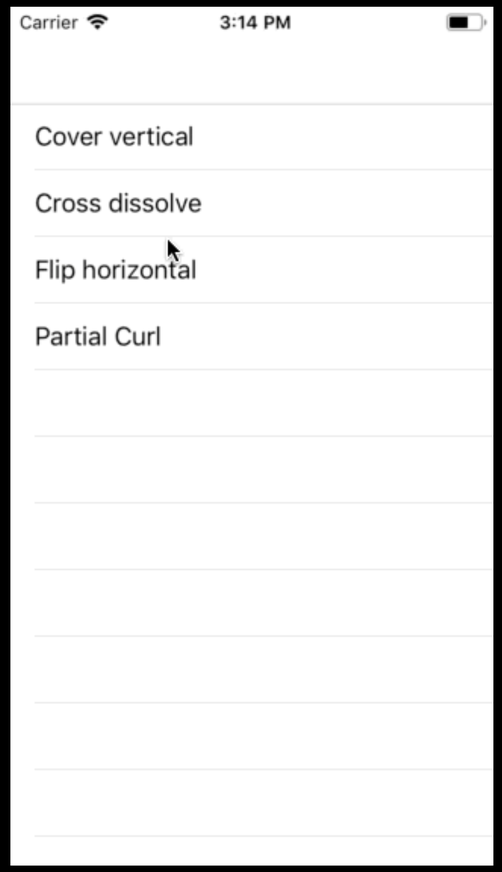
UIModalTransitionStyle



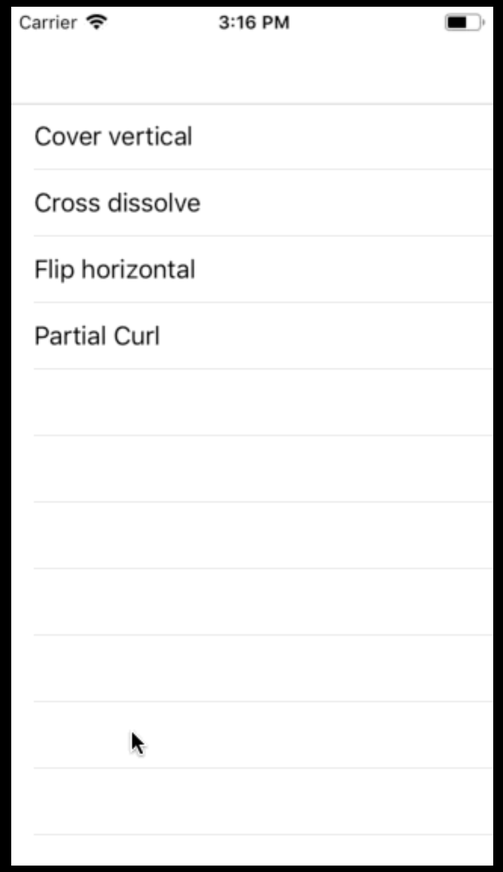
.coverVertical



.crossDissolve

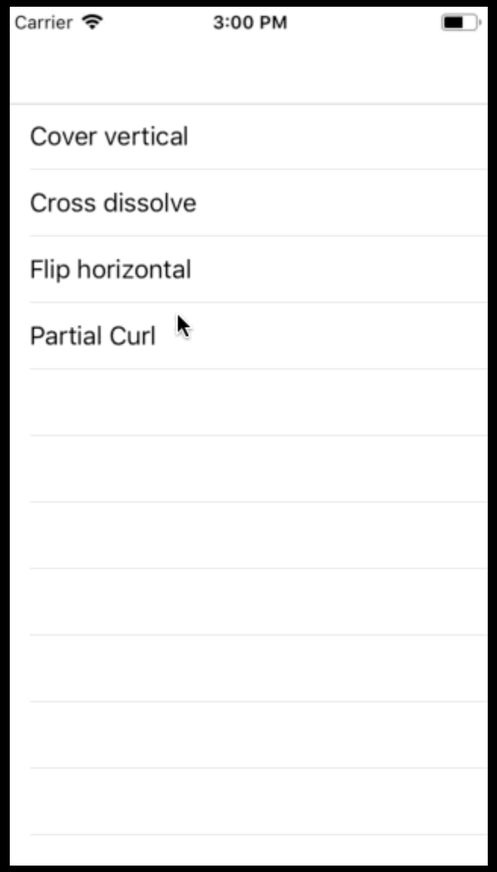


.flipHorizontal

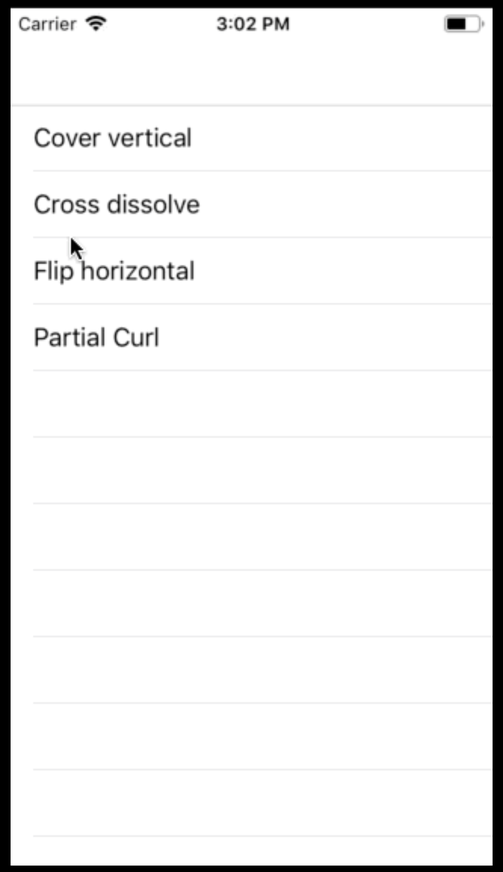


.partialCurl

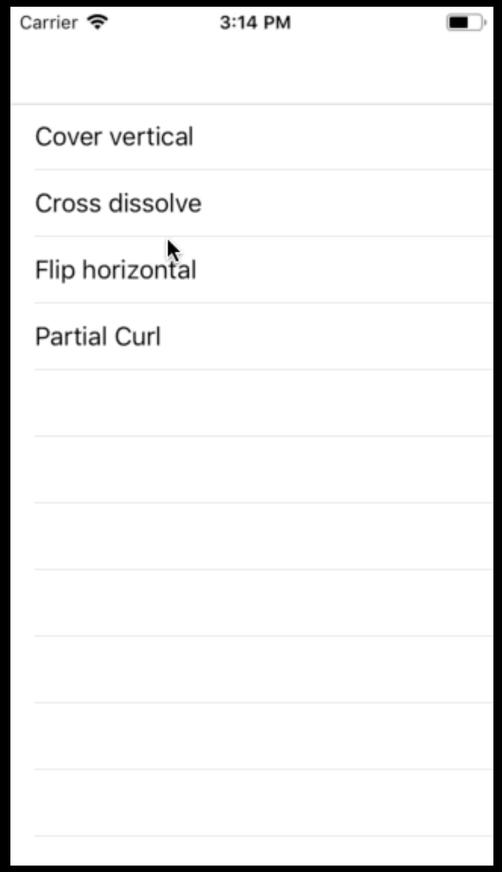
UIModalTransitionStyle



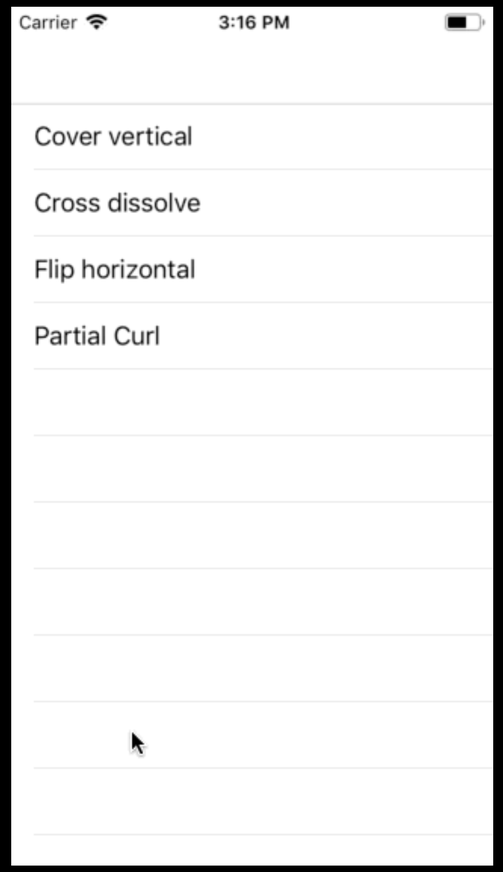
.coverVertical



.crossDissolve



.flipHorizontal



.partialCurl

Custom Presentation and Transition

```
let vc = MyViewController()

vc.modalPresentationStyle = .custom



present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()


vc.modalPresentationStyle = .custom
vc.modalTransitionStyle = .custom?



present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()


vc.modalPresentationStyle = .custom
vc.modalTransitionStyle = .custom



present(vc)
```

Custom Presentation and Transition

```
let vc = MyViewController()


vc.modalPresentationStyle = .custom
vc.


present(vc)
```

Custom Presentation and Transition

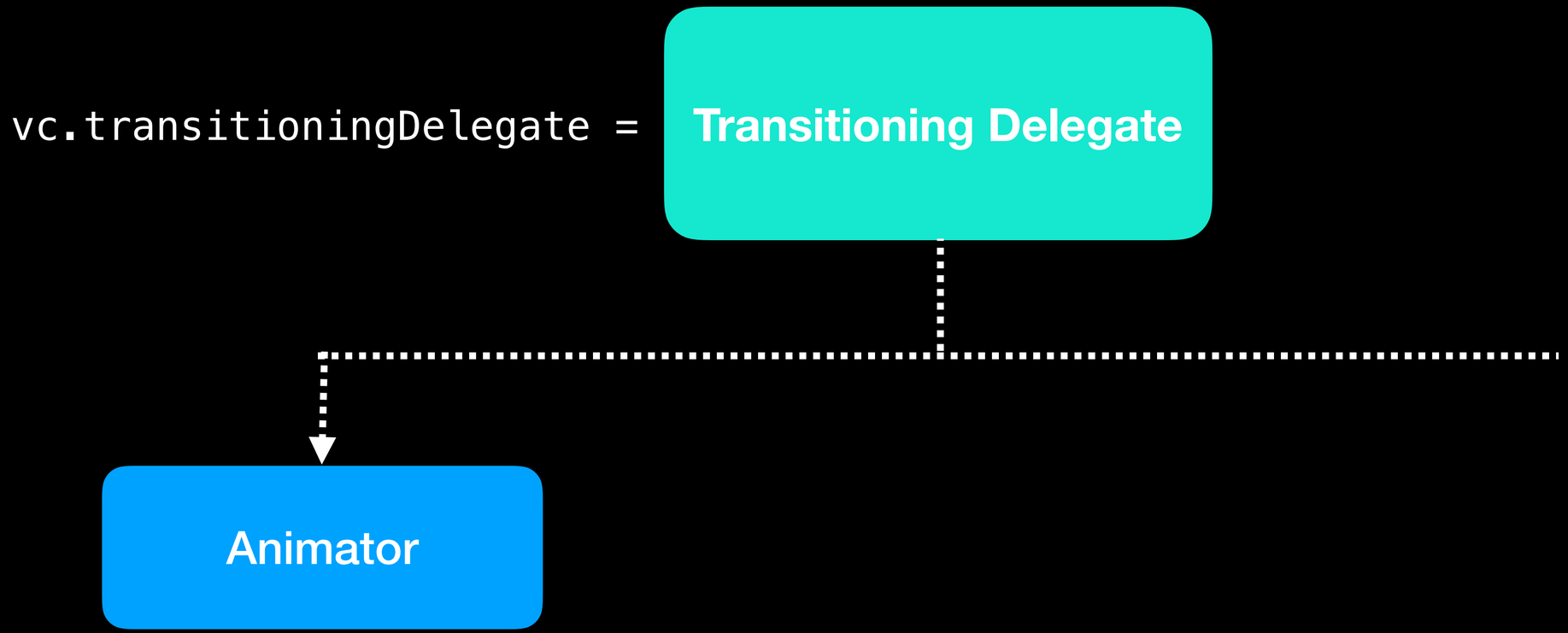
```
let vc = MyViewController()


vc.modalPresentationStyle = .custom
vc.transitioningDelegate = transitionManager


present(vc)
```

Custom Presentation and Transition

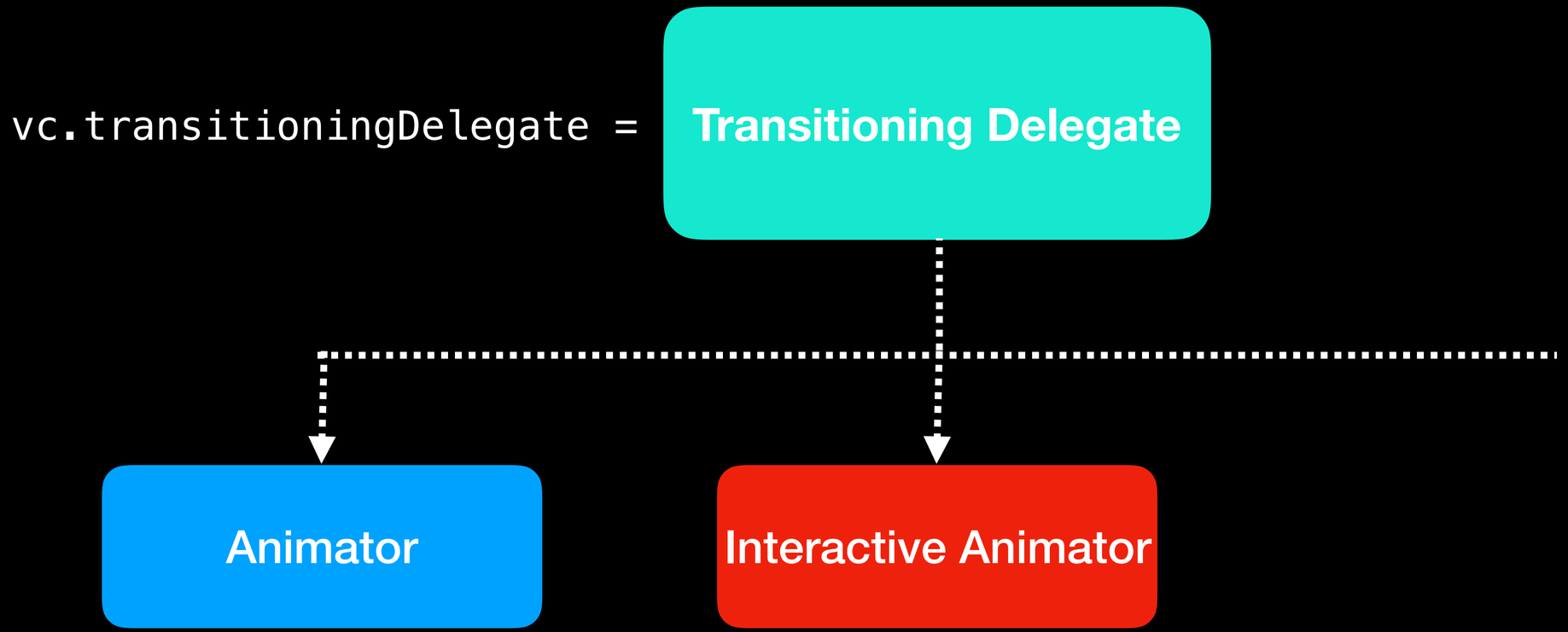
```
vc.transitioningDelegate =
```

Transitioning Delegate

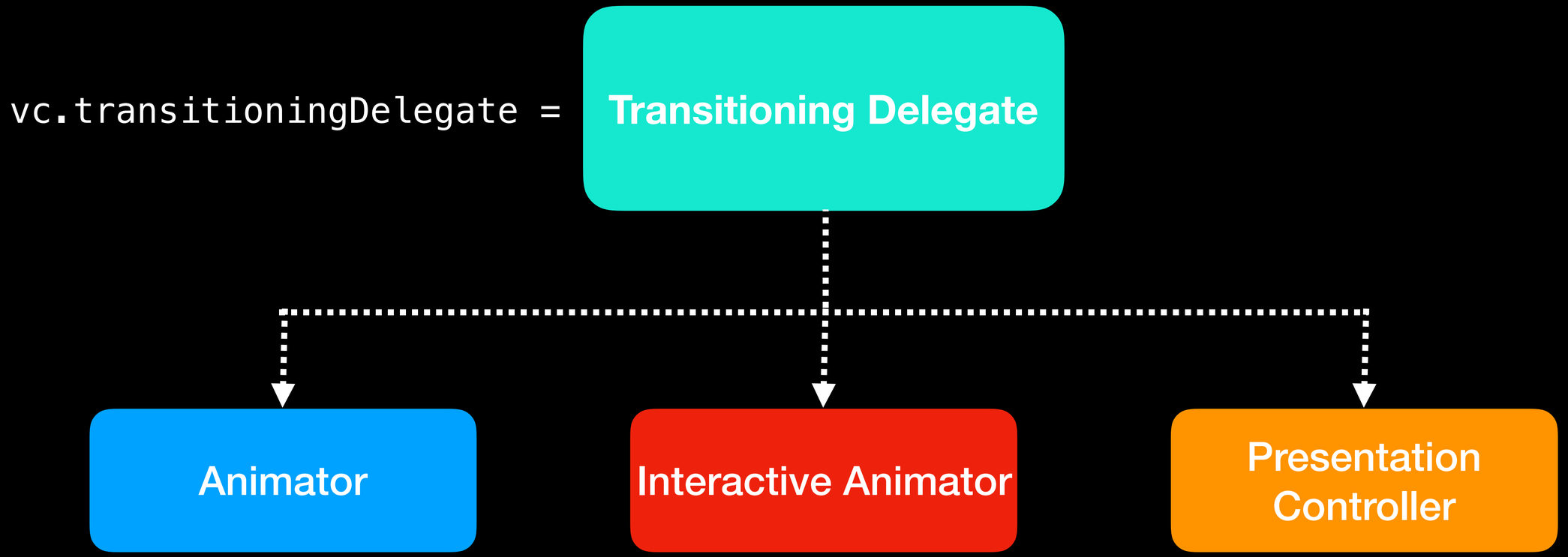
Custom Presentation and Transition



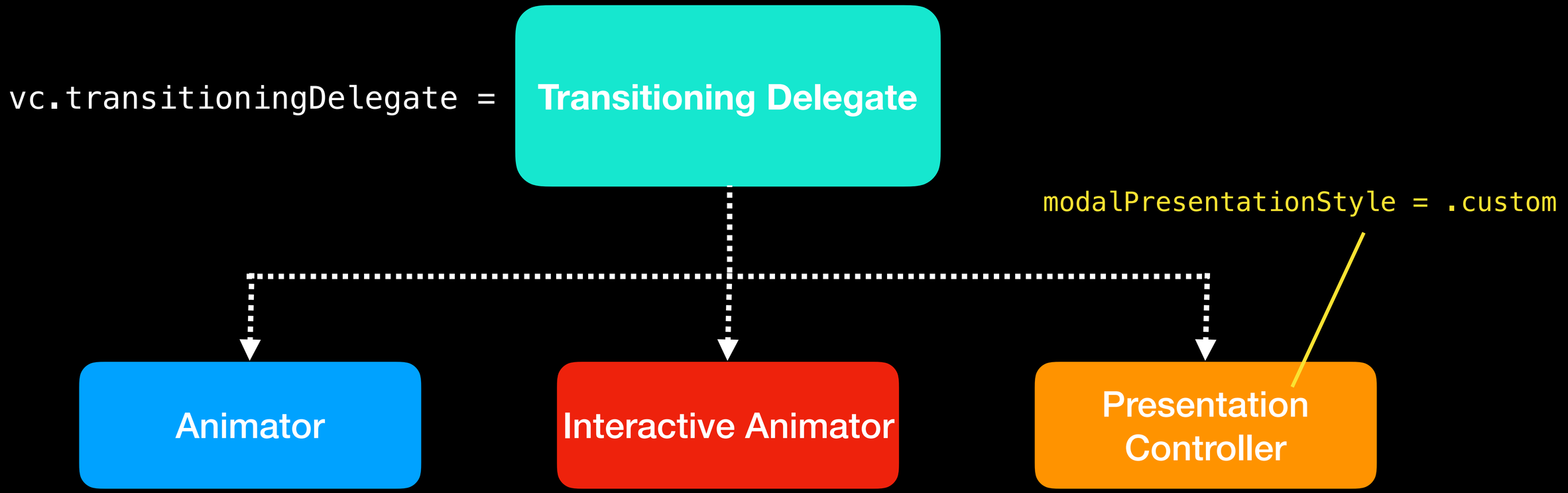
Custom Presentation and Transition



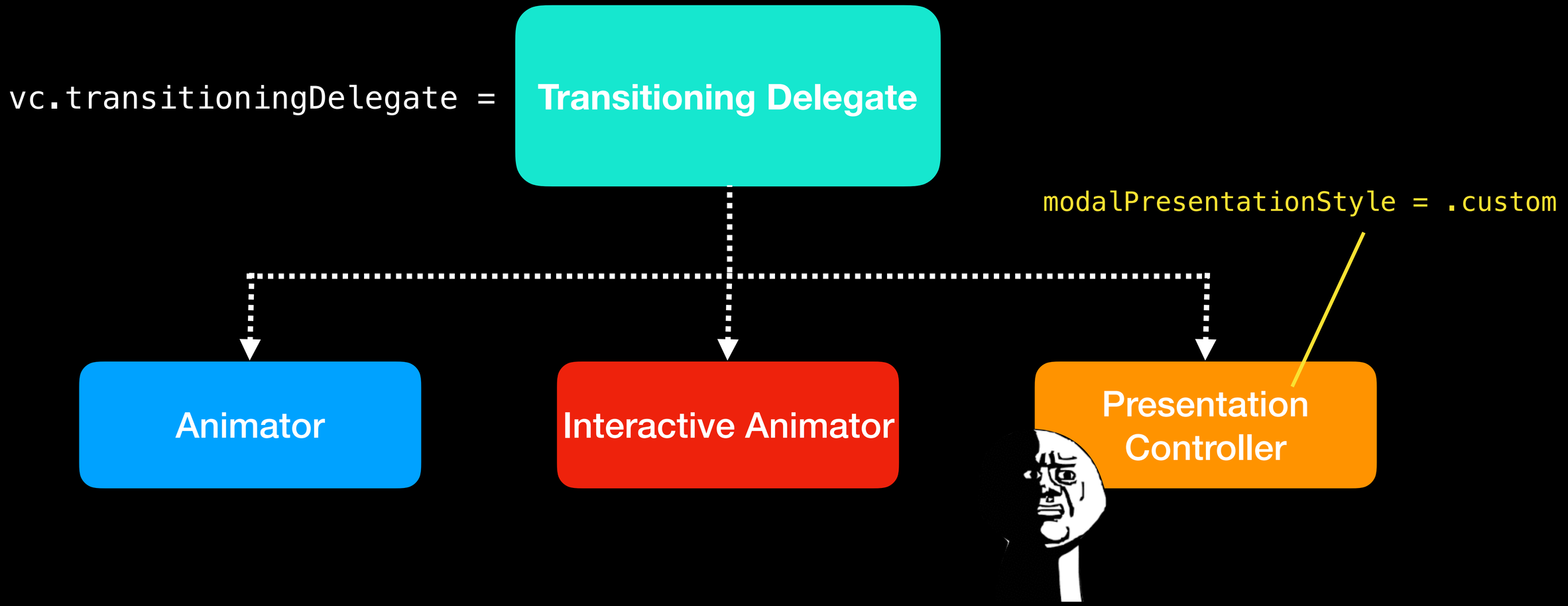
Custom Presentation and Transition



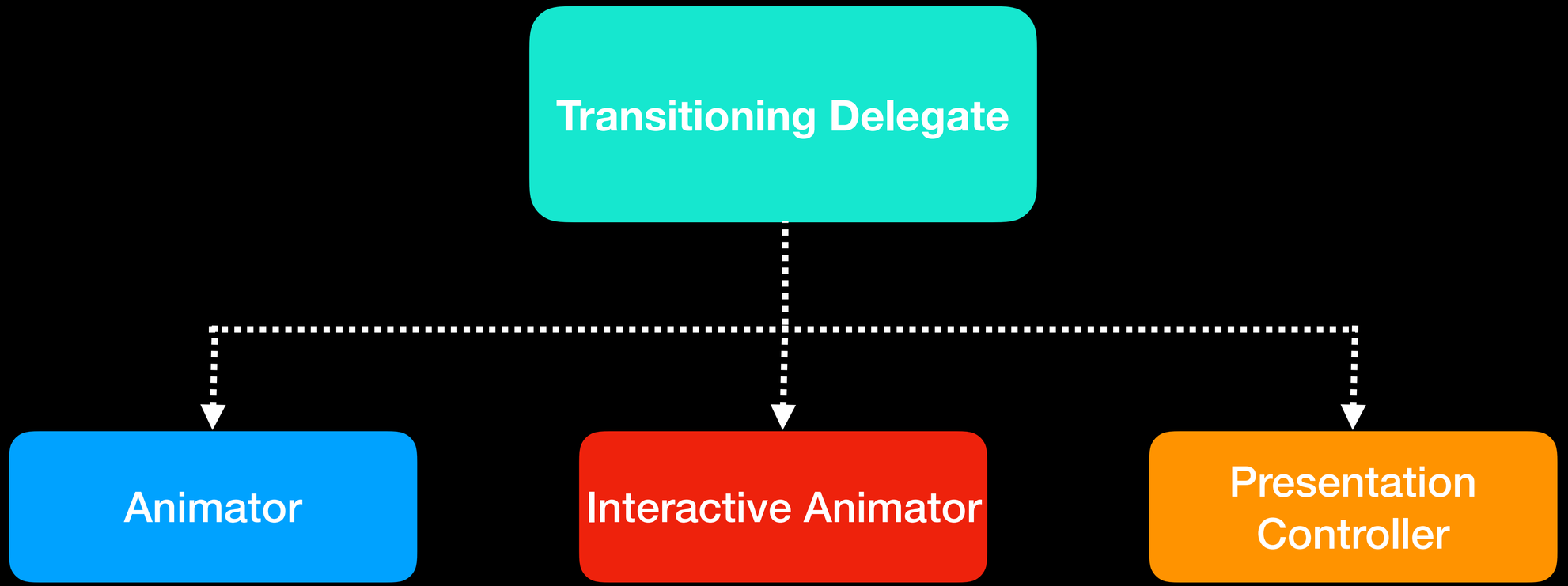
Custom Presentation and Transition

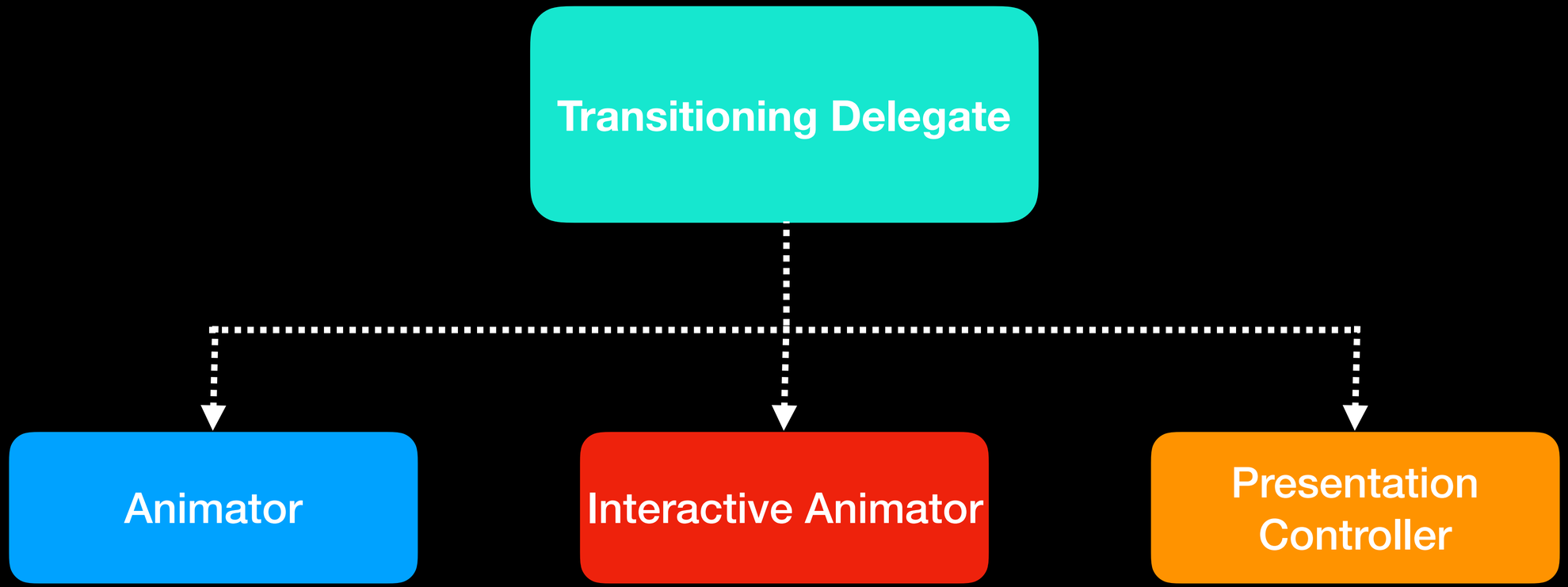


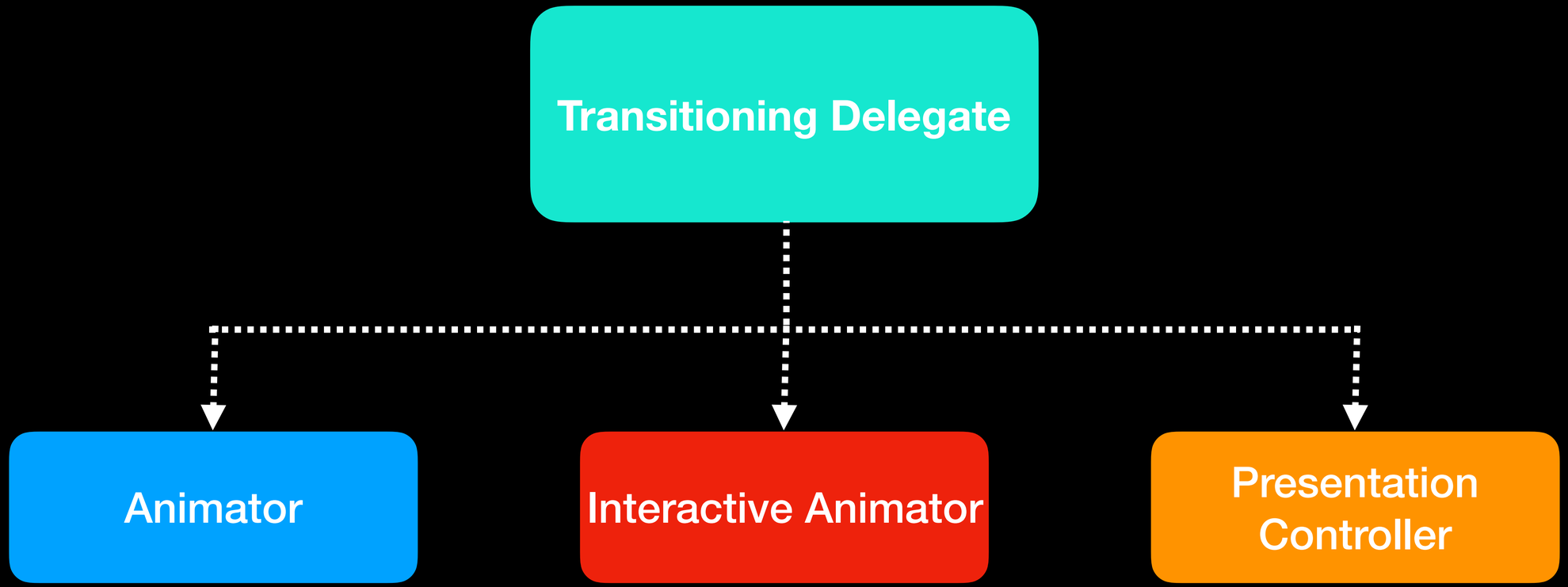
Custom Presentation and Transition

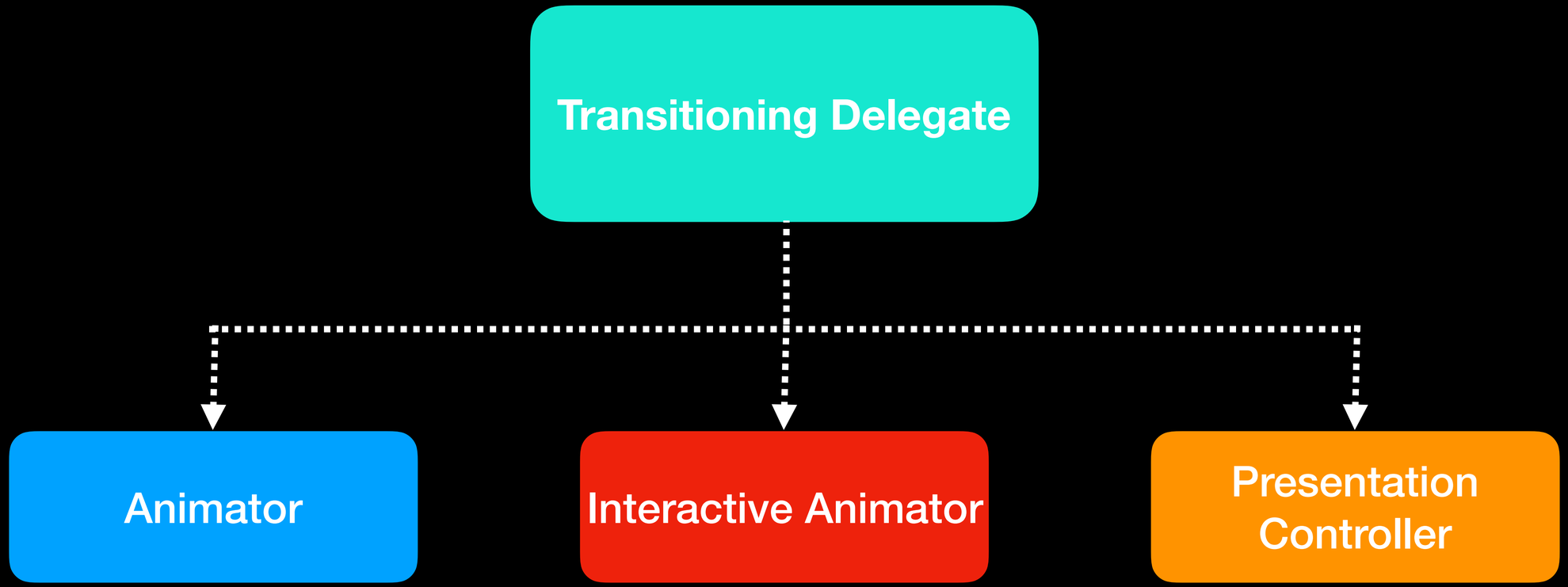


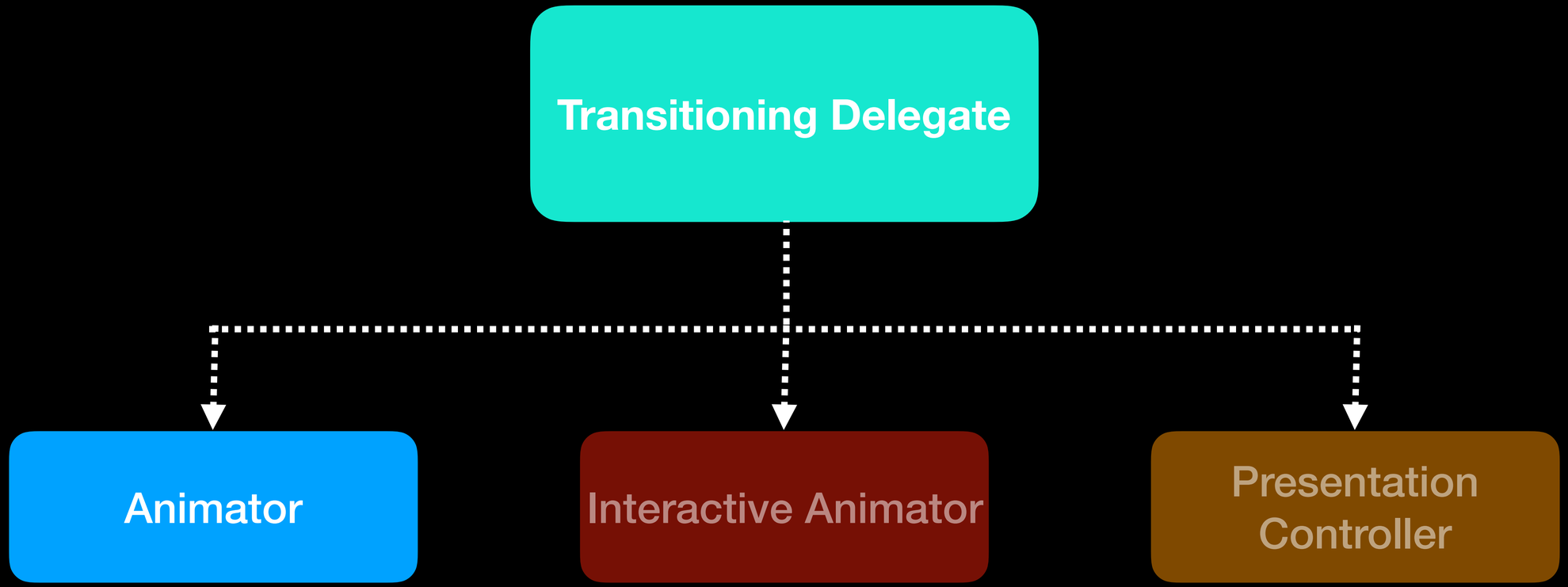
Custom Presentation and Transition



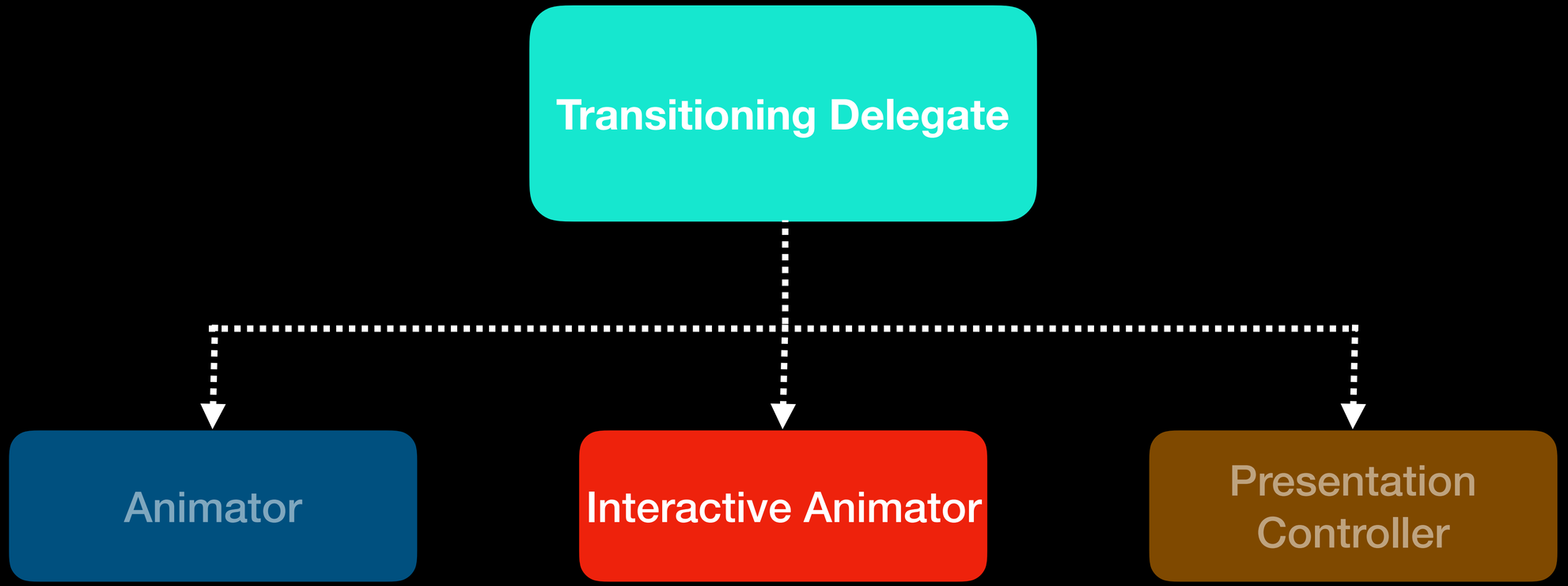

Transitioning Delegate
Animator
Interactive Animator
Presentation Controller


Transitioning Delegate
Animator
Interactive Animator
Presentation Controller

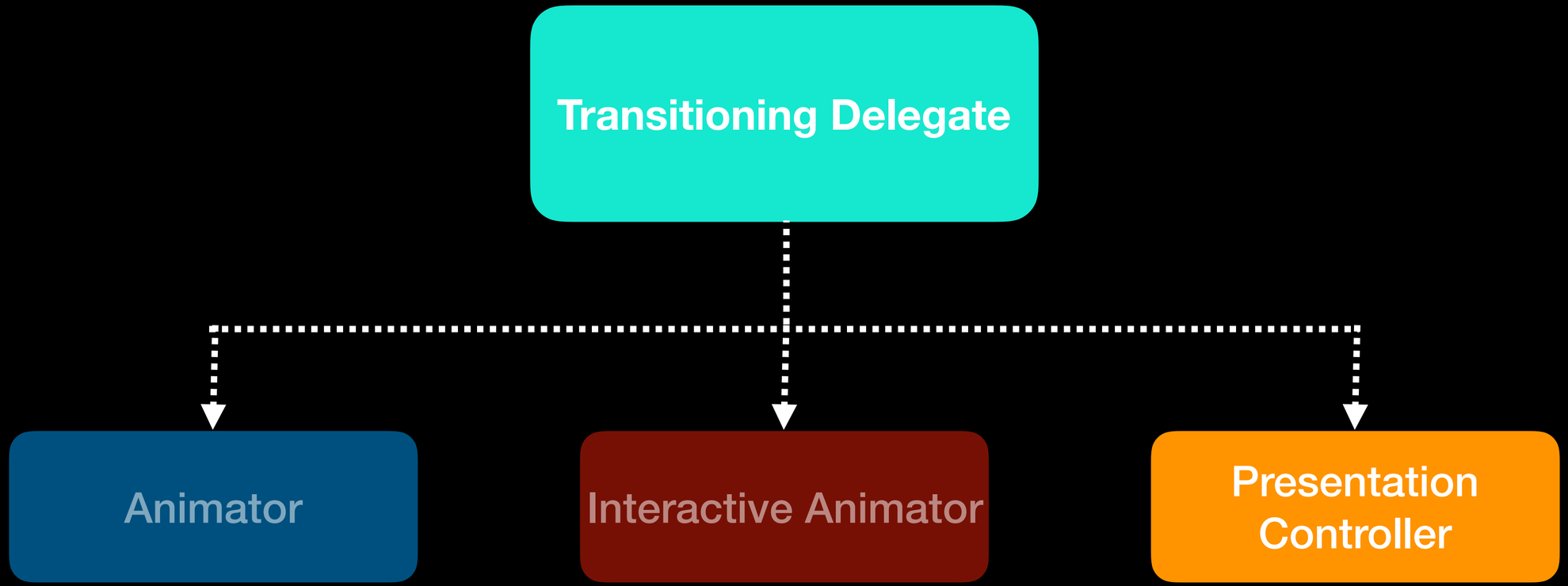

Transitioning Delegate
Animator
Interactive Animator
Presentation Controller




```
func animationController(forPresented …) -> UIViewControllerAnimatedTransitioning?

func animationController(forDismissed …) -> UIViewControllerAnimatedTransitioning?
```




```
func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning?

func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning?
```




```
func presentationController(…) -> UIPresentationController?
```

“TransitionManager”

“TransitionManager”

```
func animationController(forPresented …) -> UIViewControllerAnimatedTransitioning?

func animationController(forDismissed …) -> UIViewControllerAnimatedTransitioning?
```

“TransitionManager”

```
func animationController(forPresented …) -> UIViewControllerAnimatedTransitioning?

func animationController(forDismissed …) -> UIViewControllerAnimatedTransitioning?

func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning?

func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning?
```

“TransitionManager”

```
func animationController(forPresented …) -> UIViewControllerAnimatedTransitioning?

func animationController(forDismissed …) -> UIViewControllerAnimatedTransitioning?

func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning?

func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning?

func presentationController(…) -> UIPresentationController?
```

Show me the code!

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom

    present(vc, animated: true)
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom

    present(vc, animated: true)
}
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom

    present(vc, animated: true)
}
```

Show me the code!

```
func collectionView(_ collectionView: UICollectionView, didSelectItemAt
indexPath: IndexPath) {
    let cell = collectionView.cellForItem(at: indexPath) as!
CardCollectionViewCell
    let model = models[indexPath.row]
    let params = …

    let vc = storyboard!.instantiateViewController(withIdentifier:
"cardDetailVc") as! CardDetailViewController

    self.transition = CardTransition(params: params)

    vc.transitioningDelegate = transition
    vc.modalPresentationStyle = .custom

    present(vc, animated: true)
}
```

```
final class CardTransition: NSObject, UIViewControllerTransitioningDelegate {
    struct Params { … }
    let params: Params
    init(params: Params) {
        self.params = params
        super.init()
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return PresentCardAnimator(params: params)
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return DismissCardAnimator(params: params)
    }

    func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func presentationController(…) -> UIPresentationController? {
        return CardPresentationController(presentedViewController: presented, presenting:
presenting)
    }
}
```

```
final class CardTransition: NSObject, UIViewControllerTransitioningDelegate {
    struct Params { … }
    let params: Params
    init(params: Params) {
        self.params = params
        super.init()
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return PresentCardAnimator(params: params)
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return DismissCardAnimator(params: params)
    }

    func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func presentationController(…) -> UIPresentationController? {
        return CardPresentationController(presentedViewController: presented, presenting:
presenting)
    }
}
```

```
final class CardTransition: NSObject, UIViewControllerTransitioningDelegate {
    struct Params { … }
    let params: Params
    init(params: Params) {
        self.params = params
        super.init()
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return PresentCardAnimator(params: params)
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return DismissCardAnimator(params: params)
    }

    func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func presentationController(…) -> UIPresentationController? {
        return CardPresentationController(presentedViewController: presented, presenting:
presenting)
    }
}
```

```
final class CardTransition: NSObject, UIViewControllerTransitioningDelegate {
    struct Params { … }
    let params: Params
    init(params: Params) {
        self.params = params
        super.init()
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return PresentCardAnimator(params: params)
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return DismissCardAnimator(params: params)
    }

    func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func presentationController(…) -> UIPresentationController? {
        return CardPresentationController(presentedViewController: presented, presenting:
presenting)
    }
}
```

```
final class CardTransition: NSObject, UIViewControllerTransitioningDelegate {
    struct Params { … }
    let params: Params
    init(params: Params) {
        self.params = params
        super.init()
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return PresentCardAnimator(params: params)
    }

    func animationController(…) -> UIViewControllerAnimatedTransitioning? {
        return DismissCardAnimator(params: params)
    }

    func interactionControllerForPresentation(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func interactionControllerForDismissal(…) -> UIViewControllerInteractiveTransitioning? {
        return nil
    }

    func presentationController(…) -> UIPresentationController? {
        return CardPresentationController(presentedViewController: presented, presenting:
presenting)
    }
}
```

AnimationController (“Animator Object”)

```
final class PresentCardAnimator: NSObject, UIViewControllerAnimatedTransitioning {

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval {
        return 0.6
    }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let ctx = transitionContext
        let container = transitionContext.containerView
        let to = ctx.view(forKey: .to)!
        container.addSubview(to)

        // Setup, prepare …

        UIView.animate(withDuration: transitionDuration(using: ctx), animations: {

            // do cool card animation here!

        }) { (finished) in
            let isSuccess = !ctx.transitionWasCancelled
            ctx.completeTransition(isSuccess)
        }
    }
}
```

AnimationController (“Animator Object”)

```
final class PresentCardAnimator: NSObject, UIViewControllerAnimatedTransitioning {

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval {
        return 0.6
    }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let ctx = transitionContext
        let container = transitionContext.containerView
        let to = ctx.view(forKey: .to)!
        container.addSubview(to)

        // Setup, prepare …

        UIView.animate(withDuration: transitionDuration(using: ctx), animations: {

            // do cool card animation here!

        }) { (finished) in
            let isSuccess = !ctx.transitionWasCancelled
            ctx.completeTransition(isSuccess)
        }
    }
}
```

AnimationController (“Animator Object”)

```
final class PresentCardAnimator: NSObject, UIViewControllerAnimatedTransitioning {

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval {
        return 0.6
    }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let ctx = transitionContext
        let container = transitionContext.containerView
        let to = ctx.view(forKey: .to)!
        container.addSubview(to)

        // Setup, prepare …

        UIView.animate(withDuration: transitionDuration(using: ctx), animations: {

            // do cool card animation here!

        }) { (finished) in
            let isSuccess = !ctx.transitionWasCancelled
            ctx.completeTransition(isSuccess)
        }
    }
}
```

AnimationController (“Animator Object”)

```
final class PresentCardAnimator: NSObject, UIViewControllerAnimatedTransitioning {

    func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval {
        return 0.6
    }

    func animateTransition(using transitionContext: UIViewControllerContextTransitioning) {
        let ctx = transitionContext
        let container = transitionContext.containerView
        let to = ctx.view(forKey: .to)!
        container.addSubview(to)

        // Setup, prepare …

        UIView.animate(withDuration: transitionDuration(using: ctx), animations: {

            // do cool card animation here!

        }) { (finished) in
            let isSuccess = !ctx.transitionWasCancelled
            ctx.completeTransition(isSuccess)
        }
    }
}
```

General concepts are nice…

Details, details, details!

Let’s deep dive!

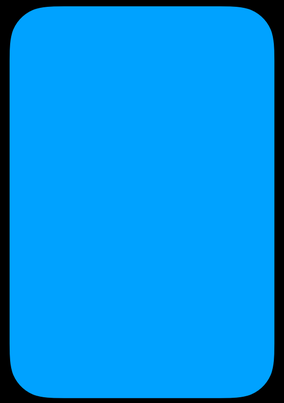
5 Phases of Interaction

5 Phases of Interaction

1. Highlight

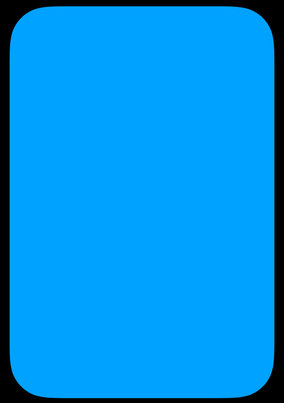
5 Phases of Interaction

1. Highlight



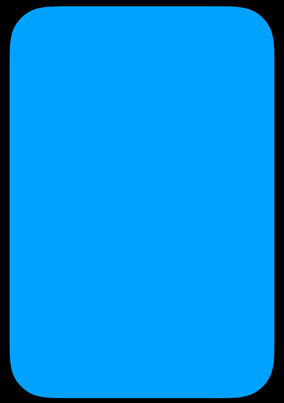
5 Phases of Interaction

1. Highlight



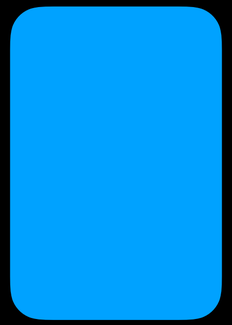
5 Phases of Interaction

2. Before present



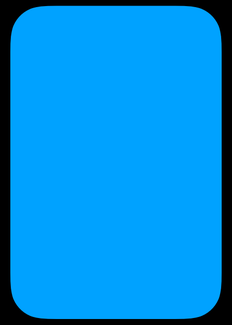
5 Phases of Interaction

2. Before present



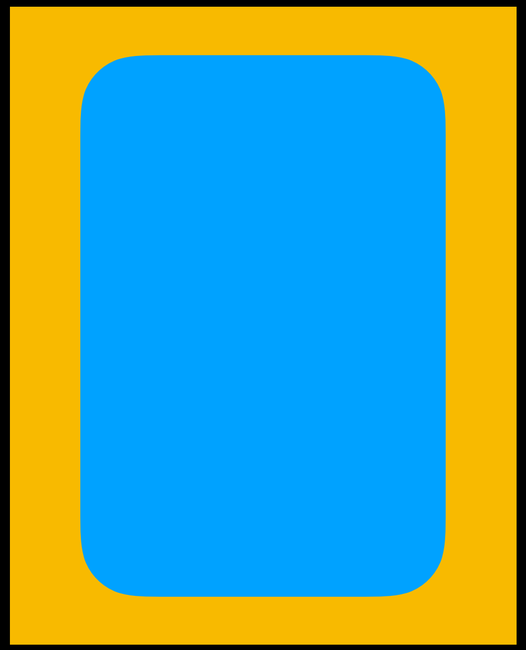
5 Phases of Interaction

3. Present



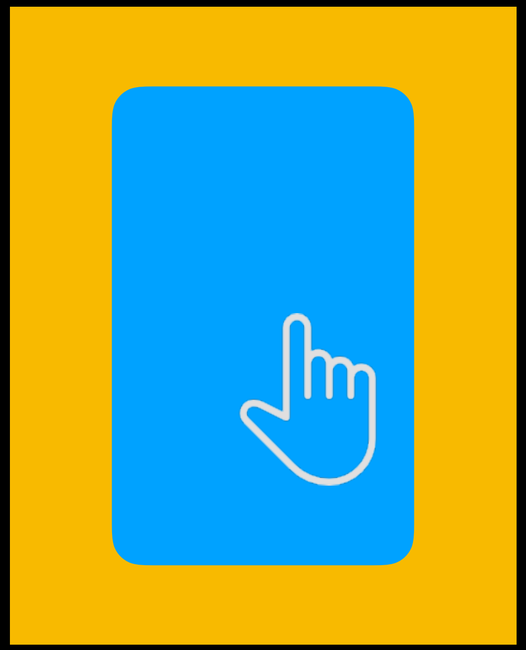
5 Phases of Interaction

3. Present



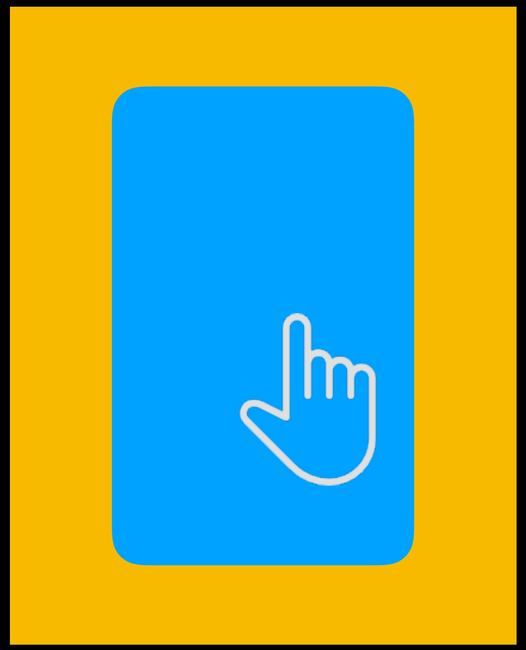
5 Phases of Interaction

4. Interactively dismiss



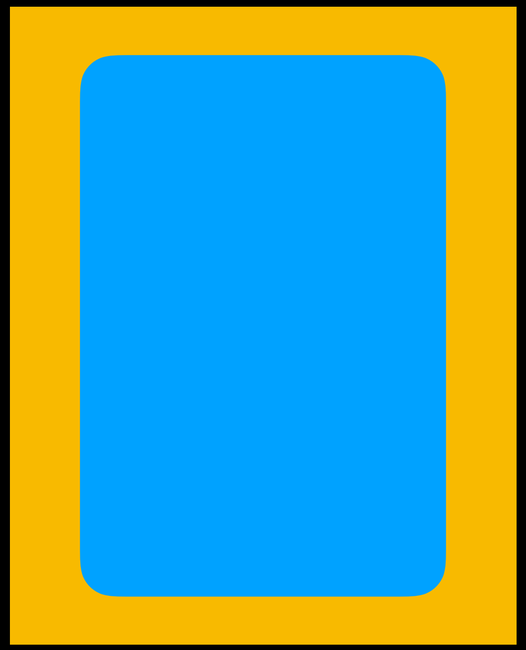
5 Phases of Interaction

4. Interactively dismiss



5 Phases of Interaction

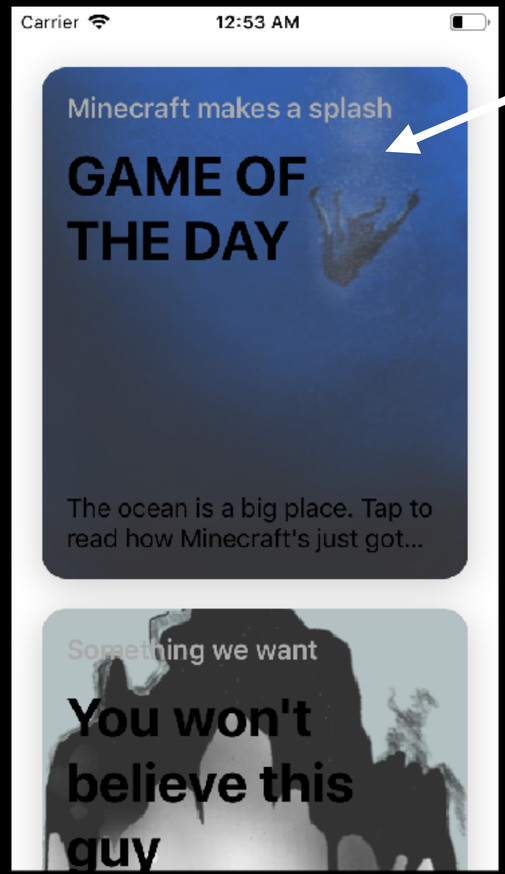
5. Dismiss



5 Phases of Interaction

5. Dismiss

Terminologies

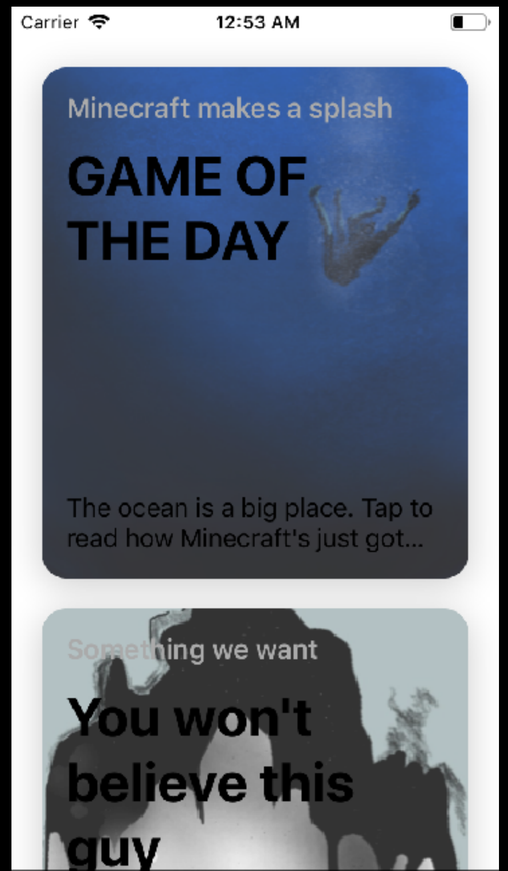


Card

Terminologies

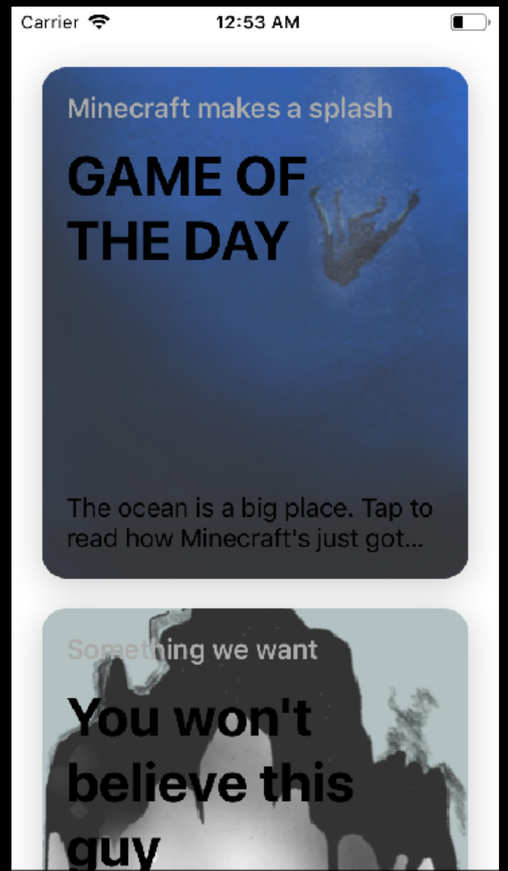
Terminologies

Home



Terminologies

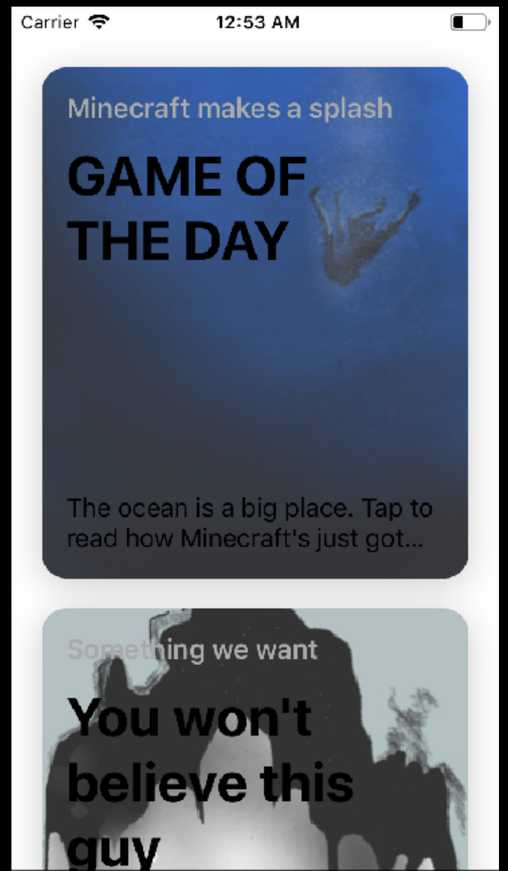
Home



present

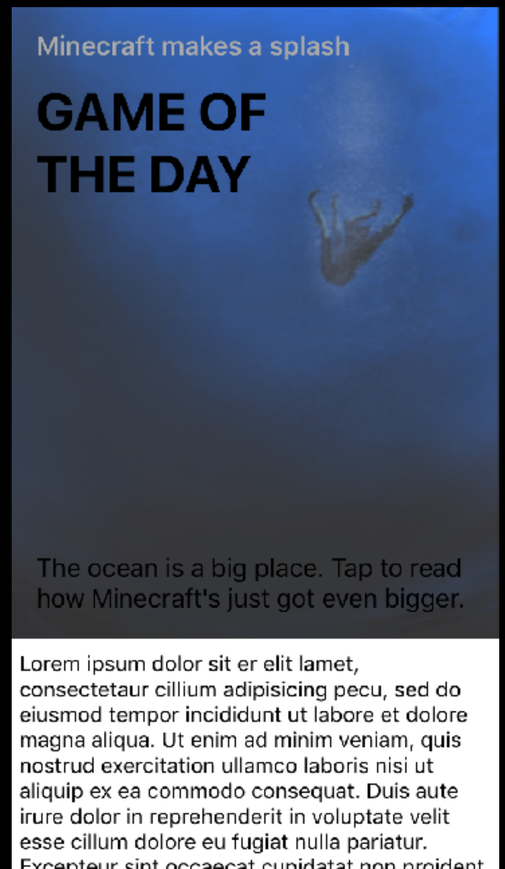
Terminologies

Home



present

DetailPage



1 Highlighting

1 Highlighting

Scale down

1 Highlighting

Scale down

```
UIView.animate(withDuration: 0.5,
               delay: 0,
               usingSpringWithDamping: 1,
               initialSpringVelocity: 0,
               options: animationOptions, animations:
{
    self.transform = .init(scaleX: 0.96, y: 0.96)
}, completion: completion)
```

1 Highlighting

Scale back up

1 Highlighting

Scale back up

```
UIView.animate(withDuration: 0.5,
               delay: 0,
               usingSpringWithDamping: 1,
               initialSpringVelocity: 0,
               options: animationOptions, animations:
{
    self.transform = .identity
}, completion: completion)
```

1 Highlighting

Allows user interaction

1 Highlighting

Allows user interaction

```
let animationOptions: UIViewAnimationOptions = [.allowUserInteraction]
```

1 Highlighting

Allows user interaction

```
let animationOptions: UIViewAnimationOptions = [.allowUserInteraction]
```

Or you can’t scroll during the animation!

1 Highlighting

Animate on touch events

```
override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesCancelled(_ touches: Set<UITouch>, with
event: UIEvent?) {…}
```

1 Highlighting

Animate on touch events

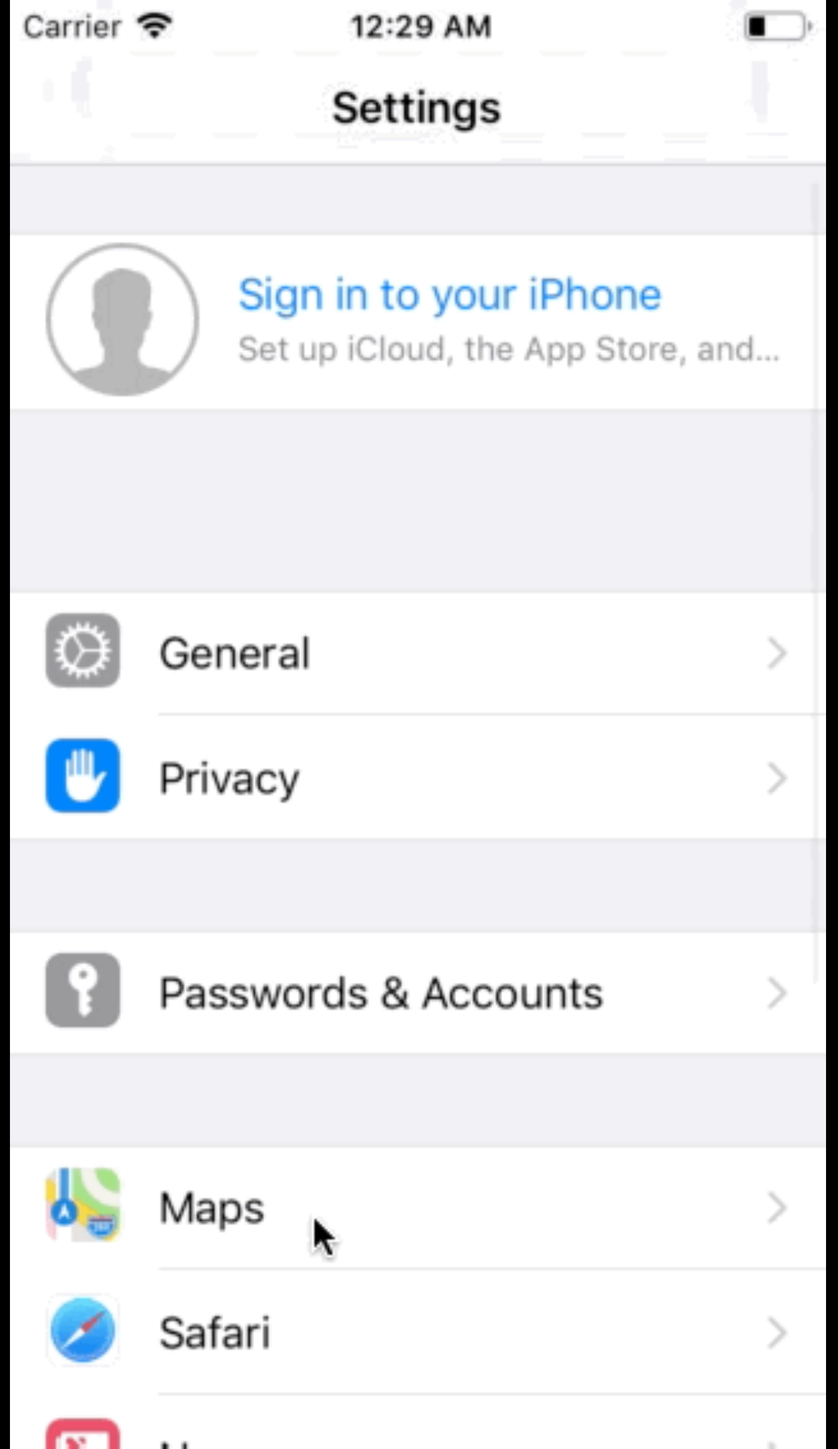
```
override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesCancelled(_ touches: Set<UITouch>, with
event: UIEvent?) {…}
```

1 Highlighting

Animate on touch events

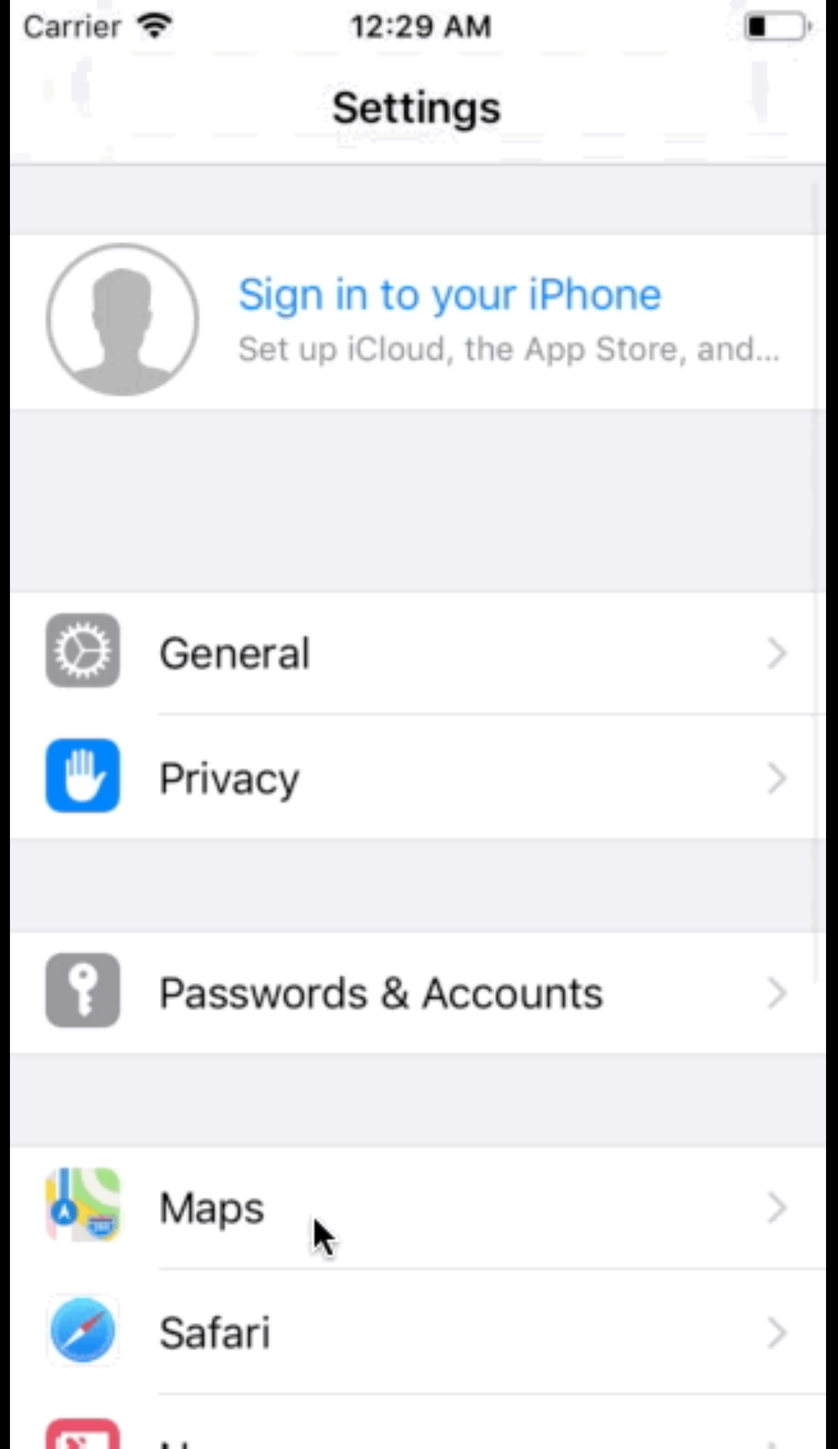
```
override func touchesBegan(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesEnded(_ touches: Set<UITouch>, with event:
UIEvent?) {…}

override func touchesCancelled(_ touches: Set<UITouch>, with
event: UIEvent?) {…}
```

1 Highlighting



Touch to the scroll view's content is **delayed** by default

1 Highlighting



Touch to the scroll view's content is **delayed** by default

1 Highlighting

Disable delay content touches

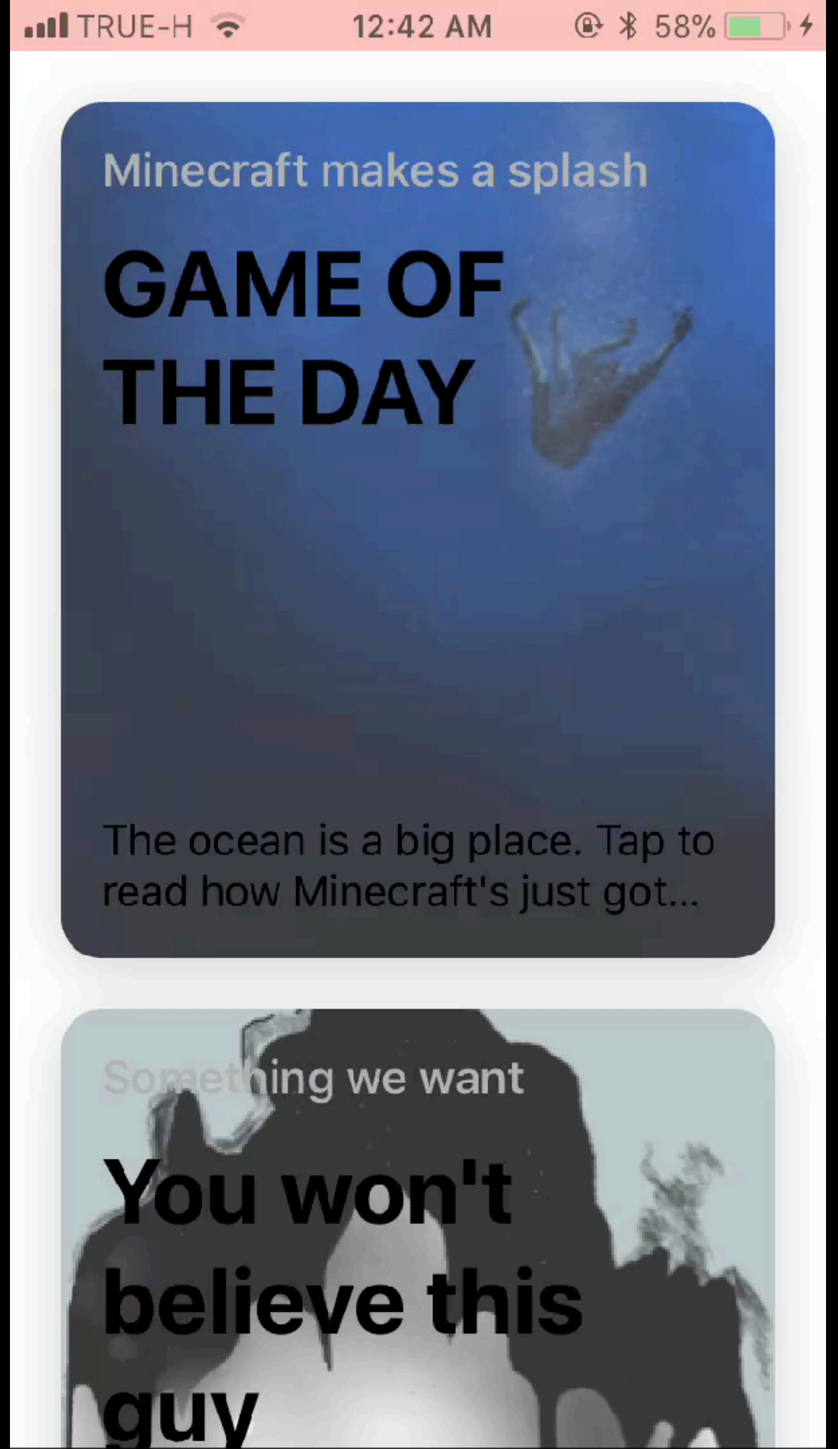
1 Highlighting

Disable delay content touches

```
scrollView.delaysContentTouches = false
```

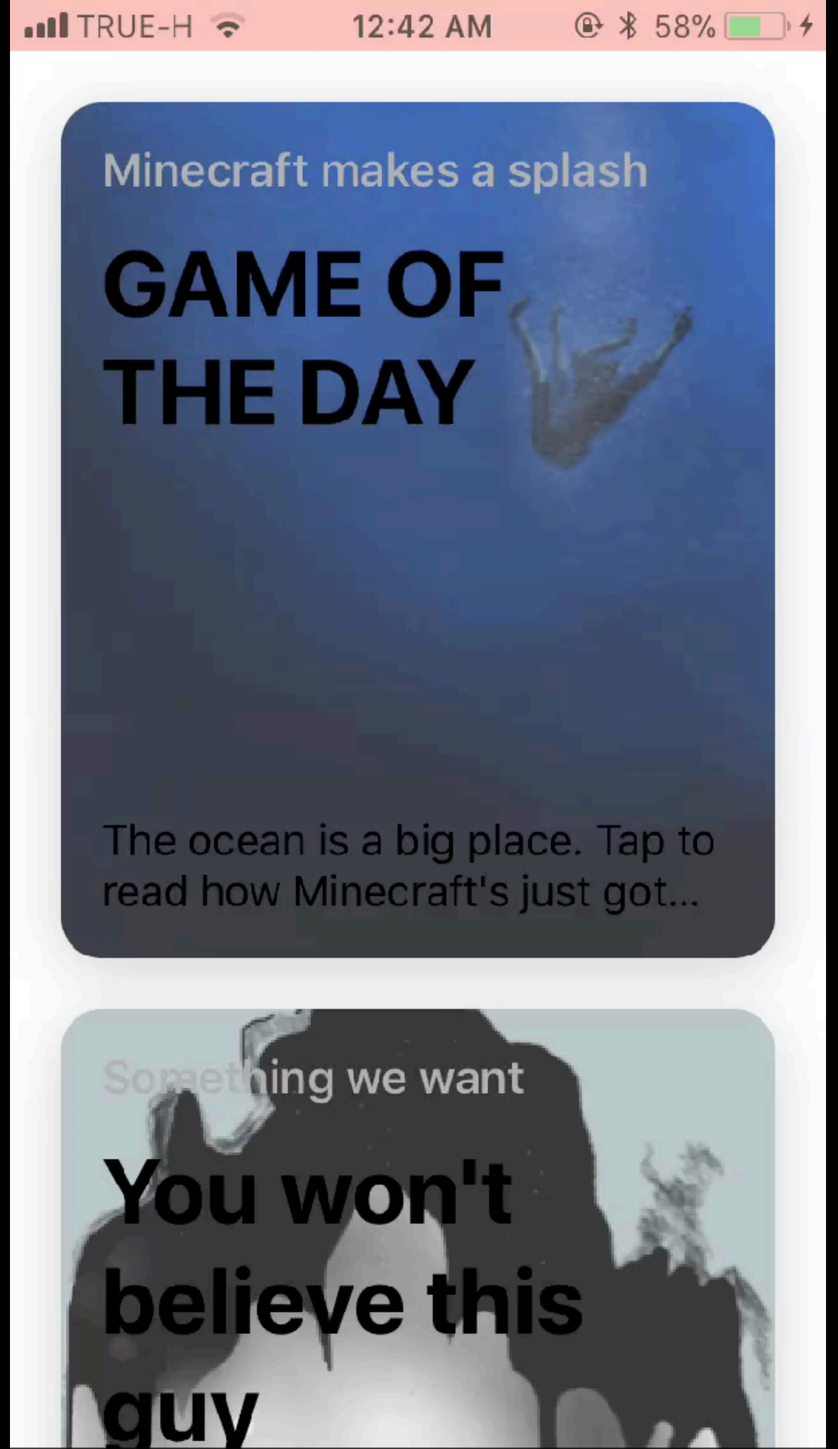
1 Highlighting

Final result



1 Highlighting

Final result



2 Before Presenting

2 Before Presenting

Freeze the highlight animation

2 Before Presenting

Freeze the highlight animation

```
// Custom flag to prevent highlight animation
cell.disabledHighlightedAnimation = true
```

2 Before Presenting

Freeze the highlight animation

```
// Custom flag to prevent highlight animation
cell.disabledHighlightedAnimation = true


// Stop current animation
cell.layer.removeAllAnimations()
```

2 Before Presenting

Get layer’s current frame on screen

2 Before Presenting

Get layer’s current frame on screen

```
let cellFrame = cell.layer.presentation()!.frame
```

2 Before Presenting

Get layer’s current frame on screen

```
let cellFrame = cell.layer.presentation()!.frame
```

```
let cellFrameOnScreen =
cell.superview!.convert(cellFrame, to: nil)
```

2 Before Presenting

Prepare

2 Before Presenting

Prepare

Add detail view on container view

2 Before Presenting

Prepare

Add detail view on container view

Place it at original card’s frame with AutoLayout

2 Before Presenting

Prepare

Add detail view on container view

Place it at original card’s frame with AutoLayout

Hide original card cell

2 Before Presenting

Layout if needed

2 Before Presenting

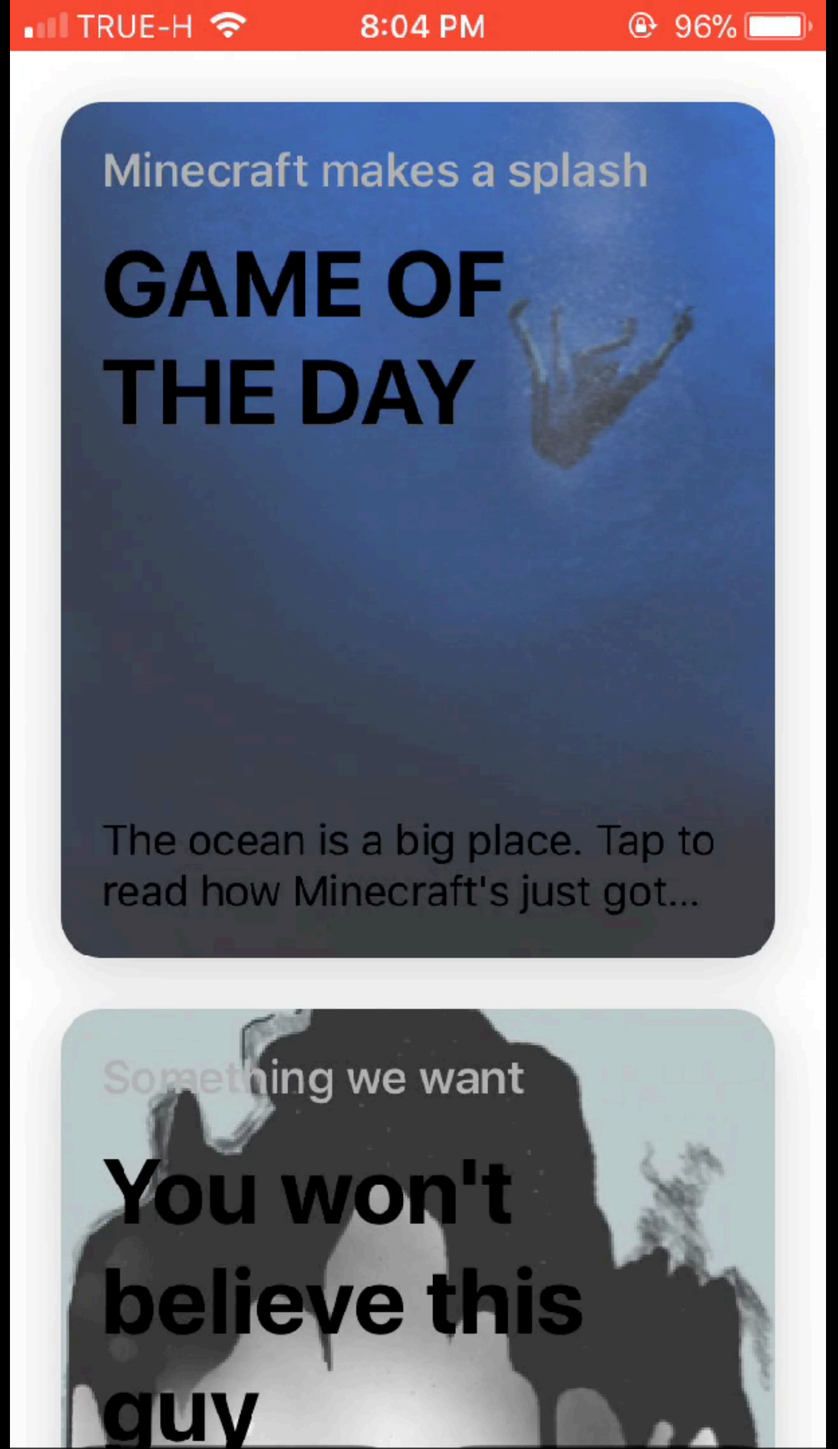
Layout if needed

```
container.layoutIfNeeded()
```

3 Presenting

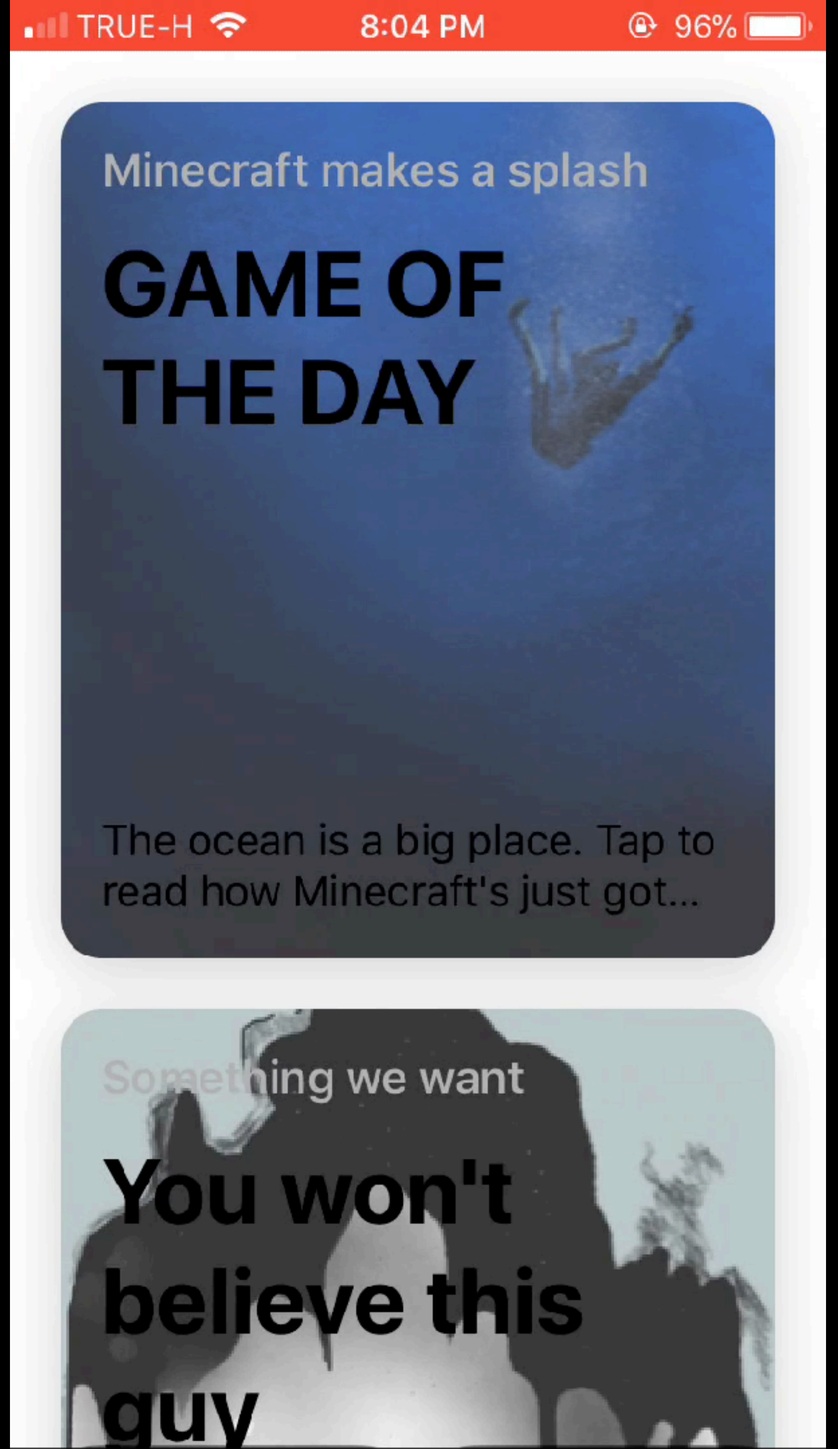
3 Presenting

Animation



3 Presenting

Animation



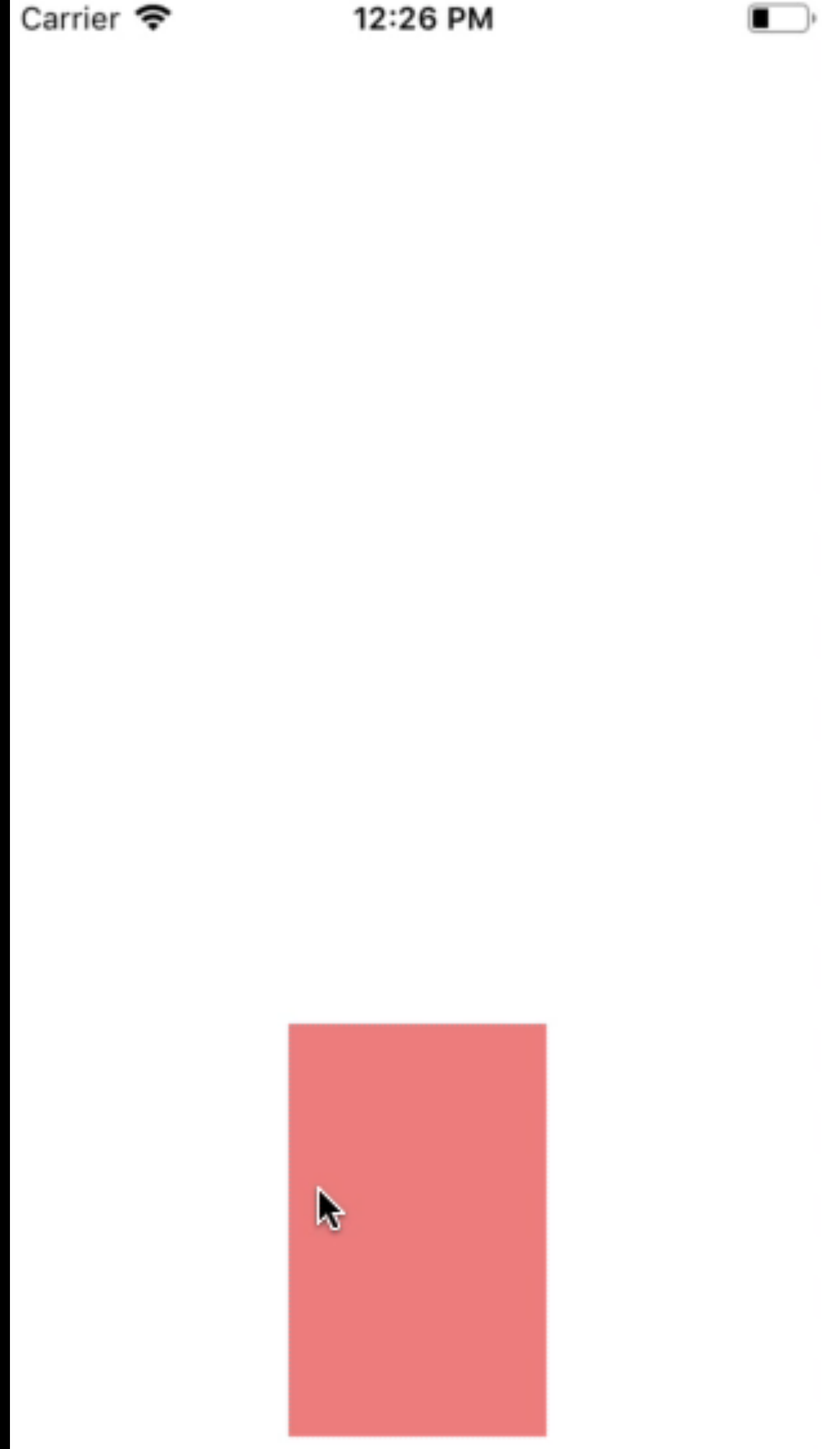
3 Presenting

Animate frames with spring?

```
UIView.animate(withDuration: 1,
               delay: 0,
               usingSpringWithDamping: 0.5,
               initialSpringVelocity: 0,
               options: [], animations:
        {
            self.bottomConstraint = -200
            self.widthConstraint = 200
            self.heightConstraint = 320
            container.layoutIfNeeded()
        }) { … }
```

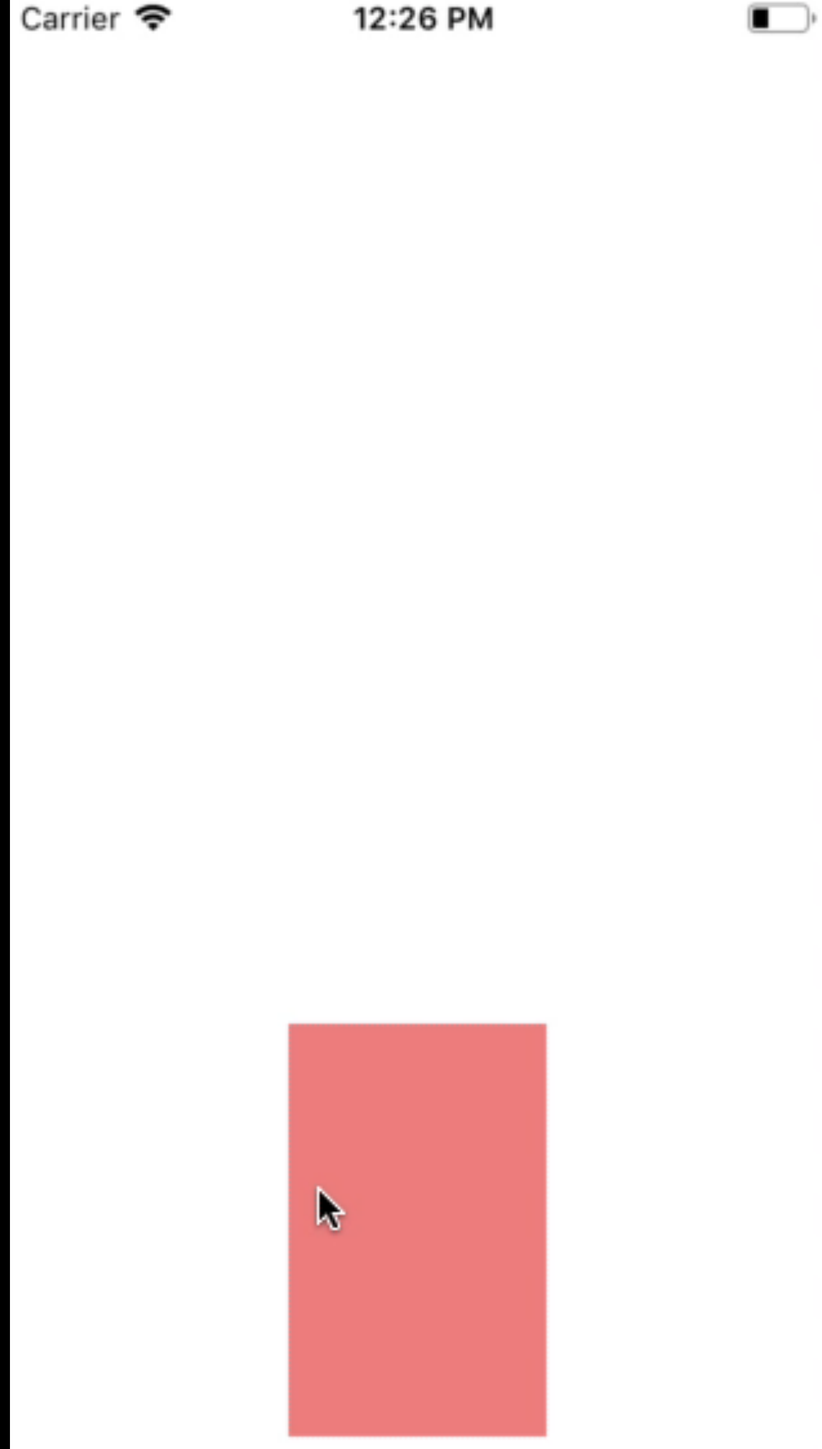
3 Presenting

Animate frames with spring?



3 Presenting

Animate frames with spring?



3 Presenting

Linear sizing + spring moving up

3 Presenting

Linear sizing + spring moving up

```
UIView.animate(withDuration: 0.8) {

    ???

    container.layoutIfNeeded()
}
```

3 Presenting

Linear sizing + spring moving up

Two animation curves for AutoLayout animation?

3 Presenting

Linear sizing + spring moving up

3 Presenting

Linear sizing + spring moving up

```
UIView.animate(withDuration: 0.6 * 0.8) {
    self.widthConstraint.constant = 200
    self.heightConstraint.constant = 320
    container.layoutIfNeeded()
}
```

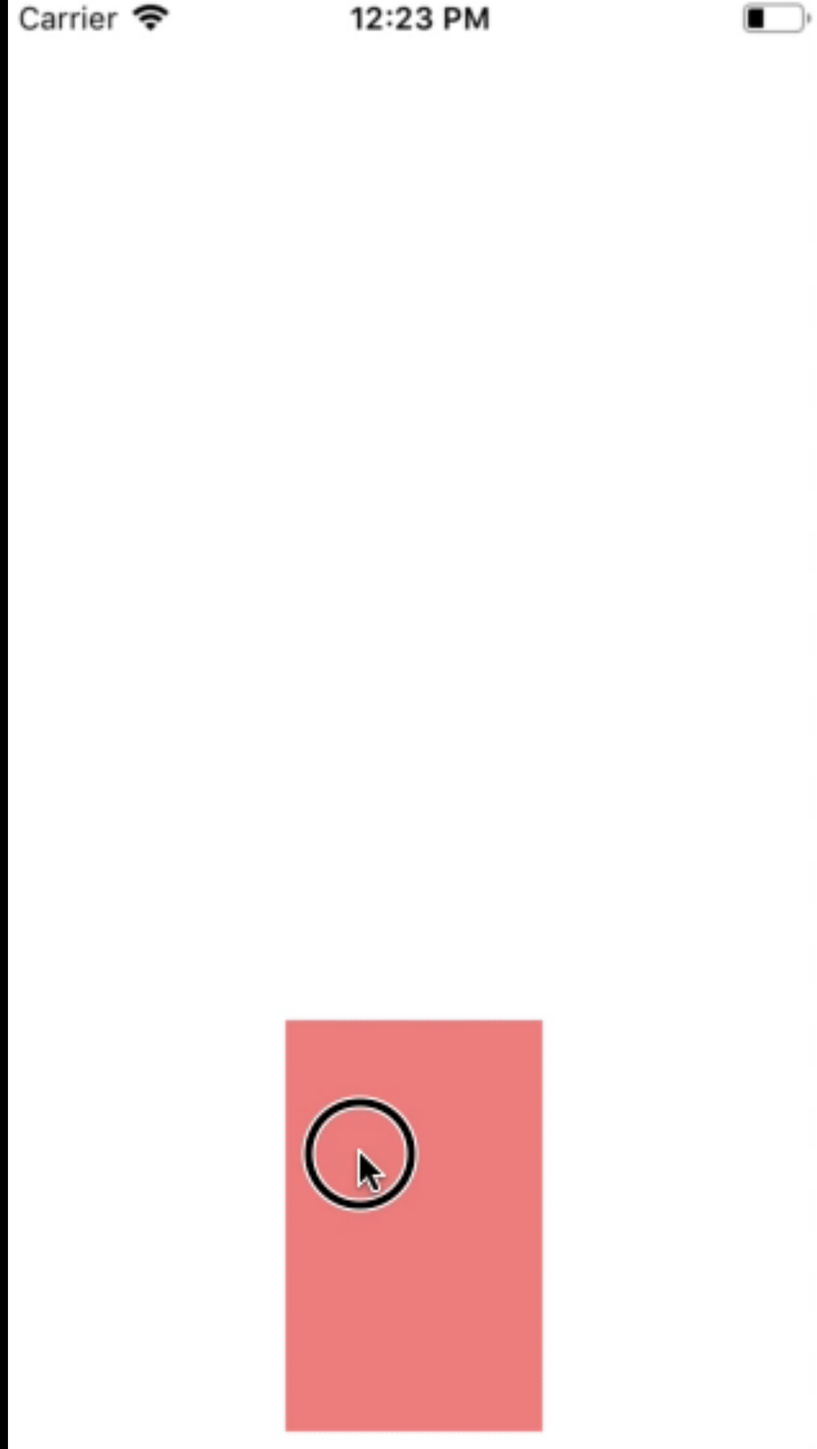
3 Presenting

Linear sizing + spring moving up

```
UIView.animate(withDuration: 0.6 * 0.8) {
    self.widthConstraint.constant = 200
    self.heightConstraint.constant = 320
    container.layoutIfNeeded()
}

UIView.animate(withDuration: 0.8, delay: 0, usingSpringWithDamping: 0.5,
initialSpringVelocity: 0, options: [], animations: {
    self.bottomConstraint.constant = -200
    container.layoutIfNeeded()
})
```

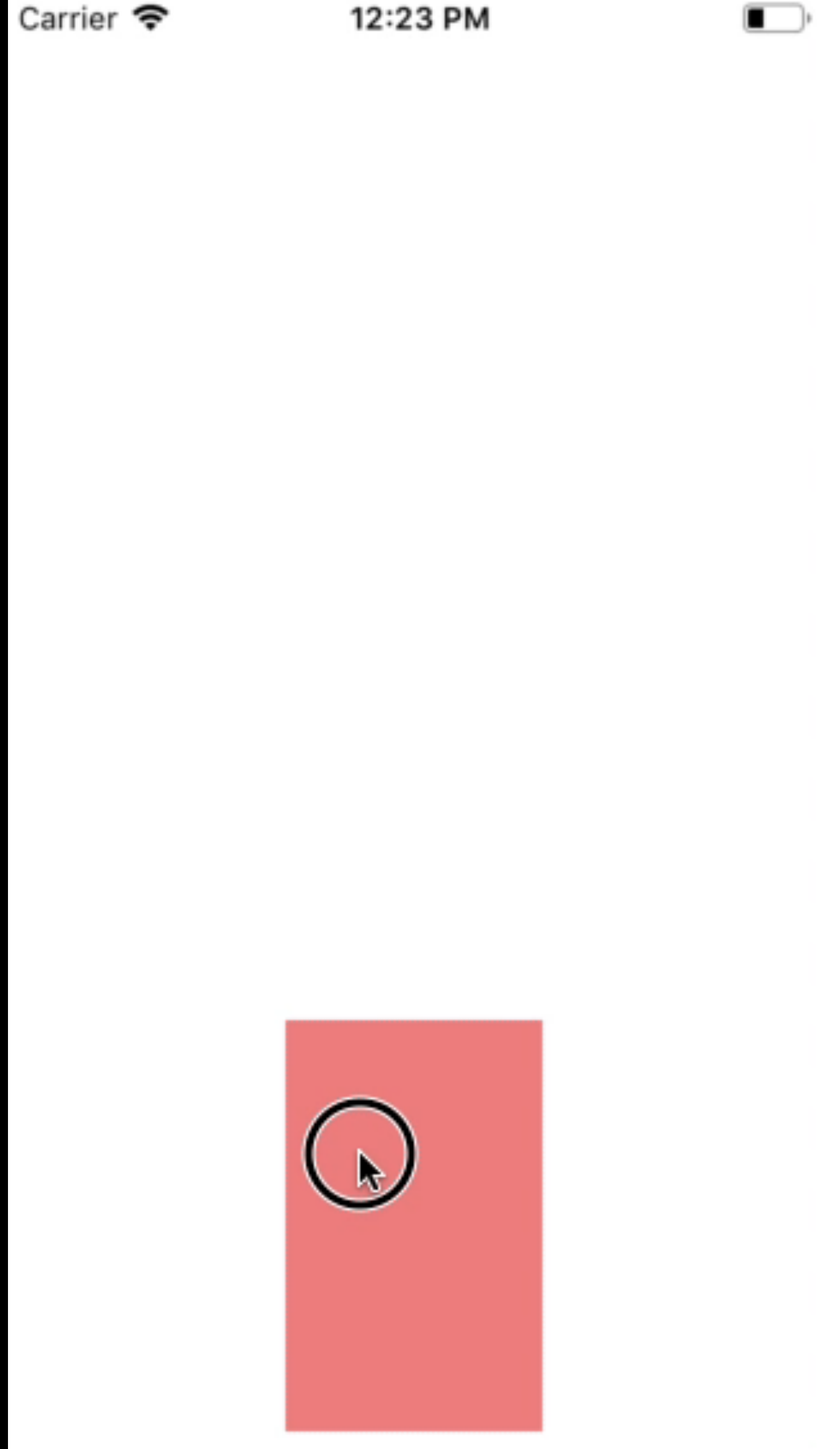
3 Presenting

Linear sizing + spring moving up



3 Presenting

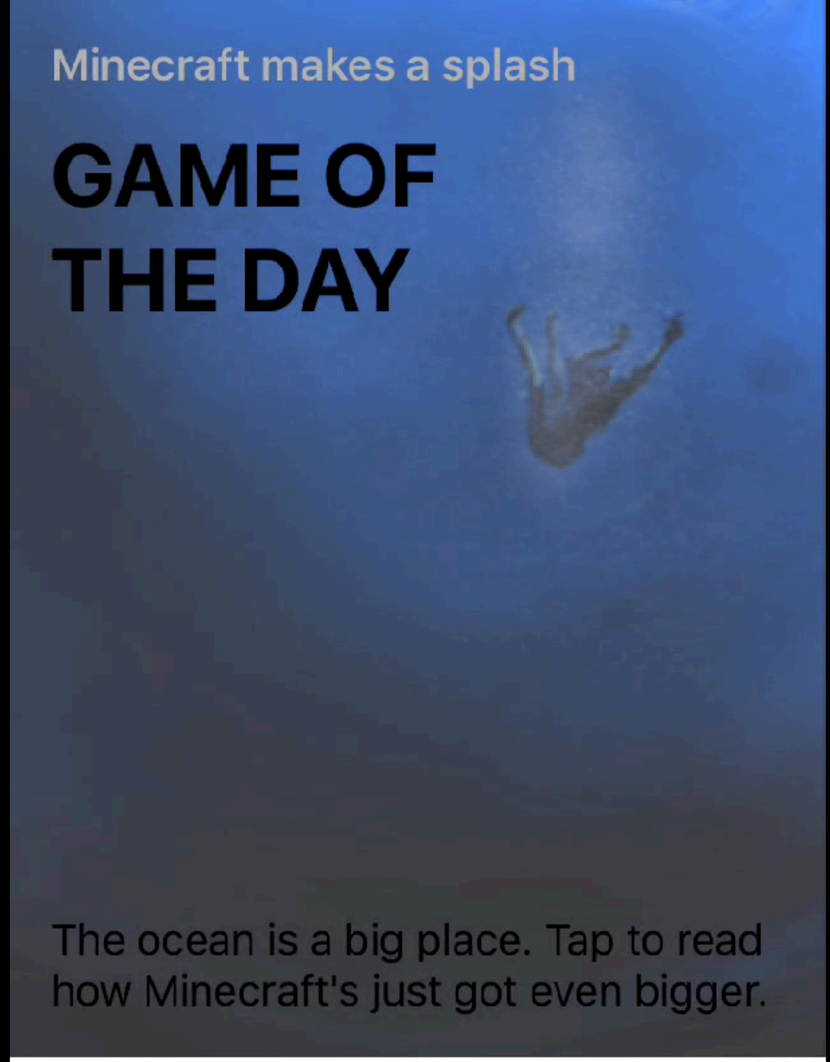
Linear sizing + spring moving up



4 Interactive Dismissing

4 Interactive Dismissing

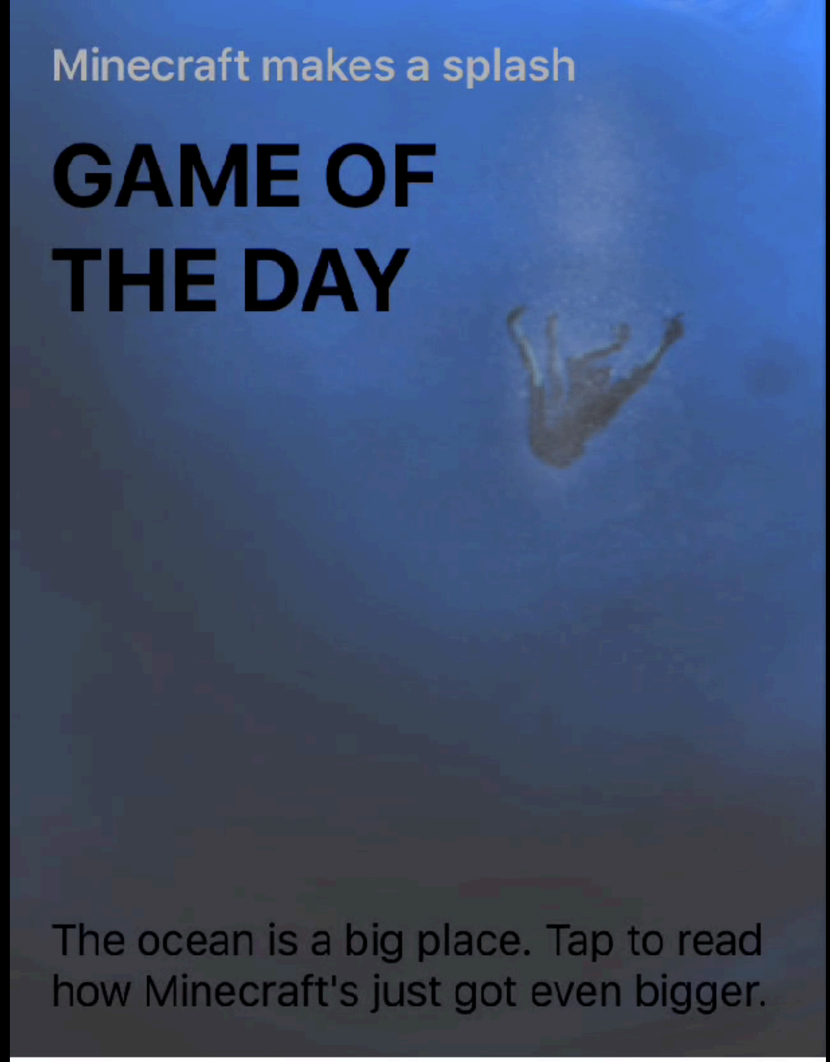
At scroll top and left edge



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

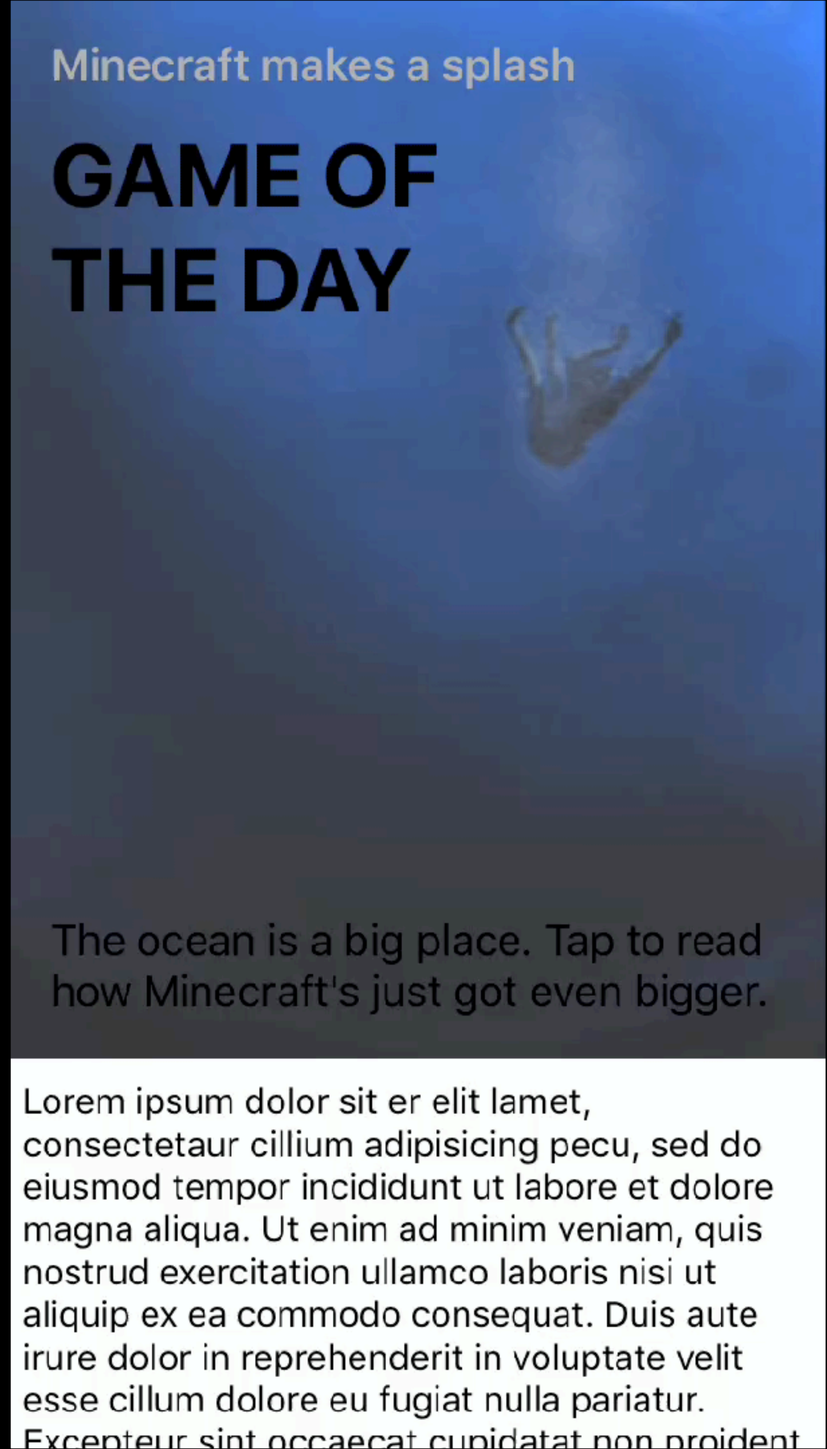
At scroll top and left edge



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

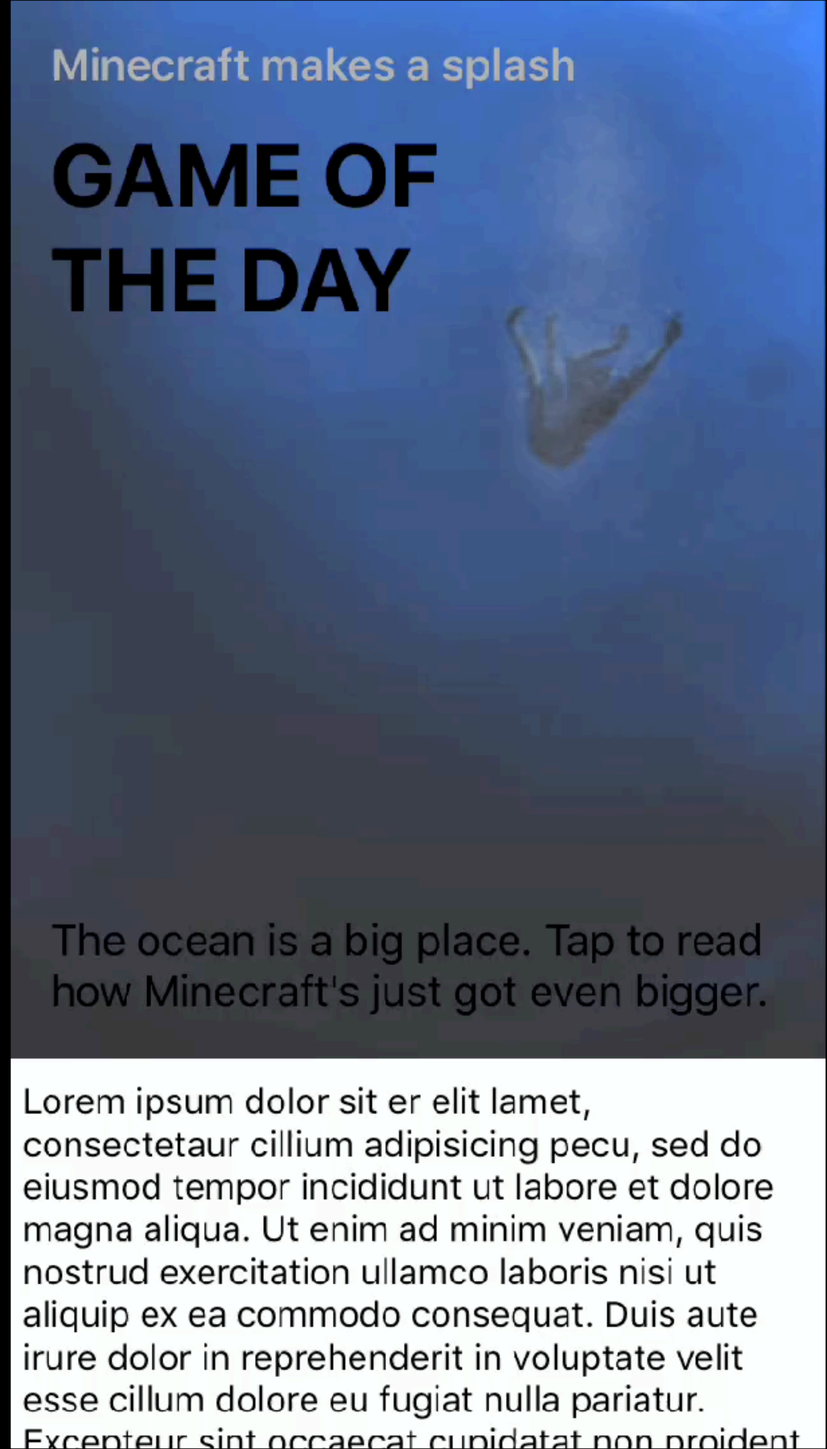
4 Interactive Dismissing

Dismissing



4 Interactive Dismissing

Dismissing



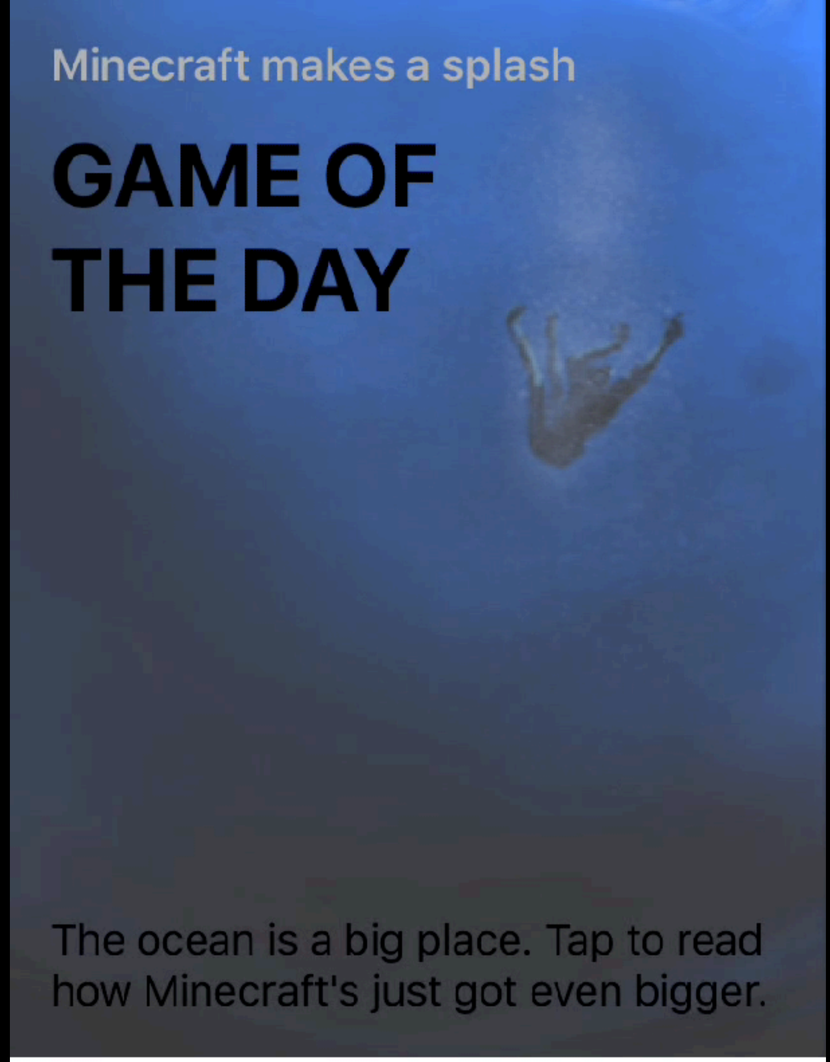
4 Interactive Dismissing

Didn’t use InteractionController…

4 Interactive Dismissing

Cancel dismissal by dragging up

Continue the scrolling!

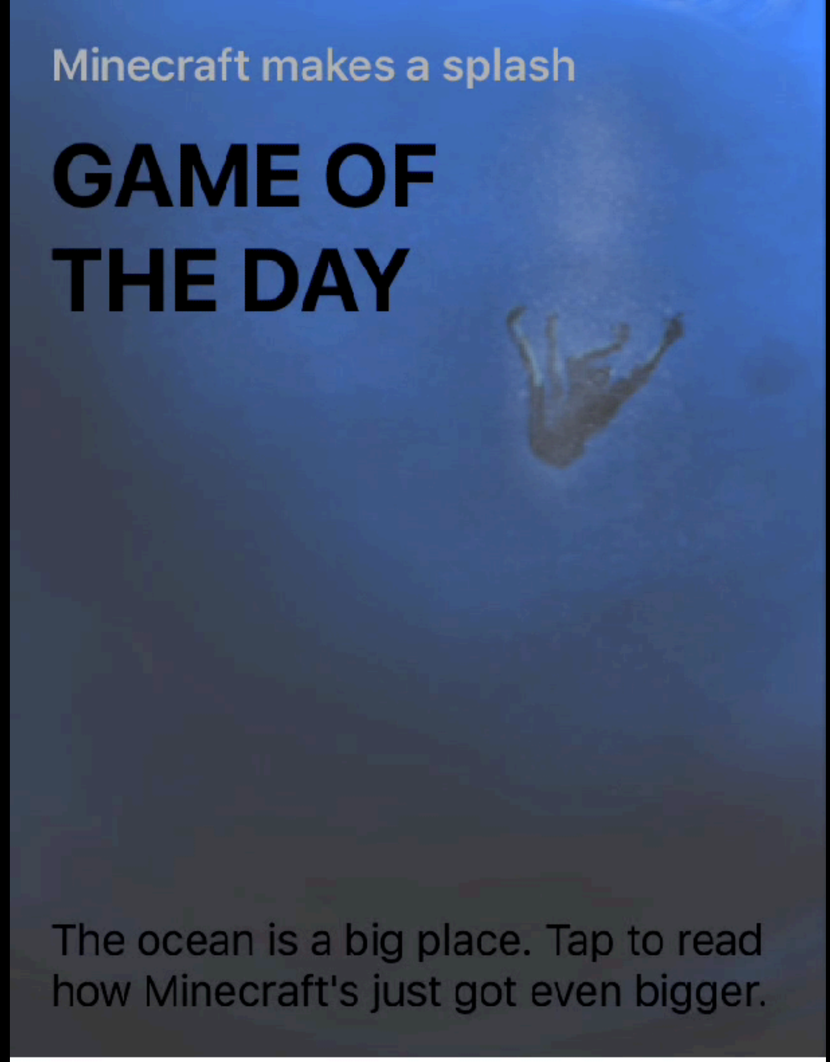


Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

Cancel dismissal by dragging up

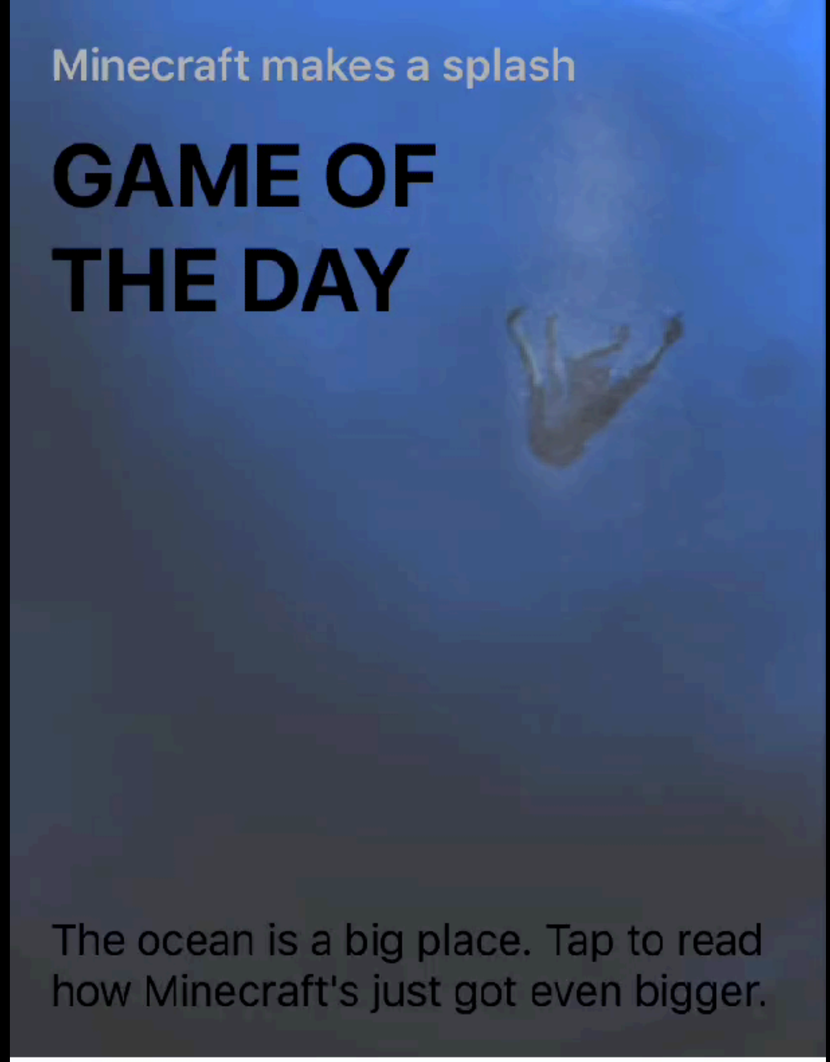
Continue the scrolling!



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

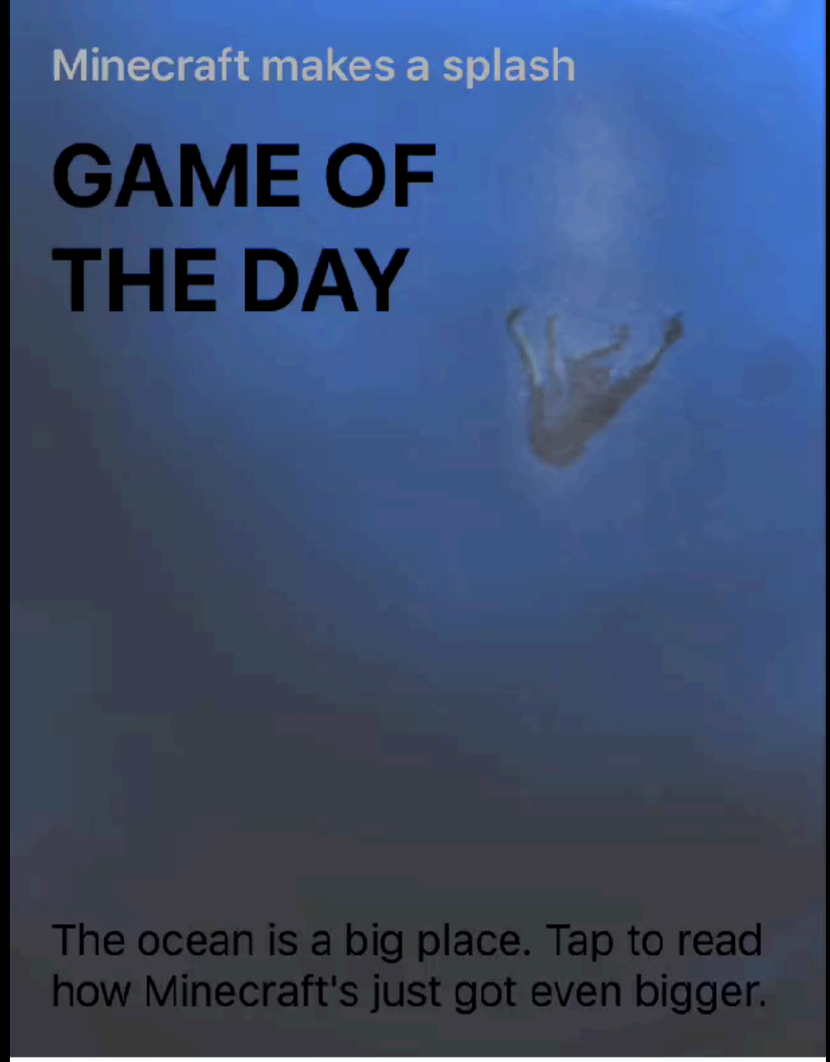
Very responsive shrinking



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

Very responsive shrinking



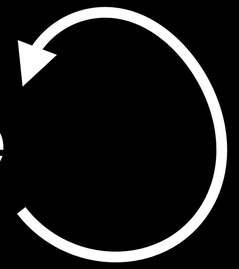
Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

Then how?

4 Interactive Dismissing

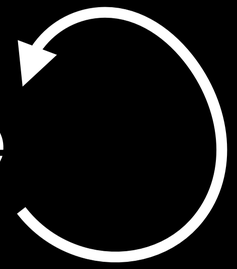
Then how?

Interactive phase 

4 Interactive Dismissing

Then how?

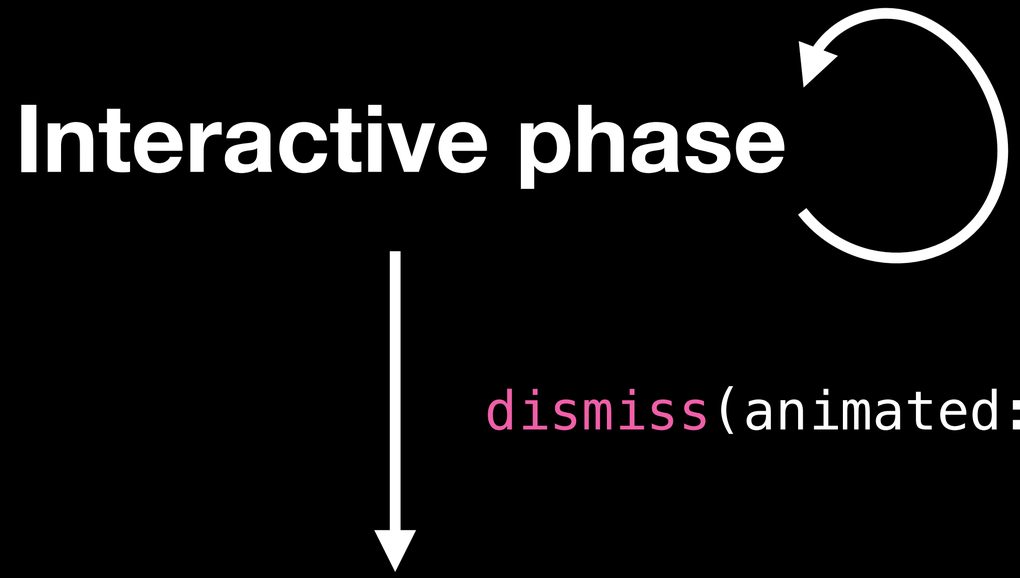
Interactive phase



Card detail VC

4 Interactive Dismissing

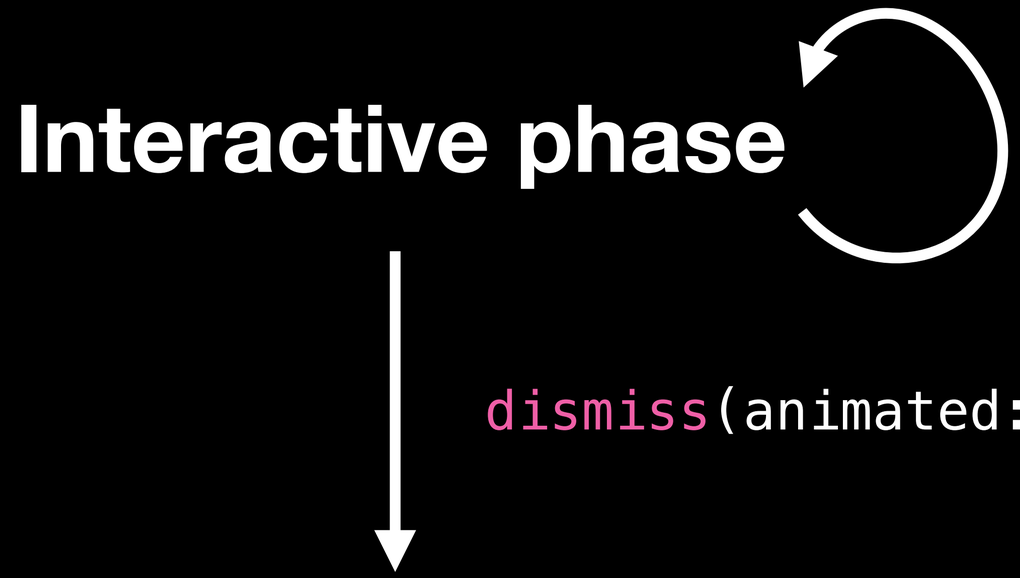
Then how?



Card detail VC

4 Interactive Dismissing

Then how?

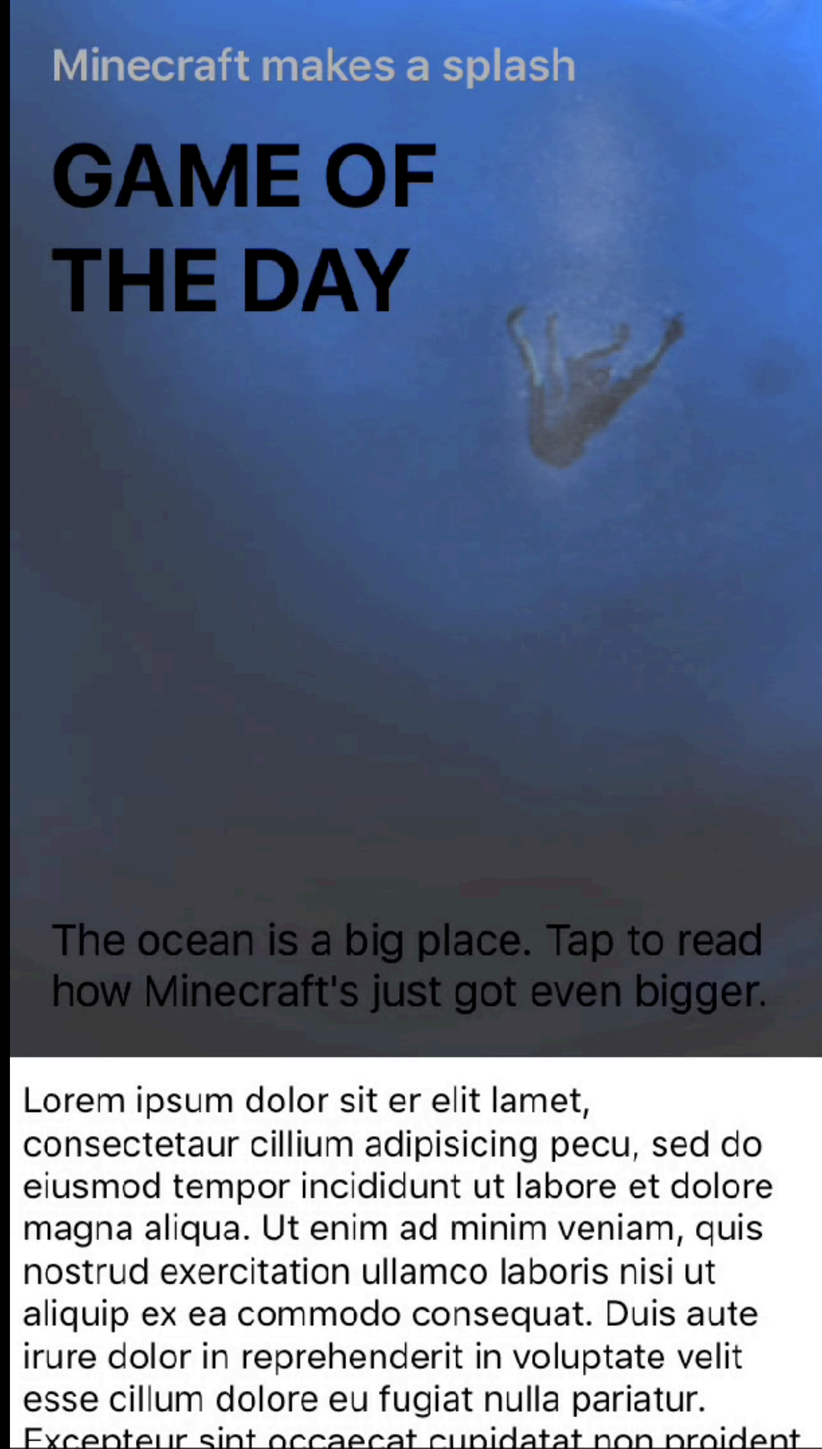


Card detail VC

Animator

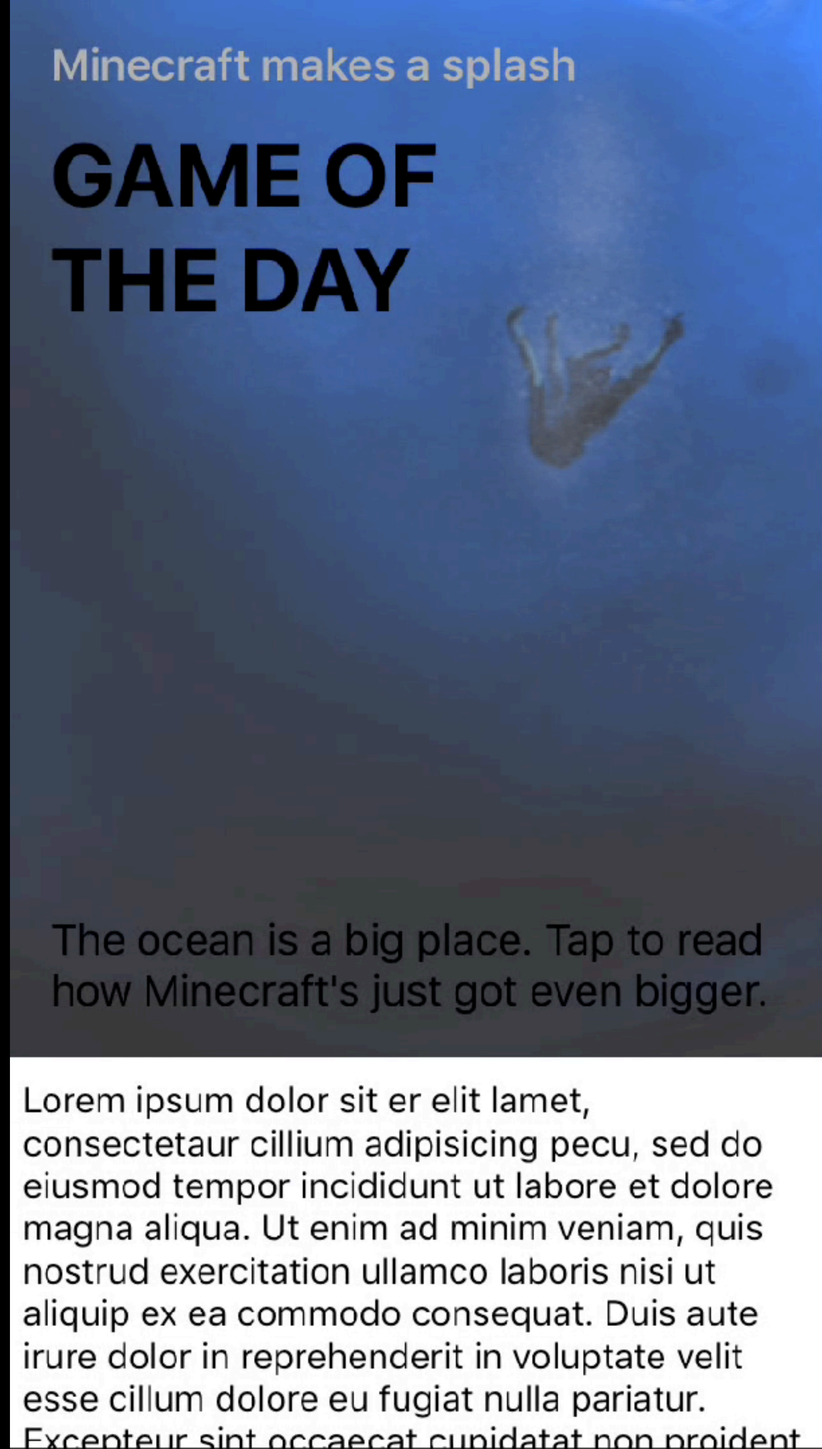
4 Interactive Dismissing

How many gesture recognizers?



4 Interactive Dismissing

How many gesture recognizers?

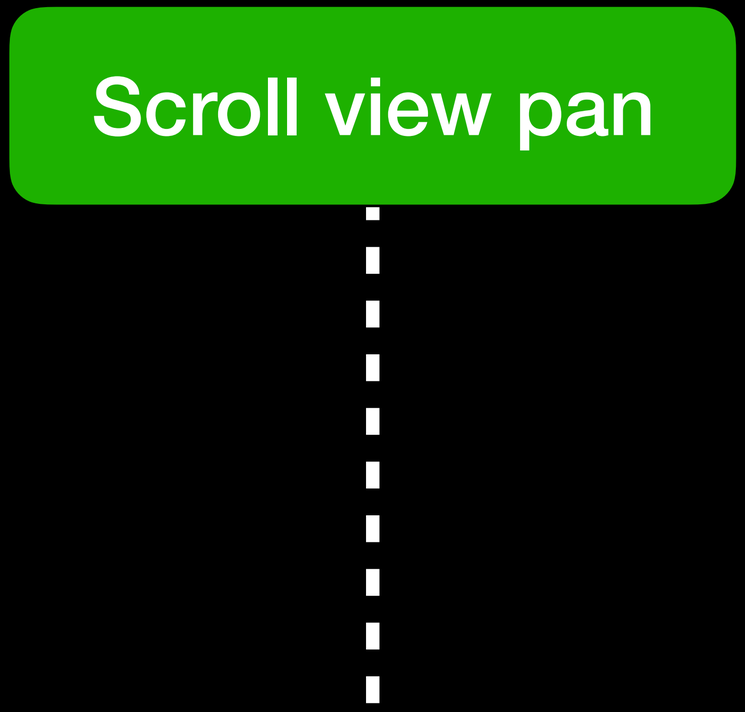


4 Interactive Dismissing

Three pan gesture recognizers, simultaneously

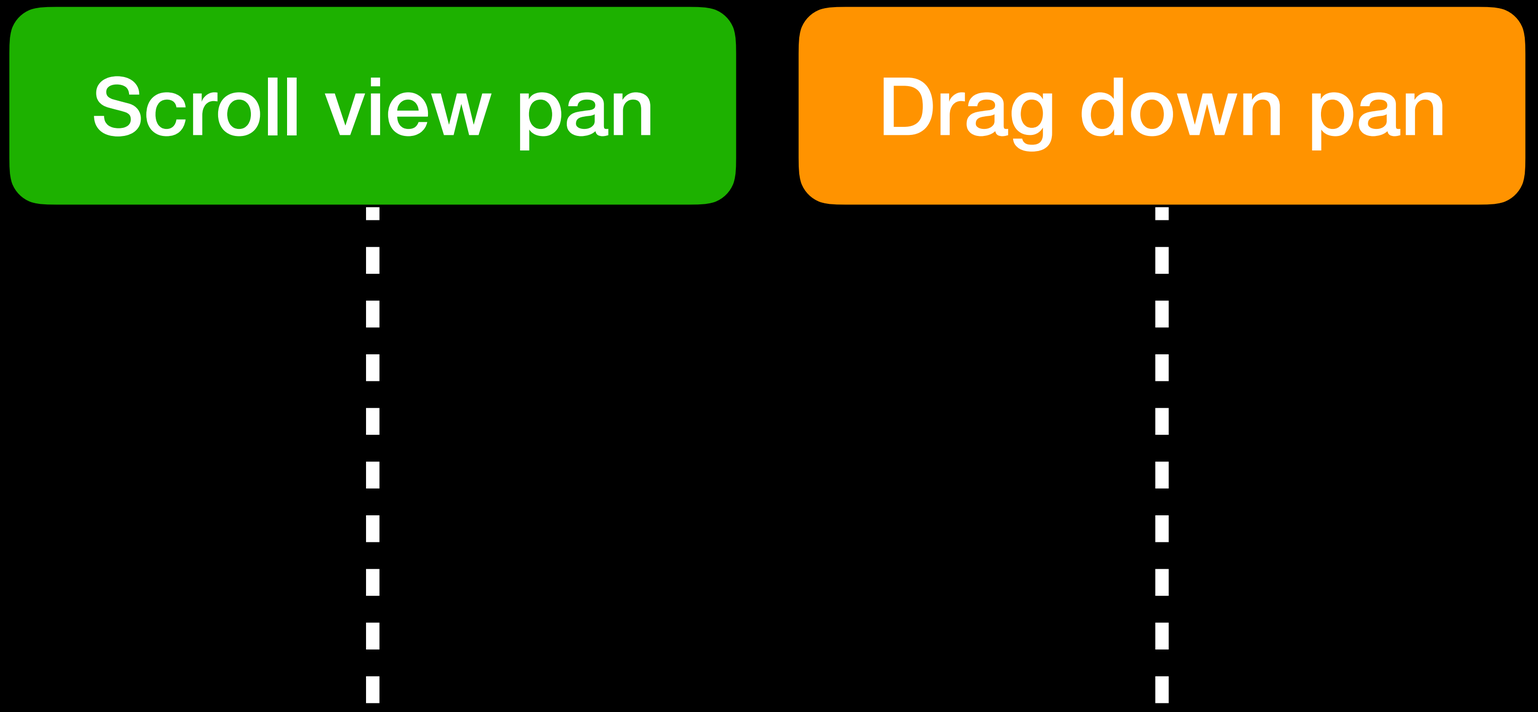
4 Interactive Dismissing

Three pan gesture recognizers, simultaneously



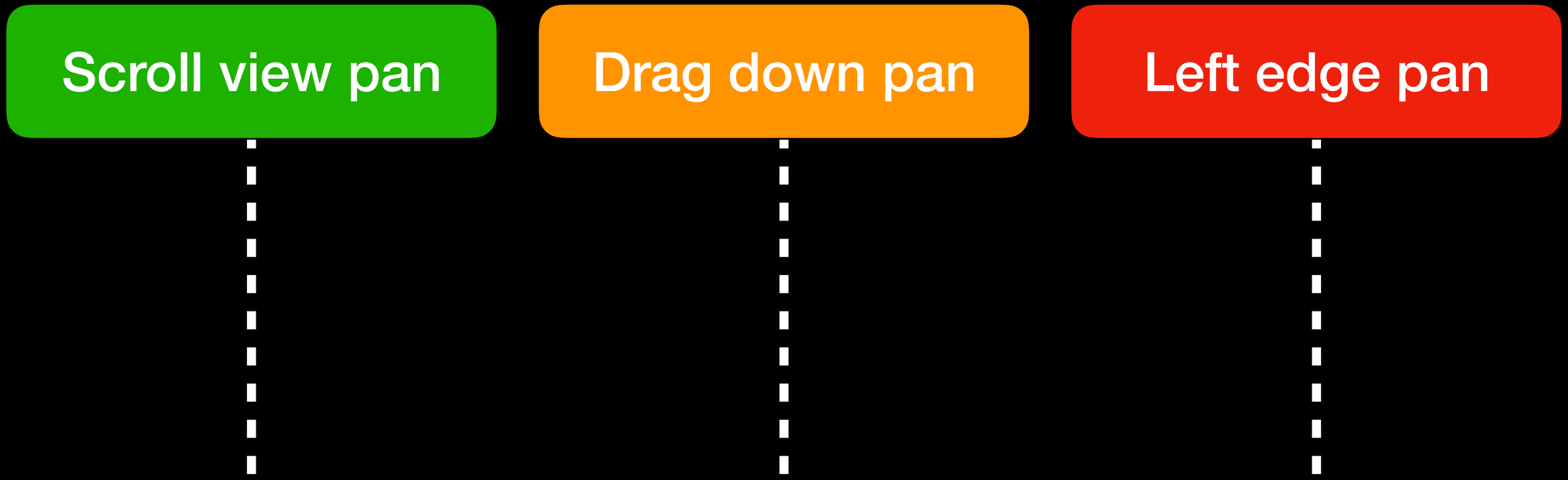
4 Interactive Dismissing

Three pan gesture recognizers, simultaneously



4 Interactive Dismissing

Three pan gesture recognizers, simultaneously



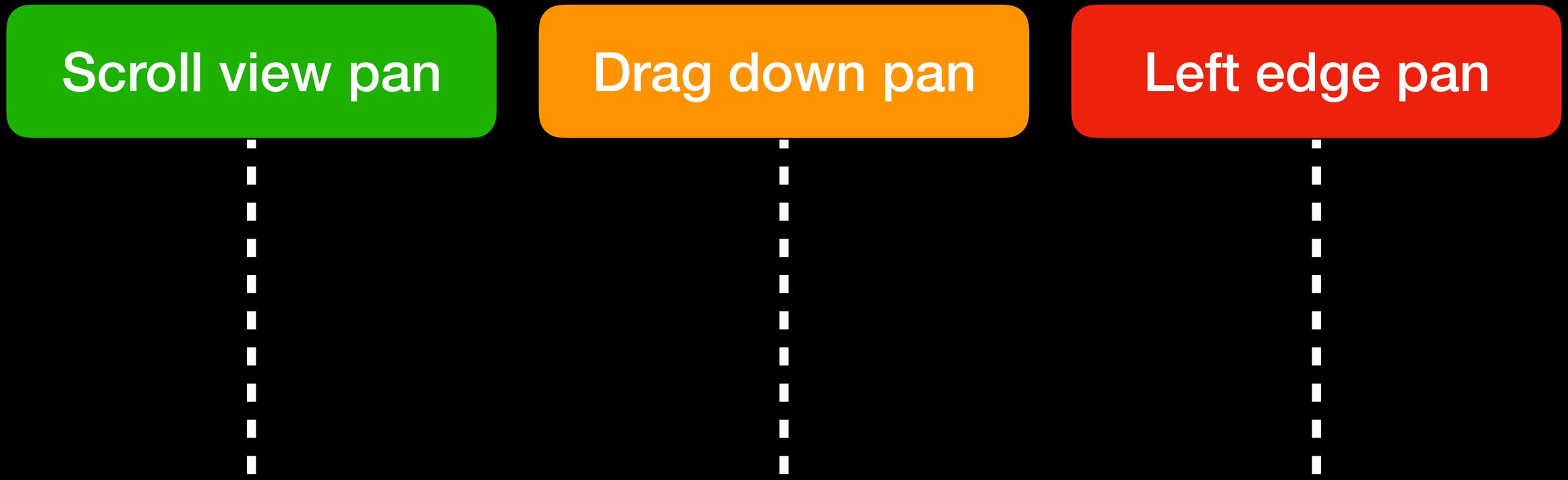
4 Interactive Dismissing

Three pan gesture recognizers, simultaneously

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
shouldRecognizeSimultaneouslyWith otherGestureRecognizer:
UIGestureRecognizer) -> Bool {
    return true
}
```

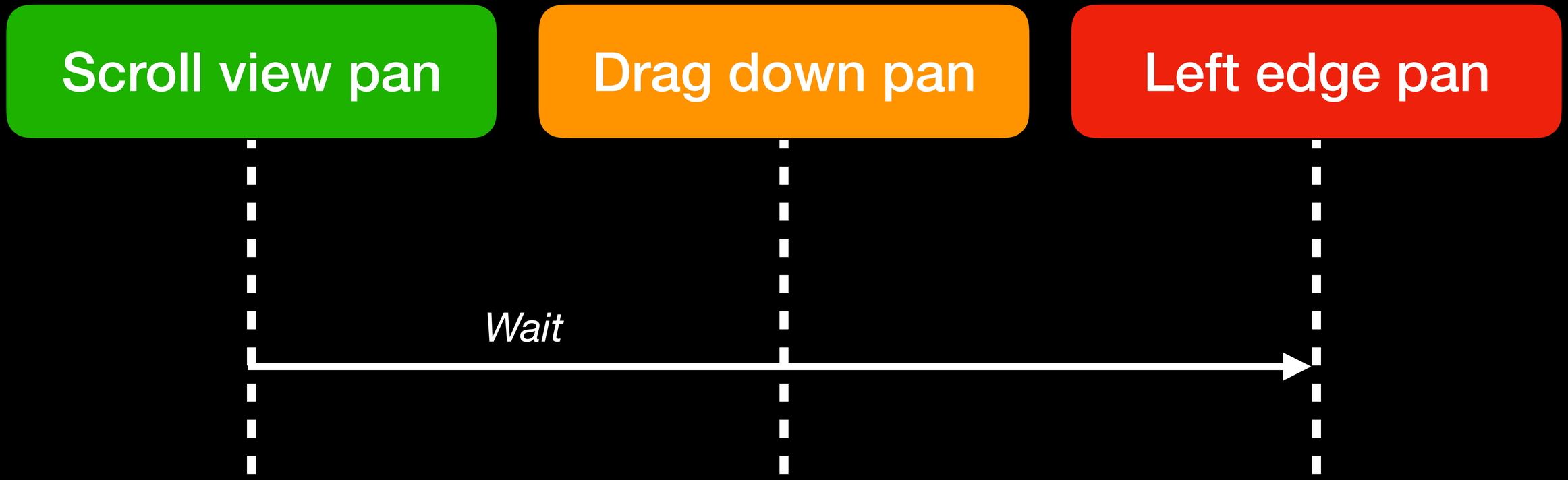
4 Interactive Dismissing

More priority to left screen edge pan



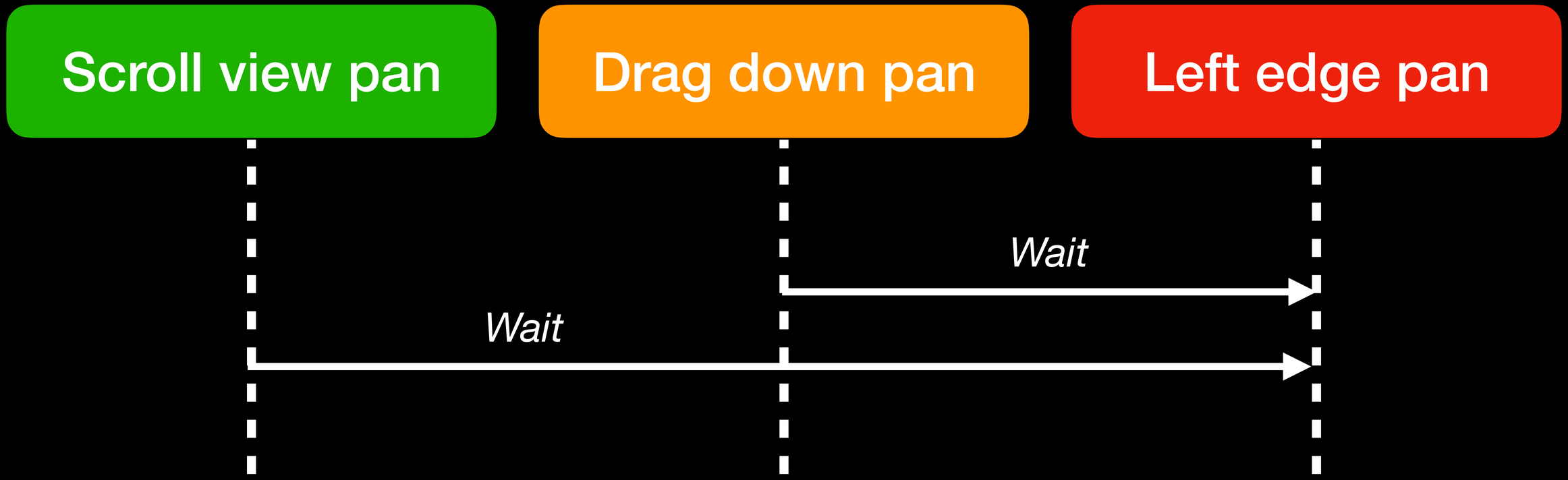
4 Interactive Dismissing

More priority to left screen edge pan



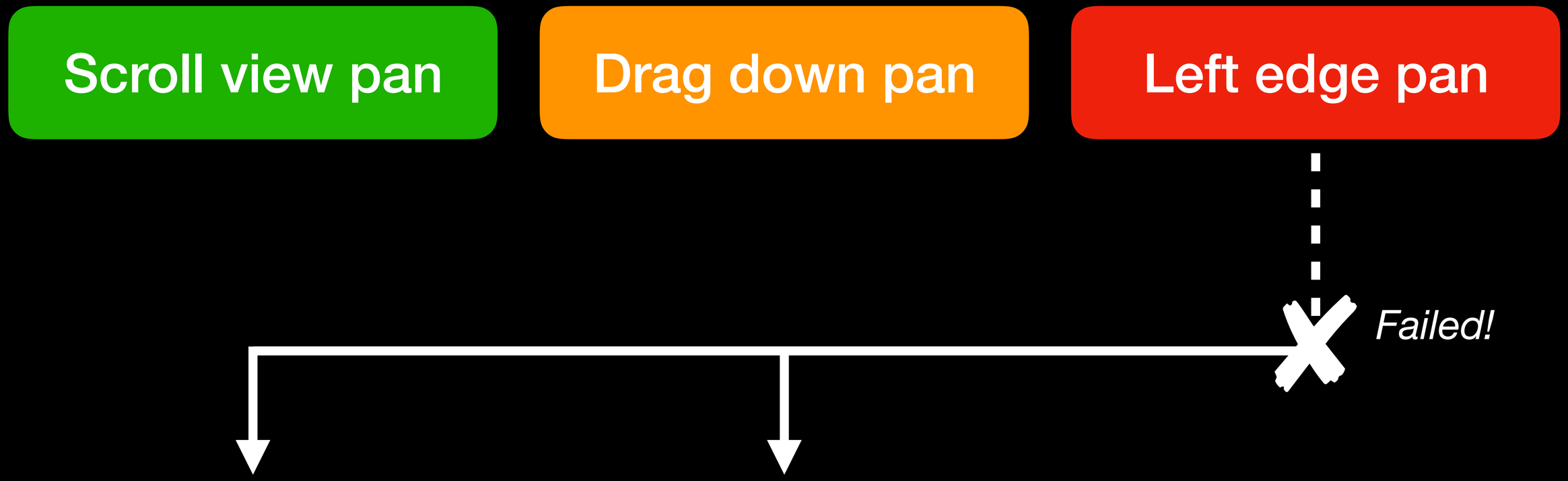
4 Interactive Dismissing

More priority to left screen edge pan



4 Interactive Dismissing

More priority to left screen edge pan



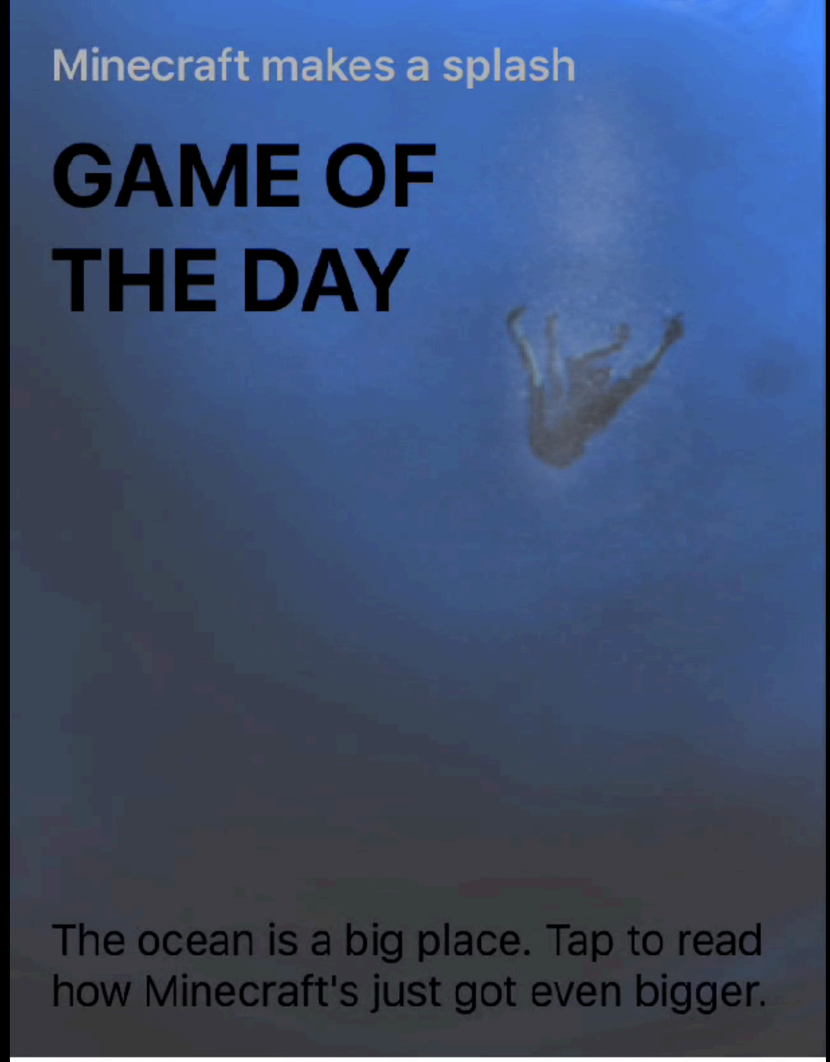
4 Interactive Dismissing

More priority to left screen edge pan

```
dragDownPan.require(toFail: leftEdgePan)

scrollView.panGestureRecognizer.require(toFail: leftEdgePan)
```

4 Interactive Dismissing

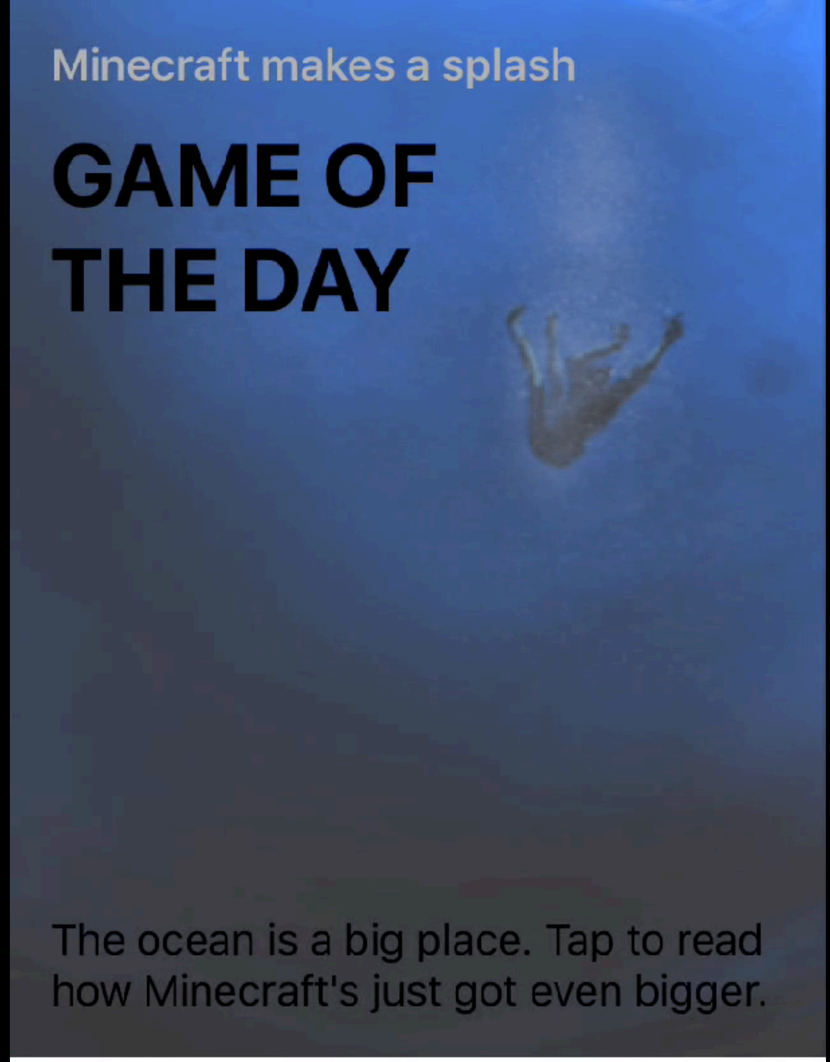
Detect mode in scrollViewDidScroll



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

Detect mode in scrollViewDidScroll



Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipisicing pecu, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false

func scrollViewDidScroll(_ scrollView: UIScrollView) {
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
        scrollView.contentOffset = .zero
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
        scrollView.contentOffset = .zero
    }
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
        scrollView.contentOffset = .zero
    }
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
        scrollView.contentOffset = .zero
    }

    scrollView.showsVerticalScrollIndicator = !draggingDownToDismiss
```

4 Interactive Dismissing

Detect mode in scrollViewDidScroll

```
var draggingDownToDismiss = false


func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if draggingDownToDismiss || (scrollView.isTracking &&
scrollView.contentOffset.y < 0) {
        draggingDownToDismiss = true
        scrollView.contentOffset = .zero
    }

    scrollView.showsVerticalScrollIndicator = !draggingDownToDismiss
}
```

4 Interactive Dismissing

Use UIViewPropertyAnimator

```
let shrinking = UIViewPropertyAnimator(duration: 0, curve: .linear,
animations: {
    self.view.transform = .init(scaleX: 0.8, y: 0.8)
    self.view.layer.cornerRadius = 16
})

shrinking.pauseAnimation()
```

4 Interactive Dismissing

Calculate progress

4 Interactive Dismissing

Calculate progress

```
let edgePanProgress = gesture.translation(in: view).x / 100)
```

4 Interactive Dismissing

Calculate progress

```
let edgePanProgress = gesture.translation(in: view).x / 100)

let dragDownProgress = (currentLoc.y - startingLoc.y) / 100
```

4 Interactive Dismissing

Calculate progress

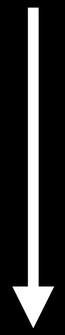
```
let edgePanProgress = gesture.translation(in: view).x / 100)

let dragDownProgress = (currentLoc.y - startingLoc.y) / 100
```

```
gesture.location(in: nil)
```

4 Interactive Dismissing

Calculate progress

```
let edgePanProgress = gesture.translation(in: view).x / 100)

let dragDownProgress = (currentLoc.y - startingLoc.y) / 100
```



```
gesture.location(in: nil)
```

4 Interactive Dismissing

Calculate progress

```
let edgePanProgress = gesture.translation(in: view).x / 100)

let dragDownProgress = (currentLoc.y - startingLoc.y) / 100
```

`gesture.location(in: nil)`

On entering drag down mode

4 Interactive Dismissing

Update shrinking progress

```
shrinking.fractionComplete = progress
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
    self.didCancelDismissalTransition()
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
    self.didCancelDismissalTransition()
    gesture.isEnabled = true
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
    self.didCancelDismissalTransition()
    gesture.isEnabled = true
}
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
    self.didCancelDismissalTransition()
    gesture.isEnabled = true
}
```

4 Interactive Dismissing

Reverse animation on gesture ended/cancelled

```
shrinking!.pauseAnimation()
shrinking!.isReversed = true

// Disable gesture until reverse closing animation finishes.
gesture.isEnabled = false
shrinking!.addCompletion { [unowned self] (pos) in
    self.didCancelDismissalTransition()
    gesture.isEnabled = true
}

shrinking!.startAnimation()
```

5 Dismissing

5 Dismissing

On reaching progress 1.0

5 Dismissing

On reaching progress 1.0

```
dismiss(animated: true)
```

5 Dismissing

On reaching progress 1.0

```
dismiss(animated: true)
```

AutoLayout animation back to original card position

Thank you!

https://github.com/aunnnn/AppStoreiOS11InteractiveTransition

https://aunnnn.com