13년 자료가 너무 오래되어서
여기도 한번 참고 해 보자
https://developer.apple.com/library/archive/featuredarticles/ViewControllerPGforiPhoneOS/index.html

Custom Transitions Using View Controllers

New capabilities, APIs and enhancements

그리고 내용이 너무 어려워서
이론만 이해하는것 보다는 직접 구현을 해보면서 체득 하는게 낫다

https://github.com/HeroTransitions/Hero 라이브러리

Session 218

Bruce D. Nilo
View Controller Mechanic

Roadmap

Roadmap

- New animation tools
- Custom view controller transitions
- Interactive view controller transitions
- Canceling and coordinating transitions

iOS 7에서 개선된 트랜지션 애니메이션

Roadmap

New animation tools

Roadmap

New animation tools

- Quick review of the block based UIView animation API

블록 기반 애니메이션 API

Roadmap

New animation tools

- Quick review of the block based UIView animation API
- Spring animations

UIView 블록기반 Spring 애니메이션

Roadmap

New animation tools

- Quick review of the block based UIView animation API
- Spring animations
- Key-frame animations

UIKit 레벨에서 keyframe 애니메이션을 다루는 것이 힘들었다.

기존에는 더 low level에서 가능했으나 이것은 더 쉽게 하겠다.

Roadmap

New animation tools

- Quick review of the block based UIView animation API
- Spring animations
- Key-frame animations
- UIKit Dynamics

UIKit Dynamics 에서도 한 이야기가 많음

(일종의 물리엔진 같은 것 같음)

이것들을 이야기 하고 (custom view control) 트랜지션이 어떻게 작동하는지 볼것이다.

Roadmap

Custom view controller transitions

Roadmap

Custom view controller transitions

- Which transitions can be customized?
  - Presentations and dismissals
  - UITabBarController
  - UINavigationController
  - UICollectionViewController layout-to-layout transitions

커스텀 가능한 트랜지션은 크게 3.5개 정도로 볼 수 있다.

Roadmap

Custom view controller transitions

- Which transitions can be customized?
  - Presentations and dismissals
  - UITabBarController
  - UINavigationController
  - UICollectionViewController layout-to-layout transitions
- What is a transition?
  - Anatomy and generalizations

모두 비슷하게 처리 가능할 것이므로
트랜지션 자체가 무엇인지 알아본다.

Roadmap

Custom view controller transitions

- Which transitions can be customized?
  - Presentations and dismissals
  - UITabBarController
  - UINavigationController
  - UICollectionViewController layout-to-layout transitions
- What is a transition?
  - Anatomy and generalizations
- API discussion with some examples

많은 API를 직접 살펴볼 예정

Roadmap

Interactive view controller transitions

Roadmap

Interactive view controller transitions

- Adding interactivity to custom transitions

iOS 6에는 페이지 뷰 컨트롤러가 있었다.
앞 뒤로 스와이프 가능한데 이것이 인터랙티브 전환의 예이다.

iOS 7의 팝 제스처도 인터랙티브이다.

우리가 직접 추가 할 수 있다.

Roadmap

Interactive view controller transitions

- Adding interactivity to custom transitions
- Special support for UICollectionViews
  - UICollectionViewTransitionLayout

컬렉션뷰를 좀 개선했고, 몇가지 다른 방식으로 인터렉티브 하게 가능하다.
그리고 그 방식을 우리의 커스텀 뷰에서 어떻게 사용할 수 있는지
뒤에서 다시 이야기 하자.

Roadmap

Interactive view controller transitions

- Adding interactivity to custom transitions
- Special support for UICollectionViews
  - UICollectionViewTransitionLayout
- Canceling transitions

상호작용 중에는 마음을 바꿔 다시 되돌릴 수 있다.

손을 반쯤 움직였다가 되돌리는 것 처럼

Roadmap

Interactive view controller transitions

- Adding interactivity to custom transitions
- Special support for UICollectionViews
  - UICollectionViewTransitionLayout
- Canceling transitions
- UITransitionCoordinator
  - Animating alongside transitions
  - Specifying a completion handler
    - This can be used for all UINavigationController transitions!

transition coordinator 라는 것을 도입했다.
처음에는 canceling 때문에 도입했으나 더 많은 활용이 가능하다.

New UIView Animation APIs

Create compelling, custom, view controller transitions

새로운 애니메이션 API

New UIView Animation APIs

Quick review: Existing UIView block based API

New UIView Animation APIs

Quick review: Existing UIView block based API

```
+ (void) beginAnimations:context:
+ (void) commitAnimations
```

ios2 에 두가지 메서드는 흔히 기반 API 가 아님

New UIView Animation APIs

Quick review: Existing UIView block based API

```
+ (void) beginAnimations:context:
+ (void) commitAnimations

+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

iOS4 부터 블록기반 애니메이션을 제공하고 있다.

이 메서드들이 Core Animation과 어떤 관계가 있을까?

New UIView Animation APIs

Quick review: Relationship to core animation

```
[UIView animationWithDuration:delay:options:animations:^{

} completion: nil];
```

UI처 애니메이션 API을 사용하게위해 Core Animation을 알 필요는 없지만
깊게 이해하는데 도움을 줄다

New UIView Animation APIs

Quick review: Relationship to core animation

```
[UIView animationWithDuration:delay:options:animations:^{

} completion: nil];
```

New UIView Animation APIs

Quick review: Relationship to core animation

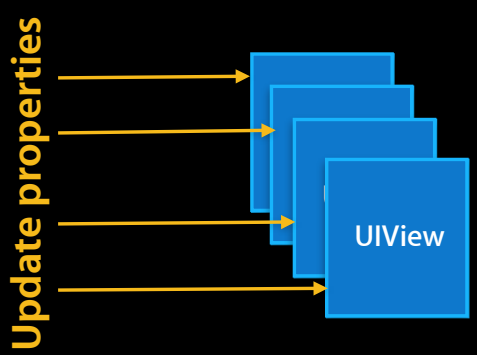
```
[UIView animationWithDuration:delay:options:animations:^{
```


```
} completion: nil];
```

New UIView Animation APIs

Quick review: Relationship to core animation

```
[UIView animationWithDuration:delay:options:animations:^{
```



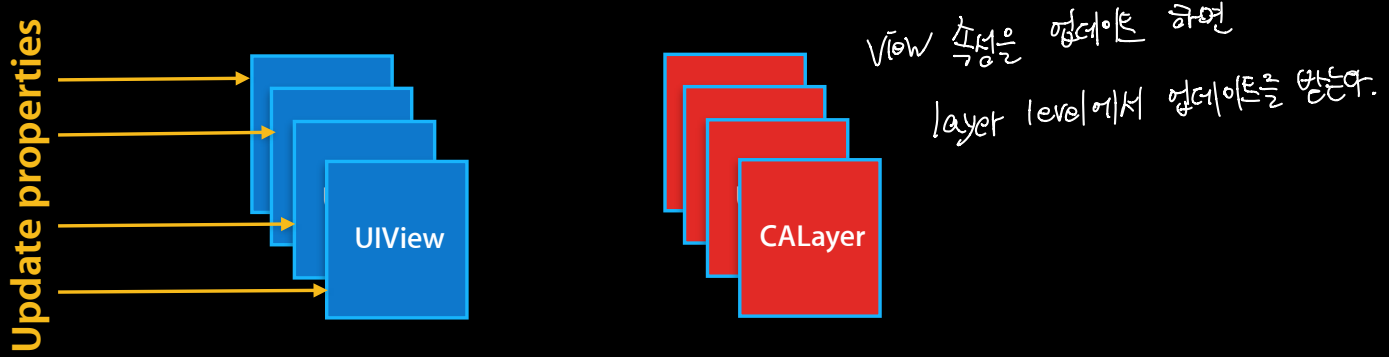
애니메이션 블록에서 View의 속성을 변경할 수 있다.

```
} completion: nil];
```

New UIView Animation APIs

Relationship to core animation

```
[UIView animationWithDuration:...animations:^{
```

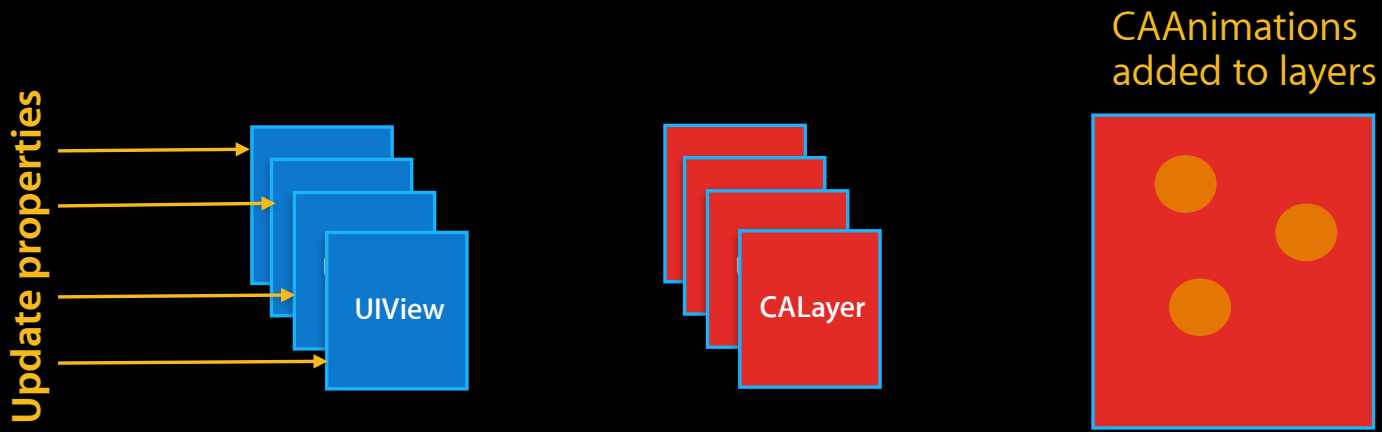



```
} completion: nil];
```

New UIView Animation APIs

Relationship to core animation

```
[UIView animationWithDuration:...animations:^{
```




```
} completion: nil];
```

그리고 Core Animation 객체가
layer에 추가되고 애니메이션을 일으킨다.

New UIView Animation APIs

UIView block based API

- Disabling and enabling animations

```
+ (void)setAnimationsEnabled:(BOOL)
```

가끔 예상하지 않은 효과가 블록 내부에서 일어난다.

애니메이션을 disable 하는 API를 추가했다.

이것을 세팅하고 다시 되돌리는것을 잊으면 버그가 터져나올 수 있다.

New UIView Animation APIs

UIView block based API

```
+ (void)performWithoutAnimation:(void ^(void))actions;
```

New UIView Animation APIs

UIView block based API

```
+ (void)performWithoutAnimation:(void ^(void))actions;
```

그래서 애니메이션 동작이 없이 블록을 실행 가능한 API를 추가 했다.

New UIView Animation APIs

Spring animations

스프링 애니메이션은 알아보자

New UIView Animation APIs

Spring animations

- Two new parameters
  - DampingRatio
    - 1.0 >= r > 0.0
  - Initial Spring Velocity
- Composes nicely with other UIView animation methods

1차원 애니메이터 구현은 알 필요가 없어서 좋다
스프링 세기와 속도만 건드리면 된다.

New UIView Animation APIs

Spring animations

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
     usingSpringWithDamping:(CGFloat)dampingRatio
      initialSpringVelocity:(CGFloat)velocity
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

아주 익숙한 API이다.

New UIView Animation APIs

Spring animations

```
+ (void)animateWithDuration:(NSTimeInterval)duration
                      delay:(NSTimeInterval)delay
     usingSpringWithDamping:(CGFloat)dampingRatio
      initialSpringVelocity:(CGFloat)velocity
                    options:(UIViewAnimationOptions)options
                 animations:(void (^)(void))animations
                 completion:(void (^)(BOOL finished))completion;
```

두가지 인자만 추가된다.

New UIView Animation APIs

Key-frame animations

New UIView Animation APIs

Key-frame animations

- animateKeyframesWithDuration…
  - is to CAKeyframeAnimation

New UIView Animation APIs

Key-frame animations

- animateKeyframesWithDuration…
  - is to CAKeyframeAnimation
- as animateWithDuration…
  - is to CABasicAnimation

New UIView Animation APIs

Key-frame animations

- animateKeyframesWithDuration…
  - is to CAKeyframeAnimation
- as animateWithDuration…
  - is to CABasicAnimation
- Specify keyframes within the animation block
  - Options augmented to include calculation mode

애니메이션 블록에서 키프레임을 생성하고 계산모드도 추가했다.

New UIView Animation APIs

Key-frame animations

- animateKeyframesWithDuration…
  - is to CAKeyframeAnimation
- as animateWithDuration…
  - is to CABasicAnimation
- Specify keyframes within the animation block
  - Options augmented to include calculation mode
- Composes nicely with other UIView animation methods

다른 애니메이션과 멋지게 같이 사용할 수 있다.

New View Based Animation APIs

Key-frame animations

```
+ (void)animateKeyframesWithDuration:(NSTimeInterval)duration
                               delay:(NSTimeInterval)delay
                             options:(UIViewKeyframeAnimationOptions)options
                          animations:(void (^)(void))animations
                          completion:(void (^)(BOOL finished))completion;

+ (void)addKeyframeWithRelativeStartTime:(double)frameStartTime
                        relativeDuration:(double)frameDuration
                              animations:(void (^)(void))animations
```

키프레임 추가 하는 법

New View Based Animation APIs

Key-frame animations

```
+ (void)animateKeyframesWithDuration:(NSTimeInterval)duration
                               delay:(NSTimeInterval)delay
                             options:(UIViewKeyframeAnimationOptions)options
                          animations:(void (^)(void))animations
                          completion:(void (^)(BOOL finished))completion;
```

```
+ (void)addKeyframeWithRelativeStartTime:(double)frameStartTime
                        relativeDuration:(double)frameDuration
                              animations:(void (^)(void))animations
```

New View Based Animation APIs

Key-frame animations

```
+ (void)animateKeyframesWithDuration:(NSTimeInterval)duration
                               delay:(NSTimeInterval)delay
                             options:(UIViewKeyframeAnimationOptions)options
                          animations:(void (^)(void))animations
                          completion:(void (^)(BOOL finished))completion;
```

```
+ (void)addKeyframeWithRelativeStartTime:(double)frameStartTime
                        relativeDuration:(double)frameDuration
                              animations:(void (^)(void))animations
```

New View Based Animation APIs

Key-frame animations

```
[UIView animateKeyframesWithDuration: .35
 delay: 0.0
 options:0
 animations:^{
     [UIView addKeyframe… animations: ^{…}];
     [UIView addKeyframe… animations:^{…}];
     [UIView addKeyframe… animations:^{
     [someView setPosition:…];
     // etc.
     }];
     }
 completion:^(BOOL finished) {…}];
```

New View Based Animation APIs

Key-frame animations

```
[UIView animateKeyframesWithDuration: .35
 delay: 0.0
 options:0
 animations:^{
     [UIView addKeyframe… animations: ^{…}];
     [UIView addKeyframe… animations:^{…}];
     [UIView addKeyframe… animations:^{
     [someView setPosition:…];
     // etc.
     }];
     }
 completion:^(BOOL finished) {…}];
```

New View Based Animation APIs

Key-frame animations

```
[UIView animateKeyframesWithDuration: .35
 delay: 0.0
 options:0
 animations:^{
     [UIView addKeyframe… animations: ^{…}];
     [UIView addKeyframe… animations:^{…}];
     [UIView addKeyframe… animations:^{
     [someView setPosition:…];
     // etc.
     }];
     }
 completion:^(BOOL finished) {…}];
```

Improved and Simplified Snapshot API

UIView snapshots

스냅샷 API를 사용하면 놀라운 전환이 가능

Improved and Simplified Snapshot API

UIView snapshots

- Snapshot API

```
- (UIView *)[UIView snapshotView]
- (UIView *)[UIView resizableSnapshotViewFromRect:(CGRect)rect
                                withCapInsets:(UIEdgeInsets)capInsets
```

Improved and Simplified Snapshot API

UIView snapshots

- Snapshot API

```
- (UIView *)[UIView snapshotView]
- (UIView *)[UIView resizableSnapshotViewFromRect:(CGRect)rect
                                withCapInsets:(UIEdgeInsets)capInsets
```

- Creating snapshots from snapshots is supported

스냅샷의 스냅샷을 만들 수 있고 매우 유용함

New View Based Animation APIs

UIKit Dynamics

New View Based Animation APIs

UIKit Dynamics

- Distinct from UIView animation APIs

New View Based Animation APIs

UIKit Dynamics

- Distinct from UIView animation APIs
  - Compatible with the new transitioning APIs
  - More in this afternoon's talk

UIKit Dynamics 새로운 트랜지션 API와 호환됨.
더 자세한건 다른 세션에서 이야기 한 듯

Customizing Your View Controller Transitions

It’s easy to use

Custom View Controller Transitions

Which transitions can be customized?

Custom View Controller Transitions

Which transitions can be customized?

- Presentations and dismissals
  - Supported presentation styles
    - `UIModalPresentationFullScreen`
    - `UIModalPresentationCustom`

Custom style이 생김

Custom View Controller Transitions

Which transitions can be customized?

- Presentations and dismissals
  - Supported presentation styles
    - `UIModalPresentationFullScreen`
    - `UIModalPresentationCustom`

```
The from view controller is not removed from the window hierarchy
```

Custom View Controller Transitions

Which transitions can be customized?

- Presentations and dismissals
  - Supported presentation styles
    - `UIModalPresentationFullScreen`
    - `UIModalPresentationCustom`

```
The from view controller is not removed from the window hierarchy

UIViewController *vc = …;
id <UIViewControllerTransitioningDelegate> transitioningDelegate;
vc.modalPresentationStyle = UIModalPresentationCustom;
[vc setTransitioningDelegate: transitioningDelegate];
[self presentViewController:vc animated: YES completion: nil];
```

presentation 스타일을 설정해 delegate를 설정하는것으로 완료된다.

Custom View Controller Transitions

Which transitions can be customized?

- Presentations and dismissals
  - Supported presentation styles
    - `UIModalPresentationFullScreen`
    - `UIModalPresentationCustom`

```
The from view controller is not removed from the window hierarchy

UIViewController *vc = …;
id <UIViewControllerTransitioningDelegate> transitioningDelegate;
vc.modalPresentationStyle = UIModalPresentationCustom;
[vc setTransitioningDelegate: transitioningDelegate];
[self presentViewController:vc animated: YES completion: nil];
```

Custom View Controller Transitions

Which transitions can be customized?

Custom View Controller Transitions

Which transitions can be customized?

- UITabBarController

```
setSelectedViewController:(UIViewController *)vc;
setSelectedIndex:(NSUInteger)idx;
```

Custom View Controller Transitions

Which transitions can be customized?

- UITabBarController

```
setSelectedViewController:(UIViewController *)vc;
setSelectedIndex:(NSUInteger)idx;
```

```
NSUInteger secondTab = 1;
self.delegate = tabBarControllerDelegate;
[self setSelectedIndex:secondTab];
```

탭바 컨트롤러

Custom View Controller Transitions

Which transitions can be customized?

Custom View Controller Transitions

Which transitions can be customized?

- UINavigationController

```
pushViewController:animated:
popViewControllerAnimated:
setViewControllers:animated:
```

Custom View Controller Transitions

Which transitions can be customized?

- UINavigationController

```
pushViewController:animated:
popViewControllerAnimated:
setViewControllers:animated:
```

```
self.delegate = navigationControllerDelegate;
[self pushViewController:vc animated:YES];
```

네비게이션 컨트롤러

Custom View Controller Transitions

UINavigationController meets UICollectionViewController

Custom View Controller Transitions

UINavigationController meets UICollectionViewController

- Layout-to-layout navigation transitions

Custom View Controller Transitions

UINavigationController meets UICollectionViewController

- Layout-to-layout navigation transitions

```
UICollectionViewLayout *layout1,*layout2,*layout3;
UICollectionViewController *cvc1, *cvc2, *cvc3;
cvc1 = [cvc1 initWithCollectionViewLayout:layout1];

…

[nav pushViewController:cvc1 animated:YES]
cvc2.useLayoutToLayoutNavigationTransitions = YES;
cvc3.useLayoutToLayoutNavigationTransitions = YES;
[nav pushViewController:cvc2 animated:YES];
[nav pushViewController:cvc3 animated:YES];
[nav popViewControllerAnimated:YES];
```

layout - layout 전환

direction
speed

Basic : slide

댐핑, 속도 ...

Spring : Bounce

성능이 좋다? - UIKit dynamics 덕?

Keyframe : Fold

Demo

Some examples of custom transitions

잡히면서 전환

Push, presentation 같은코드로 동작

단에 따른 delegate 사용

Collection view

콜렉션 item 누르면
네비게이션 전환되게
pop 제스처 가능

UIKit dynamics

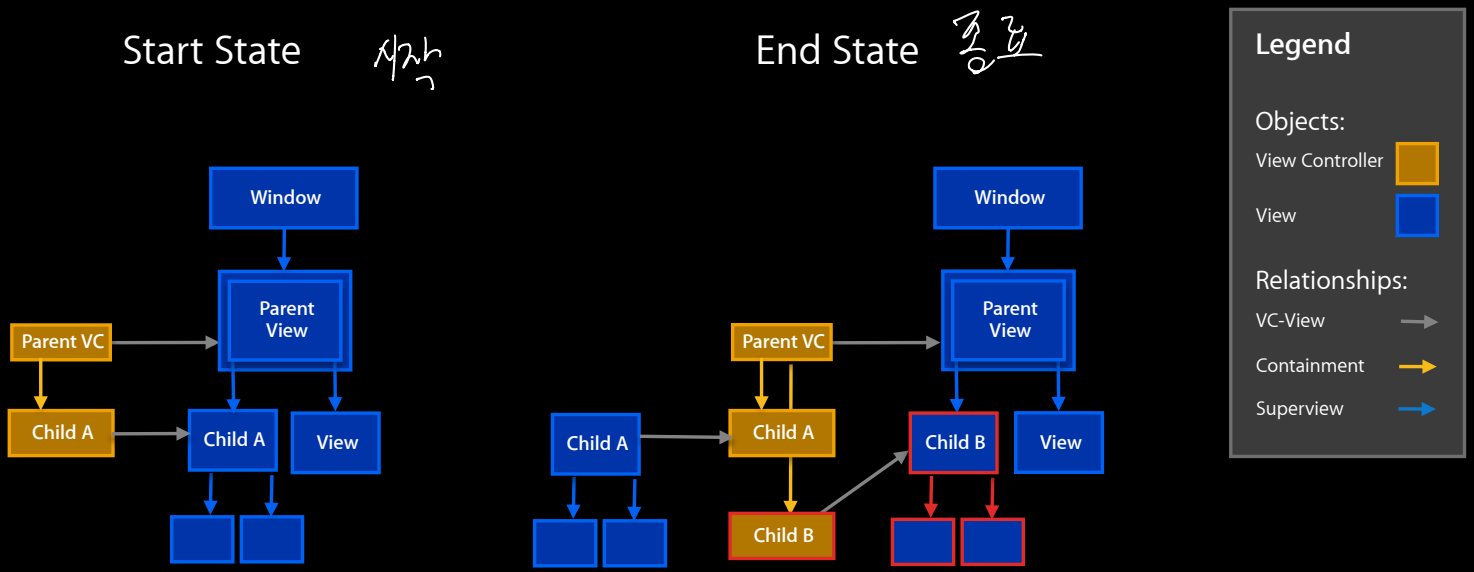
끌려오면서 떨어지는 전환

Customizing Your View Controller Transitions

Concepts and APIs

Custom View Controller Transitions

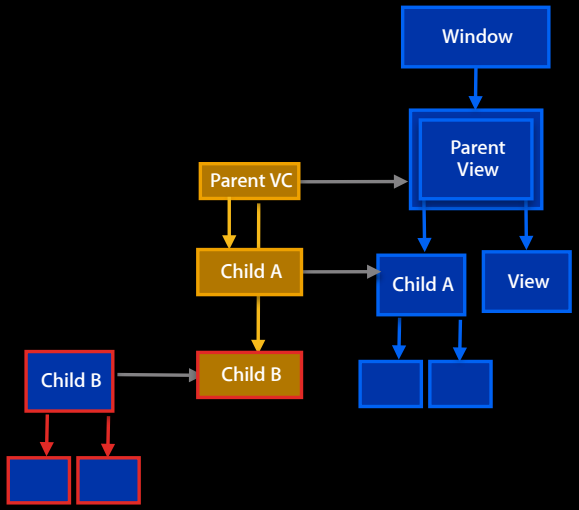
The anatomy of a transition



Custom View Controller Transitions

The anatomy of a transition

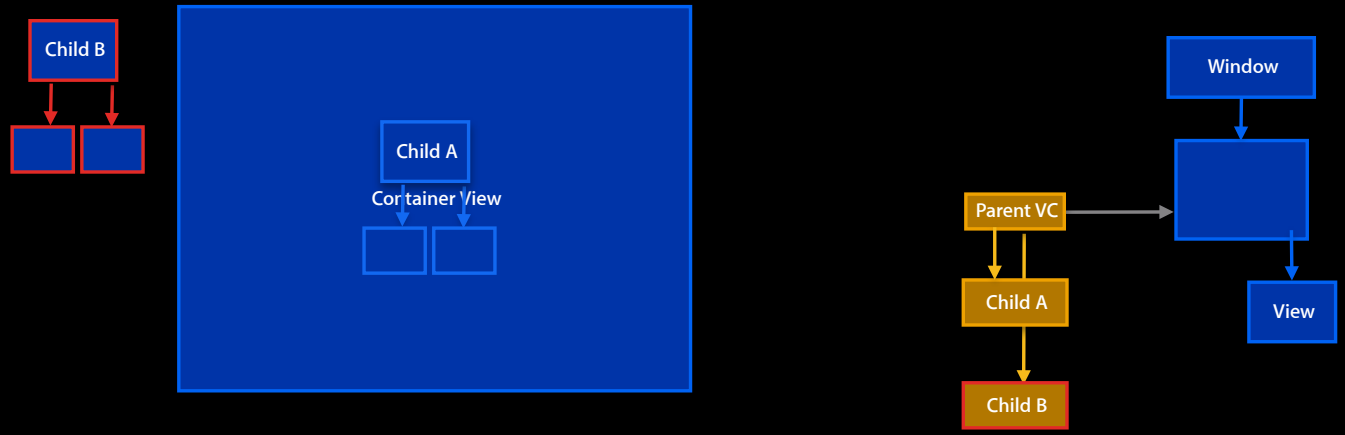
Intermediate State



Custom View Controller Transitions

The anatomy of a transition

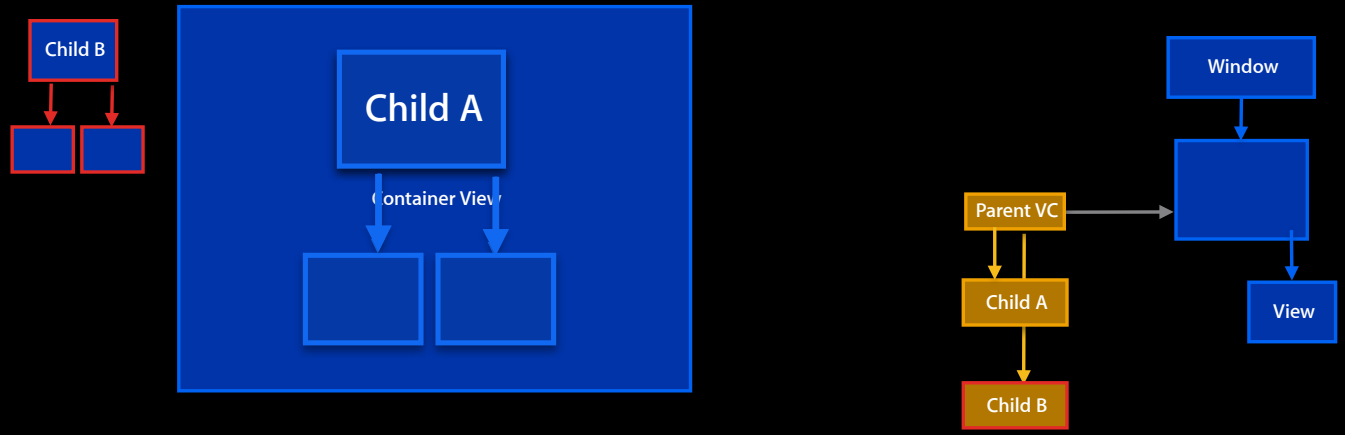
Intermediate State



Custom View Controller Transitions

The anatomy of a transition

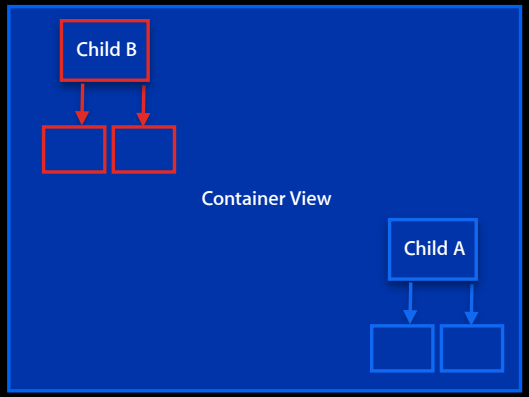
Intermediate State

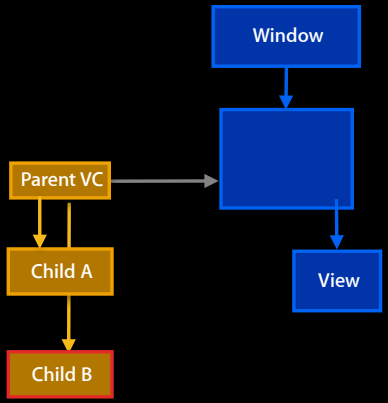


Custom View Controller Transitions

The anatomy of a transition

Intermediate State

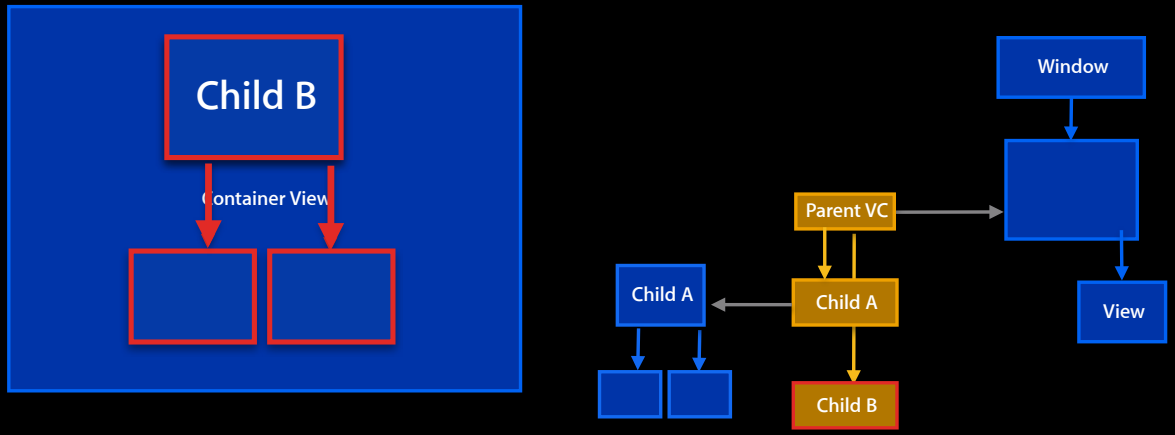




Custom View Controller Transitions

The anatomy of a transition

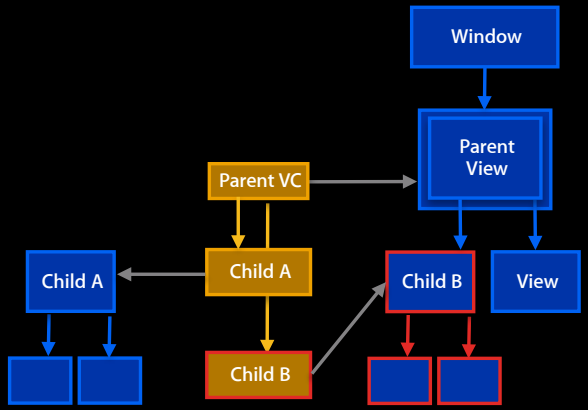
Intermediate State



Custom View Controller Transitions

The anatomy of a transition

End State



Custom View Controller Transitions

The anatomy of a transition

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.
- Container view, and start and final view positions are computed

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.
- Container view, and start and final view positions are computed
- Optional animation to end state view hierarchy is run

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.
- Container view, and start and final view positions are computed
- Optional animation to end state view hierarchy is run
- Animation completes
  - Internal structures are updated, callbacks made, etc.

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.
- Container view, and start and final view positions are computed
- Optional animation to end state view hierarchy is run
- Animation completes
  - Internal structures are updated, callbacks made, etc.
- End State
  - Consistent view controller hierarchy and view hierarchy

Custom View Controller Transitions

The anatomy of a transition

- Start state
  - Consistent view controller hierarchy and view hierarchy
- User or programmatic transition commences
- Internal structures are updated, callbacks made, etc.
- Container view, and start and final view positions are computed
- Optional animation to end state view hierarchy is run
- Animation completes
  - Internal structures are updated, callbacks made, etc.
- End State
  - Consistent view controller hierarchy and view hierarchy

Custom View Controller Transitions

<UIViewControllerContextTransitioning>

```
@protocol UIViewControllerContextTransitioning <NSObject>

// The view in which the animated transition should take place.
- (UIView *)containerView;

// Two keys for the  method below are currently defined by the system
// UITransitionContextToViewControllerKey, and
UITransitionContextFromViewControllerKey.

- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

// This MUST be called whenever a transition completes (or is cancelled.)
- (void)completeTransition:(BOOL)didComplete;
...
@end
```

Custom View Controller Transitions

<UIViewControllerContextTransitioning>

```
@protocol UIViewControllerContextTransitioning <NSObject>

// The view in which the animated transition should take place.
- (UIView *)containerView;

// Two keys for the  method below are currently defined by the system
// UITransitionContextToViewControllerKey, and
UITransitionContextFromViewControllerKey.

- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

// This MUST be called whenever a transition completes (or is cancelled.)
- (void)completeTransition:(BOOL)didComplete;
...
@end
```

Custom View Controller Transitions

<UIViewControllerContextTransitioning>

```
@protocol UIViewControllerContextTransitioning <NSObject>

// The view in which the animated transition should take place.
- (UIView *)containerView;

// Two keys for the  method below are currently defined by the system
// UITransitionContextToViewControllerKey, and
UITransitionContextFromViewControllerKey.

- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

// This MUST be called whenever a transition completes (or is cancelled.)
- (void)completeTransition:(BOOL)didComplete;
...
@end
```

시작 종료

Custom View Controller Transitions

<UIViewControllerContextTransitioning>

```
@protocol UIViewControllerContextTransitioning <NSObject>

// The view in which the animated transition should take place.
- (UIView *)containerView;

// Two keys for the  method below are currently defined by the system
// UITransitionContextToViewControllerKey, and
UITransitionContextFromViewControllerKey.

- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

// This MUST be called whenever a transition completes (or is cancelled.)
- (void)completeTransition:(BOOL)didComplete;
...
@end
```

Custom View Controller Transitions

<UIViewControllerAnimatedTransitioning>

```
@protocol UIViewControllerAnimatedTransitioning <NSObject>

- (NSTimeInterval)transitionDuration:(id <UIViewControllerContextTransitioning>)ctx;

// This method can only  be a nop if the transition is interactive and not a
percentDriven interactive transition.
- (void)animateTransition:(id <UIViewControllerContextTransitioning>)ctx;

@optional

// This is a convenience and if implemented will be invoked by the system when the
transition context's completeTransition: method is invoked.
- (void)animationEnded:(BOOL) transitionCompleted;

@end
```

Custom View Controller Transitions

<UIViewControllerAnimatedTransitioning>

```
@protocol UIViewControllerAnimatedTransitioning <NSObject>
```

전환에 걸리는 시간

```
- (NSTimeInterval)transitionDuration:(id <UIViewControllerContextTransitioning>)ctx;

// This method can only  be a nop if the transition is interactive and not a
percentDriven interactive transition.
- (void)animateTransition:(id <UIViewControllerContextTransitioning>)ctx;

@optional

// This is a convenience and if implemented will be invoked by the system when the
transition context's completeTransition: method is invoked.
- (void)animationEnded:(BOOL) transitionCompleted;

@end
```

Custom View Controller Transitions

<UIViewControllerAnimatedTransitioning>

```
@protocol UIViewControllerAnimatedTransitioning <NSObject>

- (NSTimeInterval)transitionDuration:(id <UIViewControllerContextTransitioning>)ctx;

// This method can only  be a nop if the transition is interactive and not a
percentDriven interactive transition.
- (void)animateTransition:(id <UIViewControllerContextTransitioning>)ctx;

@optional

// This is a convenience and if implemented will be invoked by the system when the
transition context's completeTransition: method is invoked.
- (void)animationEnded:(BOOL) transitionCompleted;

@end
```

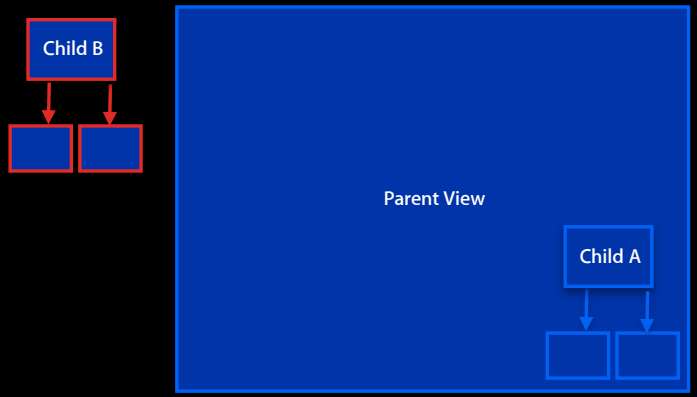
선언만 가능

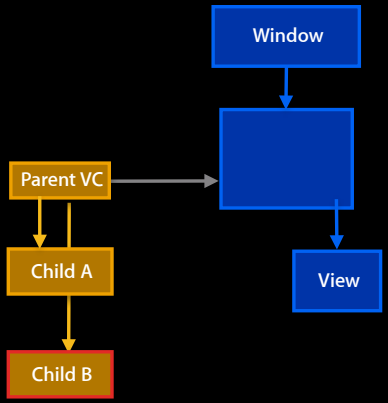
Custom View Controller Transitions

The anatomy of a transition

시작과 종료 사이로 대체 가능과

push 가 끝나면 전환 애니메이션이 동작됨

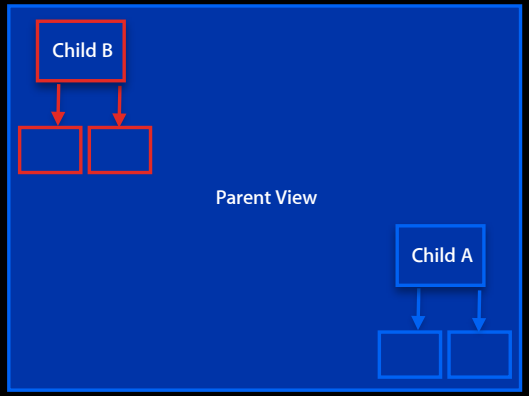


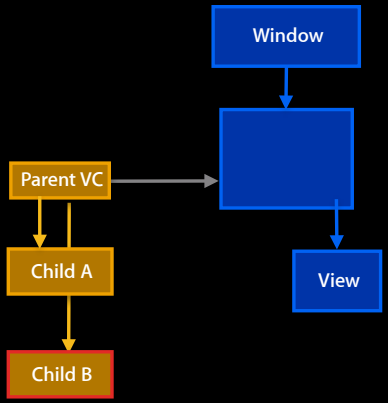


Custom View Controller Transitions

The anatomy of a transition

```
(id <UIViewControllerContexTransitioning>) context;
[animationController animateTransition: context];
```

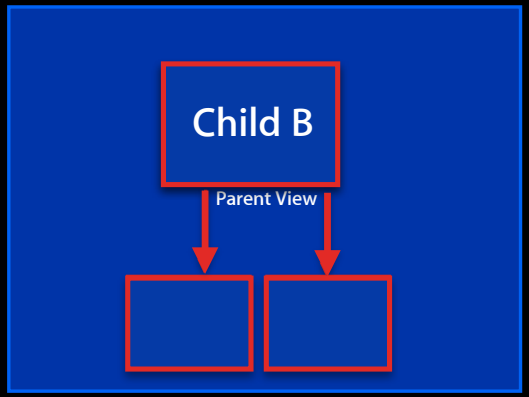


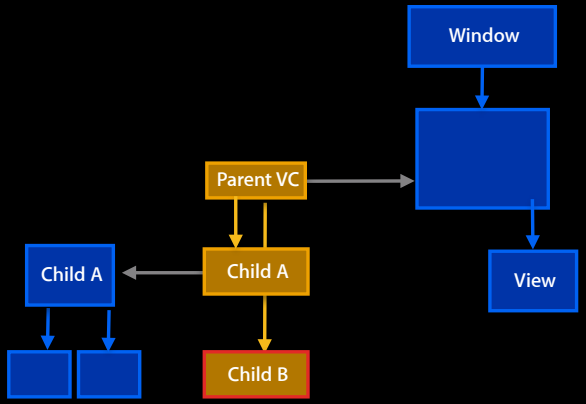


Custom View Controller Transitions

The anatomy of a transition

```
(id <UIViewControllerContexTransitioning>) context;
[animationController animateTransition: context];
```

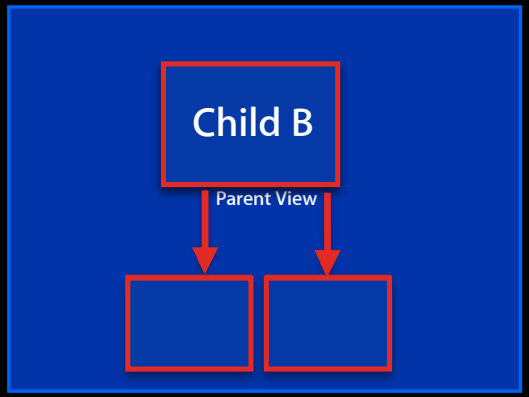




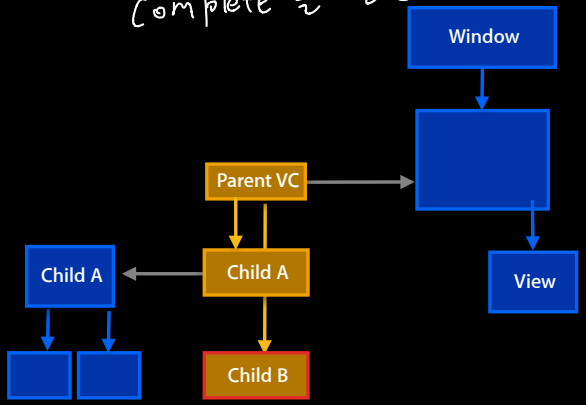
Custom View Controller Transitions

The anatomy of a transition

```
(id <UIViewControllerContexTransitioning>) context;
[context completeTransition: YES];
```

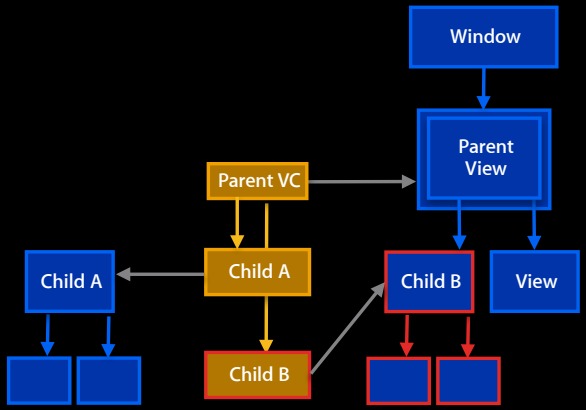


Context가
Complete 를 알림



Custom View Controller Transitions

The anatomy of a transition



Custom View Controller Transitions

Wiring it all together

Custom View Controller Transitions

Wiring it all together

- Animation and interaction controllers are vended by delegates

```
<UIViewControllerTransitioningDelegate>
<UINavigationControllerDelegate>
<UITabBarControllerDelegate>
```

관련된 object들은 delegate가 제공

Custom View Controller Transitions

Wiring it all together

- Animation and interaction controllers are vended by delegates

```
<UIViewControllerTransitioningDelegate>
<UINavigationControllerDelegate>
<UITabBarControllerDelegate>
```

- Animation controllers conform to a protocol

```
<UIViewControllerAnimatedTransitioning>
```

Custom View Controller Transitions

Wiring it all together

- Animation and interaction controllers are vended by delegates

```
<UIViewControllerTransitioningDelegate>
<UINavigationControllerDelegate>
<UITabBarControllerDelegate>
```

- Animation controllers conform to a protocol

```
<UIViewControllerAnimatedTransitioning>
```

- Interaction controllers conform to a protocol

```
<UIViewControllerInteractiveTransitioning>
```

Custom View Controller Transitions

Wiring it all together

- Animation and interaction controllers are vended by delegates

```
<UIViewControllerTransitioningDelegate>
<UINavigationControllerDelegate>
<UITabBarControllerDelegate>
```

- Animation controllers conform to a protocol

```
<UIViewControllerAnimatedTransitioning>
```

- Interaction controllers conform to a protocol

```
<UIViewControllerInteractiveTransitioning>
```

새로운 전환 protocol

- A system object passed to the controllers conforms to

```
<UIViewControllerContextTransitioning>
```

용법은 위에서

Custom View Controller Transitions

Start of a custom presentation

Presented Controller

Presenting Controller

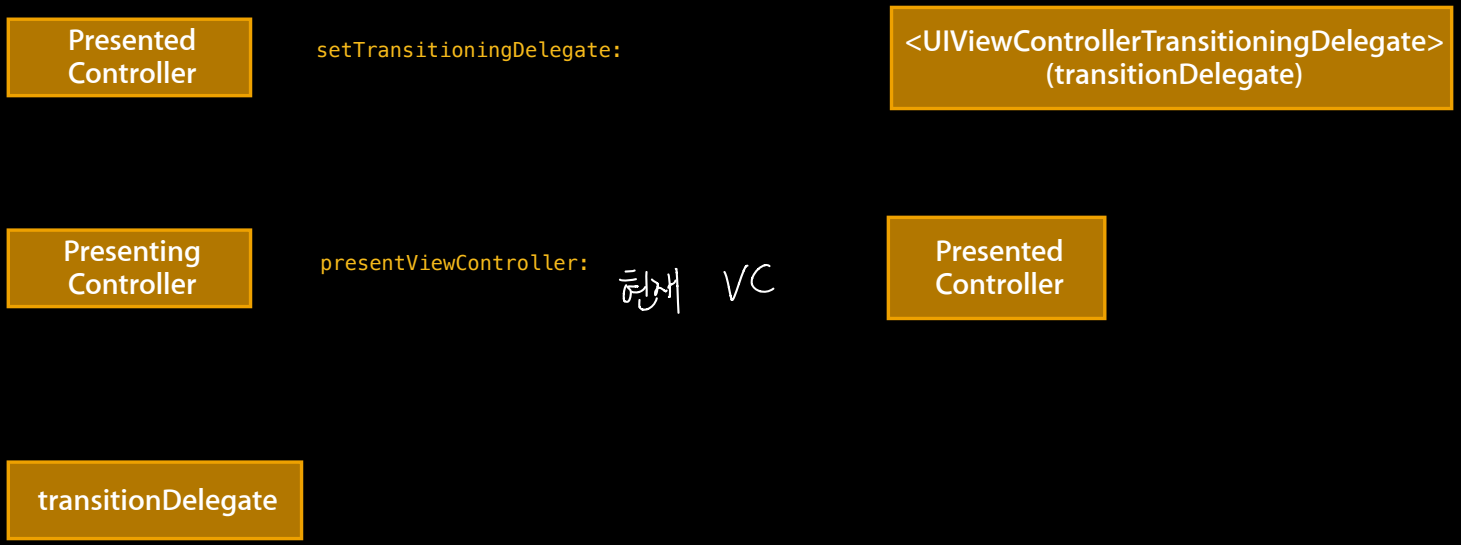
transitionDelegate

Custom View Controller Transitions

Start of a custom presentation

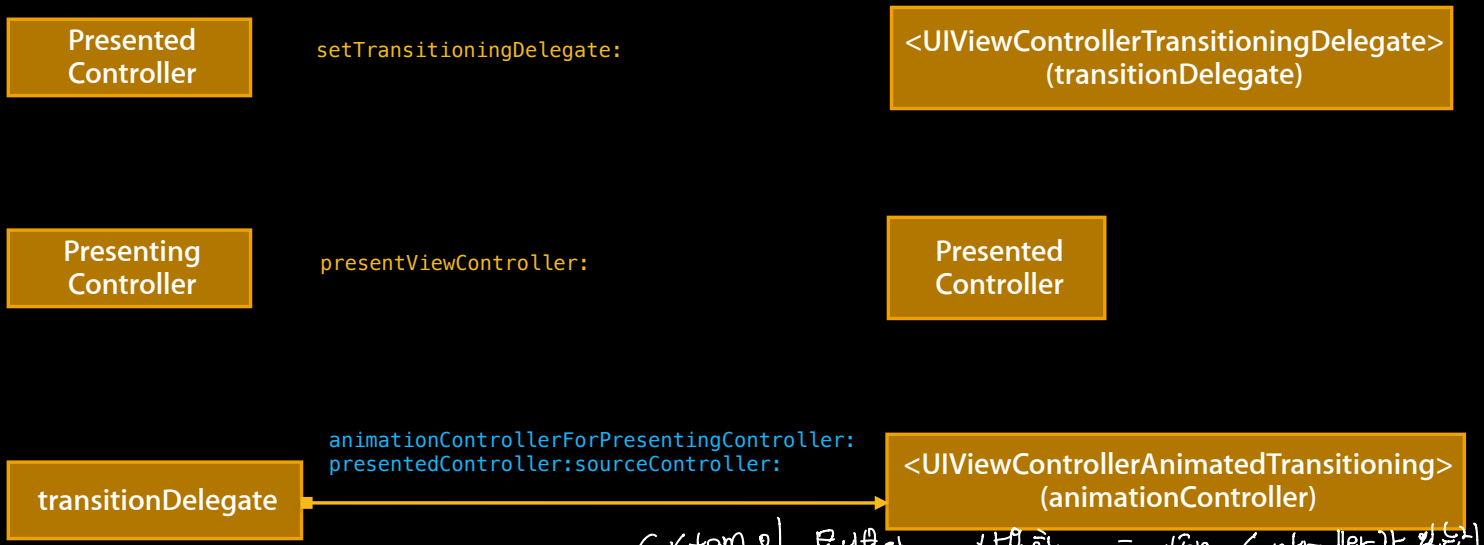
Custom View Controller Transitions

Start of a custom presentation



Custom View Controller Transitions

Start of a custom presentation



delegate에서 이것을 구현 했다면 작업을 수행한다.

Custom View Controller Transitions

End of a custom presentation

animationController

animationController

context

Custom View Controller Transitions

End of a custom presentation

animationController

transitionDuration:

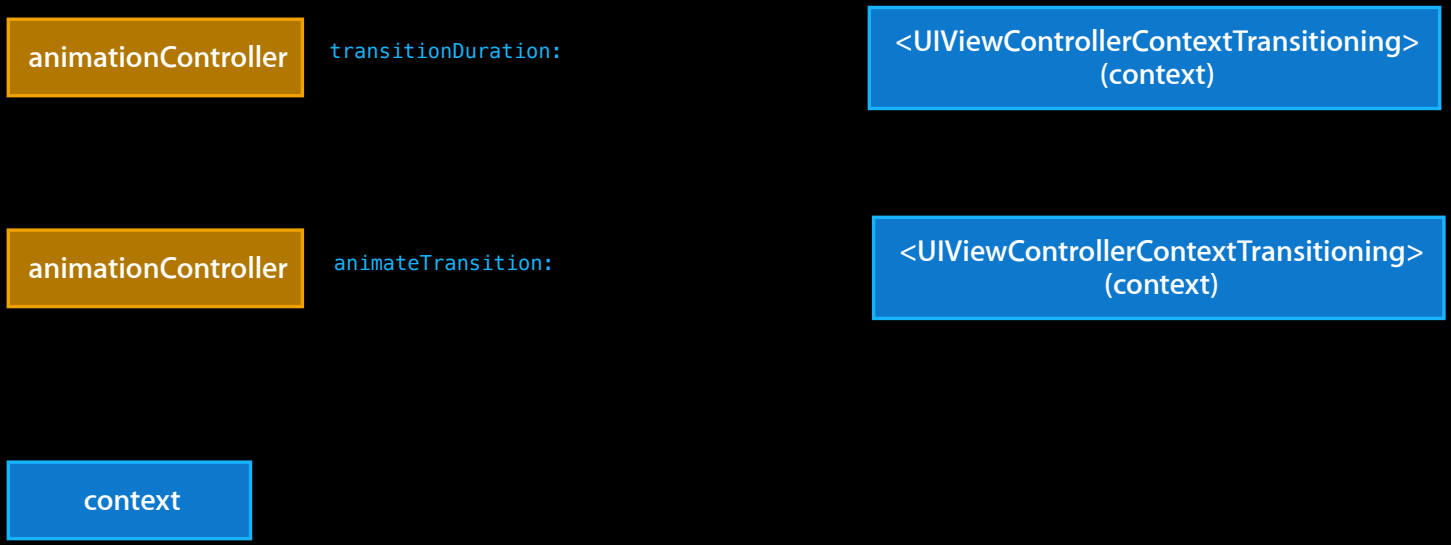
<UIViewControllerContextTransitioning>
(context)

animationController

context

Custom View Controller Transitions

End of a custom presentation



Custom View Controller Transitions

End of a custom presentation

animationController `transitionDuration:` <UIViewControllerContextTransitioning> (context)

animationController `animateTransition:` <UIViewControllerContextTransitioning> (context)

context `completeTransition:` animationController

애니메이션 완료되면 Complete 호출

Custom View Controller Transitions

Pseudo-code of a custom presentation

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
id <UIViewControllerTransitioningDelegate> delegate;
[presentedController setTransitioningDelegate:delegate];
[presentedController setModalPresentationStyle: UIModalPresentationCustom];
[self presentViewController:presentedController animated: YES
completion:nil];
```

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
id <UIViewControllerTransitioningDelegate> delegate;
[presentedController setTransitioningDelegate:delegate];
[presentedController setModalPresentationStyle: UIModalPresentationCustom];
[self presentViewController:presentedController animated: YES
completion:nil];
```

```
id <UIViewControllerAnimatedTransitioning> animationController =
[delegate animationControllerForPresentedController: presented
                                 presentingController: presenter
                                     sourceController: target];
```

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
id <UIViewControllerContextTransitioning>)ctx;
NSTimeInterval duration =  [animationController transitionDuration:ctx];
[animationController animateTransition:ctx];
```

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
id <UIViewControllerContextTransitioning>)ctx;
NSTimeInterval duration =  [animationController transitionDuration:ctx];
[animationController animateTransition:ctx];
```

실제 코드로 보자

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
- (void)animateTransition:(id <UIViewControllerContextTransitioning>ctx {
    UIView *inView = [ctx containerView];
    UIView *toView = [[ctx viewControllerForKey:…]  view];
    UIView *fromView = [[ctx viewControllerForKey: …];
    CGSize size = toEndFrame.size;

    if(self.isPresentation) {
       …
      [inView addSubview: toView];
    }
    else {
       …
      [inView insertSubview:toView belowSubview: [fromVC view]];
    }

    [UIView animateWithDuration: self.transitionDuration animations: ^ {
        if(self.isPresentation) {
            toView.center = newCenter;
            toView.bounds = newBounds;
        }
        else {
        …
        } completion: ^(BOOL finished) { [ctx completeTransition: YES];}];
}
```

프레젠트를 구현 하는곳에서 subview 추가

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
- (void)animateTransition:(id <UIViewControllerContextTransitioning>ctx {
    UIView *inView = [ctx containerView];
    UIView *toView = [[ctx viewControllerForKey:…]  view];
    UIView *fromView = [[ctx viewControllerForKey: …];
    CGSize size = toEndFrame.size;

    if(self.isPresentation) {
        …
      [inView addSubview: toView];
    }
    else {
        …
      [inView insertSubview:toView belowSubview: [fromVC view]];
    }

    [UIView animateWithDuration: self.transitionDuration animations: ^ {
        if(self.isPresentation) {
            toView.center = newCenter;
            toView.bounds = newBounds;
        }
        else {
        …
        } completion: ^(BOOL finished) { [ctx completeTransition: YES];}];
}
```

완료된 코드

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
- (void)animateTransition:(id <UIViewControllerContextTransitioning>ctx {
    UIView *inView = [ctx containerView];
    UIView *toView = [[ctx viewControllerForKey:…]  view];
    UIView *fromView = [[ctx viewControllerForKey: …];
    CGSize size = toEndFrame.size;

    if(self.isPresentation) {
        …
      [inView addSubview: toView];
    }
    else {
        …
      [inView insertSubview:toView belowSubview: [fromVC view]];
    }

    [UIView animateWithDuration: self.transitionDuration animations: ^ {
        if(self.isPresentation) {
            toView.center = newCenter;
            toView.bounds = newBounds;
        }
        else {
        …
        } completion: ^(BOOL finished) { [ctx completeTransition: YES];}];
}
```

Custom View Controller Transitions

Pseudo-code of a custom presentation

```
id <UIViewControllerContextTransitioning>)ctx;
[ctx completeTransition:YES];
```

Custom View Controller Transitions

UIViewControllerTransitioningDelegate

```
@protocol UIViewControllerTransitioningDelegate <NSObject>

@optional

- (id <UIViewControllerAnimatedTransitioning>)
       animationControllerForPresentedController:(UIVC *)presented
                            presentingController:(UIVC *)presenting
                                sourceController:(UIVC *)source;

- (id <UIViewControllerAnimatedTransitioning>)
       animationControllerForDismissedController:(UIVC *)dismissed;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForPresentation:(id <UIViewControllerAnimatedTransitioning>)a;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForDismissal:(id <UIViewControllerAnimatedTransitioning>)a;

@end
```

Custom View Controller Transitions

UIViewControllerTransitioningDelegate

```
@protocol UIViewControllerTransitioningDelegate <NSObject>

@optional

- (id <UIViewControllerAnimatedTransitioning>)
      animationControllerForPresentedController:(UIVC *)presented
                           presentingController:(UIVC *)presenting
                               sourceController:(UIVC *)source;

- (id <UIViewControllerAnimatedTransitioning>)
      animationControllerForDismissedController:(UIVC *)dismissed;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForPresentation:(id <UIViewControllerAnimatedTransitioning>)a;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForDismissal:(id <UIViewControllerAnimatedTransitioning>)a;

@end
```

Custom View Controller Transitions

UIViewControllerTransitioningDelegate

```
@protocol UIViewControllerTransitioningDelegate <NSObject>

@optional

- (id <UIViewControllerAnimatedTransitioning>)
      animationControllerForPresentedController:(UIVC *)presented
                           presentingController:(UIVC *)presenting
                               sourceController:(UIVC *)source;

- (id <UIViewControllerAnimatedTransitioning>)
      animationControllerForDismissedController:(UIVC *)dismissed;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForPresentation:(id <UIViewControllerAnimatedTransitioning>)a;

- (id <UIViewControllerInteractiveTransitioning>)
    interactionControllerForDismissal:(id <UIViewControllerAnimatedTransitioning>)a;

@end
```

Interaction

Custom View Controller Transitions

UIViewControllerTransitioningDelegate

```
@interface UIViewController(CustomTransitioning)

@property (nonatomic,retain) id <UIViewControllerTransitioningDelegate>transitionDelegate;

@end
```

Custom View Controller Transitions

UIViewControllerTransitioningDelegate

```
@interface UIViewController(CustomTransitioning)

@property (nonatomic,retain) id <UIViewControllerTransitioningDelegate>transitionDelegate;

@end
```

Custom View Controller Transitions

UINavigationControllerDelegate Extensions

```
- (id <UIViewControllerAnimatedTransitioning>)navigationController:  (UINC *)nc
          animationControllerForOperation: (UINavigationControllerOperation)op
                       fromViewController:(UIViewController *)fromVC
                         toViewController:(UIViewController *)toVC;

- (id <UIViewControllerInteractiveTransitioning>)navigationController: (UINC *)nc
interactionControllerForAnimationController: (id <UIViewControllerAnimatedTransitioning>)a;
```

Custom View Controller Transitions

UITabBarControllerDelegate Extensions

```
- (id <UIViewControllerAnimatedTransitioning>)tabBarController: (UITABC *)tbc
            animationControllerForTransitionFromViewController:(UIVC *)fromVC
                                                    toViewController:(UIVC *)toVC;

- (id <UIViewControllerInteractiveTransitioning>)tabBarController: (UITABC *)tbc
interactionControllerForAnimationController: (id <UIViewControllerAnimatedTransitioning>)a;
```

Custom View Controller Transitions

Responsibilities of the animation controller

Custom View Controller Transitions

Responsibilities of the animation controller

- Implementation of `animateTransition:` and `transitionDuration:`
  - Insertion of “to” view controller’s view into the container view

Custom View Controller Transitions

Responsibilities of the animation controller

- Implementation of `animateTransition:` and `transitionDuration:`
  - Insertion of "to" view controller's view into the container view
- When the transition animation completes
  - The "to" and "from" view controller's views need to be in their designated positions
  - The context's `completeTransition:` method must be invoked

인터랙션 전환은 완전 다른것

Interactive View Controller Transitions

Introduction

Interactive View Controller Transitions

Running transition animations interactively

- UINavigationController
  - Interactive pop gesture is pervasive on iOS 7.0

Interactive View Controller Transitions

Running transition animations interactively

- UINavigationController
  - Interactive pop gesture is pervasive on iOS 7.0
- Applications can define their own interactive transitions

Interactive View Controller Transitions

Running transition animations interactively

- UINavigationController
  - Interactive pop gesture is pervasive on iOS 7.0
- Applications can define their own interactive transitions
  - Interactive transitions need not be gesture driven

제스처 recognizer 를 꼭 사용할 필요는 없다.

Interactive View Controller Transitions

Running transition animations interactively

- UINavigationController
  - Interactive pop gesture is pervasive on iOS 7.0
- Applications can define their own interactive transitions
  - Interactive transitions need not be gesture driven
  - Interactive transitions usually run forwards and backwards
    - Often a transition can start and be cancelled

취소 가능

Interactive View Controller Transitions

Running transition animations interactively

- UINavigationController
  - Interactive pop gesture is pervasive on iOS 7.0
- Applications can define their own interactive transitions
  - Interactive transitions need not be gesture driven
  - Interactive transitions usually run forwards and backwards
    - Often a transition can start and be cancelled
- UIKit provides a concrete interaction controller class
  - UIPercentDrivenInteractiveTransition

Interactive View Controller Transitions

<UIViewControllerInteractiveTransitioning>

```
@protocol UIViewControllerInteractiveTransitioning <NSObject>

- (void)startInteractiveTransition:(id <UIViewControllerContextTransitioning>)ctx;

@optional

- (CGFloat)completionSpeed;
- (UIViewAnimationCurve)completionCurve;

@end
```

Interactive View Controller Transitions

<UIViewControllerInteractiveTransitioning>

```
@protocol UIViewControllerInteractiveTransitioning <NSObject>

- (void)startInteractiveTransition:(id <UIViewControllerContextTransitioning>)ctx;

@optional

- (CGFloat)completionSpeed;
- (UIViewAnimationCurve)completionCurve;

@end
```

Interactive View Controller Transitions

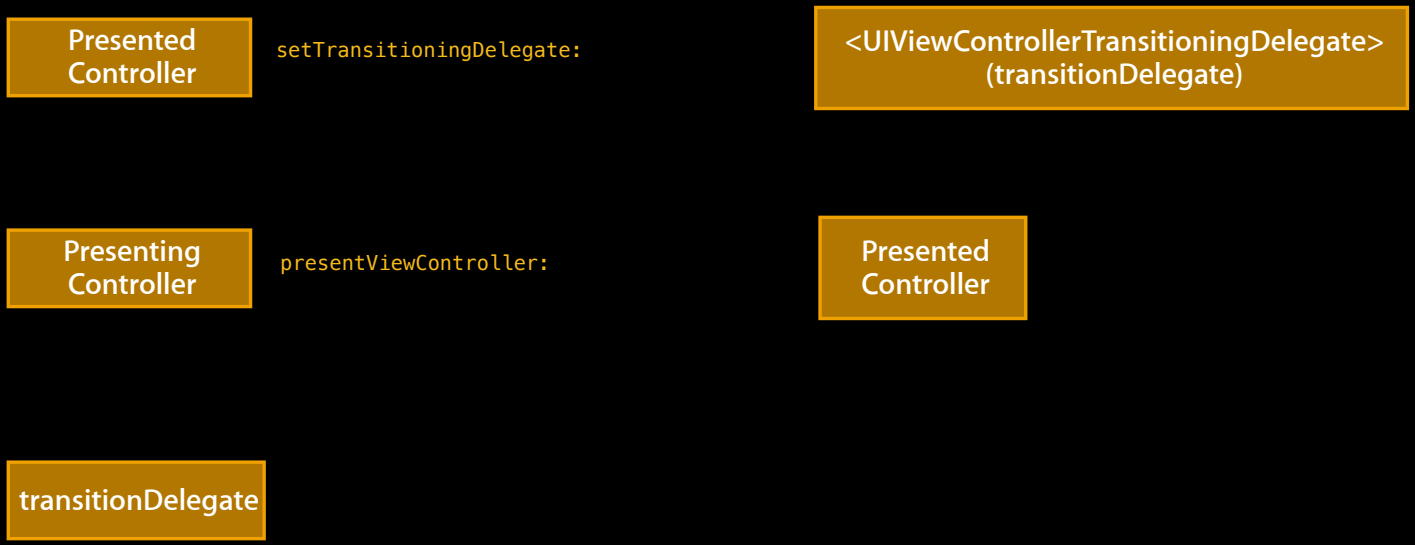
Start of an interactive presentation

Interactive View Controller Transitions

Start of an interactive presentation

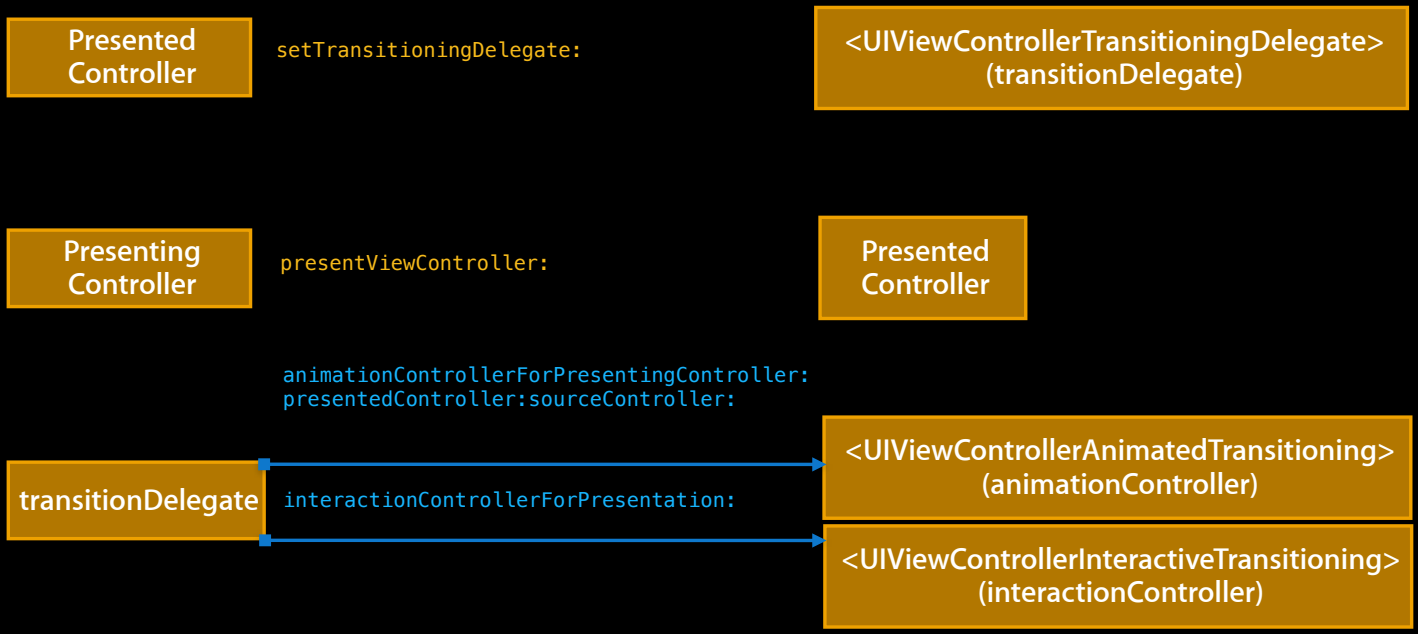
Interactive View Controller Transitions

Start of an interactive presentation



Interactive View Controller Transitions

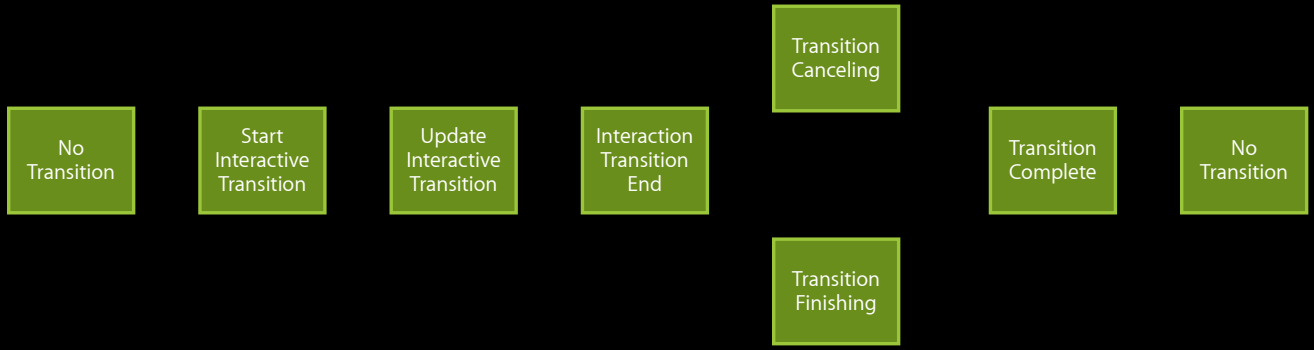
Start of an interactive presentation



UIViewControllerTransitioning

Interactive transitioning states

States

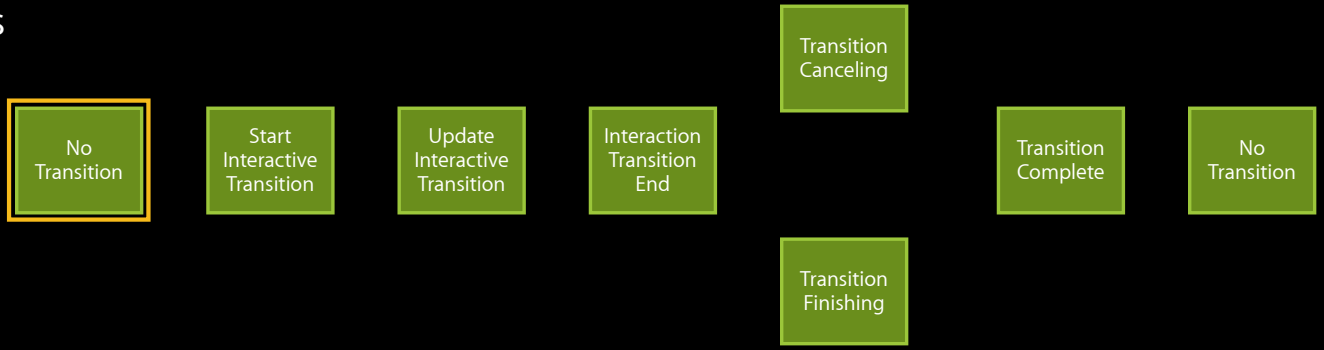


Agents

UIViewControllerTransitioning

Interactive transitioning states

States

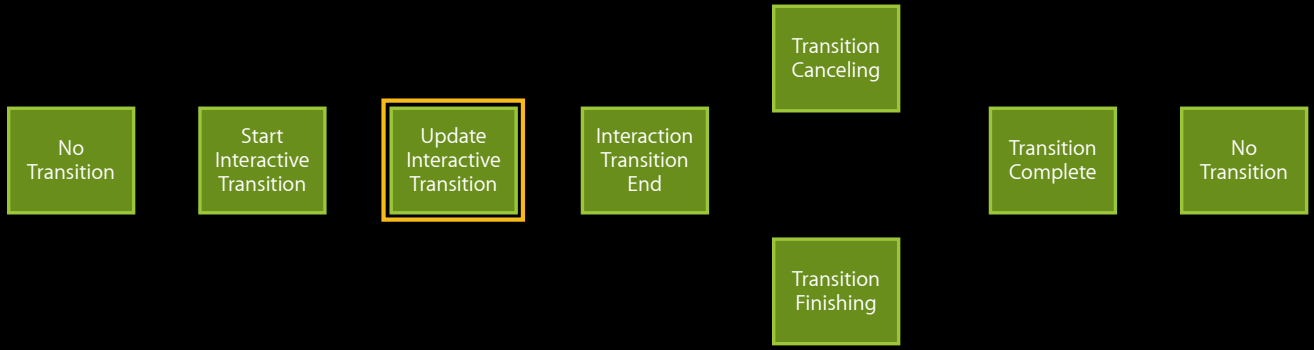


Agents

UIViewControllerTransitioning

Interactive transitioning states

States

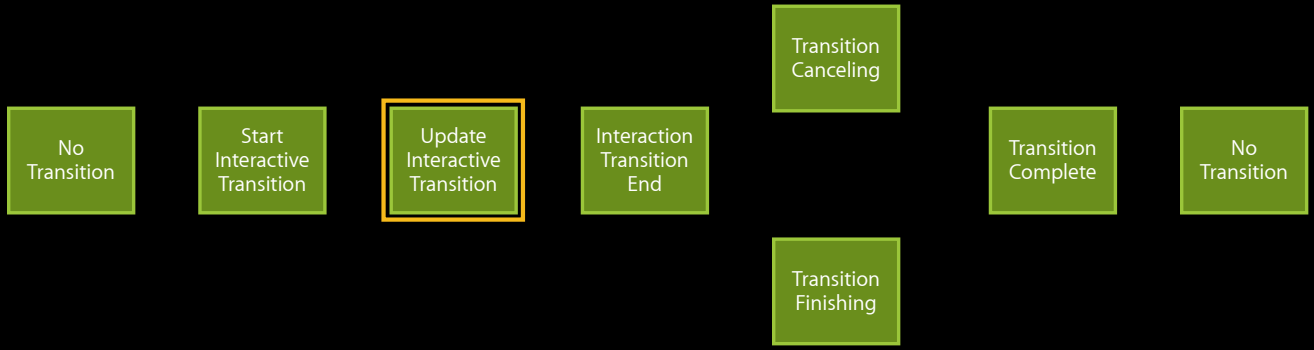


Agents

UIViewControllerTransitioning

Interactive transitioning states

States

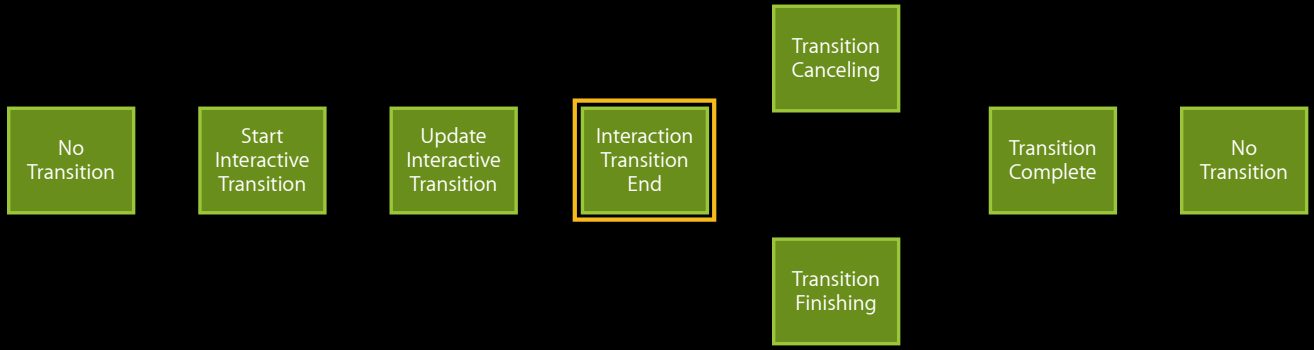


Agents

UIViewControllerTransitioning

Interactive transitioning states

States



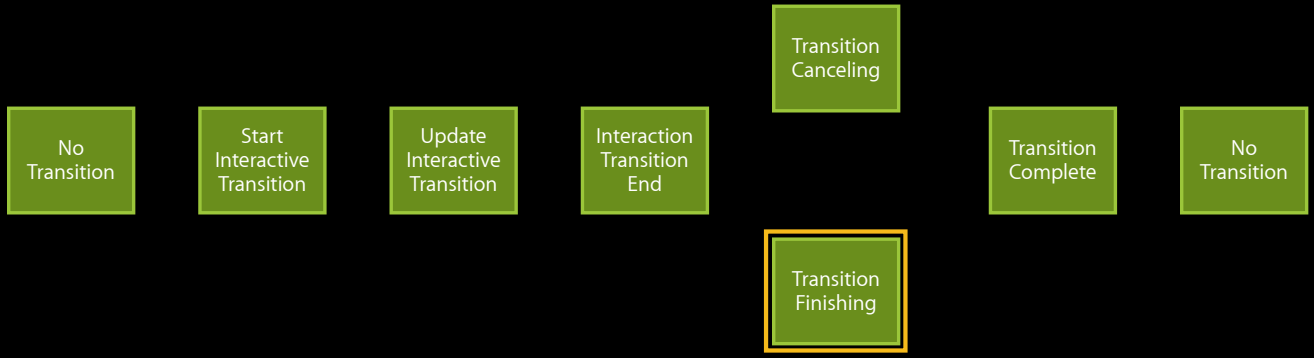
Agents

`finishInteractiveTransition`

UIViewControllerTransitioning

Interactive transitioning states

States

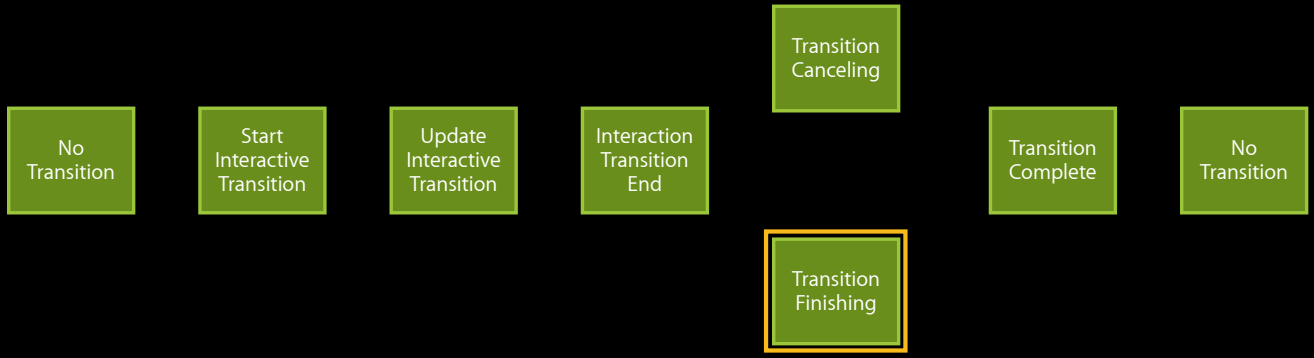


Agents

UIViewControllerTransitioning

Interactive transitioning states

States

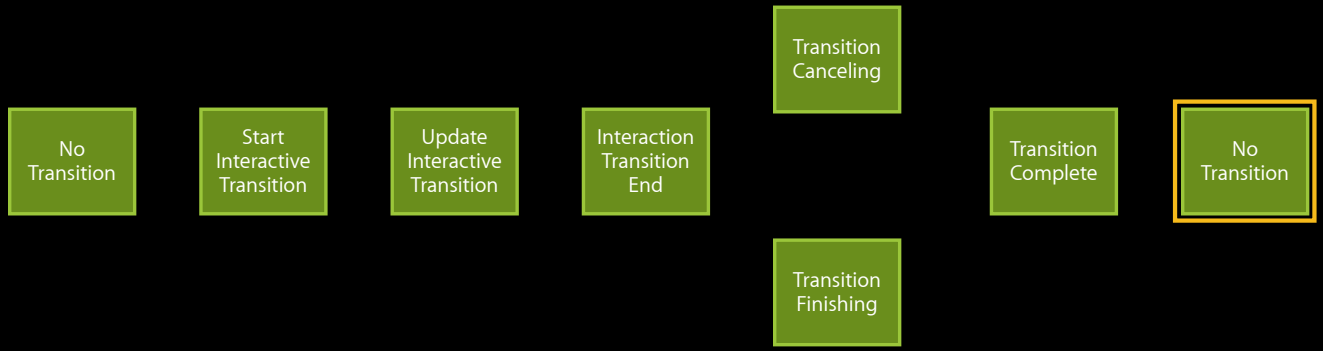


Agents

UIViewControllerTransitioning

Interactive transitioning states

States

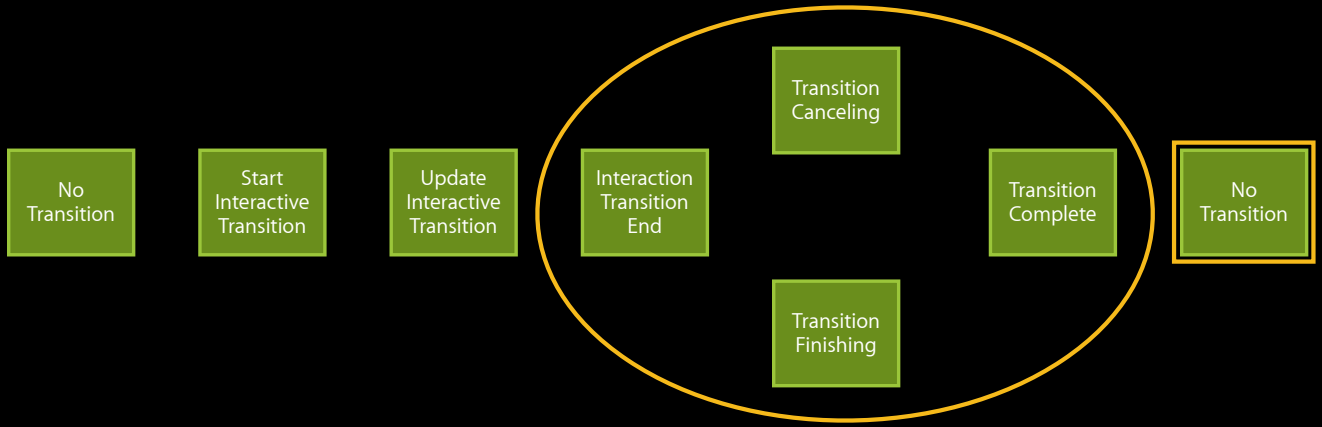


Agents

UIViewControllerTransitioning

Interactive transitioning states

States



Agents

Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition

Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition

- Implement the animation controller
  - animatePresentation: must be implemented using the UIView animation block APIs

Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition

- Implement the animation controller
  - animatePresentation: must be implemented using the UIView animation block APIs
- Implement the logic that will drive the interaction
  - e.g. The target of a gesture recognizer
  - Often this target is a subclass of UIViewControllerPercentDrivenTransition

Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition

- Implement the animation controller
  - animatePresentation: must be implemented using the UIView animation block APIs
- Implement the logic that will drive the interaction
  - e.g. The target of a gesture recognizer
  - Often this target is a subclass of UIViewControllerPercentDrivenTransition
  - The interaction logic will call
    - updateInteractiveTransition:(CGFloat)percent
    - completeInteractiveTransition or cancelInteractiveTransition
    - (Note that startInteractiveTransition is handled automatically)

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
// The associated animation controller must animate its transition using UIView animation APIs.
@interface UIPercentDrivenInteractiveTransition : NSObject <UIViewControllerInteractiveTransitioning>

@property (readonly) CGFloat duration;
// The last percentComplete value specified by updateInteractiveTransition:
@property (readonly) CGFloat percentComplete;

// completionSpeed defaults to 1.0 which corresponds to a completion duration of
// (1 - percentComplete)*duration.  It must be greater than 0.0.
@property (nonatomic,assign) CGFloat completionSpeed;

// When the interactive part of the transition has completed, this property can
// be set to indicate a different animation curve.
@property (nonatomic,assign) UIViewAnimationCurve completionCurve;

// Used instead of the corresponding context methods.
- (void)updateInteractiveTransition:(CGFloat)percentComplete;
- (void)cancelInteractiveTransition;
- (void)finishInteractiveTransition;
@end
```

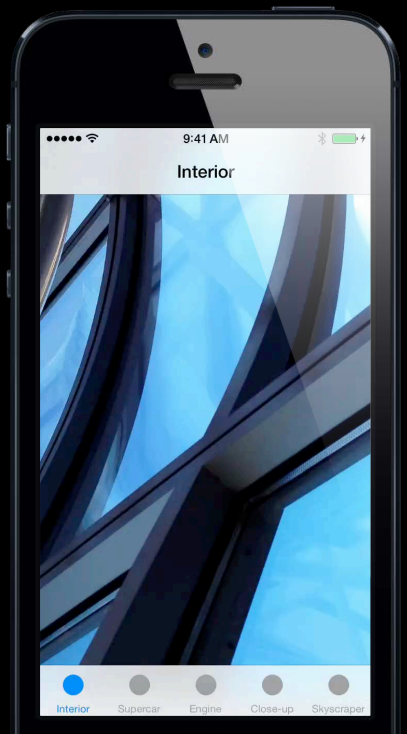
Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
// The associated animation controller must animate its transition using UIView animation APIs.
@interface UIPercentDrivenInteractiveTransition : NSObject <UIViewControllerInteractiveTransitioning>

@property (readonly) CGFloat duration;
// The last percentComplete value specified by updateInteractiveTransition:
@property (readonly) CGFloat percentComplete;

// completionSpeed defaults to 1.0 which corresponds to a completion duration of
// (1 - percentComplete)*duration.  It must be greater than 0.0.
@property (nonatomic,assign) CGFloat completionSpeed;

// When the interactive part of the transition has completed, this property can
// be set to indicate a different animation curve.
@property (nonatomic,assign) UIViewAnimationCurve completionCurve;

// Used instead of the corresponding context methods.
- (void)updateInteractiveTransition:(CGFloat)percentComplete;
- (void)cancelInteractiveTransition;
- (void)finishInteractiveTransition;
@end
```

Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition



Interactive View Controller Transitions

The easy way—use UIViewControllerPercentDrivenTransition

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
@interface YYSlideInteractor : UIPercentDrivenInteractiveTransition

- (instancetype)initWithNavigationController:(UINavigationController *)nc;

@property(nonatomic,assign) UINavigationController *parent;
@property(nonatomic,assign,getter = isInteractive) BOOL interactive;

@end
```

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
- (void)handlePinch:(UIPinchGestureRecognizer *)gr {
  CGFloat scale = [gr scale];
  switch ([gr state]) {
      case UIGestureRecognizerStateBegan:
          self.interactive = YES; _startScale = scale;
          [self.parent popViewControllerAnimated:YES];
          break;
      case UIGestureRecognizerStateChanged: {
          CGFloat percent = (1.0 - scale/_startScale);
          [self updateInteractiveTransition: (percent <= 0.0) ? 0.0 : percent];
          break;
      }
      case UIGestureRecognizerStateEnded:
      case UIGestureRecognizerStateCancelled:
          if([gr velocity] >= 0.0 || [gr state] == UIGestureRecognizerStateCancelled)
            [self cancelInteractiveTransition];
          else
            [self finishInteractiveTransition];
          self.interactive = NO;
          break;
  }
}
```

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
- (void)handlePinch:(UIPinchGestureRecognizer *)gr {
    CGFloat scale = [gr scale];
    switch ([gr state]) {
        case UIGestureRecognizerStateBegan:
            self.interactive = YES; _startScale = scale;
            [self.parent popViewControllerAnimated:YES];
            break;
        case UIGestureRecognizerStateChanged: {
            CGFloat percent = (1.0 - scale/_startScale);
            [self updateInteractiveTransition: (percent <= 0.0) ? 0.0 : percent];
            break;
        }
        case UIGestureRecognizerStateEnded:
        case UIGestureRecognizerStateCancelled:
            if([gr velocity] >= 0.0 || [gr state] == UIGestureRecognizerStateCancelled)
              [self cancelInteractiveTransition];
            else
              [self finishInteractiveTransition];
            self.interactive = NO;
            break;
    }
}
```

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
- (void)handlePinch:(UIPinchGestureRecognizer *)gr {
    CGFloat scale = [gr scale];
    switch ([gr state]) {
        case UIGestureRecognizerStateBegan:
            self.interactive = YES; _startScale = scale;
            [self.parent popViewControllerAnimated:YES];
            break;
        case UIGestureRecognizerStateChanged: {
            CGFloat percent = (1.0 - scale/_startScale);
            [self updateInteractiveTransition: (percent <= 0.0) ? 0.0 : percent];
            break;
        }
        case UIGestureRecognizerStateEnded:
        case UIGestureRecognizerStateCancelled:
            if([gr velocity] >= 0.0 || [gr state] == UIGestureRecognizerStateCancelled)
              [self cancelInteractiveTransition];
            else
              [self finishInteractiveTransition];
            self.interactive = NO;
            break;
    }
}
```

Interactive View Controller Transitions

UIPercentDrivenInteractiveTransition

```
- (void)handlePinch:(UIPinchGestureRecognizer *)gr {
    CGFloat scale = [gr scale];
    switch ([gr state]) {
        case UIGestureRecognizerStateBegan:
            self.interactive = YES; _startScale = scale;
            [self.parent popViewControllerAnimated:YES];
            break;
        case UIGestureRecognizerStateChanged: {
            CGFloat percent = (1.0 - scale/_startScale);
            [self updateInteractiveTransition: (percent <= 0.0) ? 0.0 : percent];
            break;
        }
        case UIGestureRecognizerStateEnded:
        case UIGestureRecognizerStateCancelled:
            if([gr velocity] >= 0.0 || [gr state] == UIGestureRecognizerStateCancelled)
              [self cancelInteractiveTransition];
            else
              [self finishInteractiveTransition];
            self.interactive = NO;
            break;
    }
}
```

Interactive Collection View Layout Transitions

Olivier Gutknecht

UICollectionViewTransitionLayout

UICollectionViewTransitionLayout

- A new layout interpolating between two layouts

UICollectionViewTransitionLayout

- A new layout interpolating between two layouts
- Interactively or not, with the `transitionProgress` property

UICollectionViewTransitionLayout

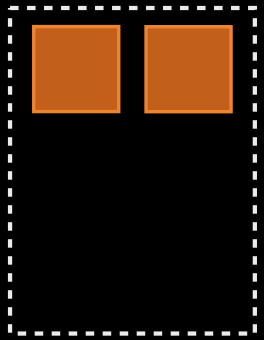
- A new layout interpolating between two layouts
- Interactively or not, with the `transitionProgress` property
- Subclassable

UICollectionViewTransitionLayout

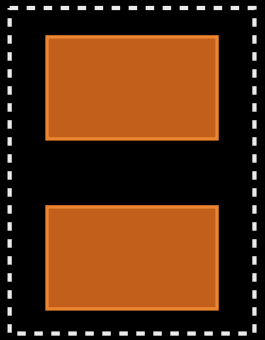
- A new layout interpolating between two layouts
- Interactively or not, with the `transitionProgress` property
- Subclassable
- Simple integration with view controller transitions

Collection Views and Transition Layout

UICollectionViewTransitionLayout



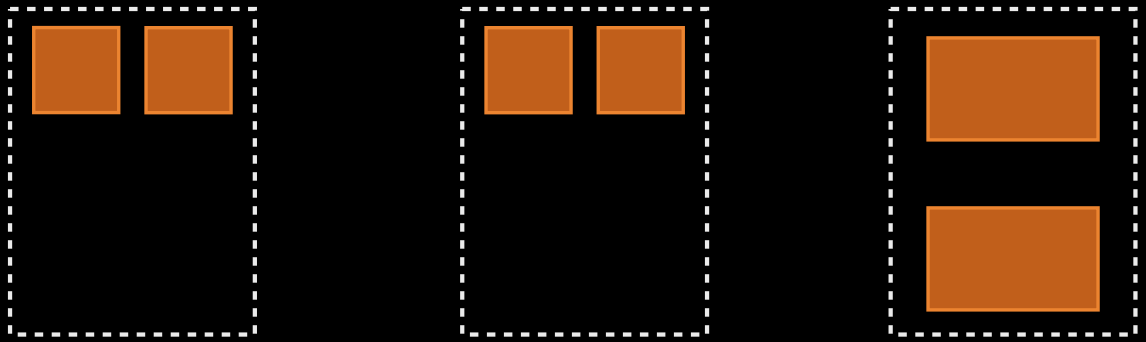
Layout A



Layout B

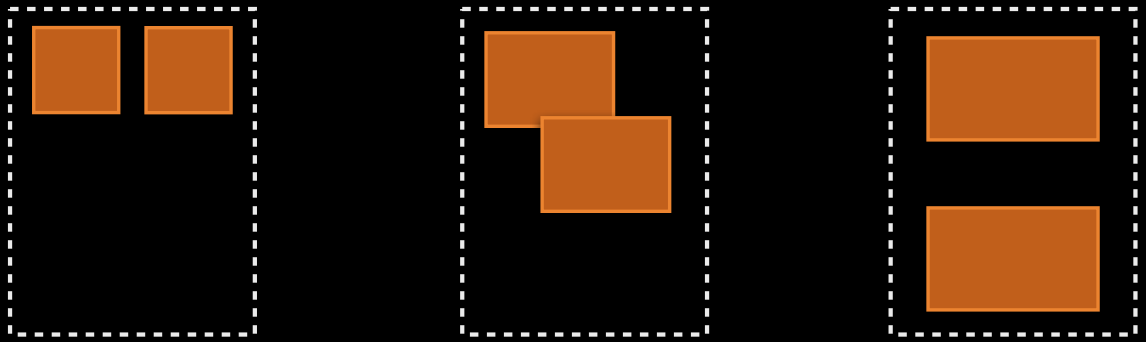
Collection Views and Transition Layout

UICollectionViewTransitionLayout



Collection Views and Transition Layout

UICollectionViewTransitionLayout



Layout A

Transition Layout

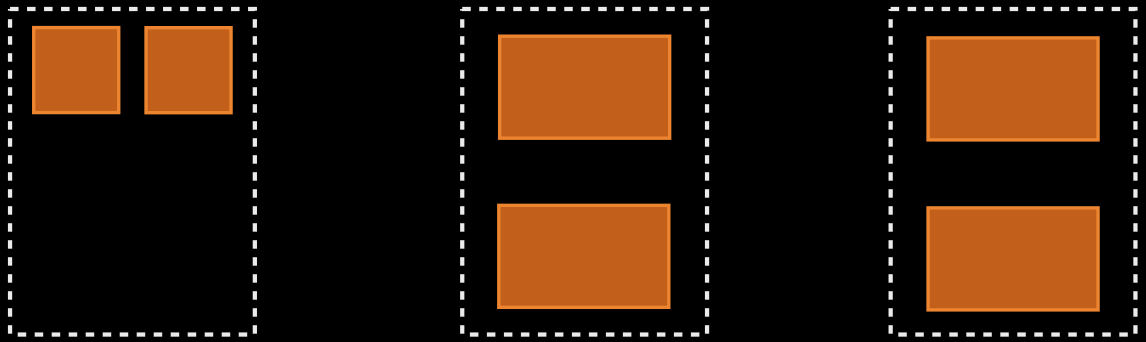
`transitionProgress`

0.5

Layout B

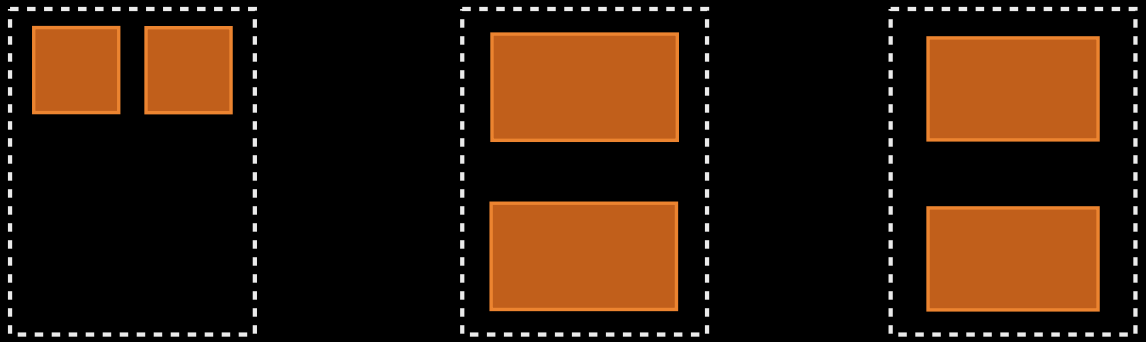
Collection Views and Transition Layout

UICollectionViewTransitionLayout



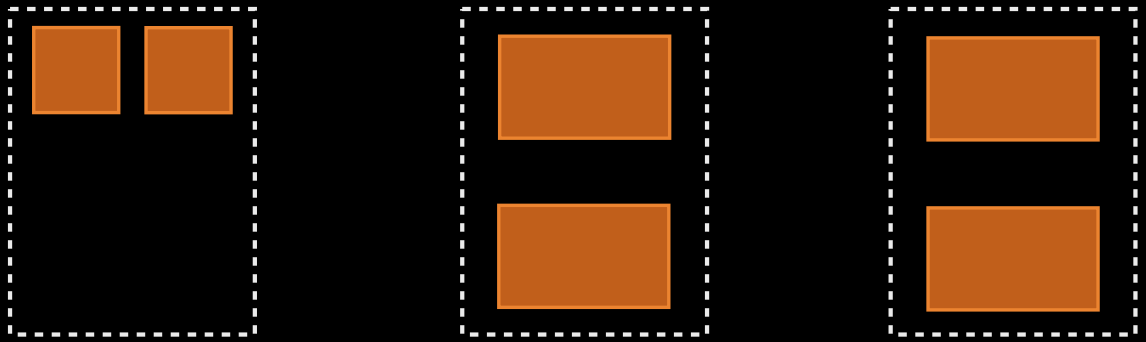
Collection Views and Transition Layout

UICollectionViewTransitionLayout



Collection Views and Transition Layout

UICollectionViewTransitionLayout



Interactive Transitions

UICollectionView

Interactive Transitions

UICollectionView

- New methods in UICollectionView

```
- (UICollectionViewTransitionLayout *)
        startInteractiveTransitionToCollectionViewLayout:completion:
- (void)finishInteractiveTransition
- (void)cancelInteractiveTransition
```

Interactive Transitions

UICollectionView

- New methods in UICollectionView

```
- (UICollectionViewTransitionLayout *)
        startInteractiveTransitionToCollectionViewLayout:completion:
- (void)finishInteractiveTransition
- (void)cancelInteractiveTransition
```

- New delegate method

```
- (UICollectionViewTransitionLayout *)collectionView:(UICollectionView*)v
                        transitionLayoutForOldLayout:(UICollectionViewLayout*)o
                                           newLayout:(UICollectionViewLayout*)n
```

Interactive Transitions

UICollectionView

- New methods in UICollectionView

```
- (UICollectionViewTransitionLayout *)
      startInteractiveTransitionToCollectionViewLayout:completion:
- (void)finishInteractiveTransition
- (void)cancelInteractiveTransition
```

- New delegate method

```
- (UICollectionViewTransitionLayout *)collectionView:(UICollectionView*)v
                        transitionLayoutForOldLayout:(UICollectionViewLayout*)o
                                           newLayout:(UICollectionViewLayout*)n
```

- Does not replace

```
- (void)setCollectionViewLayout:animated:
```

Subclassing Transition Layout

Subclassing Transition Layout

- Implement your `UICollectionViewTransitionLayout` subclass
  - e.g. update cell positions based on gesture position

Subclassing Transition Layout

- Implement your `UICollectionViewTransitionLayout` subclass
  - e.g. update cell positions based on gesture position
- Create an instance of your own class in your delegate method

Subclassing Transition Layout

- Implement your `UICollectionViewTransitionLayout` subclass
  - e.g. update cell positions based on gesture position
- Create an instance of your own class in your delegate method
- On finish or cancel, UIKit animates at correct velocity
  - Track your own parameters

    `updateValue:forAnimatedKey:`
  - UIKit will monitor velocity
  - UIKit gives you in-sync values on completion and cancel with

    `valueForAnimatedKey:`

Demo

Collection Views Transitions

Other enhancements

Collection Views Transitions

Other enhancements

- Better control of target offsets everywhere

`targetContentOffsetForProposedContentOffset:`

Collection Views Transitions

Other enhancements

- Better control of target offsets everywhere

```
targetContentOffsetForProposedContentOffset:
```

- Layouts are now notified on transitions

```
prepareForTransitionToLayout:
prepareForTransitionFromLayout:
finalizeLayoutTransition
```

Collection Views Transitions

Other enhancements

- Better control of target offsets everywhere

```
targetContentOffsetForProposedContentOffset:
```

- Layouts are now notified on transitions

```
prepareForTransitionToLayout:
prepareForTransitionFromLayout:
finalizeLayoutTransition
```

- Better animations
  - Initial and final layout attributes are now supported
  - A new completion handler

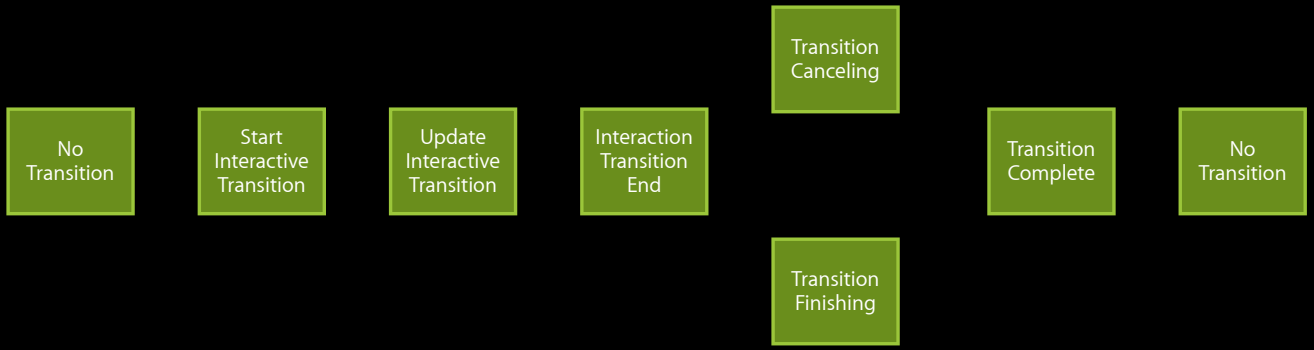
```
setCollectionViewLayout:animated:completion:
```

Interactive View Controller Transitions

Cancellation and coordinators

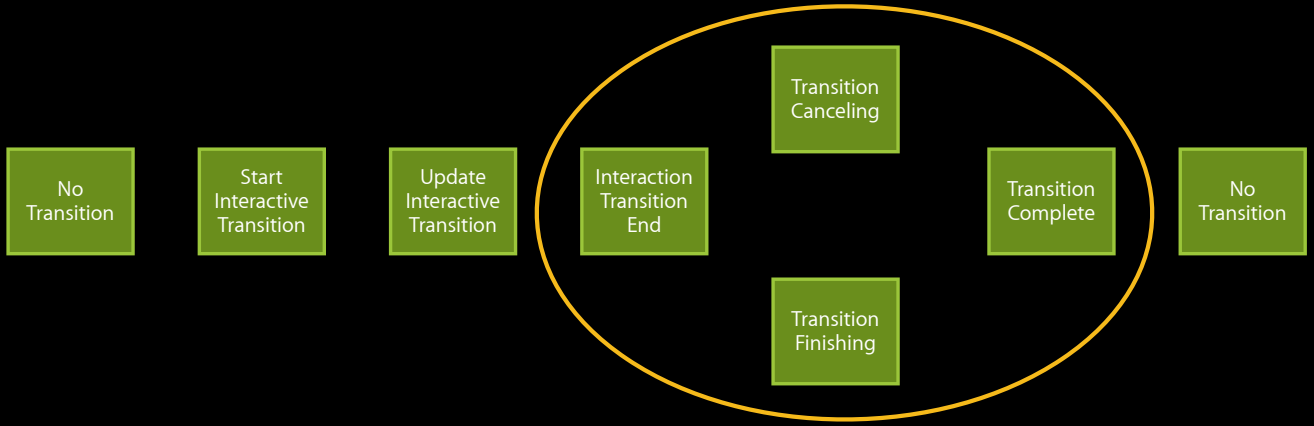
UIViewControllerTransitioning

Interactive transitioning states



UIViewControllerTransitioning

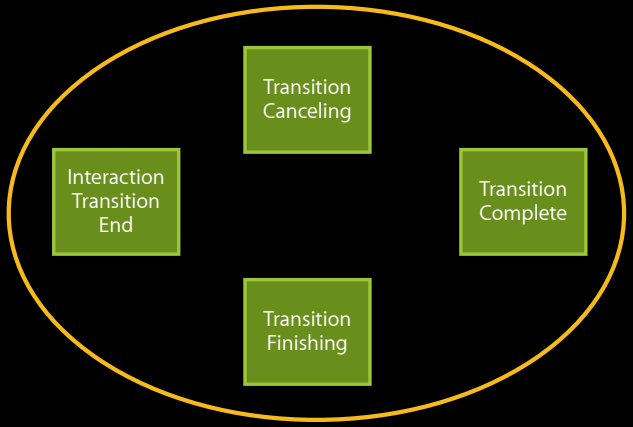
Interactive transitioning states



UIViewControllerTransitioning

Cancellation and appearance callbacks

Interactive transitioning states

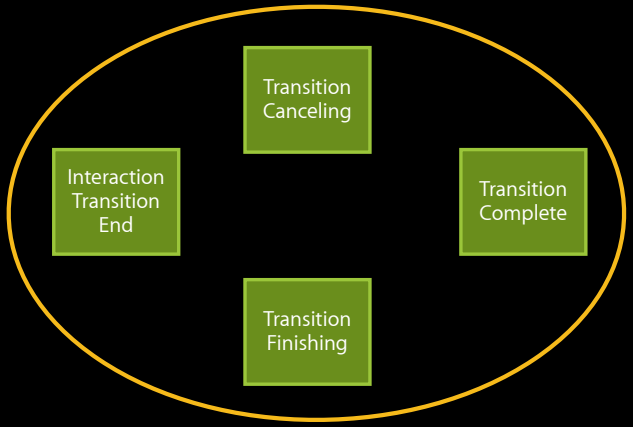


View controller appearance states

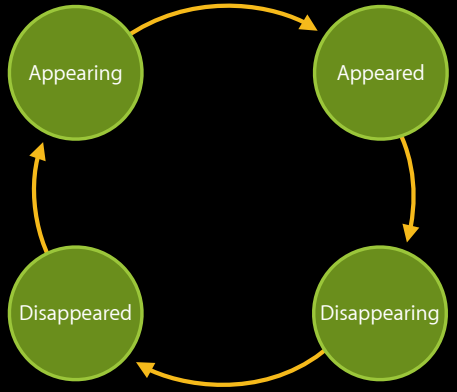
UIViewControllerTransitioning

Cancellation and appearance callbacks

Interactive transitioning states



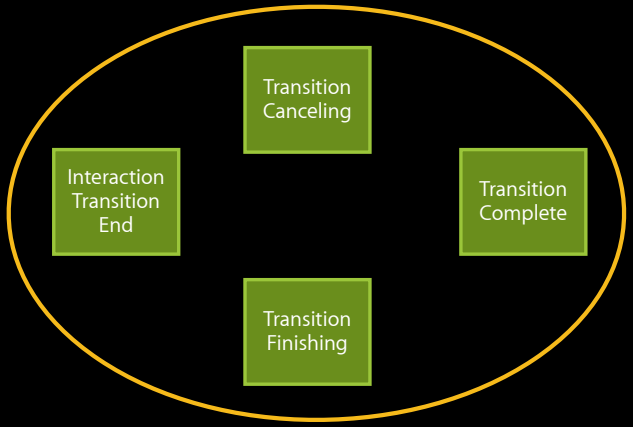
View controller appearance states



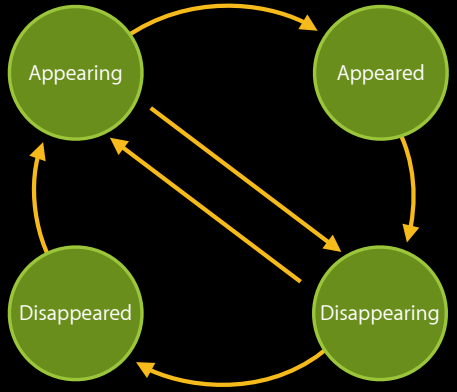
UIViewControllerTransitioning

Cancellation and appearance callbacks

Interactive transitioning states



View controller appearance states



Interactive View Controller Transitions

Canceling an interactive transition

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear
- Make sure to undo any side effects
  - There is new API to help manage this

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
@interface UIViewController(TransitionCoordinator)

@property (nonatomic,retain) id <UIViewControllerTransitionCoordinator>
                                                        transitionCoordinator;
@end
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
@interface UIViewController(TransitionCoordinator)

@property (nonatomic,retain) id <UIViewControllerTransitionCoordinator>
                                                        transitionCoordinator;
@end
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
@protocol UIViewControllerTransitionCoordinator
                    <UIViewControllerTransitionCoordinatorContext>
@optional

- (BOOL) notifyWhenInteractionEndsUsingBlock:
       (void (^ (id<UIViewControllerTransitionCoordinatorContext)handler;

- (BOOL) animatorAlongsideTransition:
                  (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                  completion:(void (^)(id<UIViewControllerTransitionCoordinatorContext)c;

- (BOOL) animatorAlongsideTransitionInView:(UIView *)view
                  animation: (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                 completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
@end
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
@protocol UIViewControllerTransitionCoordinator
                    <UIViewControllerTransitionCoordinatorContext>
@optional

- (BOOL) notifyWhenInteractionEndsUsingBlock:
       (void (^ (id<UIViewControllerTransitionCoordinatorContext)handler;

- (BOOL) animatorAlongsideTransition:
                  (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                  completion:(void (^)(id<UIViewControllerTransitionCoordinatorContext)c;

- (BOOL) animatorAlongsideTransitionInView:(UIView *)view
                  animation: (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                 completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
@end
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinatorContext>

```
@protocol UIViewControllerTransitionCoordinatorContext <NSObject>

- (UIView *)containerView;
- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

- (BOOL) isCancelled;
- (BOOL) initiallyInteractive;
- (BOOL) isInteractive;

@end
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinatorContext>

```
@protocol UIViewControllerTransitionCoordinatorContext <NSObject>

- (UIView *)containerView;
- (UIViewController *) viewControllerForKey:(NSString *)key;
- (CGRect) initialFrameForViewController:(UIViewController *)vc;
- (CGRect) finalFrameForViewController:(UIViewController *)vc;

- (BOOL) isCancelled;
- (BOOL) initiallyInteractive;
- (BOOL) isInteractive;

@end
```

Interactive View Controller Transitions

Canceling an interactive transition

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear:

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear:

```
– (void) viewWillAppear: {
     [self doSomeSideEffectsAssumingViewDidAppearIsGoingToBeCalled];

     id <UIViewControllerTransitionCoordinator> coordinator;
     coordinator = [self transitionCoordinator];

     if(coordinator && [coordinator initiallyInteractive]) {
        [transitionCoordinator notifyWhenInteractionEndsUsingBlock:
             ^(id <UIViewControllerTransitionCoordinatorContext> ctx) {
             if(ctx.isCancelled) {
                [self undoSideEffects];
             }
           }];
     }
}
```

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear:

```
- (void) viewWillAppear: {
    [self doSomeSideEffectsAssumingViewDidAppearIsGoingToBeCalled];

    id <UIViewControllerTransitionCoordinator> coordinator;
    coordinator = [self transitionCoordinator];

    if(coordinator && [coordinator initiallyInteractive]) {
       [transitionCoordinator notifyWhenInteractionEndsUsingBlock:
            ^(id <UIViewControllerTransitionCoordinatorContext> ctx) {
            if(ctx.isCancelled) {
               [self undoSideEffects];
            }
          }];
    }
}
```

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear:

```
- (void) viewWillAppear: {
     [self doSomeSideEffectsAssumingViewDidAppearIsGoingToBeCalled];

     id <UIViewControllerTransitionCoordinator> coordinator;
     coordinator = [self transitionCoordinator];

     if(coordinator && [coordinator initiallyInteractive]) {
        [transitionCoordinator notifyWhenInteractionEndsUsingBlock:
              ^(id <UIViewControllerTransitionCoordinatorContext> ctx) {
              if(ctx.isCancelled) {
                 [self undoSideEffects];
              }
            }];
     }
}
```

Interactive View Controller Transitions

Canceling an interactive transition

- Don’t assume that viewDidAppear follows viewWillAppear:

```
– (void) viewWillAppear: {
     [self doSomeSideEffectsAssumingViewDidAppearIsGoingToBeCalled];

     id <UIViewControllerTransitionCoordinator> coordinator;
     coordinator = [self transitionCoordinator];

     if(coordinator && [coordinator initiallyInteractive]) {
        [transitionCoordinator notifyWhenInteractionEndsUsingBlock:
             ^(id <UIViewControllerTransitionCoordinatorContext> ctx) {
             if(ctx.isCancelled) {
                [self undoSideEffects];
             }
           }];
     }
}
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

- The transitionCoordinator does even more
  - Allows completion handlers to be registered for transitions

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

- The transitionCoordinator does even more
  - Allows completion handlers to be registered for transitions
  - Allows other animations to run alongside the transition animation

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

- The transitionCoordinator does even more
  - Allows completion handlers to be registered for transitions
  - Allows other animations to run alongside the transition animation
- In addition to custom transitions on iOS 7
  - UINavigationController transitions have an associated transition coordinator

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

- The transitionCoordinator does even more
  - Allows completion handlers to be registered for transitions
  - Allows other animations to run alongside the transition animation
- In addition to custom transitions on iOS 7
  - UINavigationController transitions have an associated transition coordinator
  - Present and Dismiss transitions have an associated coordinator

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
- (BOOL)
  animatorAlongsideTransition:(void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                   completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
- (BOOL)
  animatorAlongsideTransitionInView:(UIView *)view
                   animation: (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                   completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
```

Interactive View Controller Transitions

<UIViewControllerTransitionCoordinator>

```
- (BOOL)
  animatorAlongsideTransition:(void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                   completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
- (BOOL)
  animatorAlongsideTransitionInView:(UIView *)view
                   animation: (void (^) (id <UIViewControllerTransitionCoordinatorContext)a;
                   completion:(void (^) (id<UIViewControllerTransitionCoordinatorContext)c;
```

Interactive View Controller Transitions

Fun with transition coordinators

```
UIViewController *vc;
[self pushViewController:vc animated: YES];

id <UIViewControllerTransitionCoordinator>coordinator;
coordinator = [viewController transitionCoordinator];

[coordinator animateAlongsideTransition:
     ^(id <UIViewControllerTransitionCoordinatorContext> c) {
        ;;; some animation
     }
     completion:(id <UIViewControllerTransitionCoordinatorContext> c) {
         ;;; Code to run after your push transition has finished.
     }];
```

Interactive View Controller Transitions

Fun with transition coordinators

```
UIViewController *vc;
[self pushViewController:vc animated: YES];
```

```
id <UIViewControllerTransitionCoordinator>coordinator;
coordinator = [viewController transitionCoordinator];

[coordinator animateAlongsideTransition:
     ^(id <UIViewControllerTransitionCoordinatorContext> c) {
        ;;; some animation
     }
     completion:(id <UIViewControllerTransitionCoordinatorContext> c) {
         ;;; Code to run after your push transition has finished.
     }];
```

Interactive View Controller Transitions

Fun with transition coordinators

```
UIViewController *vc;
[self pushViewController:vc animated: YES];
```

```
id <UIViewControllerTransitionCoordinator>coordinator;
coordinator = [viewController transitionCoordinator];
```

```
[coordinator animateAlongsideTransition:
     ^(id <UIViewControllerTransitionCoordinatorContext> c) {
        ;;; some animation
     }
     completion:(id <UIViewControllerTransitionCoordinatorContext> c) {
         ;;; Code to run after your push transition has finished.
     }];
```

Interactive View Controller Transitions

Fun with transition coordinators

```
UIViewController *vc;
[self pushViewController:vc animated: YES];

id <UIViewControllerTransitionCoordinator>coordinator;
coordinator = [viewController transitionCoordinator];

[coordinator animateAlongsideTransition:
     ^(id <UIViewControllerTransitionCoordinatorContext> c) {
        ;;; some animation
     }
     completion:(id <UIViewControllerTransitionCoordinatorContext> c) {
         ;;; Code to run after your push transition has finished.
     }];
```

Concluding Remarks

"With great power comes greater responsibility"

Concluding Remarks

“With great power comes greater responsibility”

Concluding Remarks

“With great power comes greater responsibility”

Concluding Remarks

"With great power comes greater responsibility"

Concluding Remarks

“With great power comes greater responsibility”

- Powerful new animation and snapshot APIs can be used to create awesome transition animations

Concluding Remarks

“With great power comes greater responsibility”

- Powerful new animation and snapshot APIs can be used to create awesome transition animations
- Many view controller transition animations can be customized
  - `UICollectionViewControllers` and `UICollectionViews` can be easily used to define custom transitions
  - The protocol based expression of this API is very flexible

Concluding Remarks

"With great power comes greater responsibility"

- Powerful new animation and snapshot APIs can be used to create awesome transition animations
- Many view controller transition animations can be customized
  - `UICollectionViewControllers` and `UICollectionViews` can be easily used to define custom transitions
  - The protocol based expression of this API is very flexible
- View controller transitions can be interactive
  - Is it `viewWillAppear:` or `viewWillProbablyAppear:`?

Concluding Remarks

"With great power comes greater responsibility"

- Powerful new animation and snapshot APIs can be used to create awesome transition animations
- Many view controller transition animations can be customized
  - `UICollectionViewControllers` and `UICollectionViews` can be easily used to define custom transitions
  - The protocol based expression of this API is very flexible
- View controller transitions can be interactive
  - Is it `viewWillAppear:` or `viewWillProbablyAppear:`?
- A transition coordinator can be used with/without custom transitions
  - `animateAlongsideTransition:completion:`
  - `notifyWhenInteractionEndsUsingBlock:`

More Information

Jake Behrens
App Frameworks Evangelist
behrens@apple.com

Documentation and Sample Code
iOS Dev Center
http://developer.apple.com

Apple Developer Forums
http://devforums.apple.com

Related Sessions

Building User Interfaces for iOS 7	Presidio Tuesday 10:15AM
Getting Started with UIKit Dynamics	Presidio Tuesday 4:30PM
Advance Techniques with UIKit Dynamics	Presidio Thursday 3:15PM
Best Practices for Great iOS UI Design	Presidio Friday 10:15AM

Labs

Cocoa Touch Animation Lab	Frameworks Lab B Thursday 2:00PM


WWDC2013