

Introduction to Core Animation

Ben Chen

CocoaHeads Meeting @ Beijing

Agenda

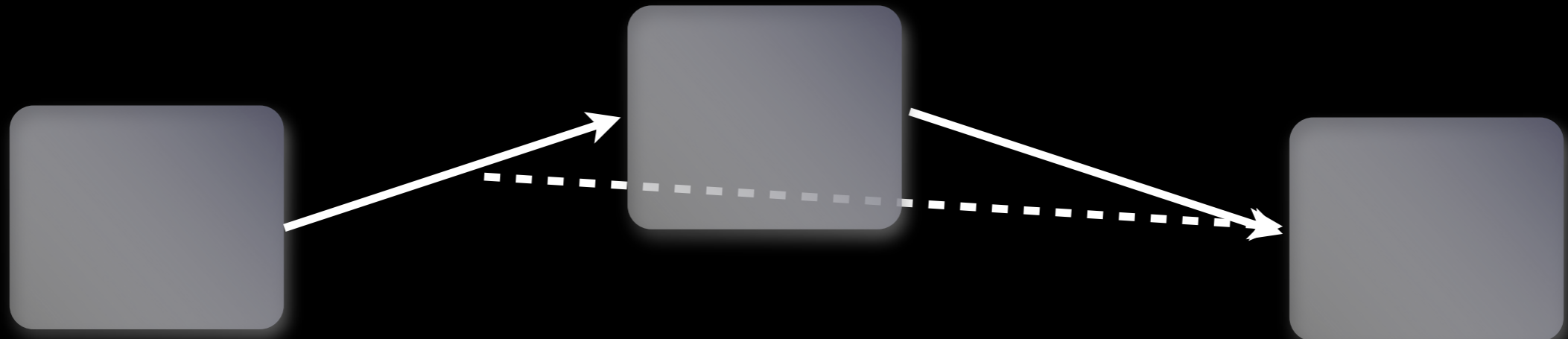
- Why Core Animation?
- Getting Started With Core Animation
- AppKit & UIKit support
- Tips when using explicit animation
- Core Animation and Gaming
- Q&A

Why Core Animation



Animations are hard

- Calculate the position for every frame
- Drawing code gets called at frame rate
 - Needs to setup a timer
 - May consume default Run Loop power
- Create a separate thread
 - Painful!
- Harder to handle intermediate states
- Harder to animate multiple properties (size, position, etc.)



Core Animation to Rescue

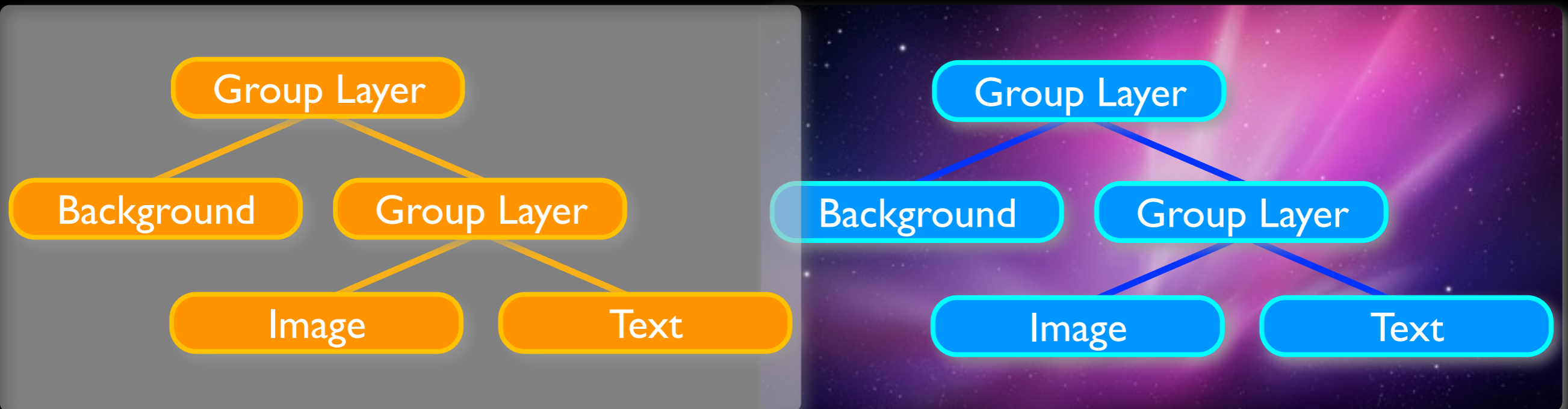
- High performance compositing and animation
- Familiar recursive abstraction
- Provide abstract interface for animations
- Animation
 - Happens on a separate thread
 - Independent of runloop workload

Performance

- Does minimize redraw
- Does minimize application code run during resize/relayout
- Eliminates application code that runs at frame-rate
- Ensures consistent frame-rate

Two Trees

- Separated Data Tree and Render Tree
- Data Tree is the interface for developer
 - We change the data tree by setting each layer's properties
 - The changes get committed to render tree, and animations are done in render tree.
- Render Tree is opaque to developers, and all rendering is done in a separate thread.



Availability

- Available starting in Mac OS X 10.5 Leopard and iPhone OS 2.0
- Some enhancements in Mac OS X 10.6 Snow Leopard and iPhone OS 3.0

Getting Started

```
CALayer *l = [CALayer layer];  
l.frame = CGRectMake(0,0,300,300);  
[parent addSublayer:l];  
l.backgroundColor = [[UIColor redColor]  
CGColor];  
l.content = [[UIImage imageNamed:@"img"]  
CGImage];  
l.cornerRadius = 40;  
l.opacity = 0.5;
```



Custom Drawing

- Define your drawing method
 - By subclassing CALayer

```
- (void)drawInContext:(CGContextRef)ctx {  
    // draw something into ctx...
```

- Or by setting a delegate object on the layer

```
- (void)drawLayer:(CALayer*)l inContext:  
(CGContextRef)ctx {  
    // draw something into ctx...
```

- Call `-setNeedsDisplay` or `-setNeedsDisplayInRect:`
- Read context's clip rect to optimize drawing
- Set the layer's opaque property to YES if alpha not needed

Animation Types

- Types
 - CABasicAnimation
 - CAKeyframeAnimation
 - CATransition
 - CAAnimationGroup

Control Animations

- Timing
 - Start, duration, speed, repeat, reverse
 - Timing function (ease-in/out)

Explicit Animations

```
CABasicAnimation *anim = [CABasicAnimation  
animationWithKeyPath:@"bounds"];  
anim.fromValue = [NSValue valueWithCGRect:smallRect];  
anim.toValue = [NSValue valueWithCGRect:bigRect];  
anim.duration = 2.0;  
anim.timingFunction = [CAMediaTimingFunction  
functionWithName:kCAMediaTimingFunctionEaseOut];  
anim.delegate = anObj;  
[layer addAnimation:anim forKey:@"someAnim"];  
  
...  
  
- (void)animationDidStop:(CAAnimation*)anim finished:  
(BOOL)flag {  
    // Do something  
    [layer removeAnimation:anim forKey:@"someAnim"];
```

Explicit Animations



Implicit Animations

- `layer.bounds = bigRect;`
- This will trigger this

```
- (id <CAAction>)actionForKey:(NSString*)key
```

- The action is normally a `CABasicAnimation` with:

- `action.duration = 0.25;`
- `action.timingFunction = [CAMediaTimingFunction
functionWithName:kCAMediaTimingFunctionLinear];`
- `action.fromValue = [[layer presentationLayer]
valueForKey:@"bounds"];`
- `action.toValue = [NSValue valueWithCGRect:bigRect];`

- Then the action gets added to the layer

```
[layer addAnimation:action forKey:someKey];
```

AppKit Support

```
[[view animator] setFrame:rect];
```

UIKit Support

```
[UIView beginAnimations:...];  
view.position = newPosition;  
[UIView commitAnimations];
```

Demo

How to use UIKit support to take advantage of Core Animation.

Tips for using explicit animations

- Disable Actions

```
[CATransaction setDisableActions:YES];  
- (id <CAAction>)actionForKey:(NSString*)key {  
    return nil;  
}
```

- Use `CAKeyFrameAnimation` whenever possible

- Presentation Layer

```
[layer presentationLayer];
```

- Avoid blinks

```
anim.removedOnCompletion = NO;  
anim.fillMode = kCAFillModeForwards;  
// remove animation and set final value in delegate method
```

- Customize `CAAnimations` to store more information (remember to conform to `NSCopy` protocol)

Demo

Example of the shooting game done in Core Animation

My take on Core Animation for Gaming

- It's possible and sometimes easier
- However if your game is doing frequent update (e.g. a shooting game we just saw), it might be better to use Open GL directly.
- If the updating is only driven by user events (like a more casual game), and you don't need any real 3D animations, and you don't have lots of resource files, try Core Animation first.

Q&A

