

**CSC 471 / 371**  
**Mobile Application Development for iOS**



Prof. Xiaoping Jia  
 School of Computing, CDM  
 DePaul University  
[xjia@cdm.depaul.edu](mailto:xjia@cdm.depaul.edu)  
[@DePaulSWEEng](https://twitter.com/DePaulSWEEng)



1

**Animations & Transitions**



2

**Outline**

- View animations
- View transitions




DEPAUL UNIVERSITY

3

**View Animation**



4

**View Animation**

- Many properties of the view objects are *animatable*.
  - center, frame, bounds
  - alpha, backgroundColor
  - transform
- Any changes to these properties can be animated automatically by the system
  - Through *interpolation*



DEPAUL UNIVERSITY

5

**View Animation Demo**

- Simple view animation
  - Movement of view object
  - Changing the frame property
- Animation parameters:
  - Duration, delay
  - Animation curve
- Spring animation




DEPAUL UNIVERSITY

6

## Create View Programmatically

- Create view instances

```
class ViewController: UIViewController {
    enum Direction { case up, down }

    let soccer = UIImageView(image: UIImage(named: "..."))
    let basketball = UIImageView(image: UIImage(named: "..."))
    let volleyball = UIImageView(image: UIImage(named: "..."))
    let tennis = UIImageView(image: UIImage(named: "..."))

    let size: CGFloat = 50
    var direction = Direction.up

    override func viewDidLoad() { ... }
    @IBAction func startAnimation(_ sender: UIButton) { ... }
}
```

DEPAUL UNIVERSITY

7

## Create View Programmatically

- Set the frame property of each view
- Add the views to the view hierarchy

```
override func viewDidLoad() {
    super.viewDidLoad()
    let y: CGFloat = view.bounds.height - size * 2
    var x: CGFloat = 25

    soccer.frame = CGRect(x: x, y: y,
                          width: size, height: size)
    view.addSubview(soccer)
    ...
}
```

DEPAUL UNIVERSITY

8

## View Animation

```
@IBAction func startAnimation(_ sender: UIButton) {
    let duration = 2.0 // 2 seconds
    var x: CGFloat = 25
    let y: CGFloat = direction == .up ? size * 2 :
        view.bounds.height - size * 2
    direction = direction == .up ? .down : .up

    UIView.animate(withDuration: duration, animations: {
        self.soccer.frame = CGRect(x: x, y: y,
                                    width: self.size, height: self.size)
    })
    ...
}
```

Set the frame property of the after position

DEPAUL UNIVERSITY

9

## View Animation – Animation Curve

- With an ease-in/ease-out animation curve

```
@IBAction func startAnimation(sender: UIButton) {
    ...
    x += (size + CGFloat(25))
    let optionEase = UIView.AnimationOptions.curveEaseInOut

    UIView.animate(withDuration: duration, delay: 0.0,
                  options: optionEase, animations: {
        self.basketball.frame = CGRect(x: x, y: y,
                                       width: self.size, height: self.size)
    }, completion: nil)
    ...
}
```

DEPAUL UNIVERSITY

10

## View Animation – Animation Curve

- With a linear animation curve

```
@IBAction func startAnimation(sender: UIButton) {
    ...
    x += (size + CGFloat(25))
    let optionsLinear = UIView.AnimationOptions.curveLinear

    UIView.animate(withDuration: duration, delay: 0.0,
                  options: optionsLinear, animations: {
        self.volleyball.frame = CGRect(x: x, y: y,
                                       width: self.size, height: self.size)
    }, completion: nil)
    ...
}
```

DEPAUL UNIVERSITY

11

## Spring Animation

```
@IBAction func startAnimation(sender: UIButton) {
    ...
    let delay = 1.0 // 1 second
    let damping: CGFloat = 0.5
    let velocity: CGFloat = 0.5
    x += (size + CGFloat(25))

    UIView.animate(withDuration: duration, delay: delay,
                  usingSpringWithDamping: damping,
                  initialSpringVelocity: velocity,
                  options: optionEase, animations: {
        self.tennis.frame = CGRect(x: x, y: y,
                                   width: self.size, height: self.size)
    }, completion: nil)
}
```

DEPAUL UNIVERSITY

12

## Key Frame Animation

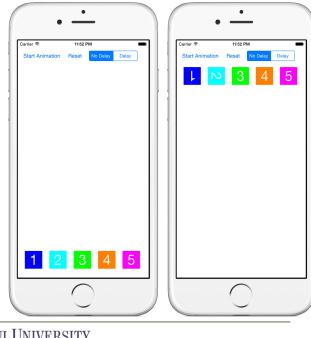
- Allows an animation to be divided into several segments.
- The consecutive segments are divided by *key frames*.
- You may specify the relative start time and duration of each stage
- How to define key frame animations?
  - Insert/add key frames at the appropriate points of the animation

 DEPAUL UNIVERSITY 13

13

## Key Frame Animation Demo

- Animating movements of numbered squares
- Move from bottom to top
- Rotate
  - ½ rotation
  - ¼ rotation
  - Full rotation, and animation



 DEPAUL UNIVERSITY

14

## Key Frame Animation – Half Rotation

```
@IBAction func startAnimation(_ sender: UIButton) {
    let duration = 2.0
    var delay = 0.0
    let dt =
        delayOptions.selectedSegmentIndex == 0 ? 0.0 : 2.0

    var x: CGFloat = 20

    UIView.animate(withDuration: duration, animations: {
        self.objects[0].frame = CGRect(x: x, y: 100,
                                       width: 50, height: 50)
        self.objects[0].transform =
            CGAffineTransform(rotationAngle: .pi)
    })
    ...
}
```

A half rotation

 DEPAUL UNIVERSITY 15

15

## Key Frame Animation – Quarter Rotation

```
@IBAction func startAnimation(_ sender: UIButton) {
    ...
    x += 70
    delay += dt
    UIView.animate(withDuration: duration, delay: delay,
                  options: UIView.AnimationOptions(),
                  animations: {
        self.objects[1].frame =
            CGRect(x: x, y: 100, width: 50, height: 50)
        self.objects[1].transform =
            CGAffineTransform(rotationAngle: .pi / 2)
    }, completion: nil)
    ...
}
```

A quarter rotation

 DEPAUL UNIVERSITY 16

16

## Key Frame Animation – Full Rotation, 1<sup>st</sup> Attempt

```
@IBAction func startAnimation(_ sender: UIButton) {
    ...
    x += 70
    delay += dt
    UIView.animate(withDuration: duration, delay: delay,
                  options: UIView.AnimationOptions(),
                  animations: {
        self.objects[2].frame =
            CGRect(x: x, y: 100, width: 50, height: 50)
        self.objects[2].transform =
            CGAffineTransform(rotationAngle: 2 * .pi)
    }, completion: nil)
    ...
}
```

The start and the end positions are the same

A full rotation. But no animation.

 DEPAUL UNIVERSITY 17

17

## Key Frame Animation – Full Rotation, 2<sup>nd</sup> Attempt

```
UIView.animateKeyframes(withDuration: duration,
                       delay: delay, options: UIView.KeyframeAnimationOptions(),
                       animations: {
    self.objects[3].frame = CGRect(x: x, y: 100,
                                   width: 50, height: 50)

    UIView.addKeyframe(withRelativeStartTime: 0,
                      relativeDuration: 1/2, animations: {
        self.objects[3].transform =
            CGAffineTransform(rotationAngle: .pi)
    })
    UIView.addKeyframe(withRelativeStartTime: 1/2,
                      relativeDuration: 1/2, animations: {
        self.objects[3].transform =
            CGAffineTransform(rotationAngle: 2 * .pi)
    })
}, completion: nil)
```

 DEPAUL UNIVERSITY 18

18

## Key Frame Animation – Full Rotation, 3<sup>rd</sup> Attempt

- Divide into three segments.
- Each segment performs a 1/3 rotation.

```
UIView.animateKeyframes(withDuration: duration,
delay: delay, options: UIView.KeyframeAnimationOptions(),
animations: {
    self.objects[4].frame = CGRect(x: x, y: 100,
width: 50, height: 50)
}
Add key frames with 1/3, 2/3, and full rotations
}, completion: nil)
```

DEPAUL UNIVERSITY



19

## The Key Frames with 1/3 Rotation

```
UIView.addKeyframe(withRelativeStartTime: 0,
                    relativeDuration: 1/3, animations: {
    self.objects[4].transform =
        CGAffineTransform(rotationAngle: .pi * 2 / 3)
})
UIView.addKeyframe(withRelativeStartTime: 1/3,
                    relativeDuration: 1/3, animations: {
    self.objects[4].transform =
        CGAffineTransform(rotationAngle: .pi * 4 / 3)
})
UIView.addKeyframe(withRelativeStartTime: 2/3,
                    relativeDuration: 1/3, animations: {
    self.objects[4].transform =
        CGAffineTransform(rotationAngle: .pi * 2)
})
```

DEPAUL UNIVERSITY

20

## View Transition



21

## View Transitions

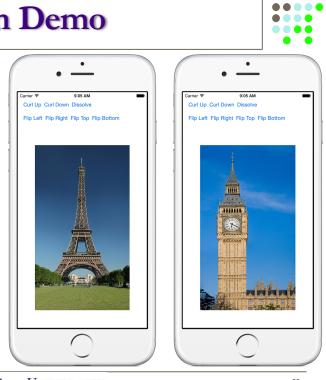
- Replace one view with another in the view hierarchy
- Animate the transition from one to the other
- Transition animation options:
  - Curl up or down
  - Flip from left to right or from right to left
  - Flip from top or from bottom
  - Dissolve

DEPAUL UNIVERSITY

22

## View Transition Demo

- A container view contains one of the two views.
- The transition from one to the other is animated



DEPAUL UNIVERSITY

23

## The Views

- Create a container view and two views

```
class ViewController: UIViewController {

    let container = UIView()
    let big_ben = UIImageView(image: UIImage(named: "big_ben"))
    let eiffel = UIImageView(image: UIImage(named: "eiffel"))

    override func viewDidLoad() { ... }

    @IBAction func changeView(_ sender: UIButton) { ... }
}
```

DEPAUL UNIVERSITY

24

## Set Up the View Hierarchy

- Add the container and *Eiffel* to the view hierarchy

```
override func viewDidLoad() {
    super.viewDidLoad()
    let w = view.bounds.width - 100
    let h = view.bounds.height - 200
    container.frame = CGRect(x: 50, y: 150,
                             width: w, height: h)
    view.addSubview(container)

    big_ben.frame = CGRect(x: 0, y: 0, width: w, height: h)
    eiffel.frame = big_ben.frame
    container.addSubview(eiffel)
}
```

DEPAUL UNIVERSITY

25

## Action to Change View – Choose a View Transition

```
@IBAction func changeView(_ sender: UIButton) {
    var transitionOptions = UIView.AnimationOptions.TransitionNone
    if let title = sender.currentTitle {
        switch title {
            case "Curl Down": transitionOptions = .transitionCurlDown
            case "Curl Up": transitionOptions = .transitionCurlUp
            case "Dissolve": transitionOptions = .transitionCrossDissolve
            case "Flip Left": transitionOptions = .transitionFlipFromLeft
            case "Flip Right": transitionOptions = .transitionFlipFromRight
            case "Flip Top": transitionOptions = .transitionFlipFromTop
            case "Flip Bottom": transitionOptions = .transitionFlipFromBottom
            default: transitionOptions = .transitionCurlUp
        }
    }
    -> The next slide
}
```

DEPAUL UNIVERSITY

26

## Action to Change View – Animate the Transition

- Change the view hierarchy, i.e., remove/add views

```
@IBAction func changeView(_ sender: UIButton) {
    var transitionOptions = "" -> The previous slide
    ...
    let view1 = big_ben.superview != nil ? big_ben : eiffel
    let view2 = big_ben.superview != nil ? eiffel : big_ben

    UIView.transition(with: self.container,
                      duration: 2.0, options: transitionOptions,
                      animations: {
            view1.removeFromSuperview()
            self.container.addSubview(view2)
        }, completion: nil)
}
```

Switch view1 with view2

DEPAUL UNIVERSITY

27

27

## Action to Change View – Animate the Transition, Alt

```
@IBAction func changeView(sender: UIButton) {
    var transitionOptions = "" -> The previous slide
    ...
    let view1 = big_ben.superview != nil ? big_ben : eiffel
    let view2 = big_ben.superview != nil ? eiffel : big_ben

    UIView.transition(from: view1, to: view2,
                      duration: 2.0, options: transitionOptions,
                      completion: nil)
}
```

Switch view1 with view2

DEPAUL UNIVERSITY

28

## Sample Code

- View Animation – SB.zip
- Key Frame Animation – SB.zip
- View Transition – SB.zip

DEPAUL UNIVERSITY

29

29

## Next ...

- Touch events
- Gestures & gesture recognizers

DEPAUL UNIVERSITY

30