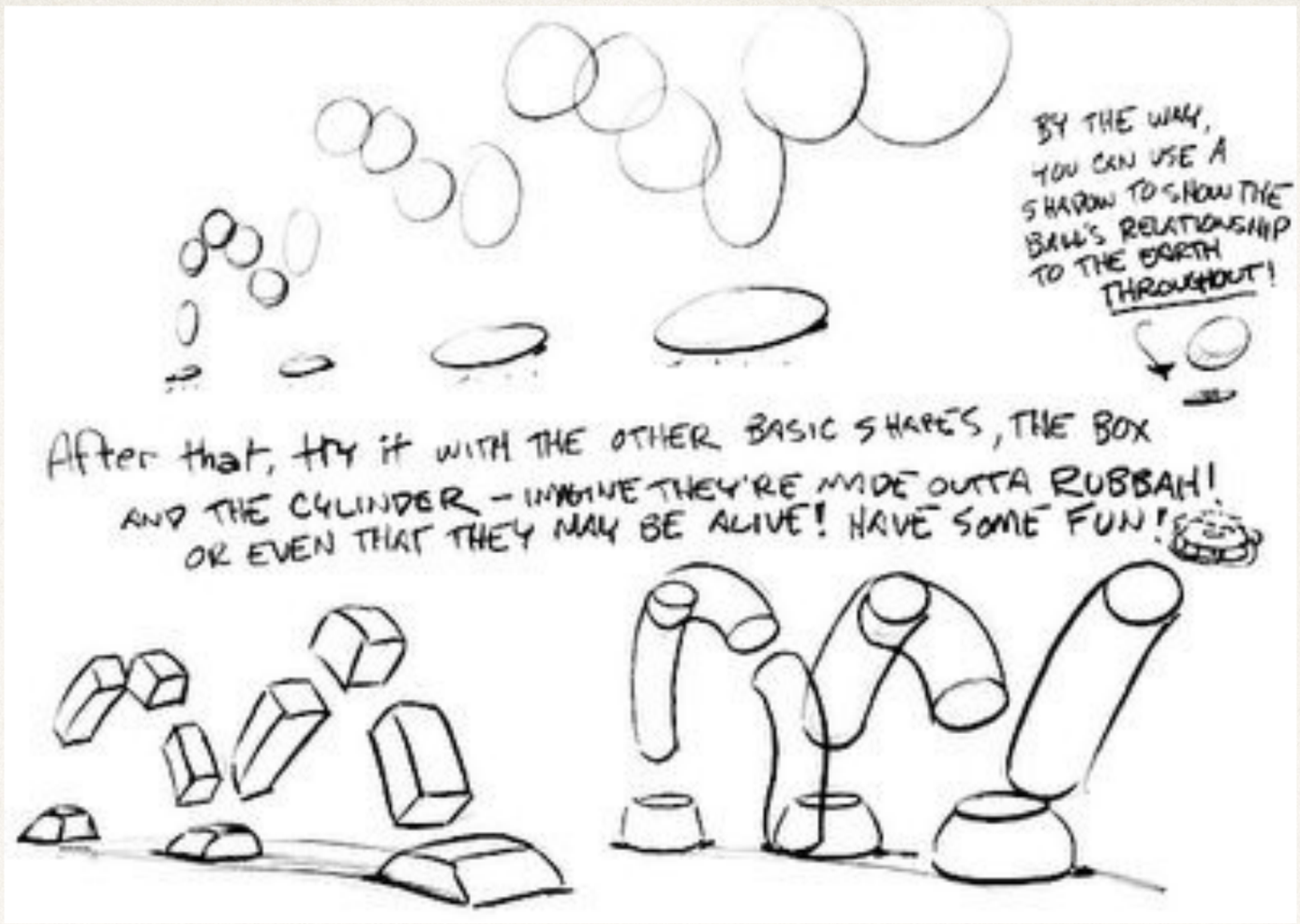


Core Animation

Dr. Sarah Abraham

University of Texas at Austin
CS329e
Spring 2020

What is Animation?

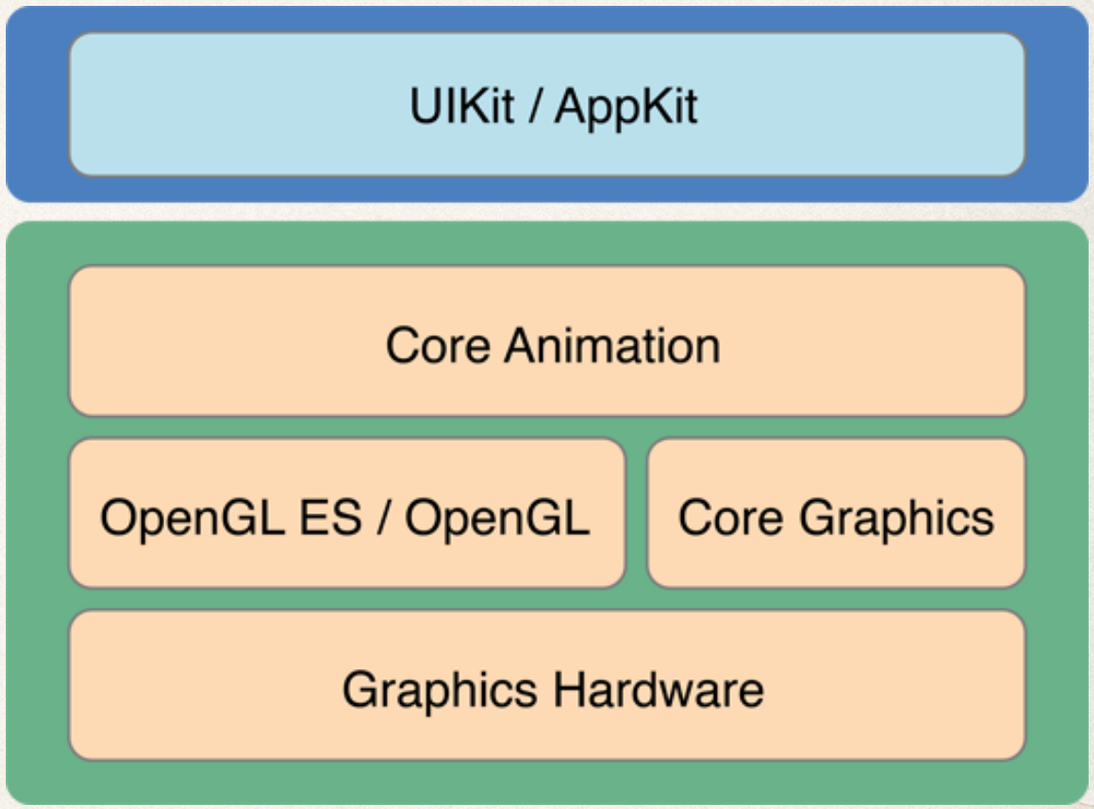
- Series of images presented in succession
- Gives the impression of continuous motion
- Mathematical interpolations determine how the animation moves

Why Use Animation?

- Animations are everywhere!
- Moving action gives a sense of “narrative”
  - Draws user’s attention
  - Indicates importance of activity
  - Adds extra polish to final app

Core Animation

- Simplifies the animation process
- Programmer configures start and end points, duration of animation, etc
- Automatically renders on the graphics hardware
- Fine-grained control of animations also possible



Producing Animations

- Core Animation accessible in UIViews (buttons, labels, etc) and UIViewControllers
- Assign changes to UIView properties over time
  - Different changes (and rate of change) change user's impression of event
- Core Animation handles the actual interpolation

UIAnimation

- UIView objects have a CALayer
  - Allows Core Animation to perform animations on them without explicitly calling on CA objects/functions
- Provides a fast, accessible way to add animations
  - Less control than using Core Animation directly

Basic UIAnimation

```
/* Set start value of selected attributes earlier
in the code */

UIView.animate(withDuration: , delay: ,
options: , animations: {

        /* Block sets final state of views and
        properties */

    }, completion: { /* Block to run upon
    completion */}

)
```

Duration and Delay

- Duration sets length of animation (in seconds)
- Delay sets time before animation starts (in seconds)

Creating Animations

- Consider current and final state of view
  - `animations` block should define final appearance of object after animation
- Change the view's animatable properties to represent final view

UIView Animatable Properties

- **frame** changes view size and position relative to its superview
- **bounds** modifies the view's size
- **center** modifies the view's position relative to its superview
- **transform** modifies the view's scale, rotation, and translation
- **alpha** modifies the view's transparency
- **backgroundColor** modifies the view's color
- **contentStretch** modifies the view's aspect ratio

Fade In/Fade Out

- View alpha adjusted over time

- Fade in: alpha is initially 0 (invisible) then increases to 1 (fully visible)

- Fade out: alpha is initially 1 (fully visible) then decreases to 0 (invisible)

Sliding

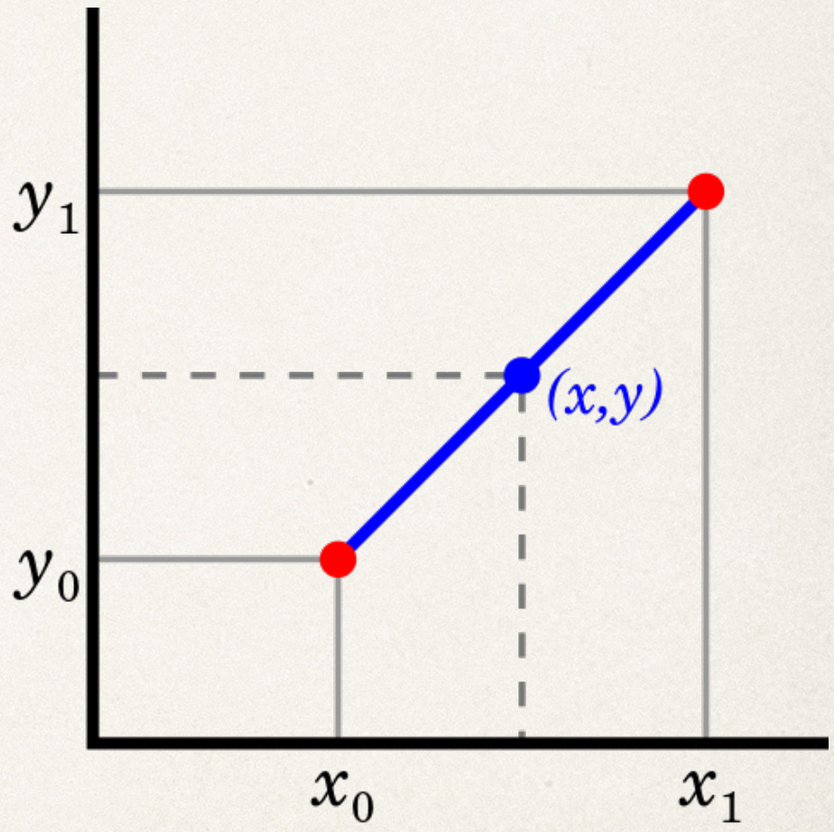
- View center position adjusted over time
- Horizontal slide: center.x increases (slide right) or decreases (slide left)
- Vertical slide: center.y increases (slide down) or decreases (slide up)

Spinning

- View transform orientation adjusted over time
  - Calculate angle of rotation using radians
- Rotate clockwise: increase angle of rotation matrix
- Rotate counterclockwise: decrease angle of rotation matrix

Linear Interpolation

- Changes over time give the appearance of an animation
- Given a starting and ending target, change by a fixed value at each time step
- Change happens at a linear rate

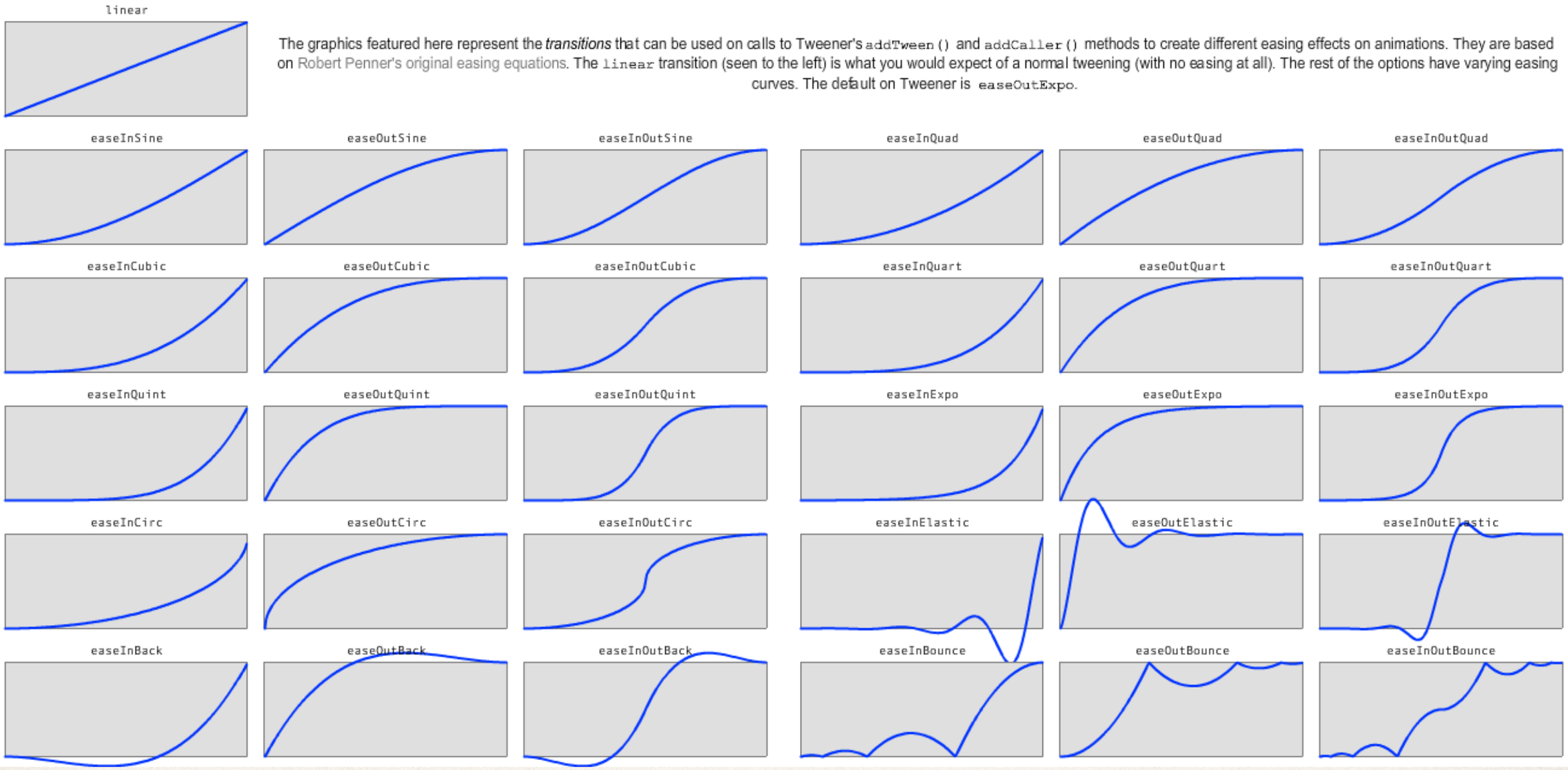


Animation Options

- `options` contains an array of values that define animation appearance
  - Easing, loops, transition style, etc
- Documentation: https://developer.apple.com/documentation/uikit/uiview/animationoptions
- Demonstrations: https://medium.com/@apmason/uiview-animation-options-9510832eedba

Easing

- Easing allows movement between two values at nonlinear increments
  - Objects can accelerate/decelerate as they approach the target
- Equation determines the fraction of the distance between the object's current and target positions that the object moves



(https://code.google.com/archive/p/tweener/)

Easing Options

- `CurveEaseInOut` causes the animation to begin slowly, accelerate through the middle of its duration, and then slow again before completing

- `CurveEaseIn` causes the animation to begin slowly, and then speed up as it progresses

- `CurveEaseOut` causes the animation to begin quickly, and then slow as it completes

- `CurveLinear` causes an animation to occur evenly over its duration (a linear interpolation)

Animated Transition Options

- Standard transitions between views
- `transitionFlipFromLeft/transitionFlipFromRight`
- `transitionCurlUp/transitionCurlDown`
- `transitionCrossDissolve`
- `transitionFlipFromTop/transitionFlipFromBottom`

Loops and Reversing

- Indefinitely plays the animation in a loop
  - `setAnimationRepeatCount()` allows you to set number of times a block should repeat
  - Can also remove the repeat animation using `removeAllAnimations()` but this will cancel additional animations as well
- Autoreverse plays then reverses the animation
  - Usually used in conjunction with repeat

Animation Demo

Instapoll Question: Animations

- What sort of animation will applying the `CGAffineTransform.translatedBy` create?
  - Fade In/Fade Out
  - Sliding
  - Spinning
  - Resizing

Using Core Animation Directly

- Modify the `CALayer` to change object's appearance
  - Can draw directly in `CAShapeLayer` as well using shapes and curves!
- Animations performed through `CAAnimation`
  - Interpolation handled automatically
  - Must define additional information about start and stop
- A little more work but a lot more control

CAAnimation

- Abstract class that provides additional animation support
  - `CABasicAnimation`
  - `CAKeyframeAnimation`
  - `CAAnimationGroup`
  - `CATransition`
- Allows for a variety of interpolations

CABasicAnimation

- Define a BasicAnimation using `CABasicAnimation(keyPath: CALayer property)`

- Define `fromValue` (initial value of CALayer property) and `toValue` (final value of CALayer property)

- `repeatCount` defines number of times to repeat the animation (-1 loops indefinitely)

- Add animation to a `CALayer` using `addAnimation()`

Additional Animations

- `CAKeyframeAnimation`
  - Allows for multiple keyframes across animation
  - Array of values defined in `values`
  - Includes additional functionality for controlling curves along transitions
- `CASpringAnimation`
  - Creates animations that have physically-based spring-like properties
  - Control animation feel via spring stiffness and damping