# IT'S AAALIVE!

## ANIMATING IOS APPLICATIONS

Błażej Marcinkiewicz

@blazejmar

marcinkiewicz.blazej@gmail.com

## AGENDA

1. Core Animation introduction
2. CAShapeLayer
3. View/layer transformations
4. Creating animatable properties
5. Driving animations manually

## WHY CORE ANIMATION?

1. Low level animation framework
2. More configuration than block animations
3. Can animate all animatable layer properties
4. Can animate non visual properties
5. Easy to dismiss

## INTRO

1. Works on layers instead of views
2. CABasicAnimation is the simples block ready to use
3. CAKeyframeAnimation can be used to create more complicated transitions

## CABASICANIMATION

1. Initial and end values or increment for animated property
2. Number of repetitions and behavior after finishing animation
3. Full timing control (begin time, repeat duration)
4. Keypath to animate, delegate and many more

## CAKEYFRAMEANIMATION

- Values in between frames
- Interpolation between key frame values

## DEMO

## TASK

- Change the animation in [SwapView startAnimating] to Core Animation
- Tweak the animation so that in second phase right image goes on top of left image

## CASHAPELAYER

What's so special about CAShapeLayer:

- CALayer objects are rendered as rectangles, CAShapeLayer can define arbitrary path
- Great to use in transition animations between views (can be used as maskLayer)
- CAShapeLayer path can be animated

## CAMEDIATIMING

- Protocol implemented by CALayers and CAAnimations
- Let's you define timing properties of animations

## DEMO

## TASK

- Rework transition between images in RefreshViewController
- Animation should remove placeholder image with animation
- Hint 1: Requires adding additional UIImageView behind imageV of RefreshTableViewCell
- Hint 2: CAShapeLayer should be used as a mask layer for view in front
- Hint 3: Placeholder image should be set to image view in front, downloaded image should be set to image in the back, animation should change path from covering full image view bounds to empty, at the end placeholder image should be replaced with downloaded image

## A BIT OF MATH

- UIView is using 2D math to compute transforms
- CALayer is using 3D math to compute transforms

## TRANSFORMATION MATRIX

Describes how to transform vector V. Result is computed by multiplying transformation matrix by a vector. If you use more than one transformation they're applied from right to left.

2D transforms are using 3x3 element matrices, 3D transforms 4x4. Transforms are applied around (0, 0, 0) point.

## TRANSLATION

$$T_{\mathbf{v}} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## SCALING

$$S_v = \begin{bmatrix} v_x & 0 & 0 & 0 \\ 0 & v_y & 0 & 0 \\ 0 & 0 & v_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

## ROTATION

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## PERSPECTIVE

Perspective is a distortion transformation. One of the factors responsible for is m34 coefficient.

## CHANGING AXIS OF ROTATION

- Cannot be done on UIView :(
- Can be done using CALayer :)
- CATransform3DMakeRotation(angle, x, y, z)

## HOW THE VIEWS ARE RENDERED

- 4 properties describe where view is rendered: frame, center, bounds, transform
- Frame is not a real property, it's only for convenience, it's computed from center and bounds
- Base properties describing position are center and bounds

## HOW TO CHANGE POINT WHERE TRANSFORMS ARE APPLIED?

- CALayer has anchor point property
- Anchor point is using normalized coordinates (top left corner is (0, 0) bottom right (1, 1))
- Important: Changing anchor point immediately changes the rendering position of view

# DEMO

## TASK

- Change animation of side menu (RootView)
- Animation should flip side menu from left side of the screen, this may require layout change

## CUSTOM PROPERTY ANIMATIONS

- Redefine setter and create and make it fire animation
- Implement CALayer methods: actionForKey and needsDisplayForKey
- Limitation of CALayer methods: it can only interpolate only CoreGraphics values (int, floats, rects, etc.)

## DISPATCHING ANIMATIONS

1. When dispatching animations we provide only initial and end value for properties
2. Render server underneath your app decodes this call
3. Render server using current time interpolates between values (can call display on each view's layer)

## HOW DOES IT WORK?

- When value is set to a property in CALayer system checks if layer has defined some action for it
- When animating asks if it should redraw the layer
- Each layer consists of model layer and presentation layer
- Model layer is updated immediately after dispatching animation
- It's the presentation layer which changes over time

## TASK

- Create animation for progressValue property of RatingLayer
- Setter should trigger animation with linear interpolation between initial and provided value

## MANUAL DRIVING ANIMATIONS

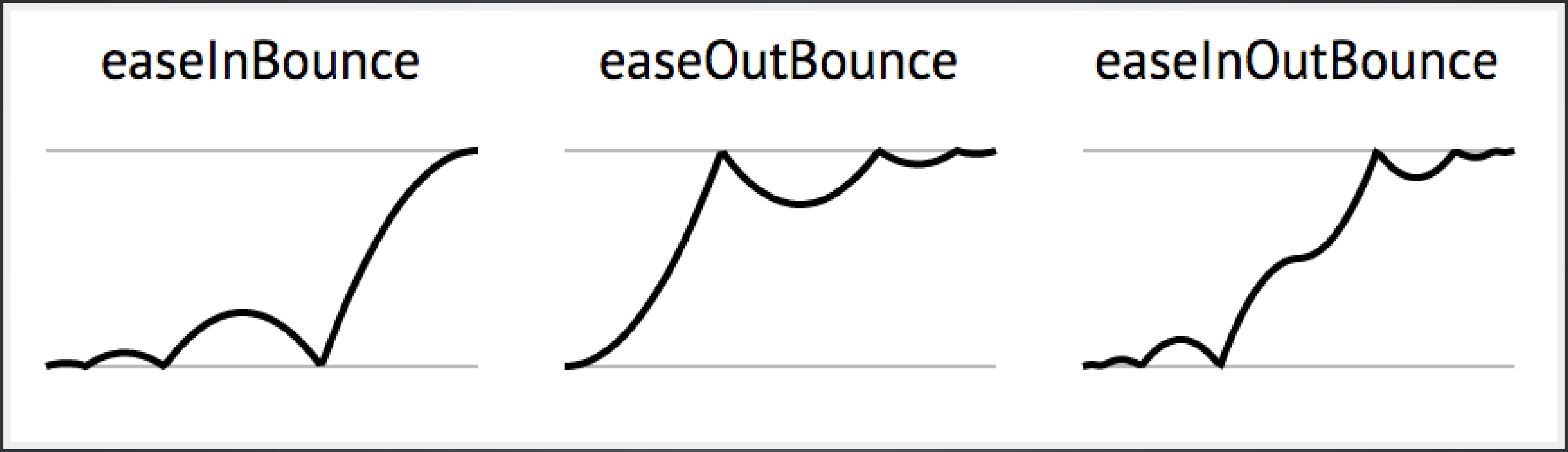
Basic animations in Core Animation are also limited, it's still much more than UIView block animation offers, but you can't do anything.

## WHAT WILL WE HAVE TO DO?

- Interpolate between our custom values
- Synchronize animation frames with rendering
- Define interpolation curve

## EASING

Easing defines how the interpolation works.



## BOUNCING ANIMATION

For iOS 7 you can use UIDynamicBehavior. Drawbacks:

- It's quite heavy to compute with collisions and gravity
- Hard to control
- Takes long time to tweak

## BOUNCING ANIMATION

How to create it manually:

- Given the current time compute proper value for easing curve (requires some math)
- Use CADisplayLink to synchronize with screen

## CADISPLAYLINK

- API looks similar to a normal timer
- Main difference from regular timer is that CADisplayLink is connected with render server
- CADisplayLink fires every time screen redraws

## TASK

- Create bounce animation for red rectangle in EasingView

## SUMMARY

- Core Animation gives you fine grain control over animations
- Using CADisplayLink is quite heavy and may cause frame rate drops
- Good math knowledge helps :)

THANK YOU!

## REFERENCES:

- Images of transform matrices - Wikipedia
- Objc.io - Issue 12