Creating 2D Games with Sprite Kit

Rich Warren

Rich Warren

- iOS Consultant, Writer and Trainer
- Writing articles for MacTech Magazine since 2006
- Wrote *Creating iOS 5 Apps: Develop and Design* and *Objective-C Bootcamp* for Peachpit Press.
- Senior instructor for About Objects

rich@freelancemadscience.com
www.freelancemadscience.com
google.com/+RichWarren
@rikiwarren on Twitter

Creating iOS Apps:
Develop And Design
Second Edition

- Fully updated for iOS 7
- Focuses on modern, best practices
    - *Storyboards*
    - *ARC*
    - *Auto Layout*
- Designed with the new UI Paradigm in mind
    - *Clean, content-focused interface*
    - *Emphasizes animation over ornamentation*

Available December 5!

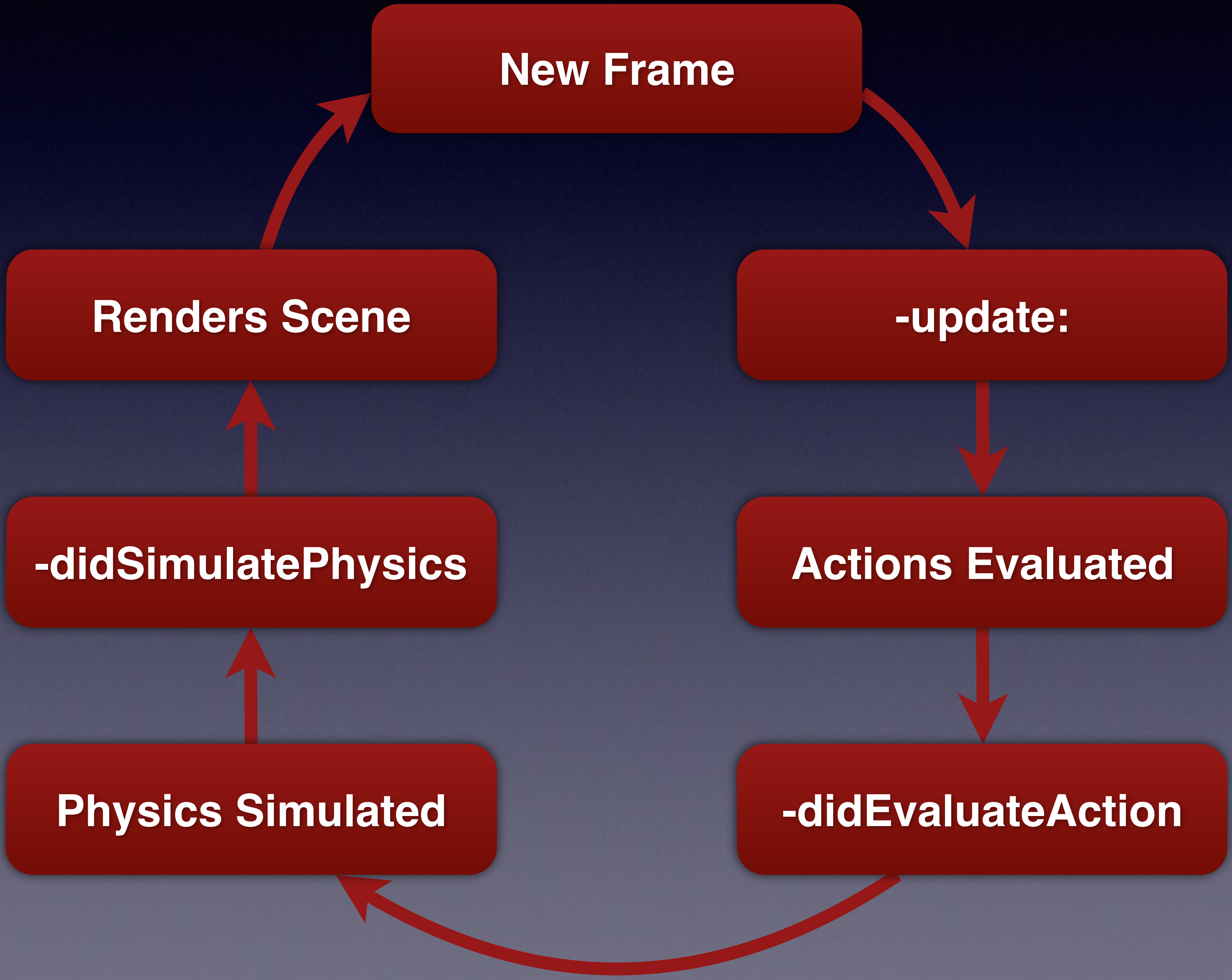
What is Sprite Kit?

- High-performance rendering and animation framework
- Designed for 2D game development
- Objective-C wrapper around OpenGL
- Built-in Physics Engine
- Runs on iOS and OS X

Major Classes

- `SKView`
- `SKScene`
- `SKNode`
- `SKTexture`
- `SKTextureAtlas`
- `SKAction`
- `SKPhysicsWorld`
- `SKPhysicsBody`
- `SKPhysicsJoint`
- `SKPhysicsContact`

Each Frame



New Frame

-update:

Actions Evaluated

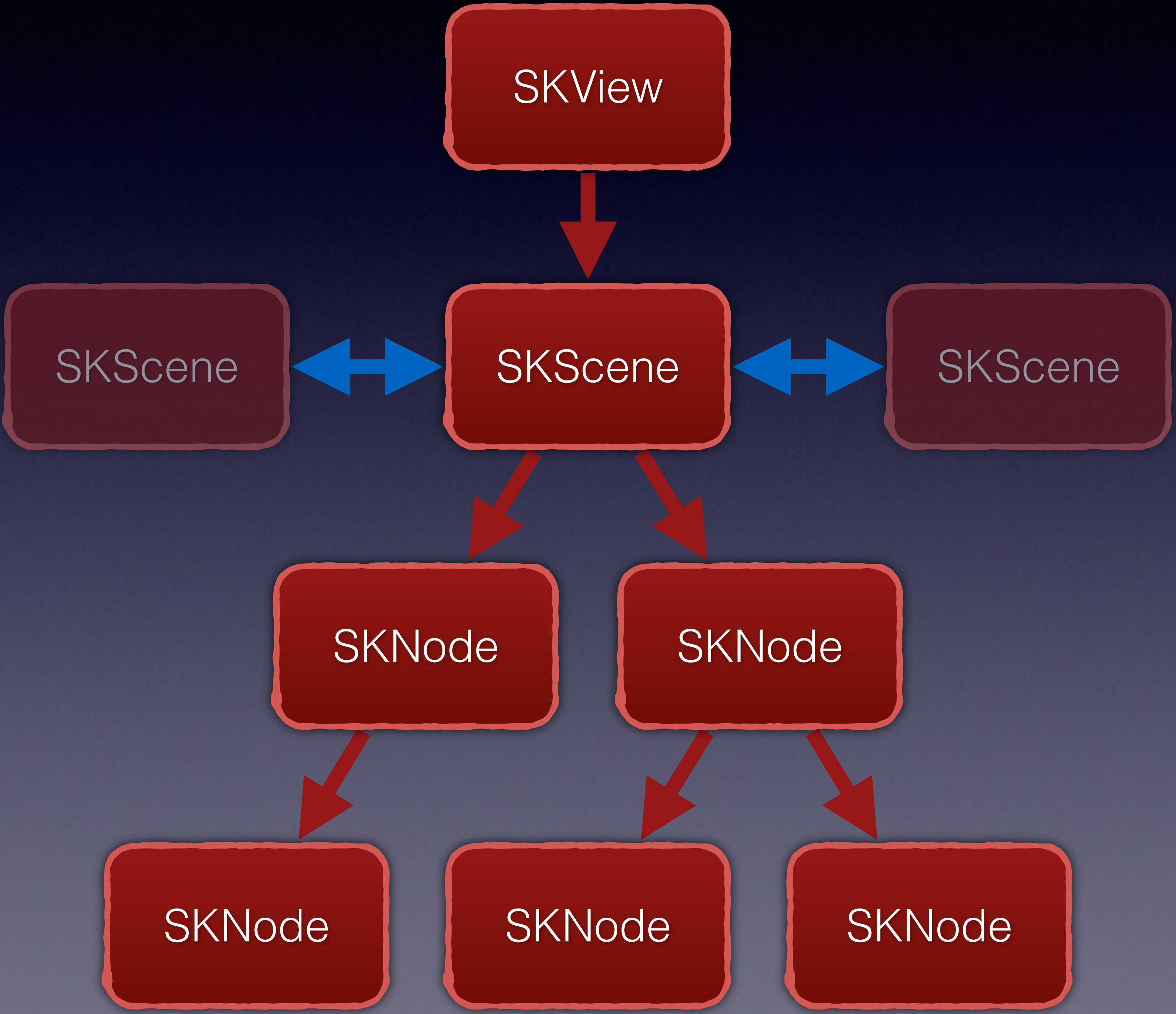
-didEvaluateAction

Physics Simulated

-didSimulatePhysics

Renders Scene

Node Tree


SKView

SKScene

SKScene

SKScene

SKNode

SKNode

SKNode

SKNode

SKNode

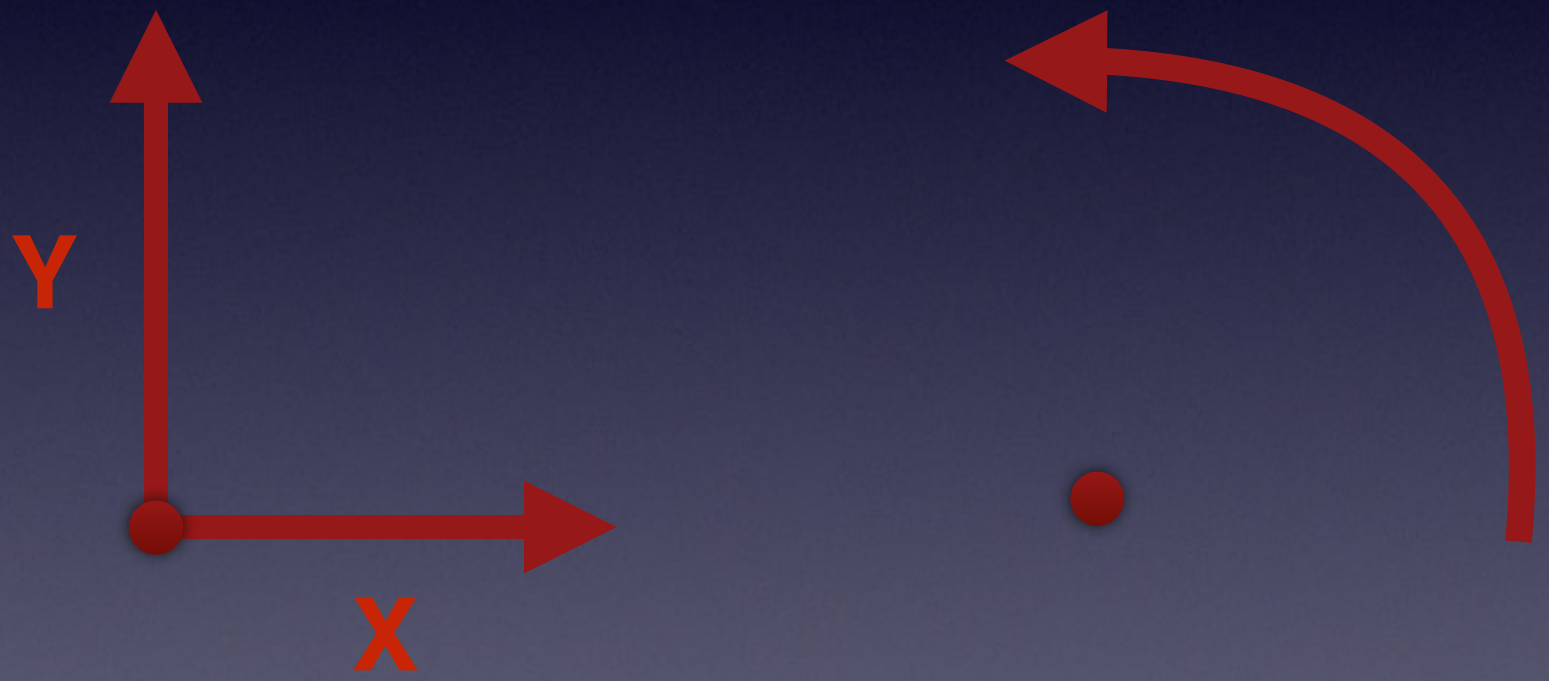
Integration

- `SKView` can be added to the view hierarchy
  - *`SKView` is opaque!*
  - *We can layer other views and controls over it, but cannot display anything behind it.*
- `SKNode` subclasses `UIResponder`/ `NSResponder`
  - *SKView automatically extends the responder chain to include the nodes in the active scene.*

Node Types

- `SKNode`
- `SKSpriteNode`
- `SKLabelNode`
- `SKShapeNode`
- `SKVideoNode`
- `SKEmitterNode`
- `SKCropNode`
- `SKEffectNode`
  - *`SKScene` is a subclass of `SKEffectNode`.*

Coordinate System



Anchor Point

- Sets the origin for the node's coordinate system
- Reference point for the node's position in its parent's coordinate system
- Node rotates around the anchor point
- `SKSpriteNode` and `SKVideoNode` defaults to `{0.5, 0.5}`
- `SKScene` defaults to `{0.0, 0.0}`
- `SKShapeNode` has an implicit anchor point

Textures

- Memory automatically managed by Sprite Kit
  - *Automatically loads texture data when necessary*
  - *When no longer on the scene or visible, Sprit Kit can deallocate the texture to free up memory*
- May need to preload textures to prevent rendering issues
- Group similar images in a Texture Atlas so they can be drawn in a single pass

SKAction

- Uses a number of private subclasses
- We create particular `SKAction` instances using class methods
- Compose complex actions using sequence, group or repeating actions
- Create once, reuse many times

Types of Actions

- Animate changes to a `SKNode`’s position, rotation, size, visibility or tint color
- Animate `SKSpriteNode` by changing its texture
- Play simple sounds
- Remove the node from the node tree
- Call a block or selector.

Demo 1

Radar

- **15420351:** Always loads images from the @2x texture atlas
- **15420479:** Race condition when preloading a large number of textures using `+[SKTexture preloadTextures: withCompletionHandler:]`

Physics Engine

What is a Physics Engine?

- Define the physical properties of your objects
  - *Size, Shape, mass, velocity, friction, elasticity and more*
- Define the physical aspects of your world
- Calculates the motion of objects over time
- Respond to collisions

How Does it Work?

- Add an `SKPhysicsBody` to a node
  - *Sets the physical properties for the object*
  - *Physics calculations performed for all nodes with physics bodies in the scene*
- Modify the scene's `SKPhysicsWorld`
  - *Sets global properties*

SKPhysicsBody

- Dynamic vs. Static
- Volume vs. Edge
- Affected by gravity
- Allows rotation
- Other physical characteristics: `mass`, `density`, `area`, `friction`, `restitution`, `linearDamping`, `angularDamping`

Contacts and Collisions

- Contacts provide notifications
- Collisions provide impulses
- We can define node categories, and define how they interact
    - `categoryBitMask`
    - `collisionBitMask`
    - `contactBitMask`

Demo 2

Best Practices

- Organize the game into scenes
- Limit the contents of the node tree
- Create subclasses to provide custom behavior
- Nodes adopt `NSCopying` and `NSCoding`
    - *Our subclass must properly handle their properties*
- Avoid adding content nodes to the scene
    - *Create layers using `SKNode`, and add object nodes to them instead.*

- Clipping and effect nodes are expensive
    - ***Use sparingly***
- Nodes that are drawn together should use the same blend mode and texture atlas
    - *This lets Sprite Kit draw them in a single drawing pass*
- Limit the number of particles on the screen
    - *Use low birthrates or short lifetimes*

- If a sprite's content is opaque (e.g., background images) use `SKBlendModeReplace`.

- Use game logic and assets that match Sprite Kit's coordinate system
    - *Orient artwork to the right*

- Test on a wide range of hardware

Creating Tools

- Archive individual nodes to make them easily accessible
- Create game levels by archiving scenes
- Save the game by archiving the current scene
- Unsaved Data
  - *Shape of a physics body*
  - *Actions that execute a block*

Sprite Kit vs Cocos2D

Sprite Kit Advantages

- Part of the iOS platform
- Integrated physics engine
- Full ARC support
- Xcode Tools
    - Automatic Texture Atlases
    - Particle Editor

Cocos2D Advantages

- Supported by older versions of iOS
- Adds features like tile maps, cameras and shaders
  - *Tools like Kobold Kit can add some of these*
- Has extensive third-party library support
  - *Many of these have been or are being rewritten to work with Sprite Kit*

Questions?

Rich Warren

rich@freelancemadscience.com
www.freelancemadscience.com
google.com/+RichWarren
@rikiwarren on Twitter