what's in an update?

Example Scenarios

Device scenarios	stationary	walking	running	automotive	cycling	unknown
On table	true	false	false	false	false	false
On runner's upper arm	false	false	true	false	false	false
In dash of idling vehicle	true	false	false	true	false	false
In dash of moving vehicle	false	false	false	true	false	false
Passenger checking email	false	false	false	false	false	false
Immediately after reboot	false	false	false	false	false	true
In zumba class	false	false	false	false	false	false

https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

Walking

Performance is fairly insensitive to location

- Detection can be suppressed when device is in hand

Relatively low latency

Very accurate, on average

- Expect intermittent transitions into and out of walking state

https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

Running

Completely insensitive to location

Shortest latency

Most accurate classification

https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

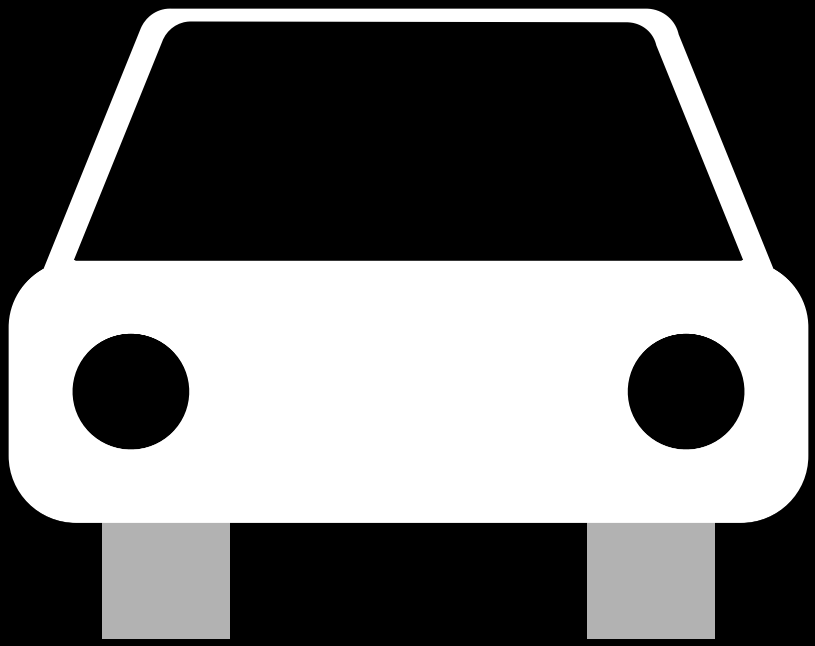
Automotive

Performance is sensitive to location

- Works best if device is mounted, or placed in dash or in cup holder

Variable latency

Relies on other information sources when available



https://developer.apple.com/videos/wwdc/2014/

what's in an update?

Motion Activity

Cycling

Performance is very sensitive to location

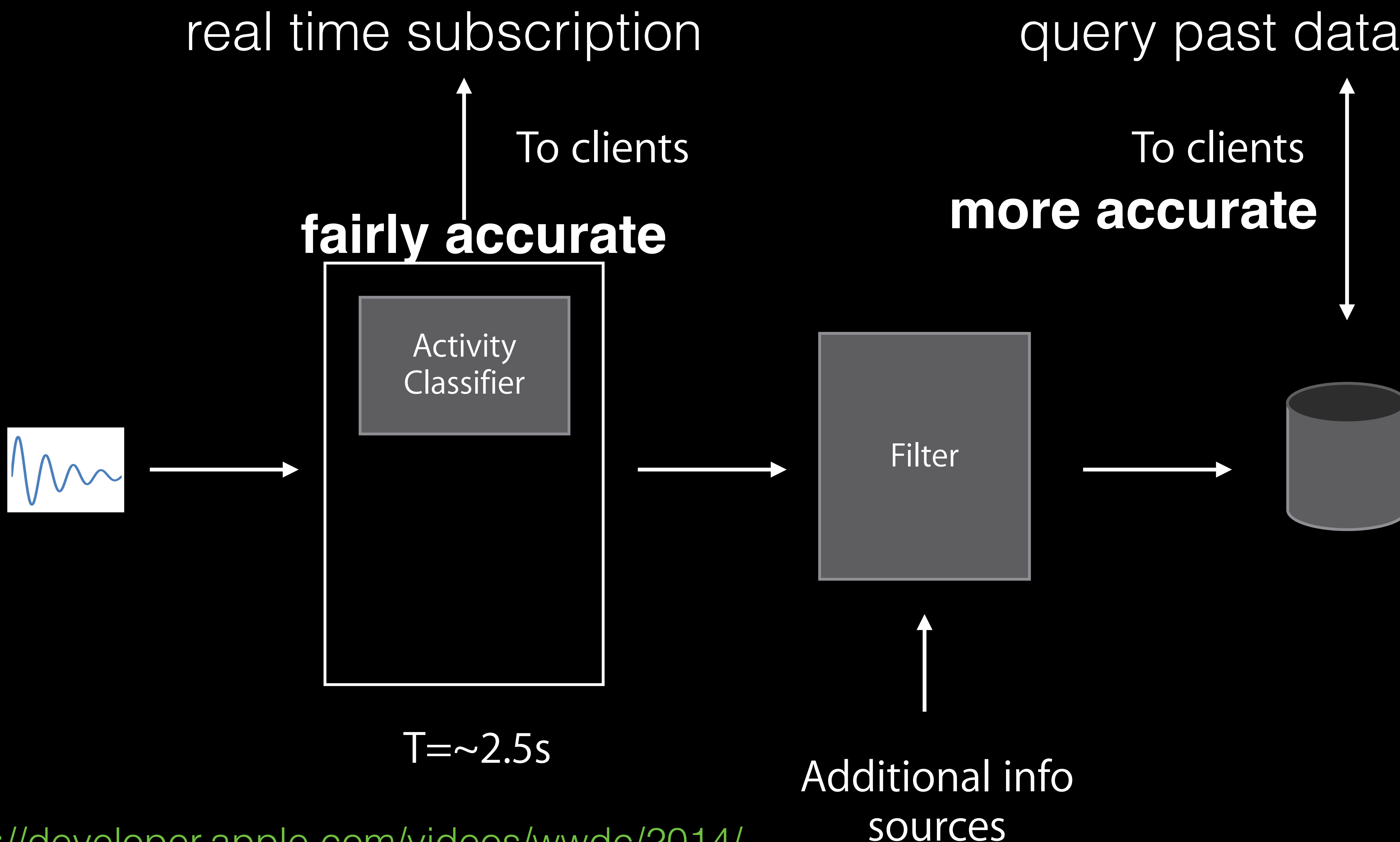
- Works best if device is worn on upper arm

Longest latency

- Best for retrospective use cases

https://developer.apple.com/videos/wwdc/2014/

Motion Processing Architecture



https://developer.apple.com/videos/wwdc/2014/

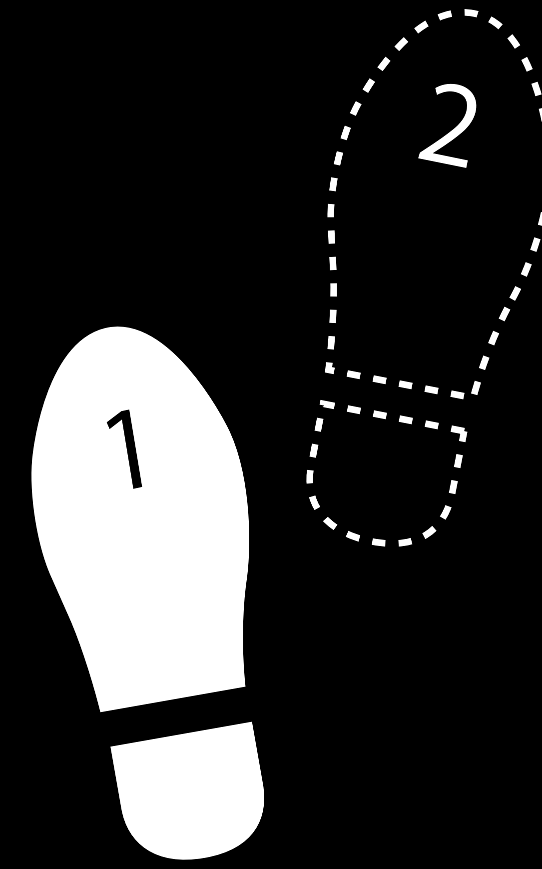
more than activity

- also tracks pedometer information during each activity
- like activity: setup as a **push** system (subscribe)

- pedometer: special handling from the A-series
    - CMPedometer

Pedometer

Step counting



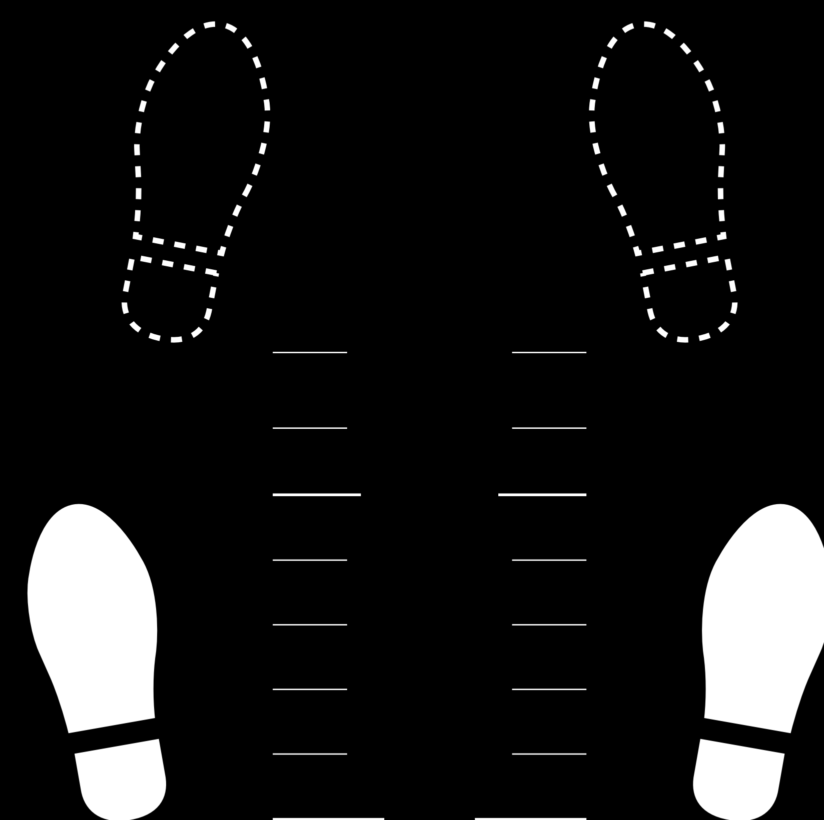
Consistent performance across body locations

Extremely accurate

Robust to extraneous motions

Pedometer

Stride estimation



Consistent performance across body locations

Consistent performance across pace

Extremely accurate

Adapts to the user over time

https://developer.apple.com/videos/wwdc/2014/

pedometer use

declare and init

available on this device?

```swift
let pedometer = CMPedometer()

if CMPedometer.isStepCountingAvailable(){
    pedometer.startPedometerUpdatesFromDate(NSDate())
        { (pedData: CMPedometerData?, error:Error?) -> Void in
            NSLog("%@",pedData.description)
        }
}
```

closure handler for updates

```swift
if CMPedometer.isStepCountingAvailable(){
    self.pedometer.stopPedometerUpdates()
```

unsubscribe

pedometer use

revisiting

```swift
let pedometer = CMPedometer()

if CMPedometer.isStepCountingAvailable(){
    pedometer.startPedometerUpdatesFromDate(Date())
        { (pedData: CMPedometerData?, error:Error?) -> Void in
            NSLog("%@",pedData.description)
        }
}


if CMPedometer.isStepCountingAvailable(){
    self.pedometer.stopPedometerUpdates()
```

declare and init

available on this device?

properties from step counter

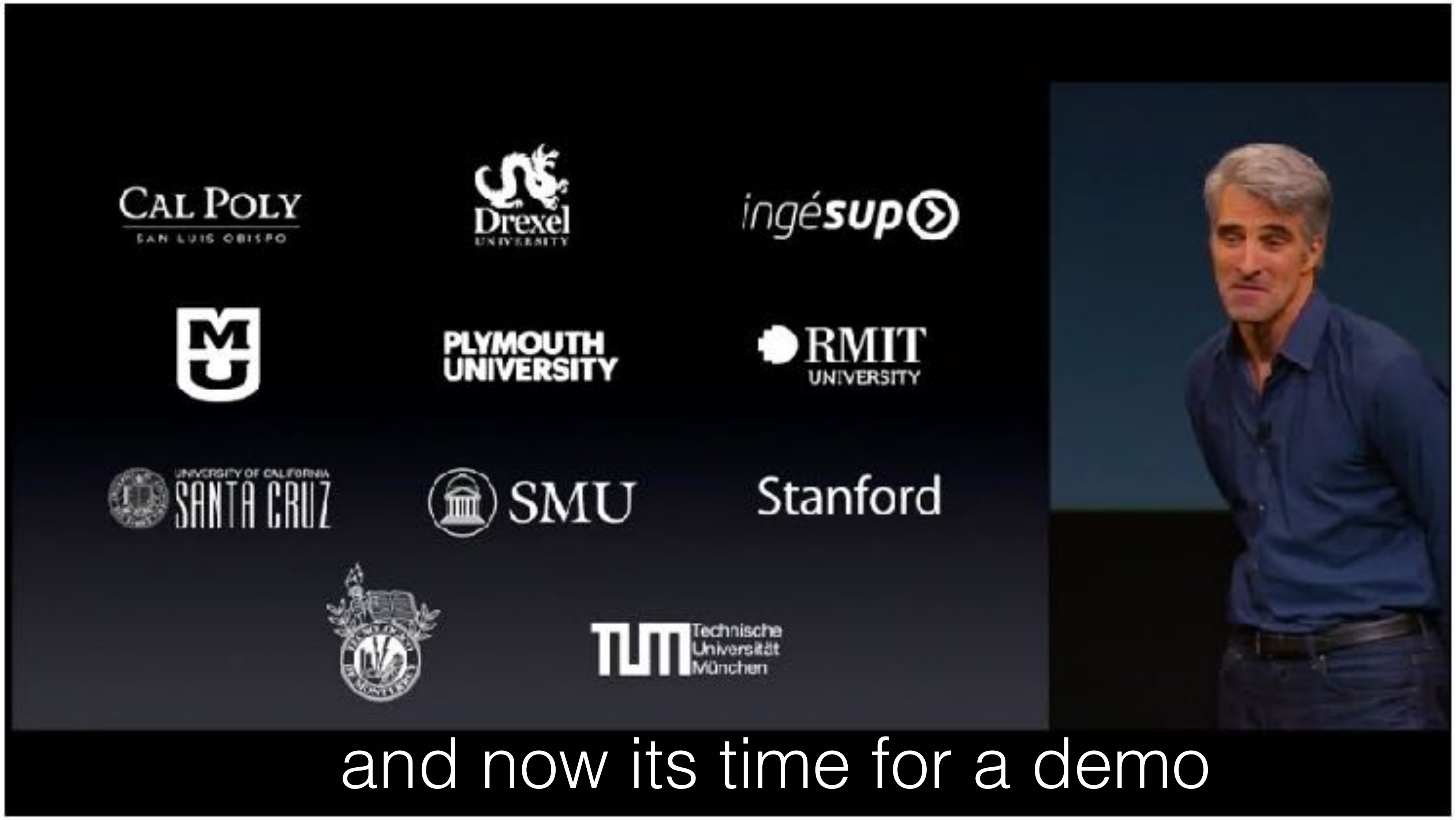
unsubscribe

CMPedometerData,<startDate 2021-09-21 13:56:54 +0000 endDate 2021-09-21 13:57:17 +0000 steps 35 distance 27.57728308765218 floorsAscended 0 floorsDescended 0 currentPace 0.5944125511973894 currentCadence 2.172188043594366 averageActivePace 0.616343178495018>

querying past steps

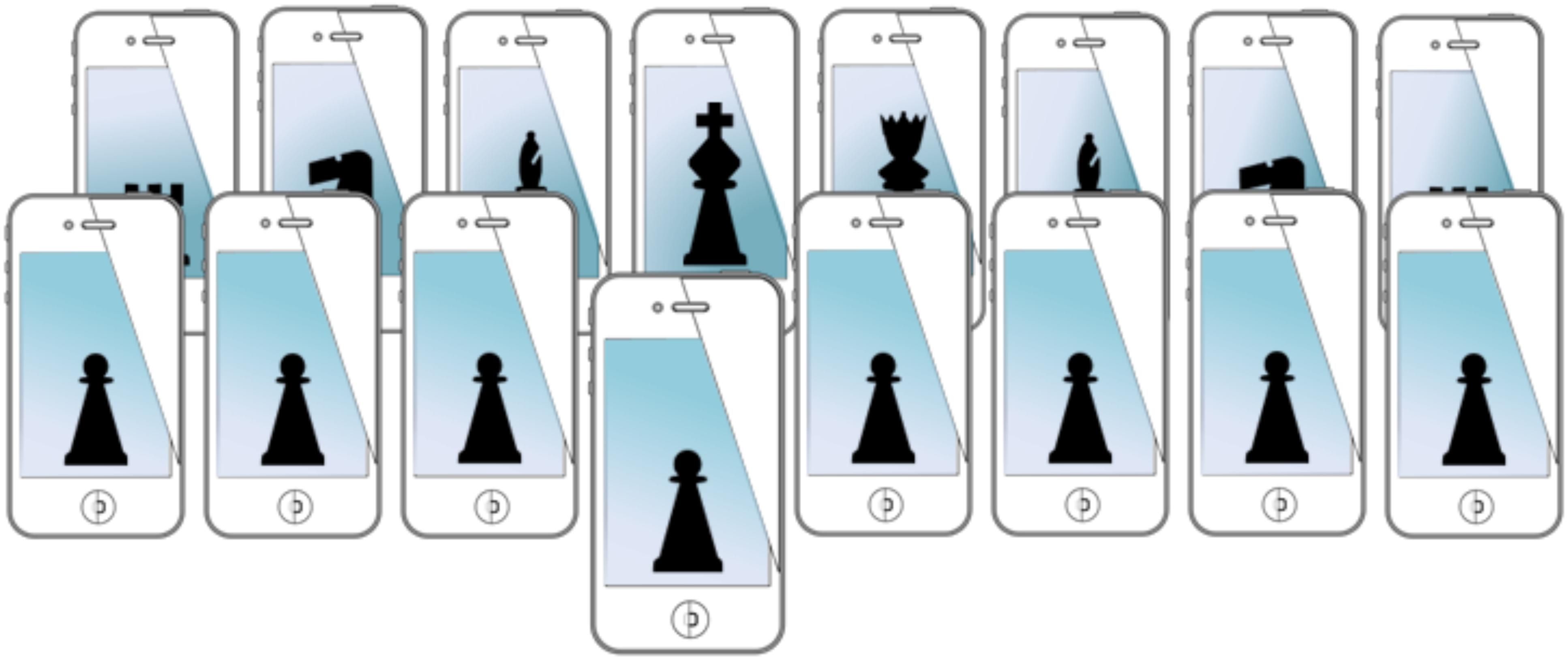

```
let now = Date()
let from = now.dateByAddingTimeInterval(-60*60*24)

self.pedometer.queryPedometerDataFromDate(from, toDate: now)
{ (pedData: CMPedometerData?, error: Error?) -> Void in

    let aggregated_string = "Steps: \(pedData.numberOfSteps) \n
           Distance \(pedData.distance) \n
           Floors: \(pedData.floorsAscended.integerValue)"

    dispatch_async(dispatch_get_main_queue()){
        self.activityLabel.text = aggregated_string
    }
}
```

pedometer/activity demo



if time!

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

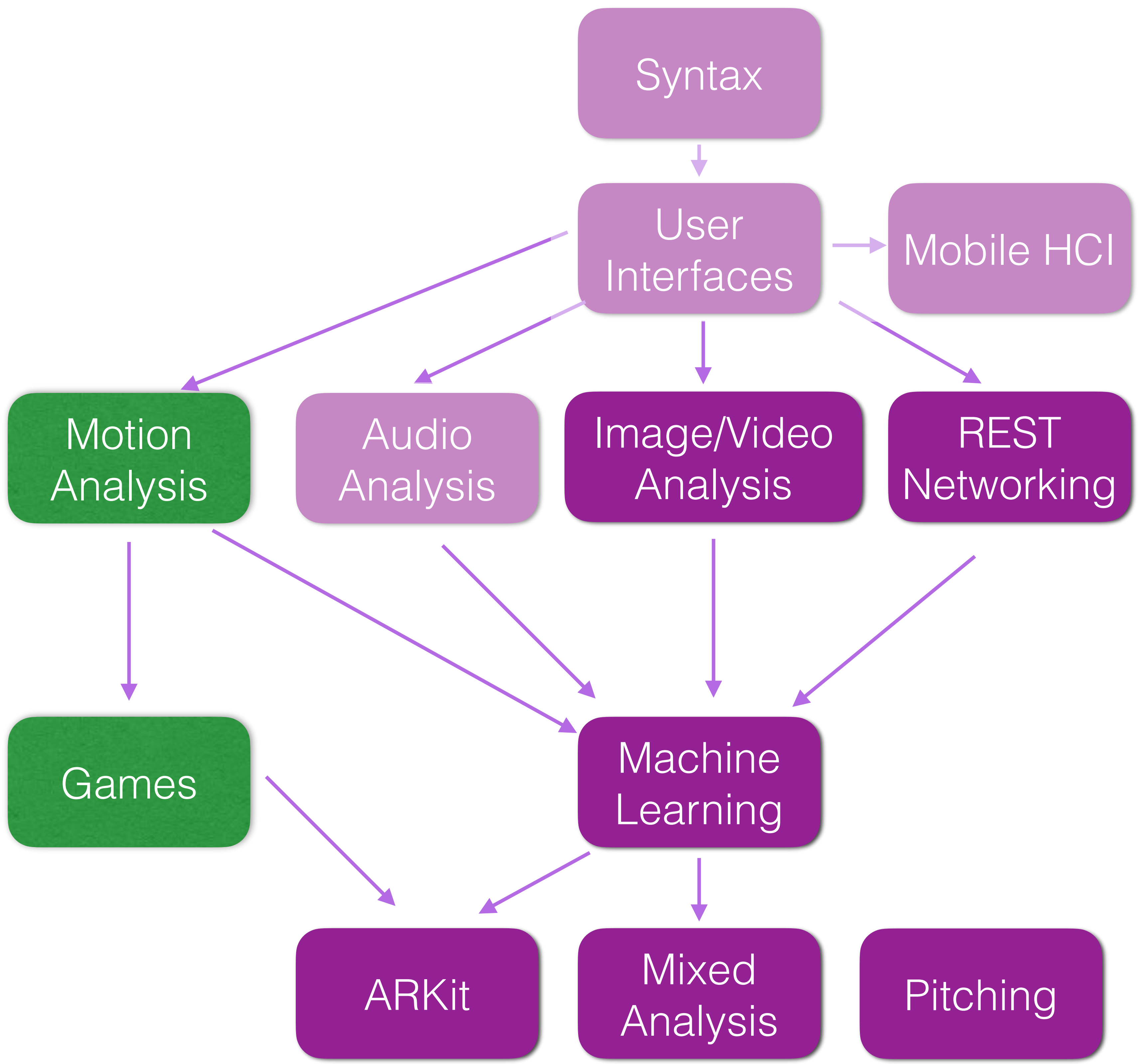
activity, pedometers, and motion sensing

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

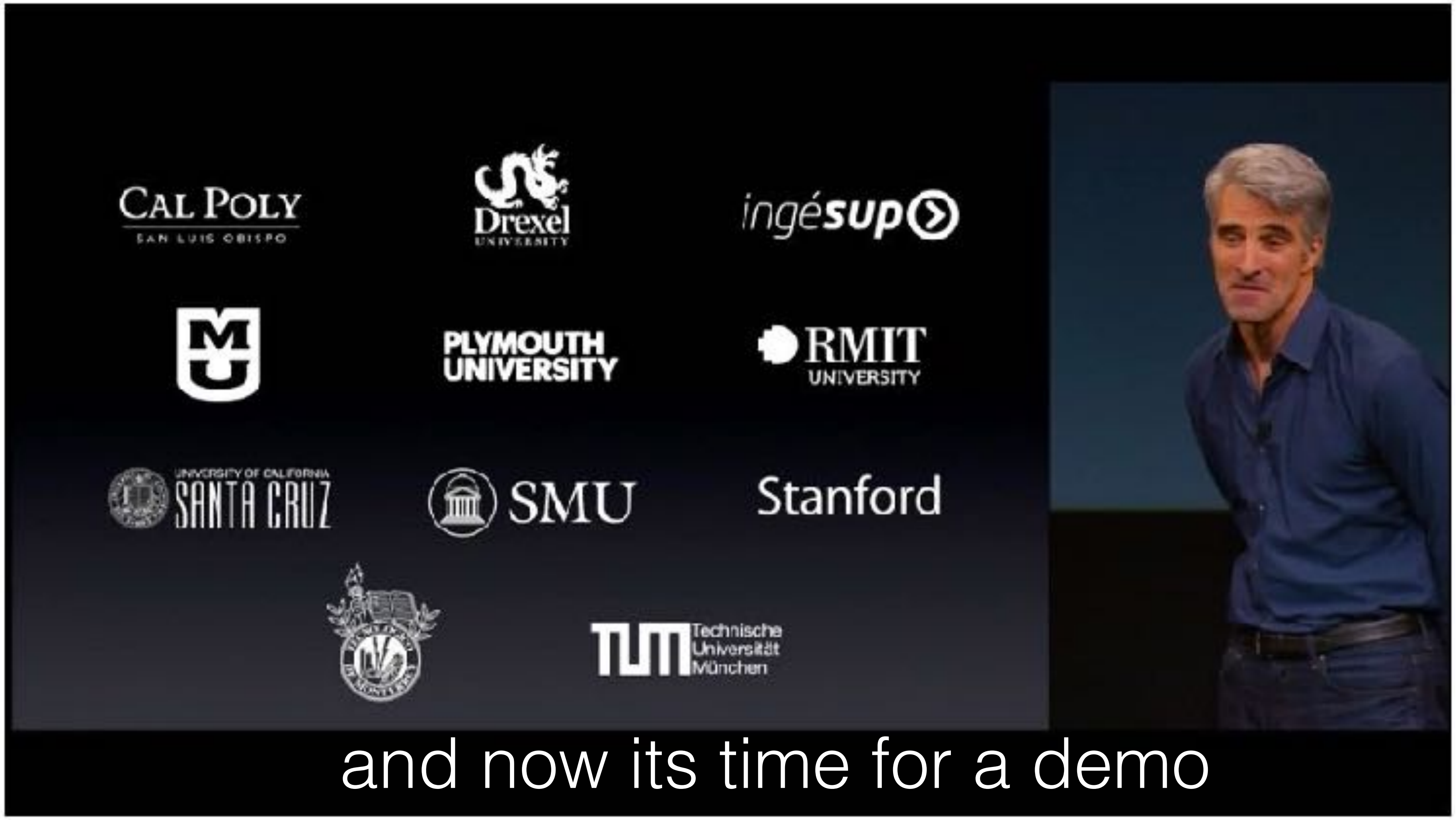
logistics and agenda

- Logistics:
  - A2 due soon, grading
- agenda:
  - core motion (continued)
    - A-series
    - demo
  - accelerometers, gyros, and magnetometers
  - SpriteKit
  - SceneKit

class overview



pedometer/activity demo



“continue” demo!

“raw” motion data

Barometer

The barometer senses air pressure to determine your relative elevation. So as you move, you can keep track of the elevation you've gained. It can even measure stairs climbed or hills conquered.

Accelerometer

The accelerometer can measure your distance for walking and running. And by using GPS to calibrate for your running stride, the sensor more accurately captures your movement.

Gyroscope

In addition to knowing whether you're on the move or stationary, M8 works with the gyroscope to detect when you're driving. It also kicks into action when you're taking panoramic photos or playing games that react to your movement.

“raw” motion data

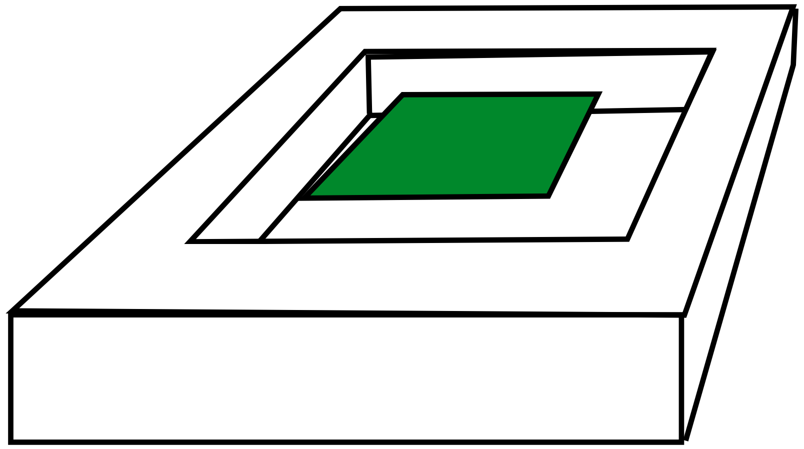
- A-series mediates access to data
- much lower battery consumption

iPhone 5	At 100Hz		At 20Hz	
	Total	Application	Total	Application
DeviceMotion	65%	20%	65%	10%
Accelerometer	50%	15%	46%	5%
Accel + Gyro	51%	10%	50%	5%

iPhone 5s	4%	1%
iPhone 6, 6S	~2%	1%
iPhone 7	~?%	?%

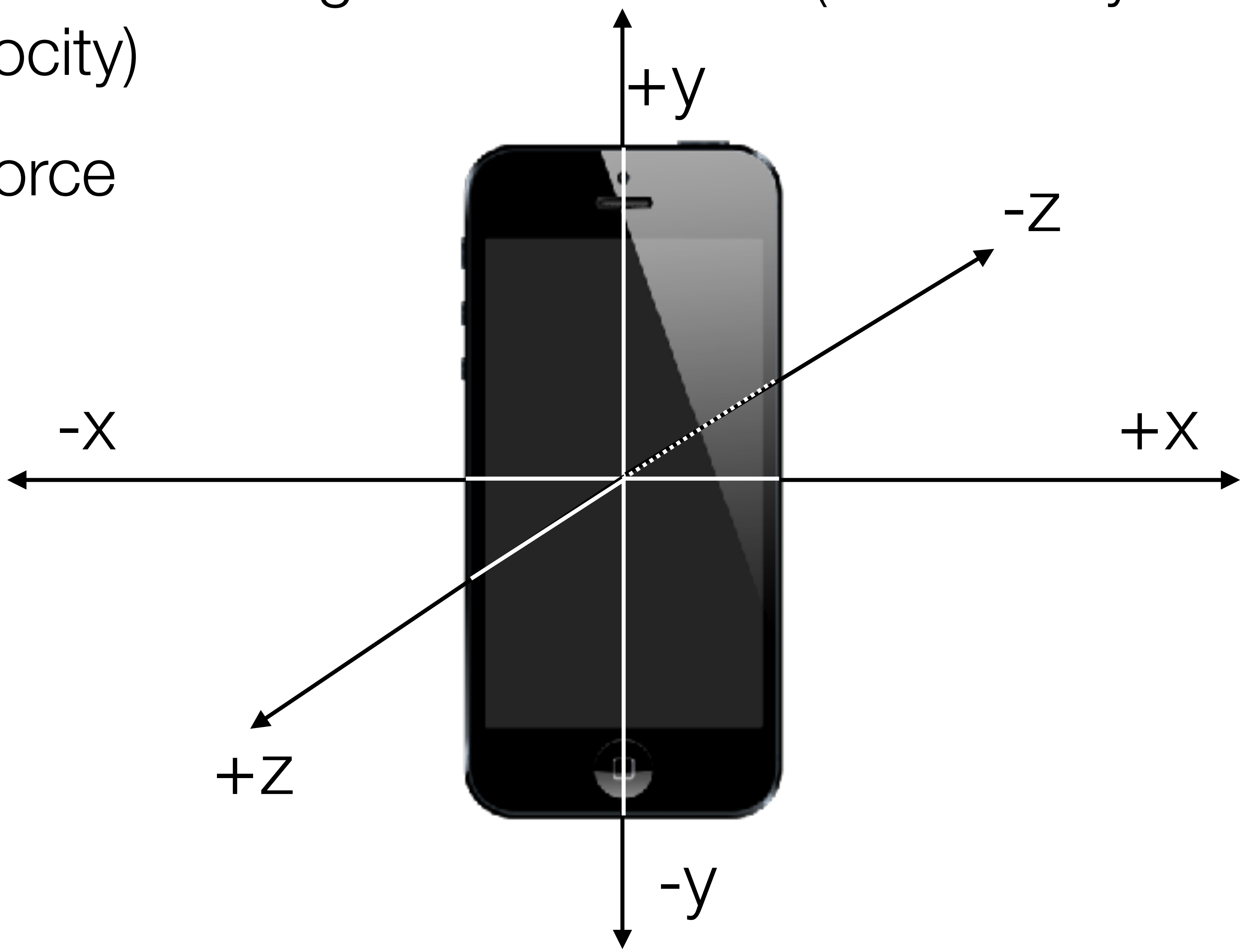
accelerometers

- how does it work?
- solid state device (fabricated on a chip)
- it has specs (not made public by Apple)
  - swing
    - +-8g (force)
  - bias and variance
    - bias can be high, easy to zero out
  - resolution
    - 20 bits or 0.000015g
  - bandwidth
    - 100Hz sampling is highest recommended



accelerometer

- measures “proper acceleration”
  - due to the weight of the device (not exactly derivative of velocity)
  - g-force



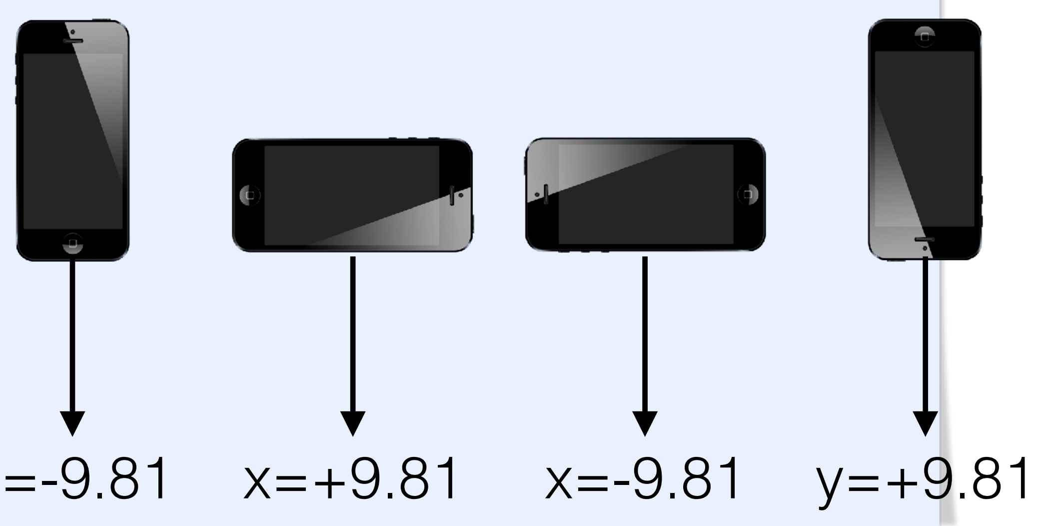
accessing the accelerometer

- usually don’t want the raw accelerometer value
- gravity is always pulling “down” on the device at a constant force of ~9.81g
- the core motion API automatically subtracts gravity from the user acceleration

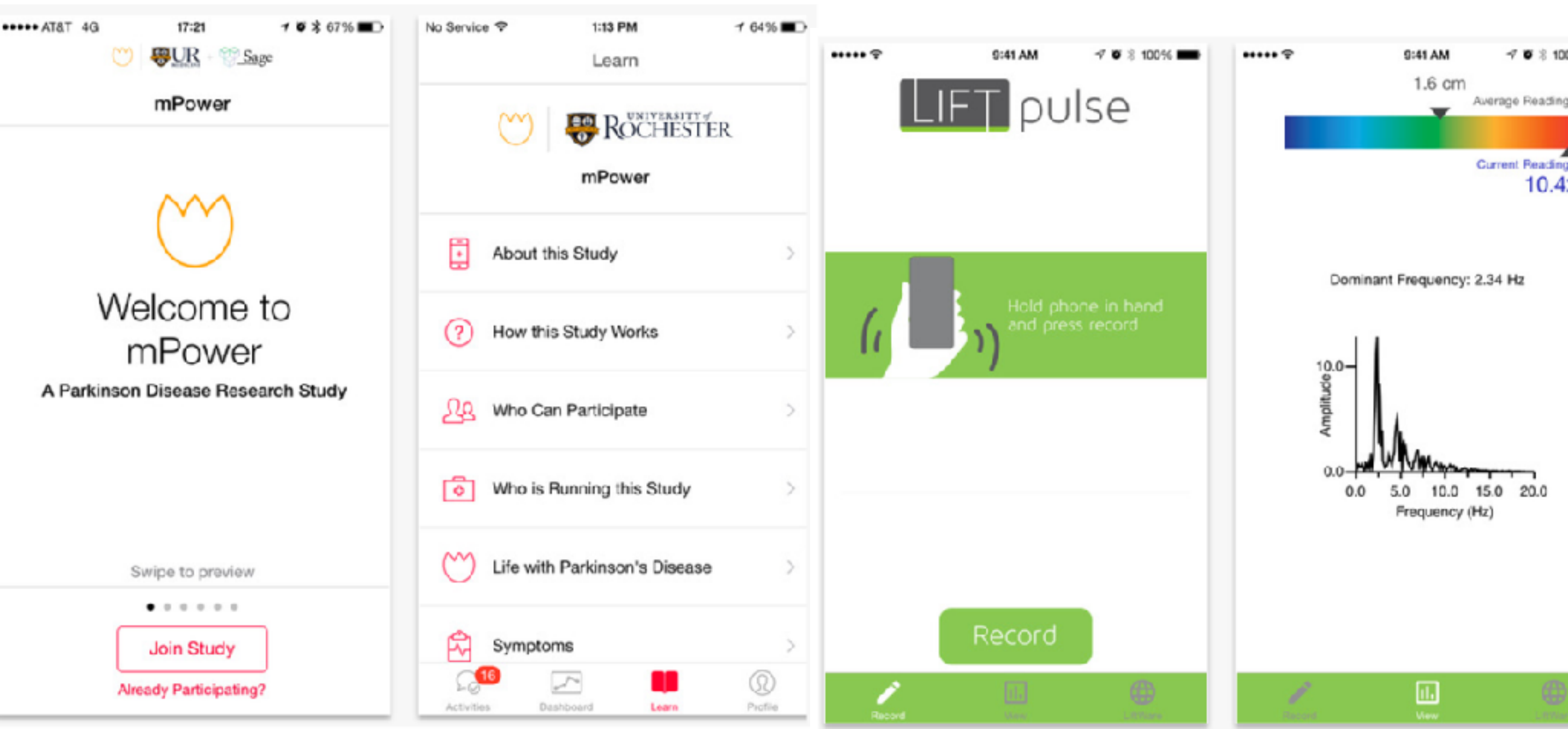
```
CMDeviceMotion *deviceMotion

deviceMotion.gravity
deviceMotion.userAcceleration

CMAcceleration gravity, CMAcceleration userAcceleration

gravity.x;
gravity.y;
gravity.z;

userAcceleration.x;
userAcceleration.y;
userAcceleration.z;
```

user movement

access through a different field!



a cool example



another cool example

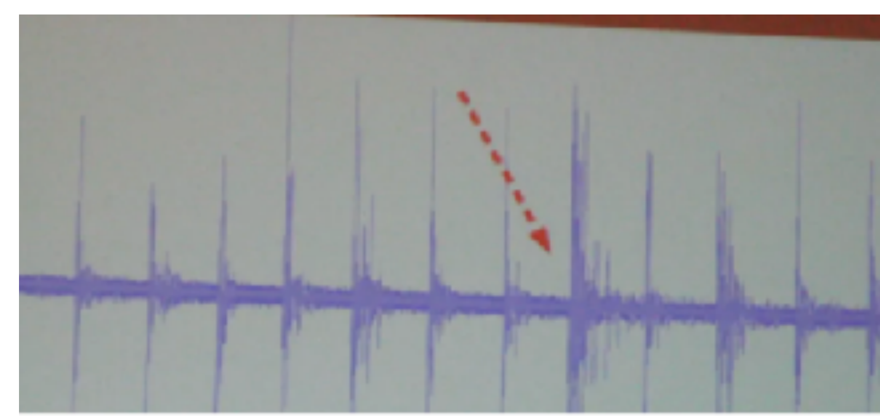
NEWS

SMU researchers find a new way to snoop with smartphones. But should you be worried?

SMU researchers used smartphones to figure out what someone's typing based on vibrations from the table, with a fourth of the words being "perfectly translated."

SMU researchers used a conference room to look into how well a couple smartphones can decipher what someone's typing on their computer nearby. While the phones are close to the laptop in this image, the researchers examined the feasibility with phones that were as far as 5-6 feet away. (Guy Rogers III / SMU)

Multiple Phones:
Audio + Acceleration



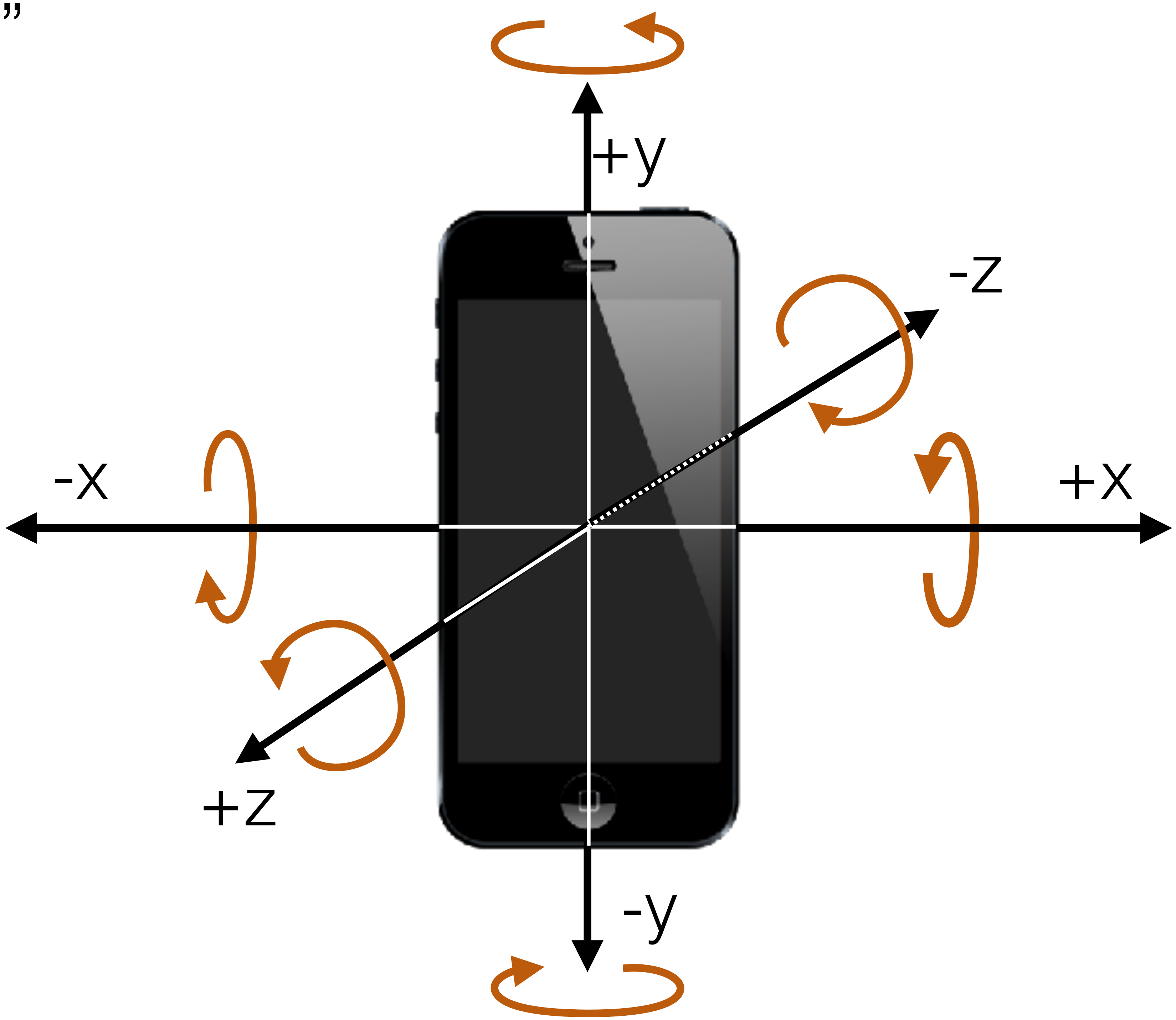
gyroscope

- measures the rate of rotation of the device
- MEMs device
  - essentially a microscopic, vibrating plate that resists motion

so it knows force in any rotating direction

gyroscope

- the "right hand rule"



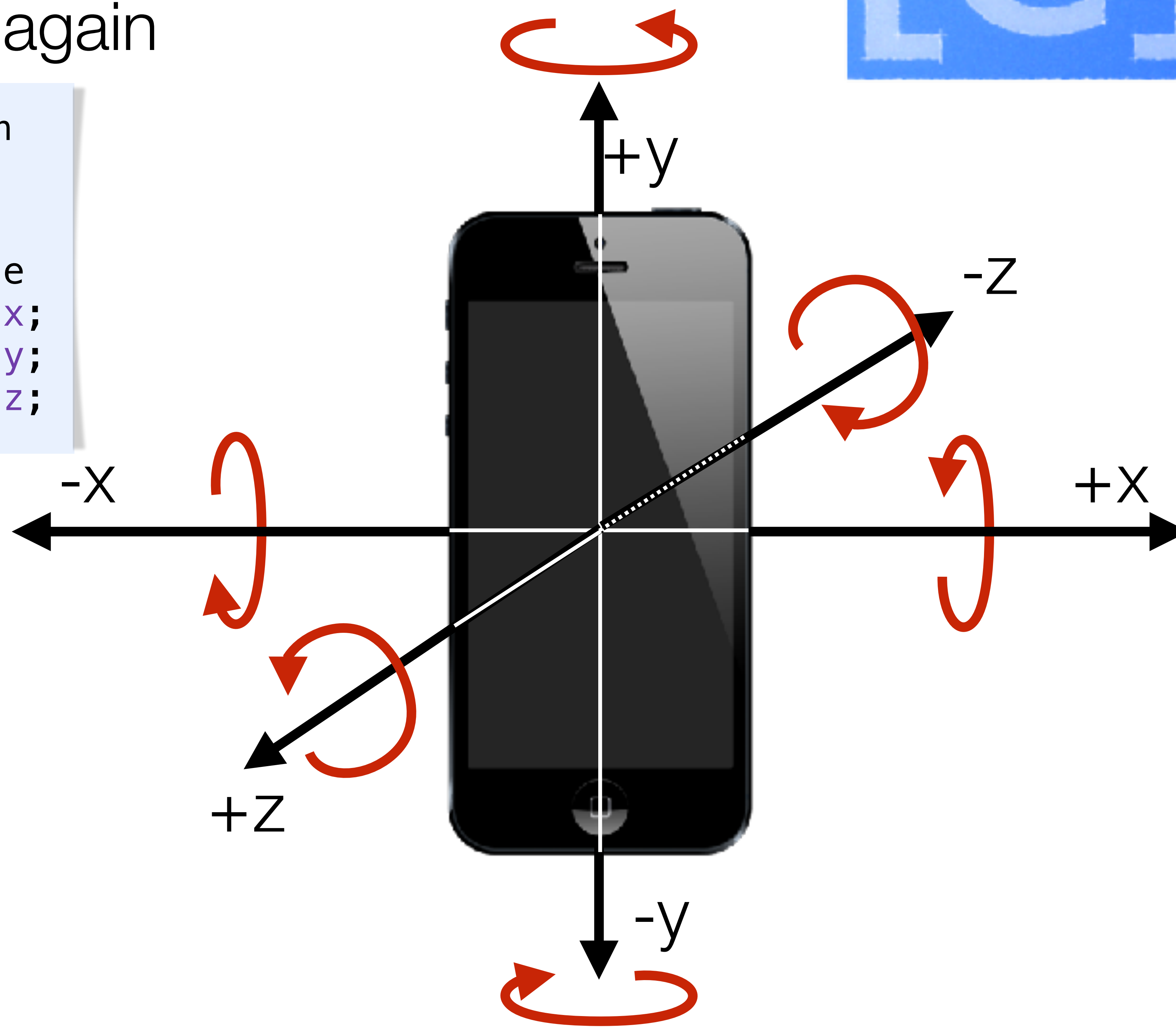
accessing the gyro

- use device motion again

```
CMDeviceMotion *deviceMotion

deviceMotion.rotationRate

CMRotationRate rotationRate
rotX[head] = rotationRate.x;
rotY[head] = rotationRate.y;
rotZ[head] = rotationRate.z;
```

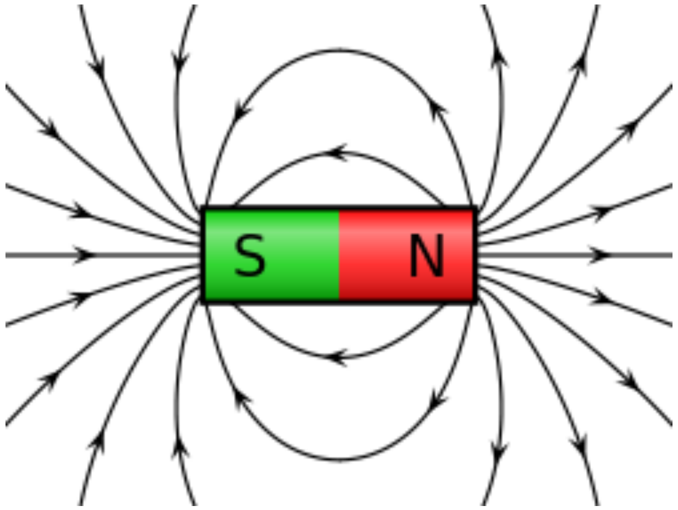
measures **rate** of motion

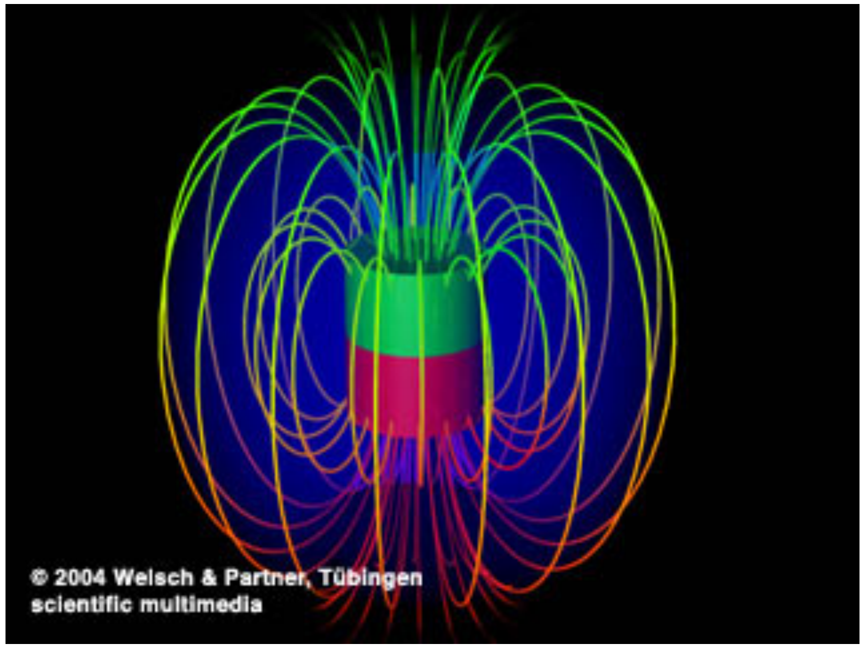
in this example, saves it to array



magnetometers

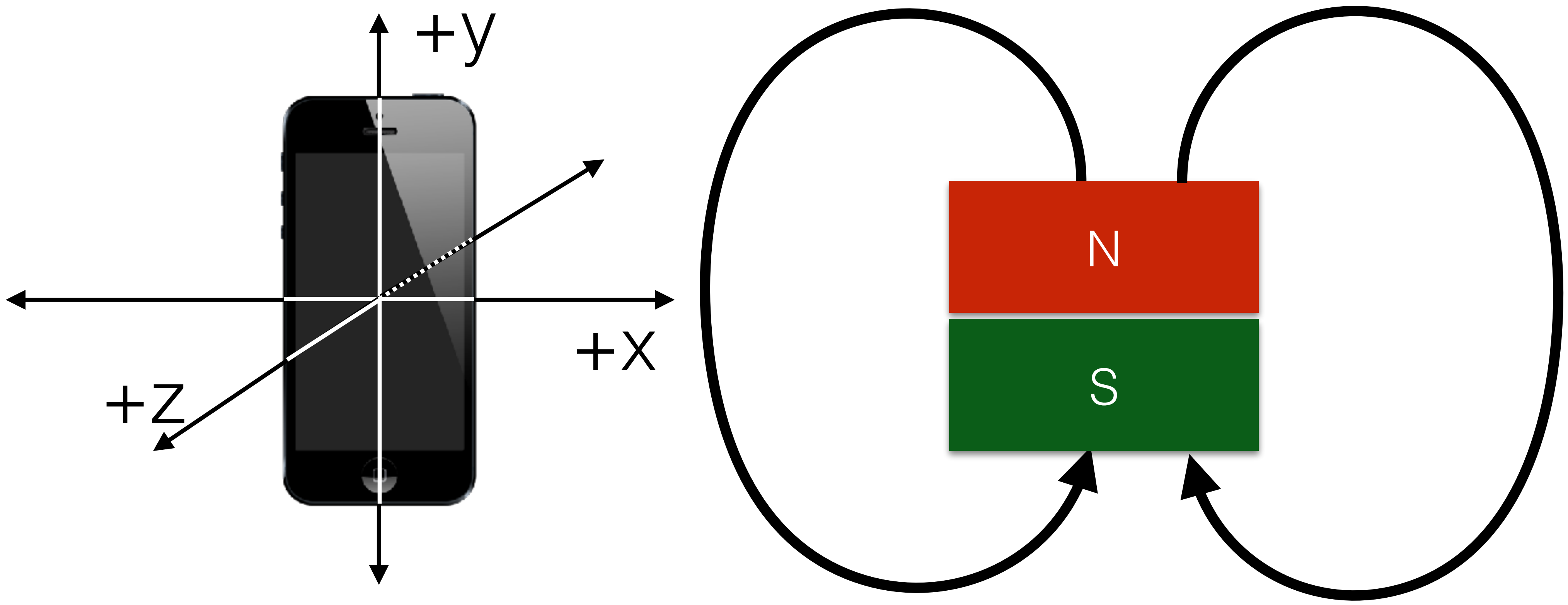
- measure magnetic fields
- magnets are measured in tesla (T)
  - **how**: essentially, there is a tight coupling between electricity flow and magnetic fields
- earth's magnetic field varies, but is around 50 uT
- iPhone can measure up to 1T with a resolution of about 8uT
- magnetic fields have direction!





magnetic fields

- measure magnetic field along axis, towards “south”

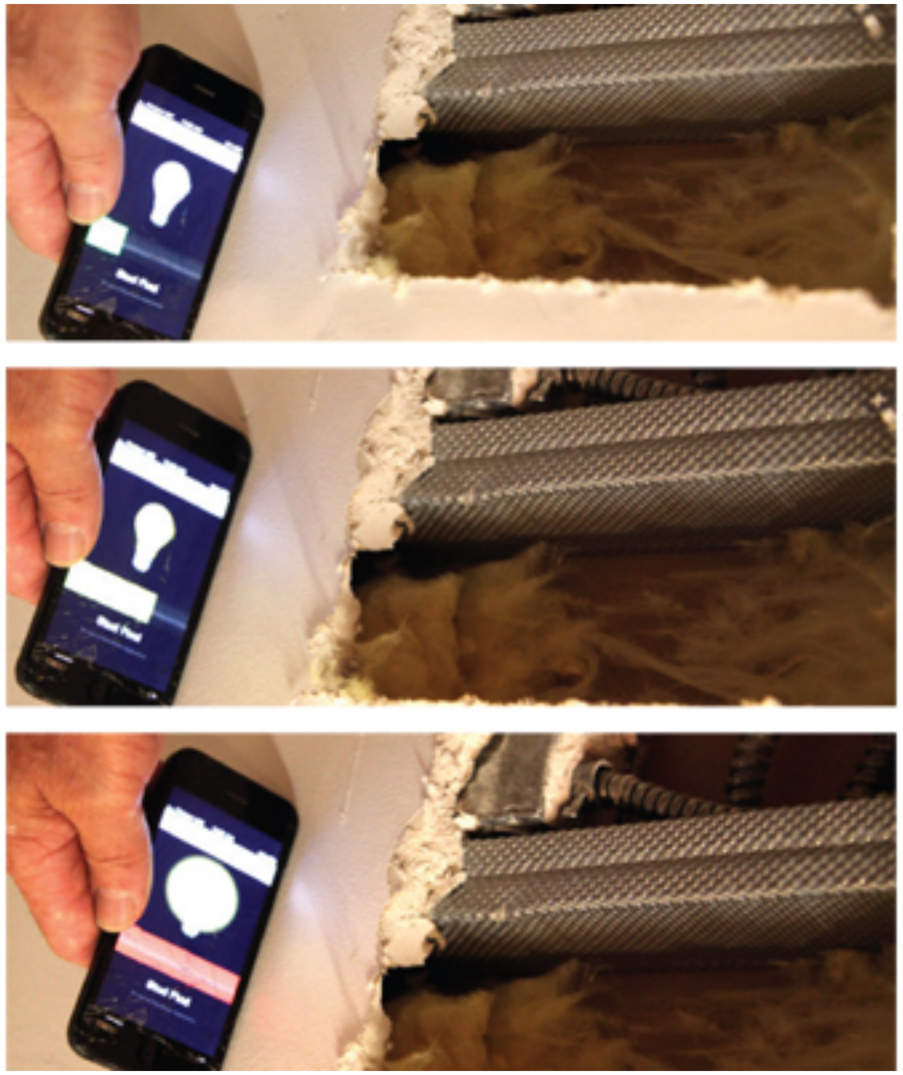


but iPhone has magnetic bias

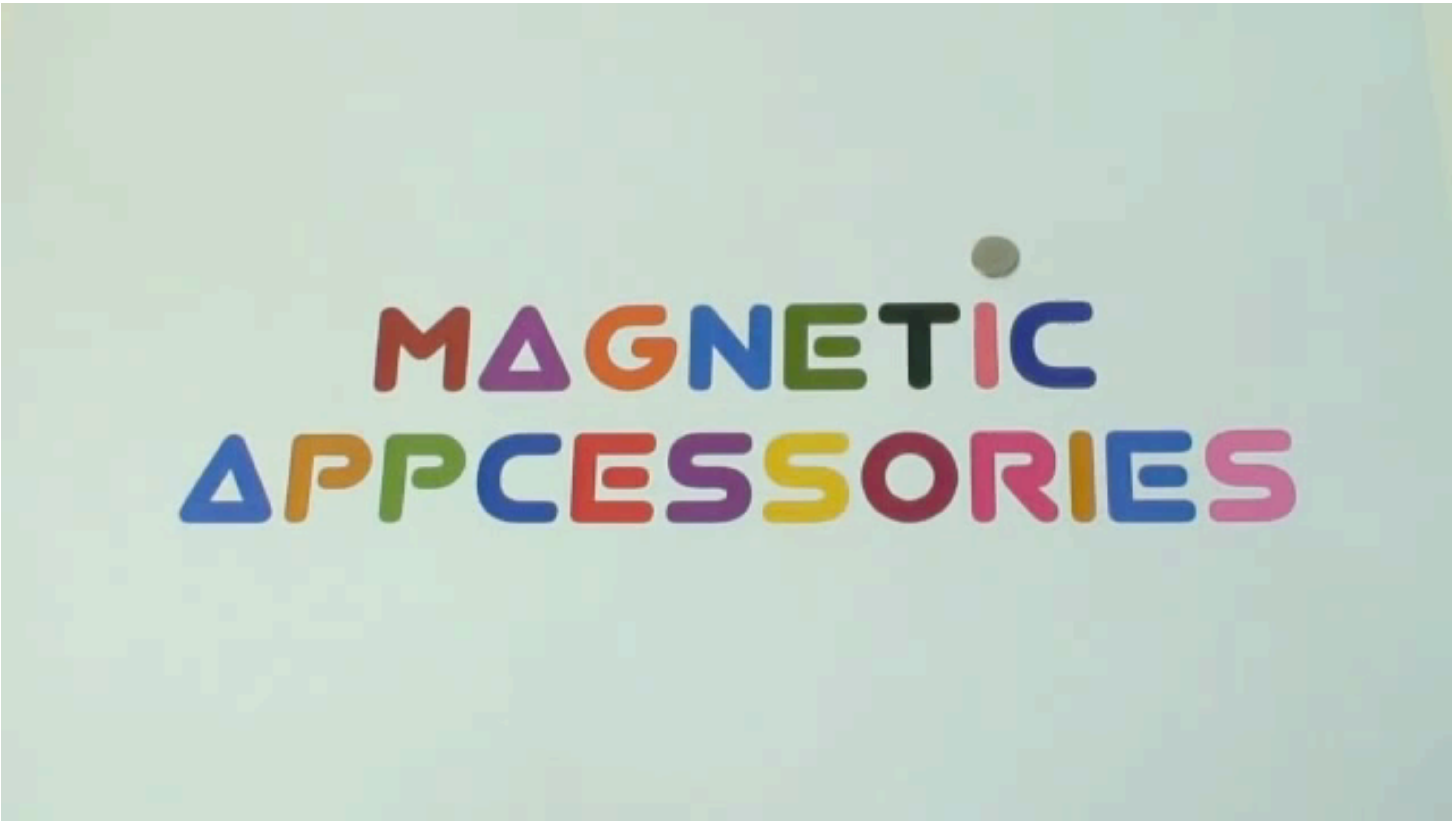
- the phone uses electricity and therefore is a magnet
  - good thing Apple subtracts that out for us!

```
CMDeviceMotion *deviceMotion

deviceMotion.magneticField
CMCalibratedMagneticField magneticField;

magneticField.field.x
magneticField.field.y
magneticField.field.z

magneticField.accuracy


CMMagneticFieldCalibrationAccuracyUncalibrated = -1,
   CMMagneticFieldCalibrationAccuracyLow,
   CMMagneticFieldCalibrationAccuracyMedium,
   CMMagneticFieldCalibrationAccuracyHigh
```

a cool example

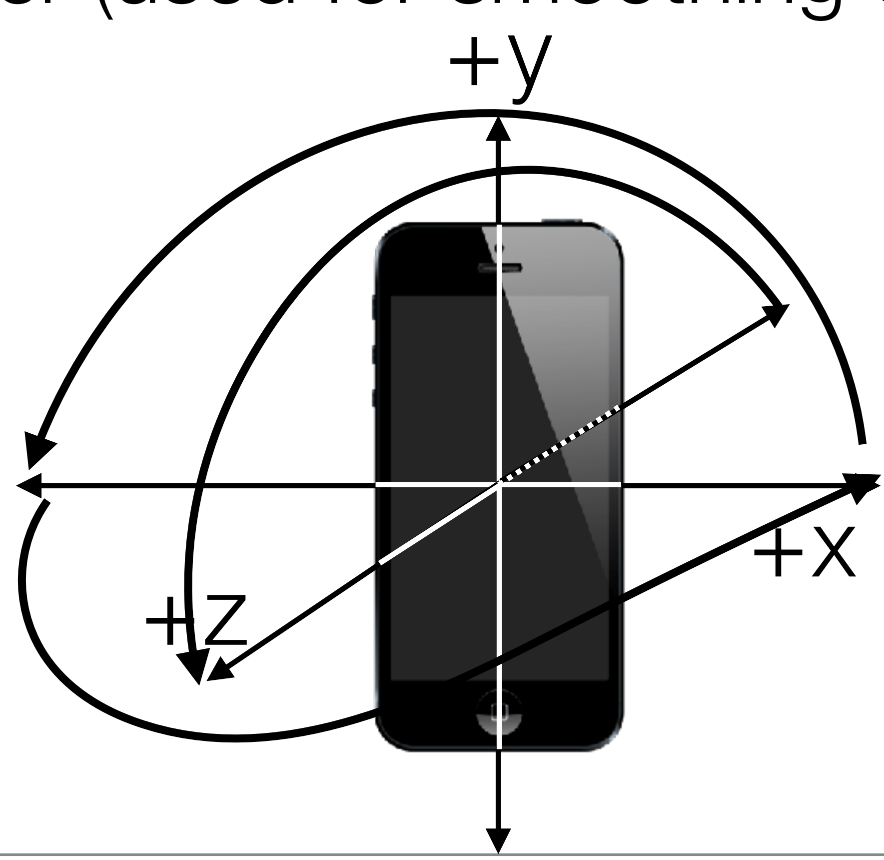


a cool example



attitude

- attitude is roll, pitch, and yaw (position)
- these are "fused" measures of the device from
  - the magnetometer (used as a compass)
  - gyroscope (used for detecting quick rotations)
  - accelerometer (used for smoothing out the gyro)



yaw in x/y plane
pitch in y/z plane
roll in x/z plane

getting updates

```
// for getting access to the fused motion data (best practice, filtered)
@property (nonatomic,strong) CMMotionManager *mManager;


self.mManager = [[CMMotionManager alloc] init];

  if([self.mManager isDeviceMotionAvailable])
  {

      [self.mManager setDeviceMotionUpdateInterval:yourSamplingIntervalInSeconds];
      [self.mManager startDeviceMotionUpdatesToQueue:[NSOperationQueue mainQueue]
withHandler:^(CMDeviceMotion *deviceMotion, NSError *error) {

          //Access to all the data…
           deviceMotion.attitude,
           deviceMotion.rotationRate,
           deviceMotion.gravity,
           deviceMotion.userAcceleration,
           deviceMotion.magneticField,


      }];
  }
```

declare

instantiate

if device is capable

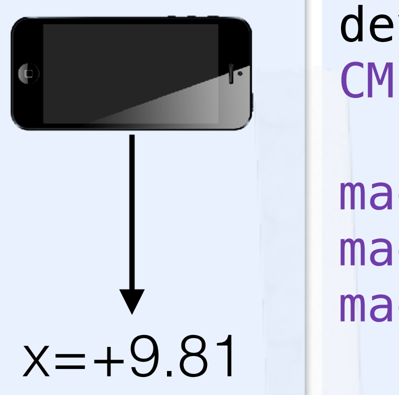
queue to run on

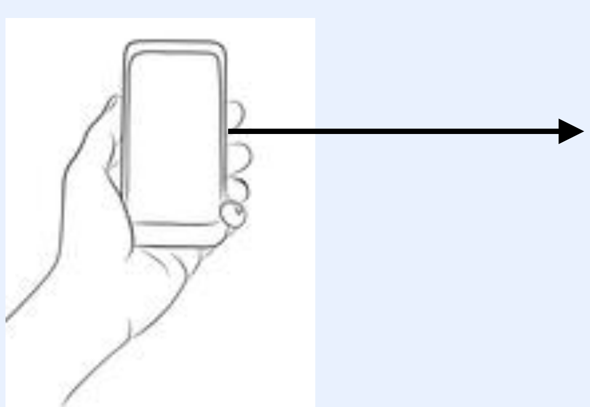
how often to push updates

the data

summary

```
CMDeviceMotion *deviceMotion

deviceMotion.gravity
deviceMotion.userAcceleration

CMAcceleration gravity,
CMAcceleration userAcceleration

gravity.x;
gravity.y;
gravity.z;

userAcceleration.x;
userAcceleration.y;
userAcceleration.z;
```

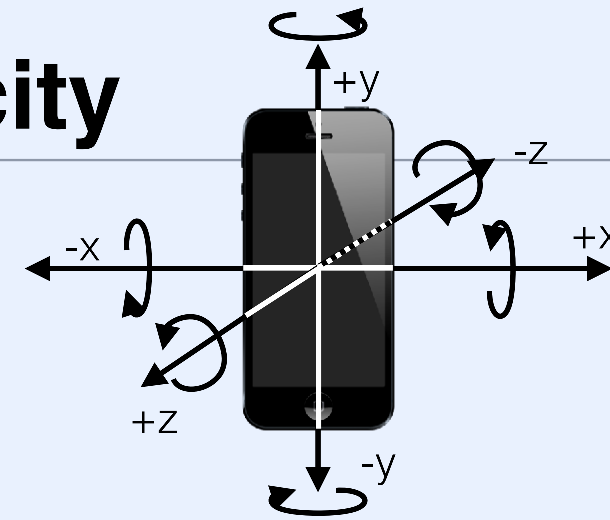




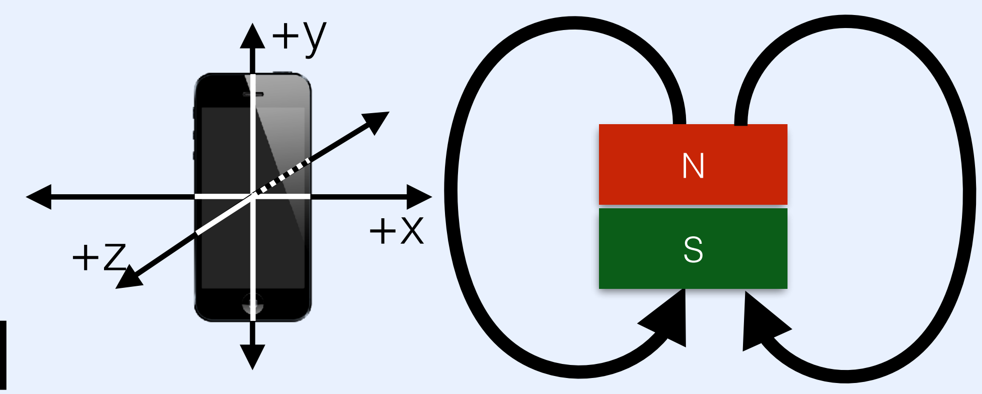
acceleration

rotation velocity

```
deviceMotion.rotationRate
CMRotationRate rotationRate
rotationRate.x;
rotationRate.y;
rotationRate.z;
```

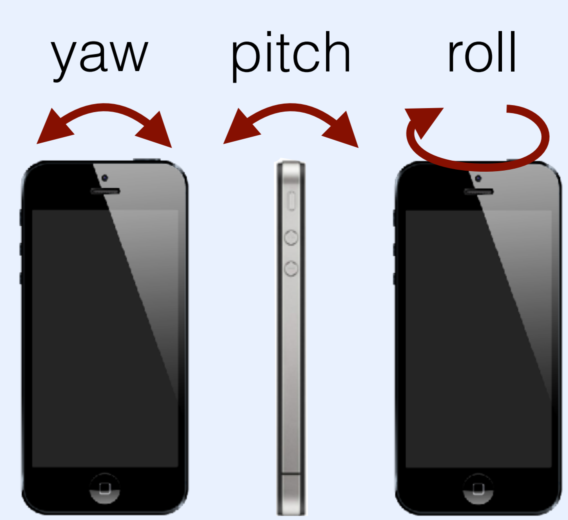



```
deviceMotion.magneticField
CMCalibratedMagneticField magneticField;

magneticField.field.x
magneticField.field.y
magneticField.field.z

magneticField.accuracy
```



magnetic field

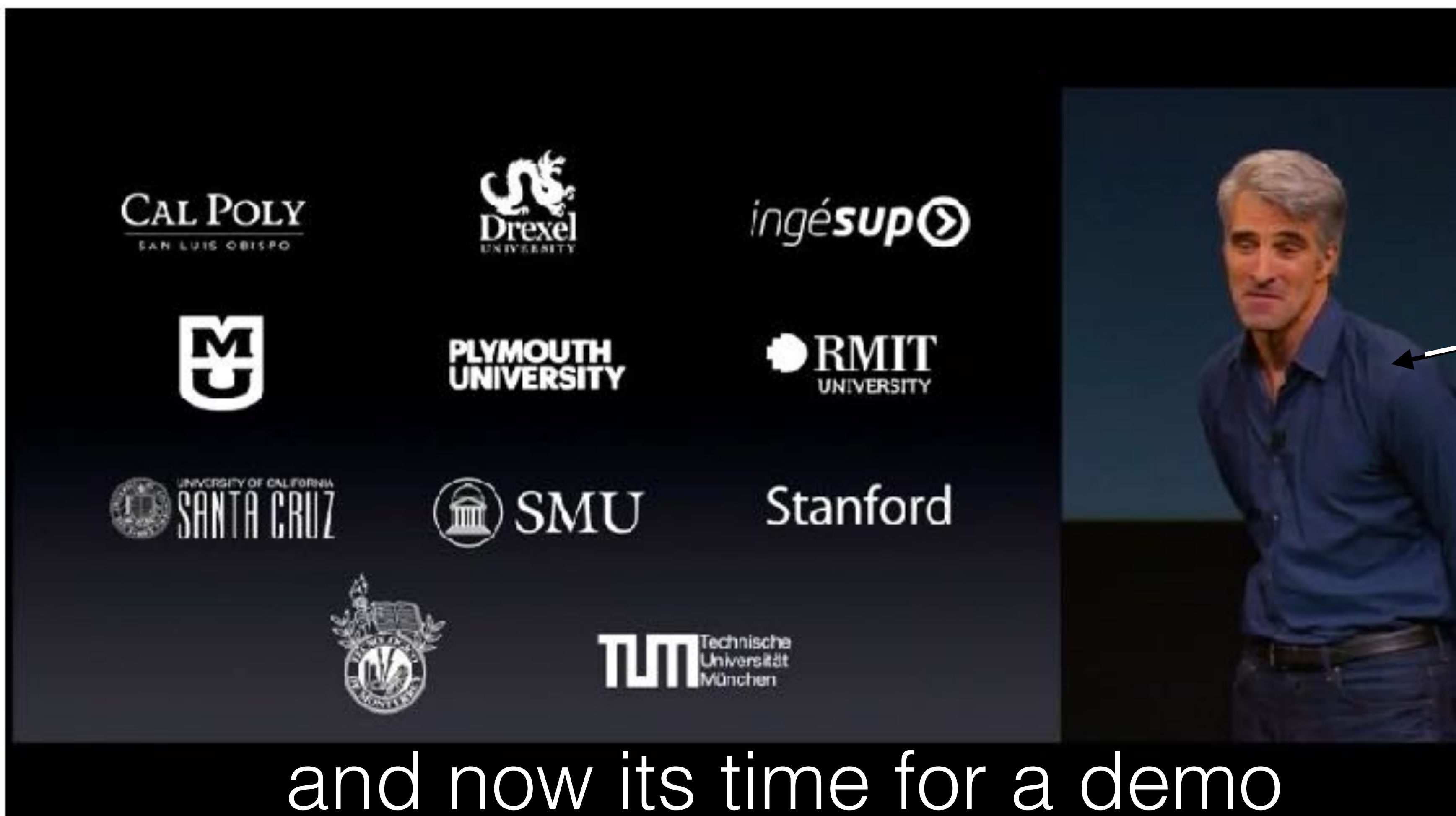
```
deviceMotion.attitude

CMAttitude* attitude

attitude.roll;
attitude.pitch;
attitude.yaw;
```

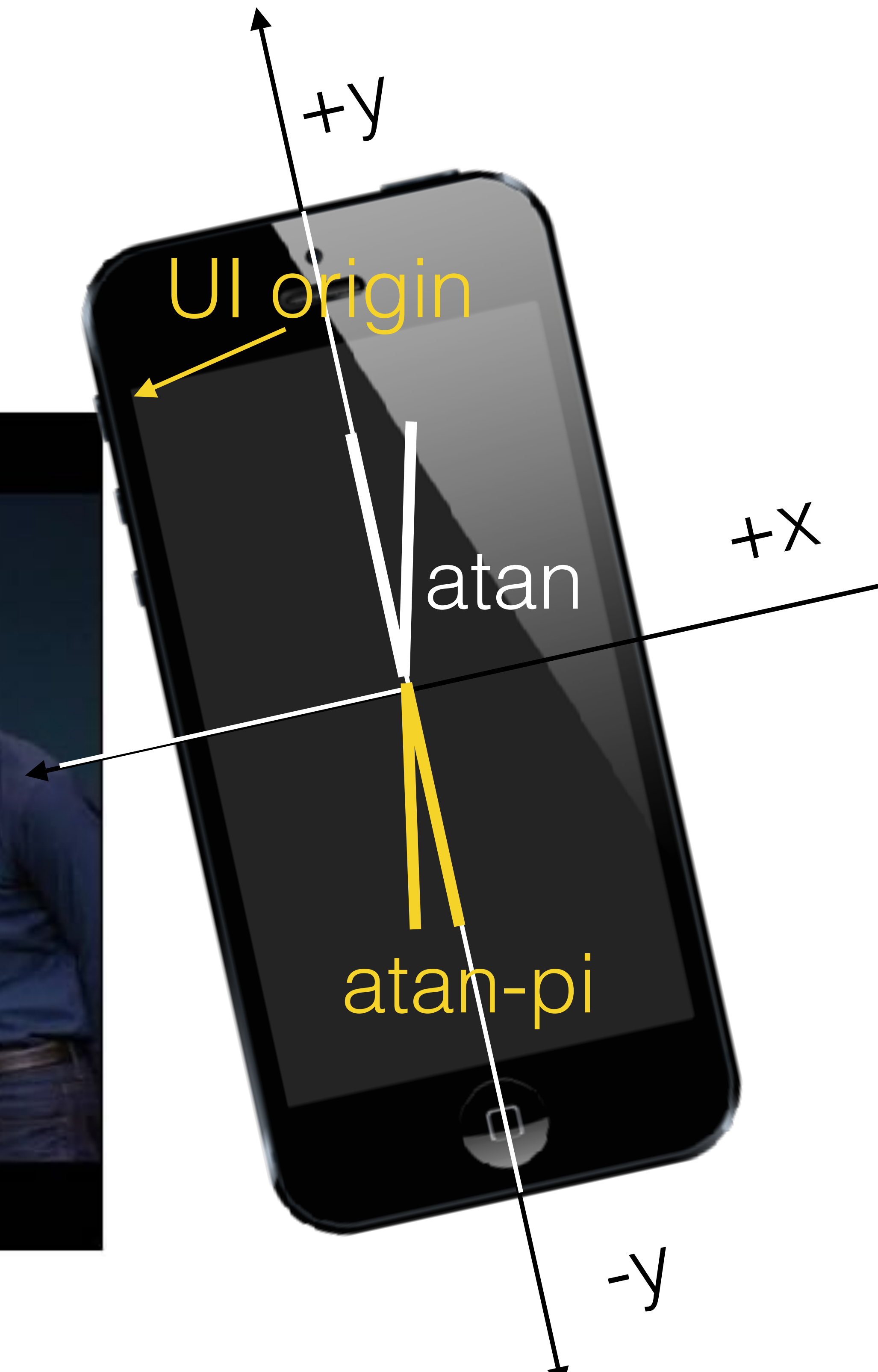


device position

device motion demo

- lets build something
  - to start: take that gravity!



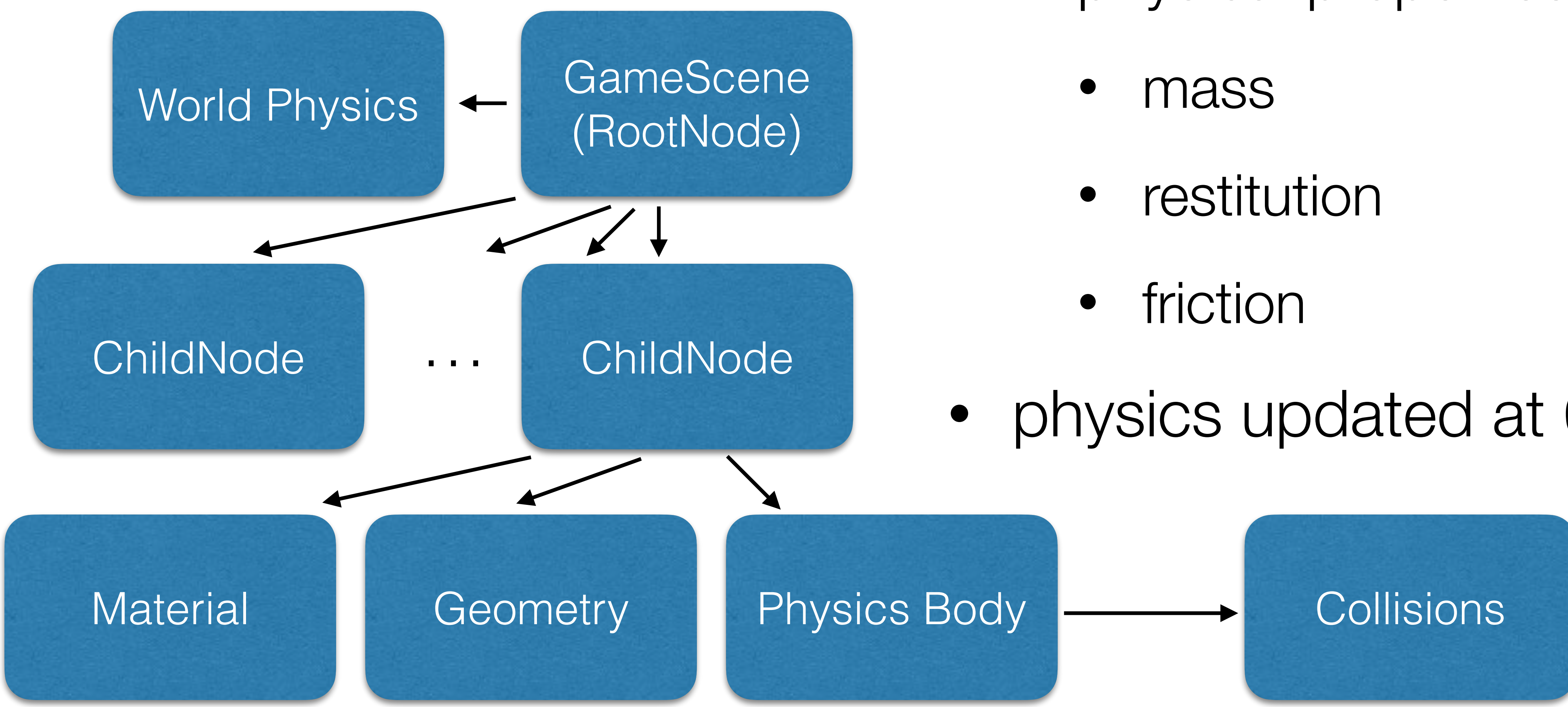


something more?

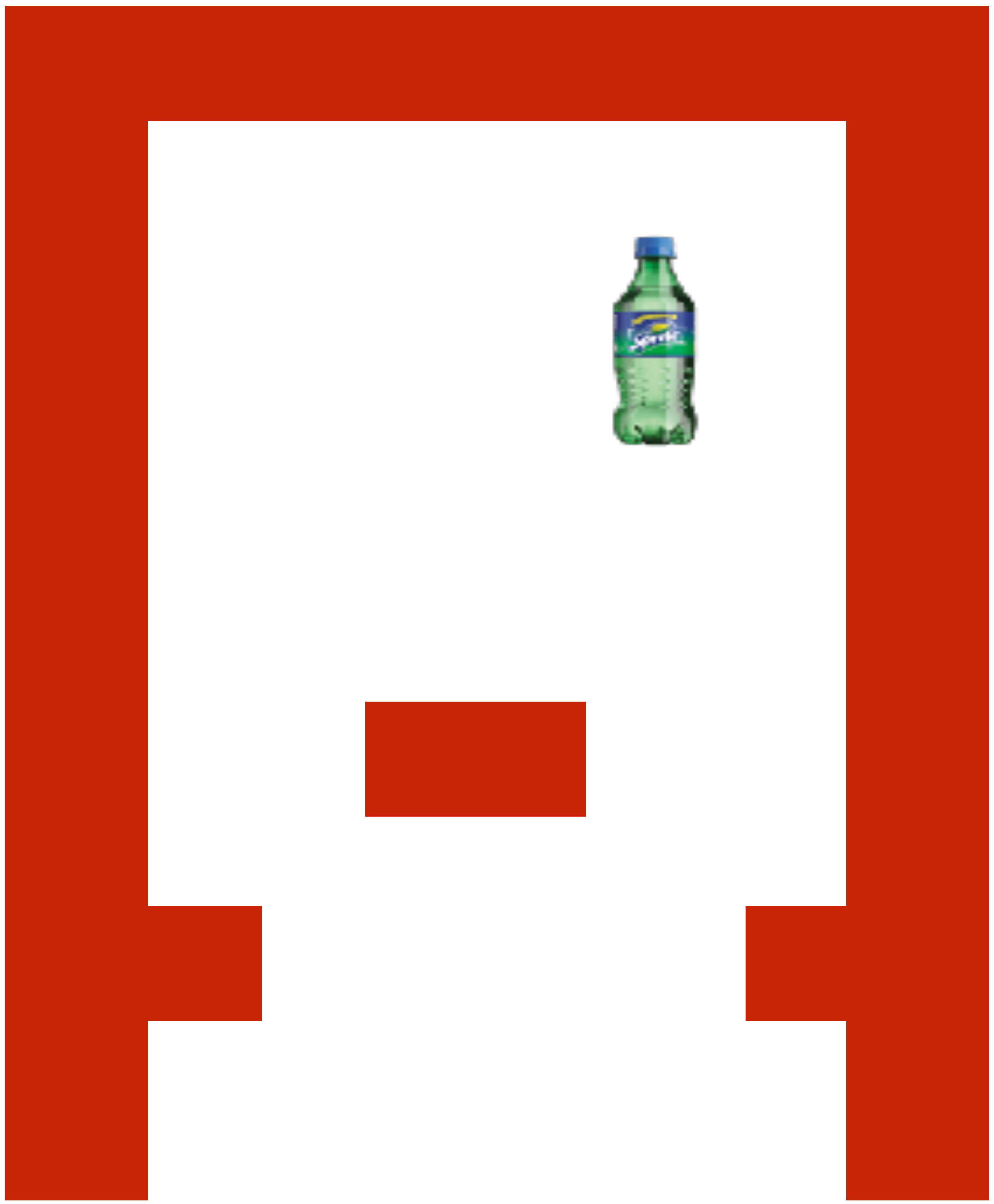
- 2D Physics Engine?
- Enter SpriteKit:
  - SK abbreviated
  - real time physics engine for game applications
  - …and 2D games in general
- how about a 3D physics engine?
  - Enter SceneKit

SpriteKit

- setup game scene
- create sprites
  - color/texture
  - physical properties
    - mass
    - restitution
    - friction
- physics updated at 60 Hz



SpriteKit



create "blocks"

create "sides/top"

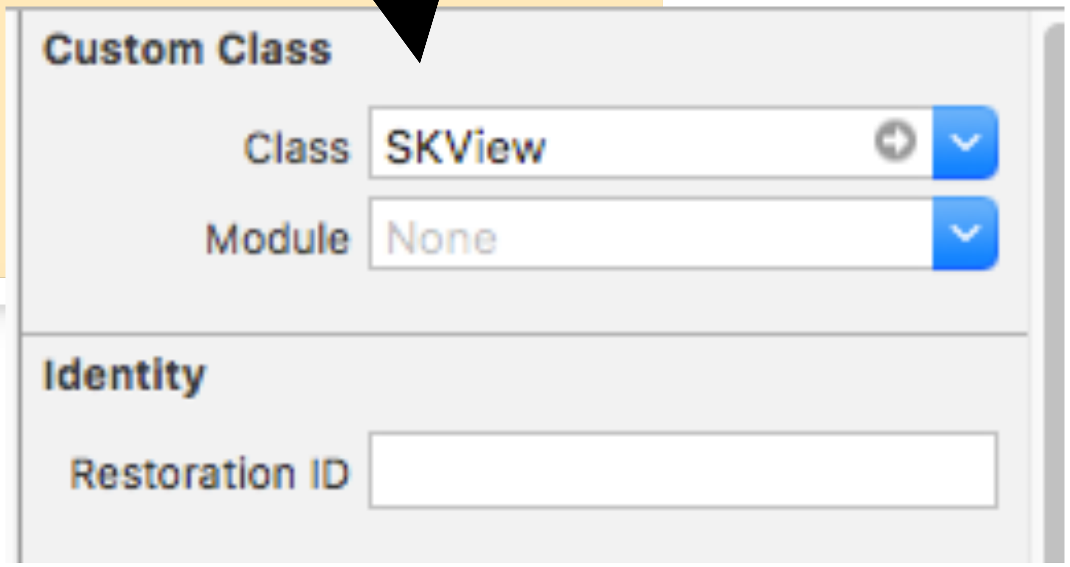
create "bouncy" sprite

make actual gravity
== game gravity

user must move phone
to keep sprite bouncing
on target

setup view controller

```swift
class GameViewController: UIViewController {


    override func viewDidLoad() {
        super.viewDidLoad()

        //setup game scene
        let scene = GameScene(size: view.bounds.size)
        let skView = view as! SKView // must be an SKView
        skView.showsFPS = true
        skView.showsNodeCount = true
        skView.ignoresSiblingOrder = true
        scene.scaleMode = .ResizeFill
        skView.presentScene(scene)
    }
}
```



set gravity

```
let motion = CMMotionManager()
func startMotionUpdates(){
    // some internal inconsistency here:
    // we need to ask the device manager for device

    if self.motion.deviceMotionAvailable{
        self.motion.deviceMotionUpdateInterval = 0.1
        self.motion.startDeviceMotionUpdatesToQueue(NSOperationQueue.mainQueue(),
                                            withHandler: self.handleMotion)
    }
}

func handleMotion(motionData:CMDeviceMotion?, error:NSError?){
    if let gravity = motionData?.gravity {
        self.physicsWorld.gravity = CGVectorMake(CGFloat(9.8*gravity.x),
                                                 CGFloat(9.8*gravity.y))
    }
}
```

start motion

adjust physics

build sprites example

add image texture

```
func addSpriteBottle(){
        let spriteA = SKSpriteNode(imageNamed: "sprite")

        spriteA.size = CGSize(width:size.width*0.1,height:size.height * 0.1)

        let randNumber = random(min: CGFloat(0.1), max: CGFloat(0.9))
        spriteA.position = CGPoint(x: size.width * randNumber, y: size.height * 0.75)

        spriteA.physicsBody = SKPhysicsBody(rectangleOf:spriteA.size)
        spriteA.physicsBody?.restitution = random(min: CGFloat(1.0), max: CGFloat(1.5))
        spriteA.physicsBody?.isDynamic = true
        spriteA.physicsBody?.contactTestBitMask = 0x00000001
        spriteA.physicsBody?.collisionBitMask = 0x00000001
        spriteA.physicsBody?.categoryBitMask = 0x00000001

        self.addChild(spriteA)
    }
```

interaction physics

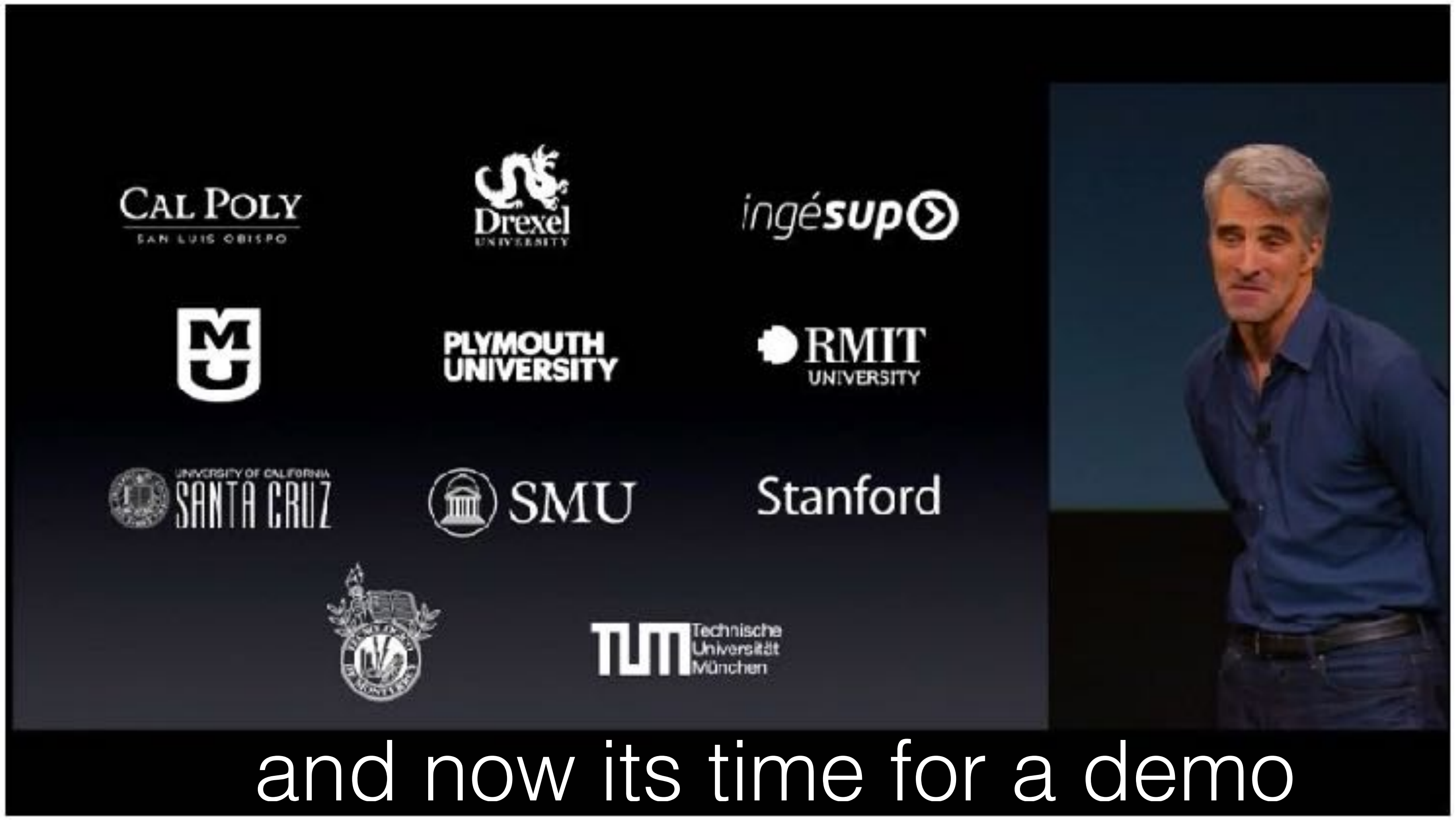
add to scene

Physics Body Types

Static bodies are unaffected by forces and collisions and cannot move.
Dynamic bodies are affected by forces and collisions with other body types.
Kinematic bodies are not affected by forces/collisions, by moving them directly you can cause collisions on dynamic bodies.

device motion demo 2

- lemon lime bounce
- pre-made demo
- Let's add something to the game

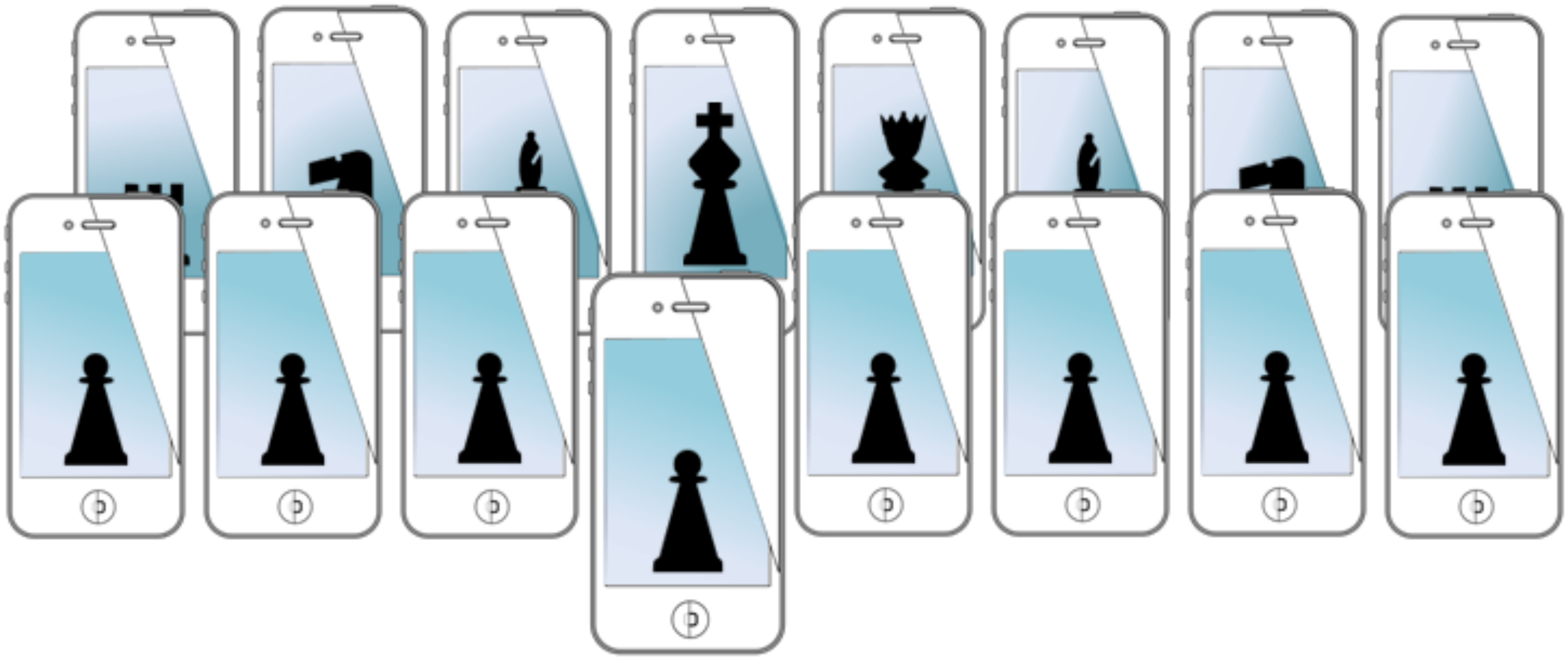


for next time…

- SceneKit

SpriteKit SceneKit

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

activity, pedometers, and motion sensing

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University