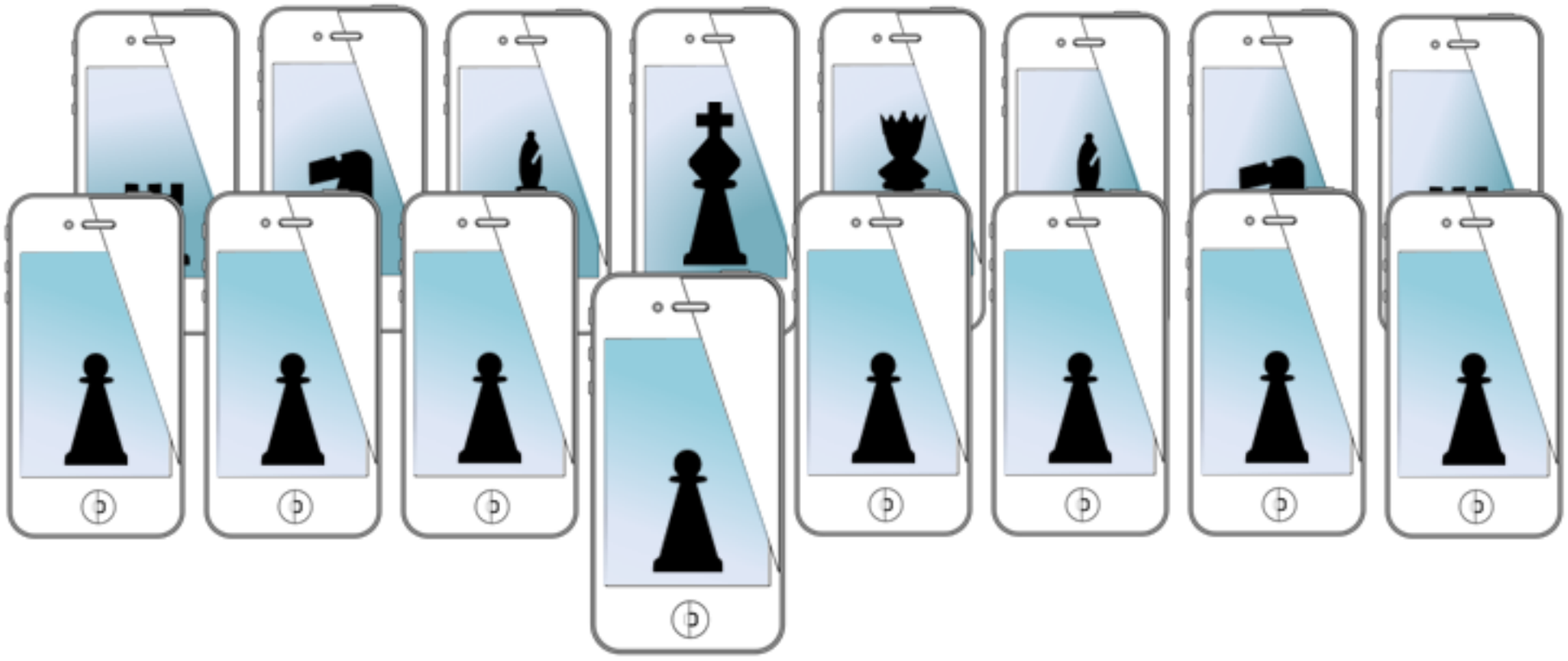
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

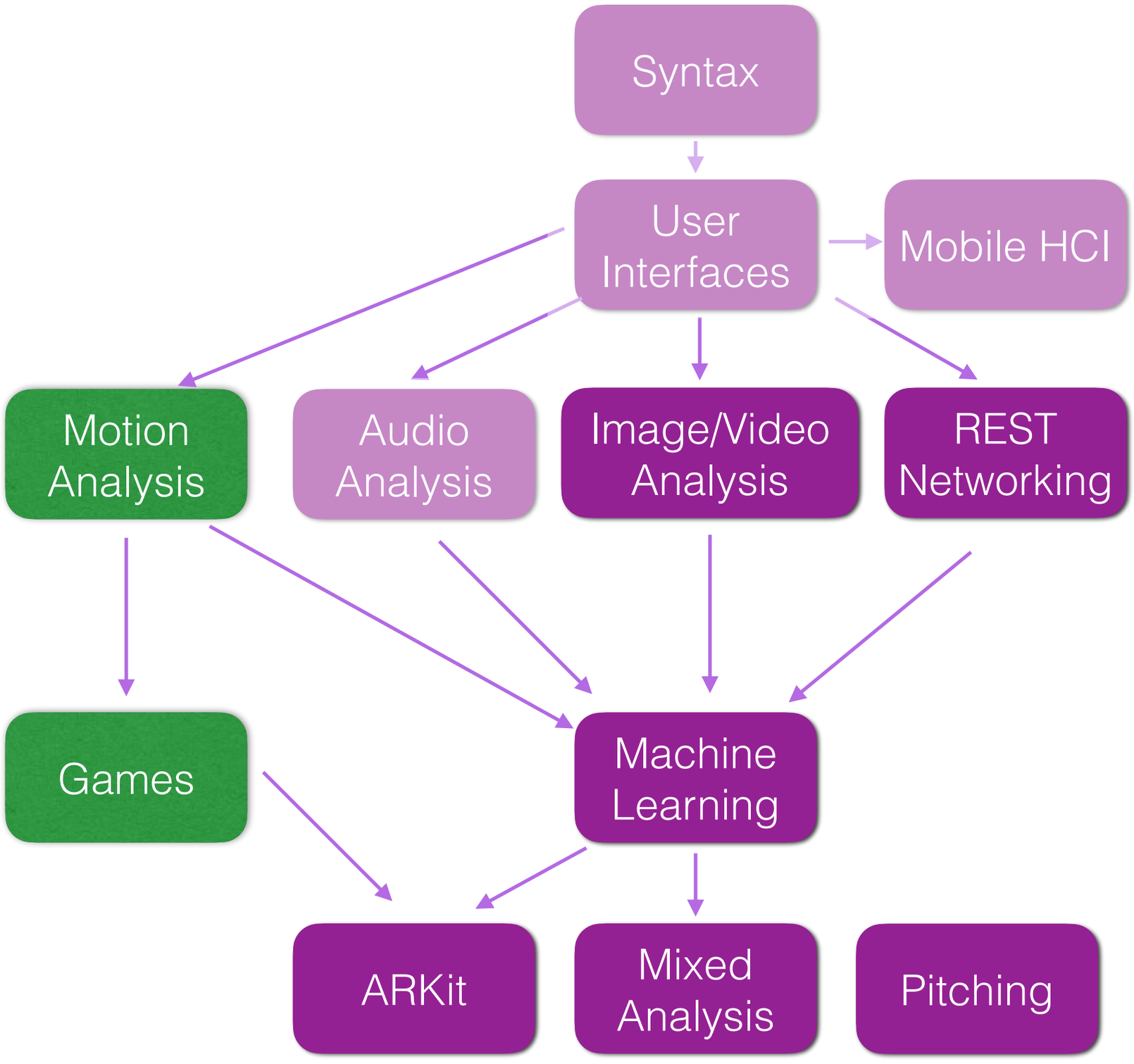
activity, pedometers, and motion sensing

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



storing persistent defaults

- iOS supports UserDefaults for primitives and encapsulated data (or lists of)

```
// standardUserDefaults variable
let defaults = UserDefaults.standardUserDefaults()

// saving
defaults.setInteger(252, forKey:@“primitiveInteger")
defaults.setDouble(3.14, forKey:@“primitiveDouble")
defaults.setFloat
defaults.setBool
defaults.setURL

// saving an object
defaults.setObject("Coding Explorer", forKey: “userNameKey")

if let name = defaults.stringForKey("userNameKey") {
    print(name)
}
```

import defaults

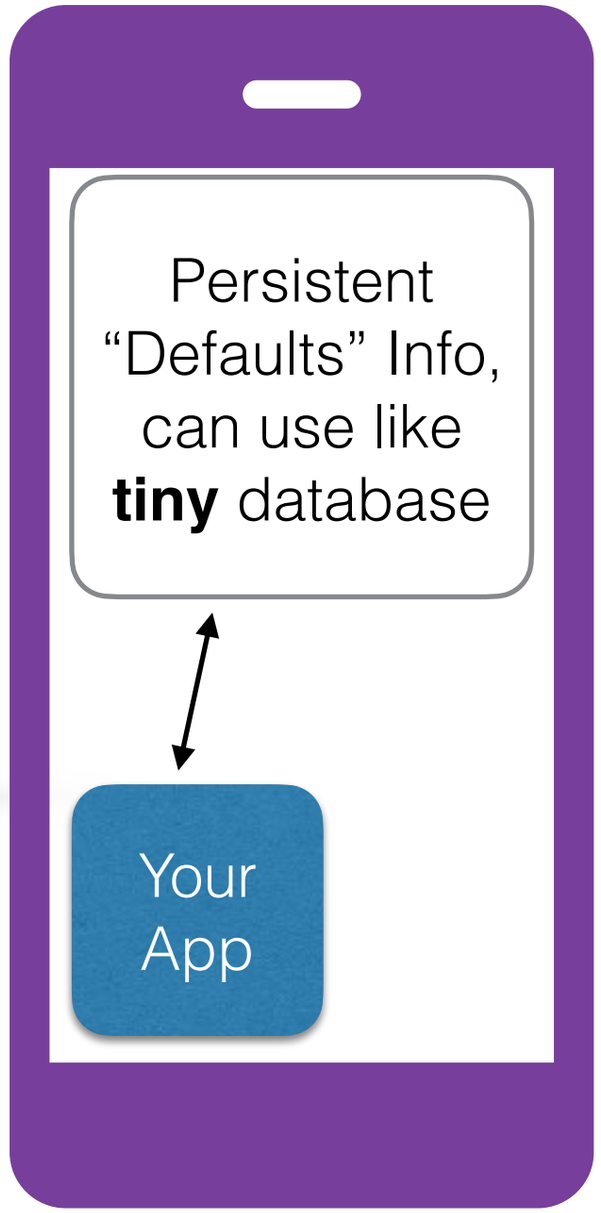
primitives

objects

access saved objects

```
boolForKey          ->  Bool
integerForKey       ->  Int
dataForKey          ->  Data?
objectForKey        ->  AnyObject?
arrayForKey         ->  [AnyObject]?
stringArrayForKey   ->  [String]?
dictionaryForKey    ->  {String:AnyObject}?
```

supported datatypes



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

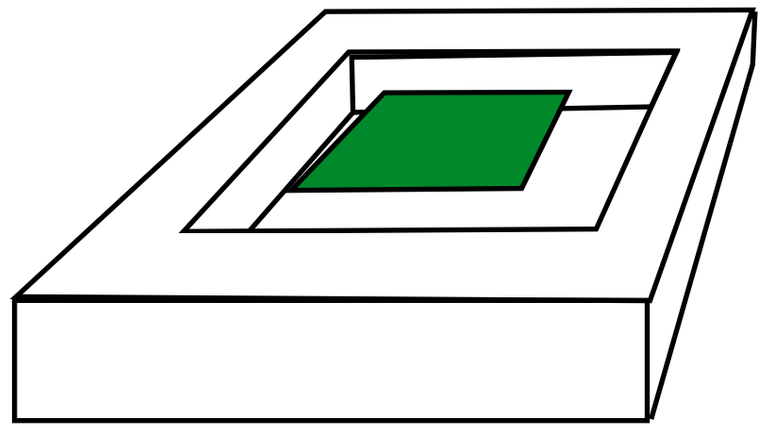
- A-series mediates access to data
- much lower battery consumption

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	~?%	?%

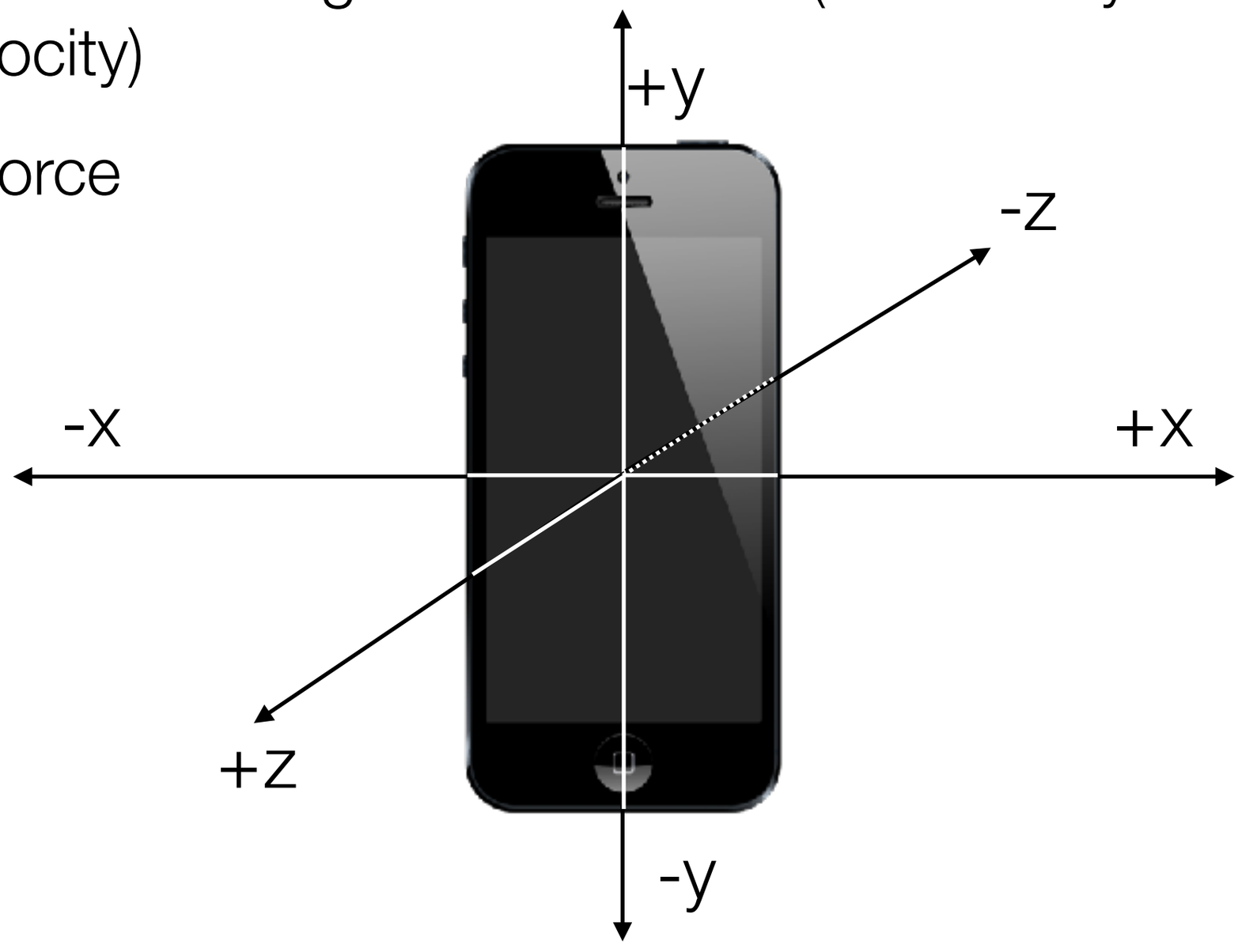
accelerometers

- 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



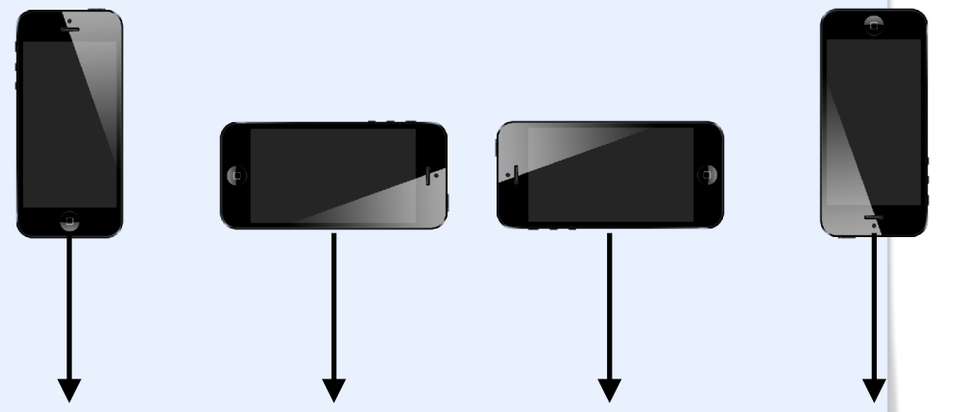
accessing the accelerometer

- 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

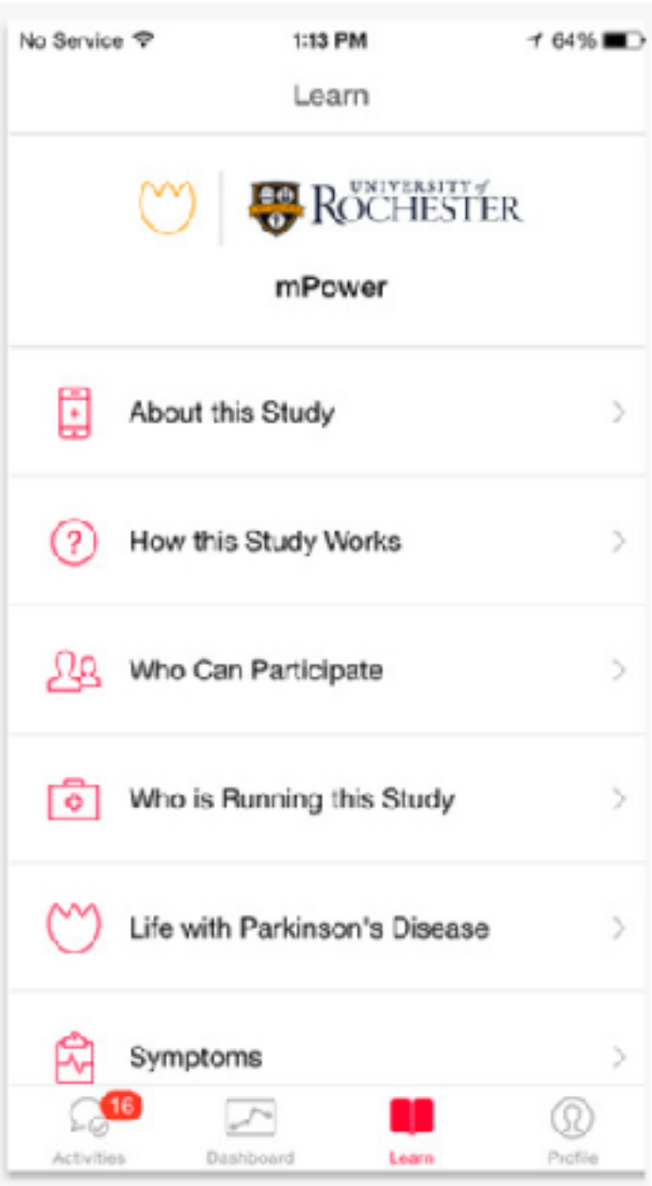
```
CMDeviceMotion *deviceMotion

deviceMotion.gravity
deviceMotion.userAcceleration

CMAcceleration gravity, CMAcceleration userAcceleration

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

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

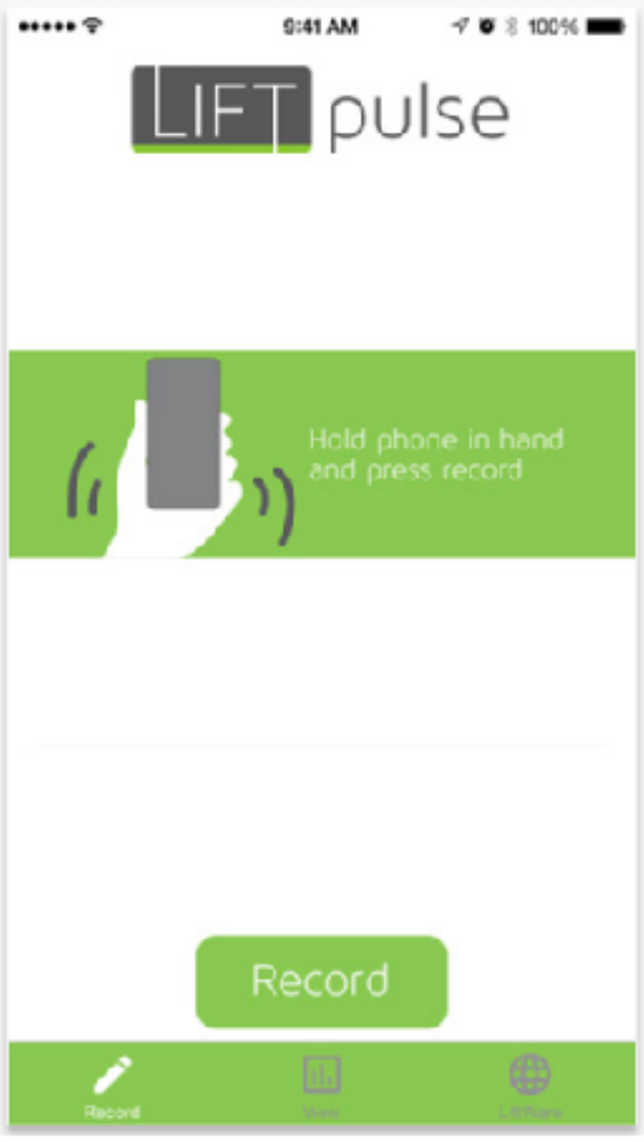
access through a different field!

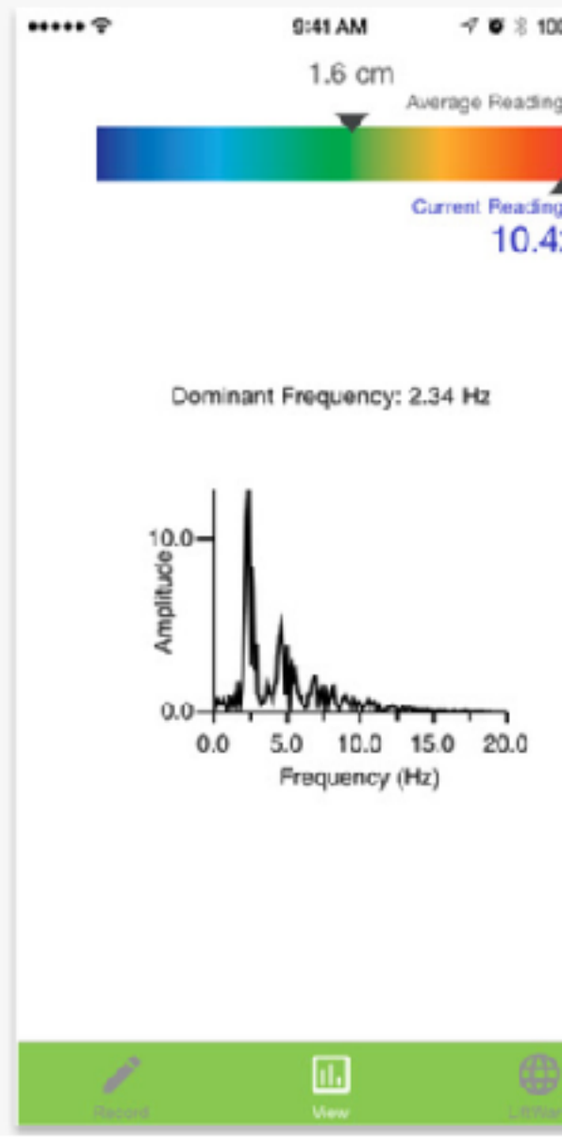
user movement



a cool example







another cool example

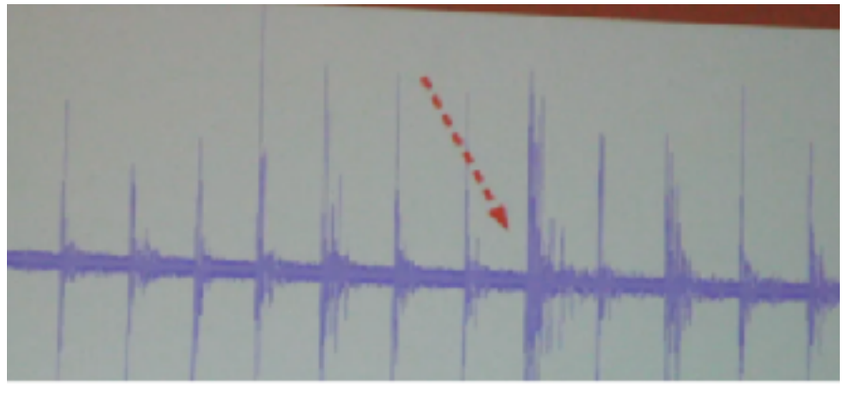
NEWS

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 II / SMU)

Multiple Phones:
Audio + Acceleration



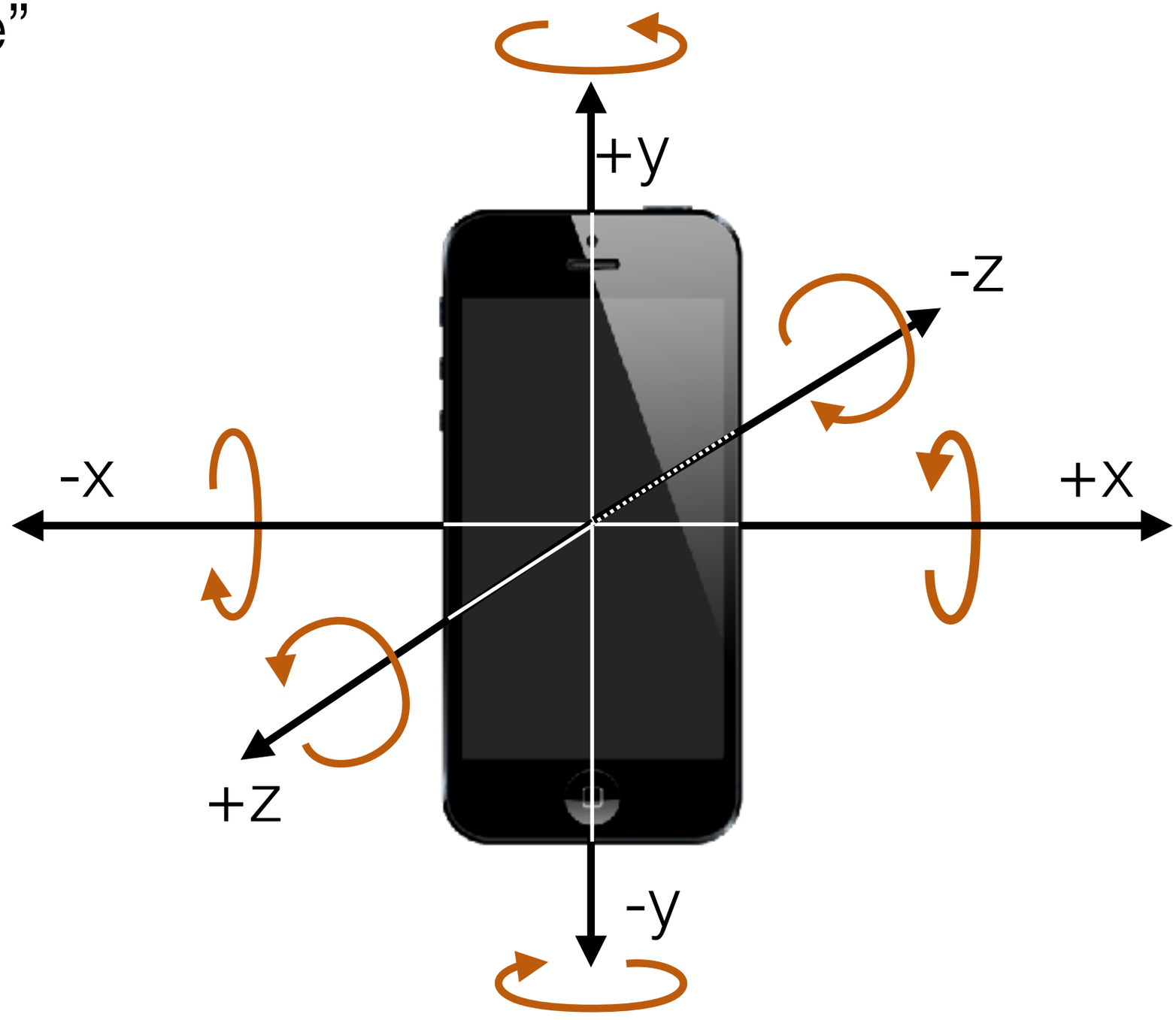
gyroscope

- 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”



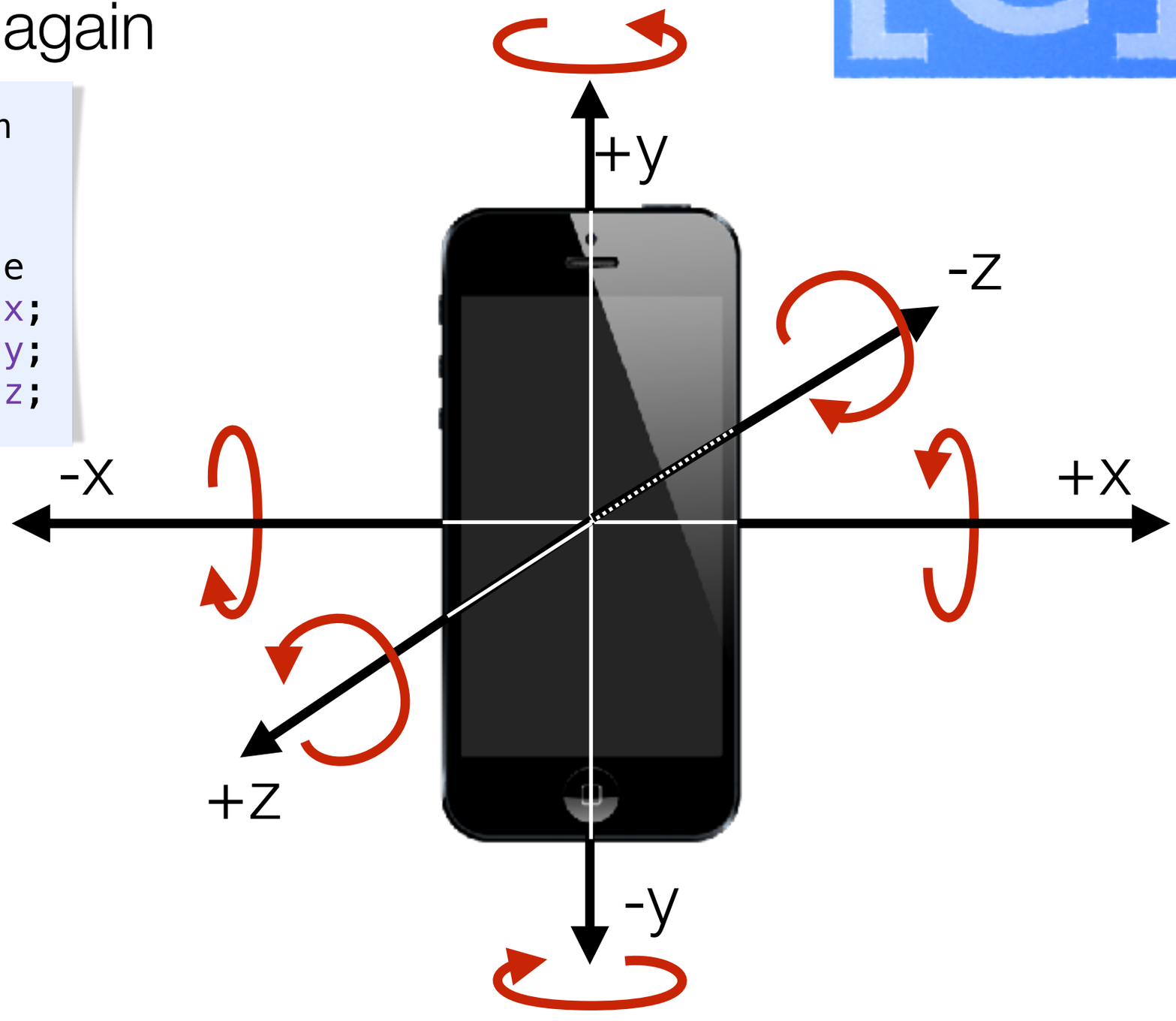
accessing the gyro

- use device motion again

```
CMDeviceMotion *deviceMotion

deviceMotion.rotationRate

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

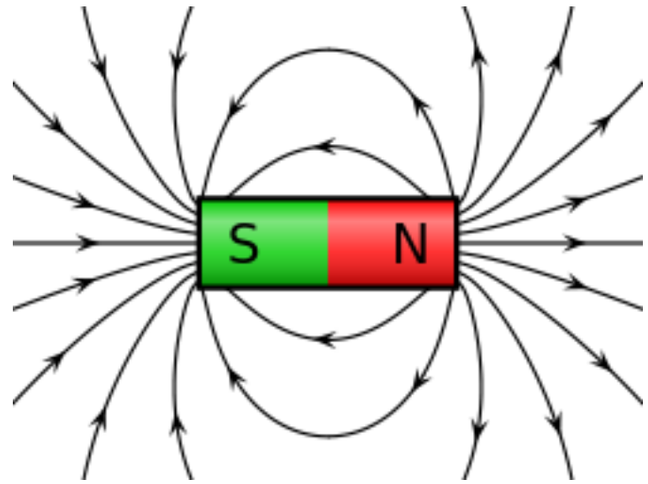
measures **rate** of motion

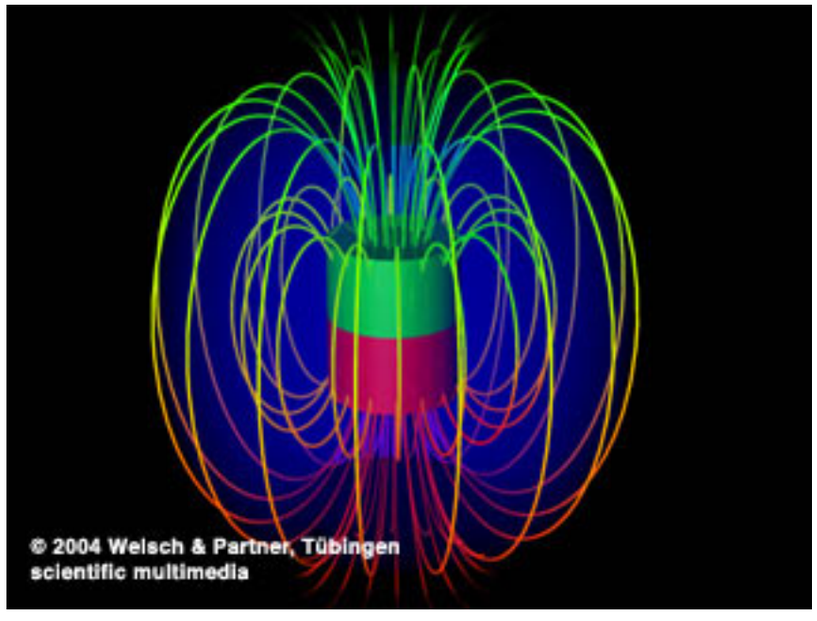
in this example, saves it to array



magnetometers

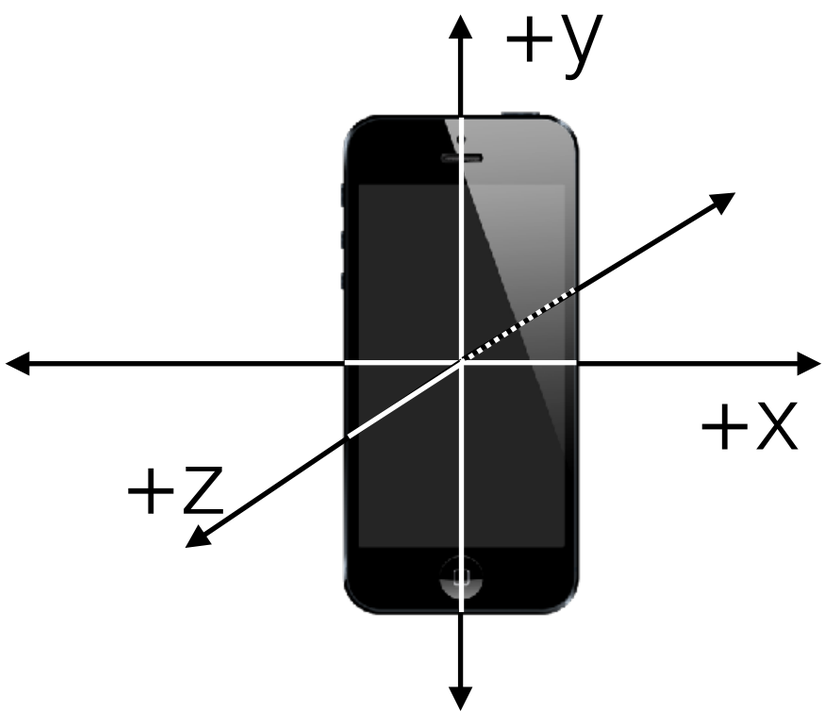
- measure magnetic fields
- magnets are measured in tesla (T)
  - **how**: essentially, there is a tight coupling between electricity flow and magnetic fields
- earth's magnetic field varies, but is around 50 uT
- iPhone can measure up to 1T with a resolution of about 8uT
- magnetic fields have direction!

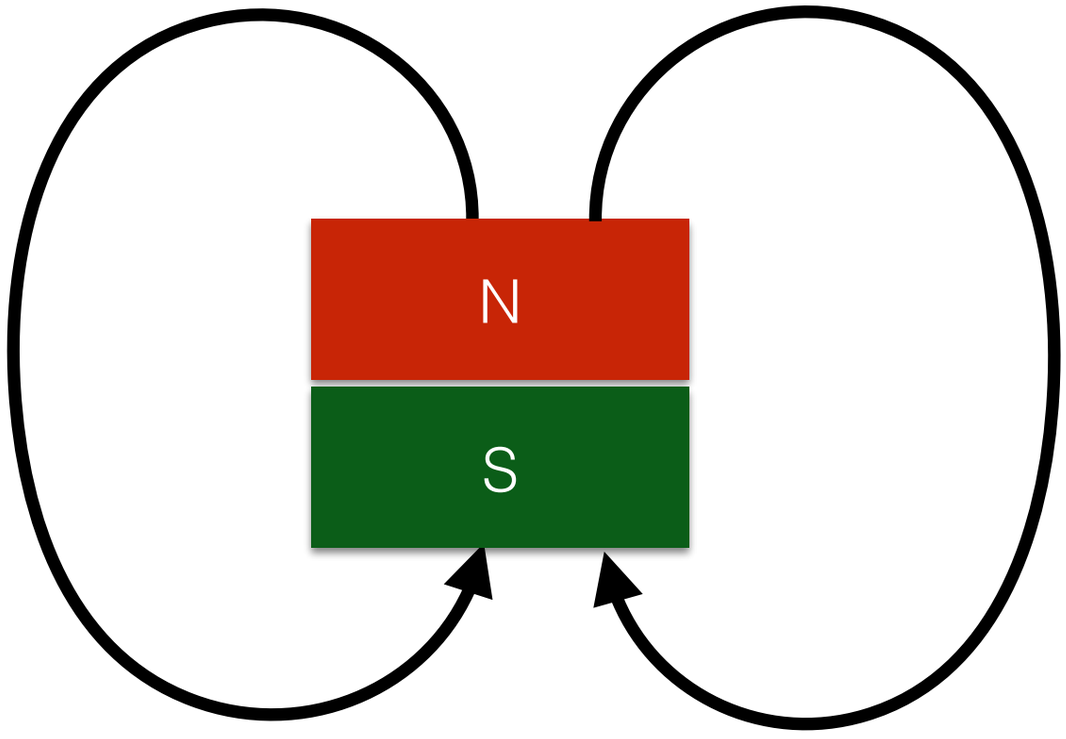




magnetic fields

- measure magnetic field along axis, towards “south”



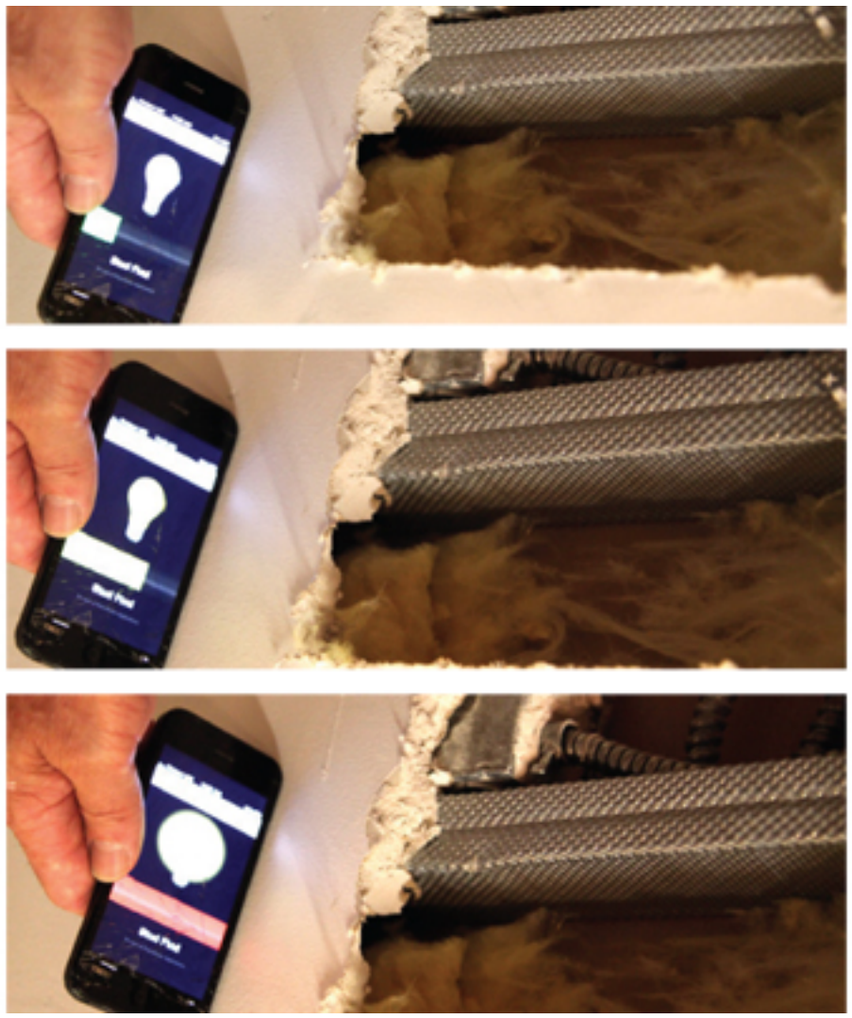


but iPhone has magnetic bias

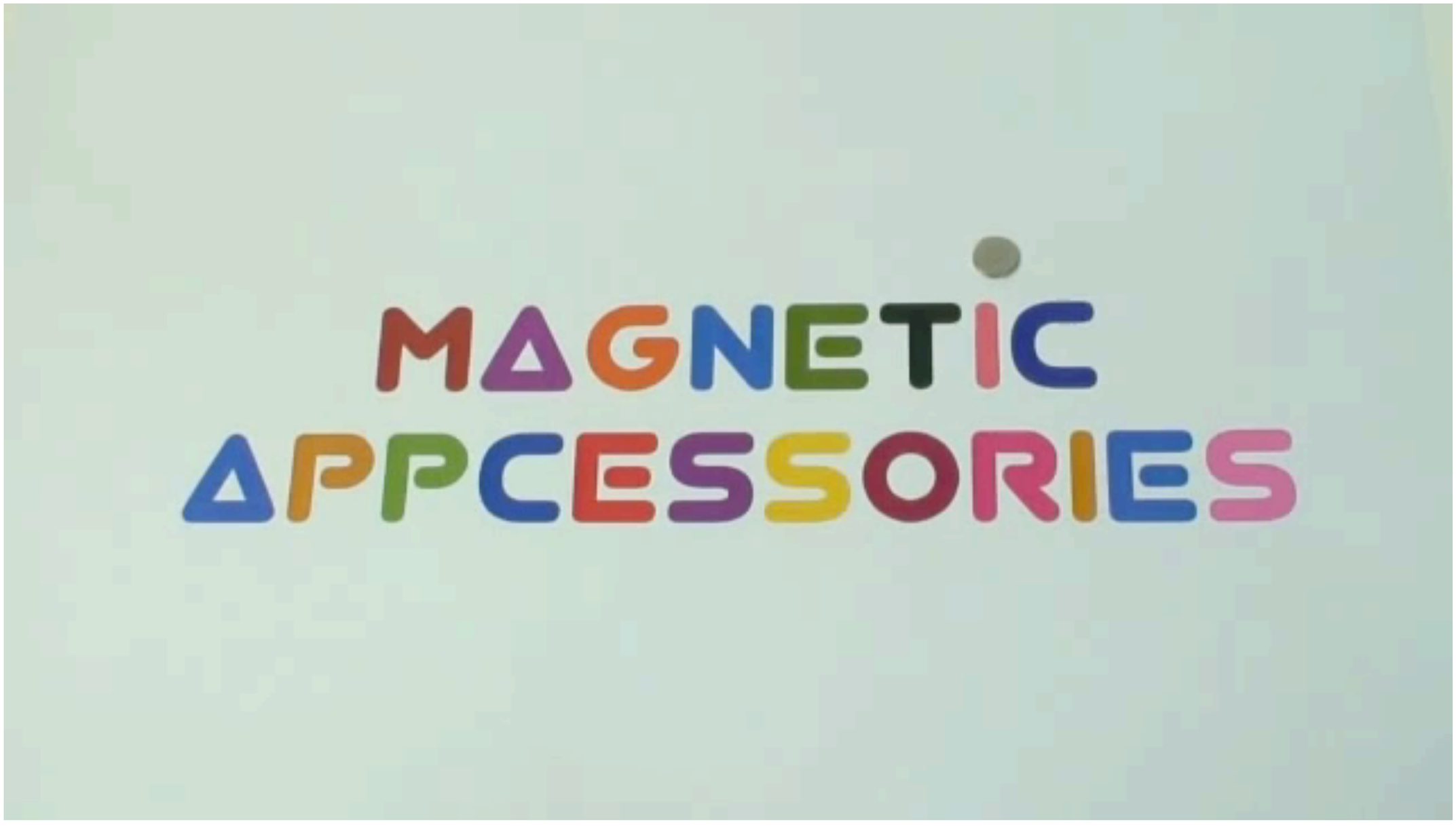
- the phone uses electricity and therefore is a magnet
  - good thing Apple subtracts that out for us!

```
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

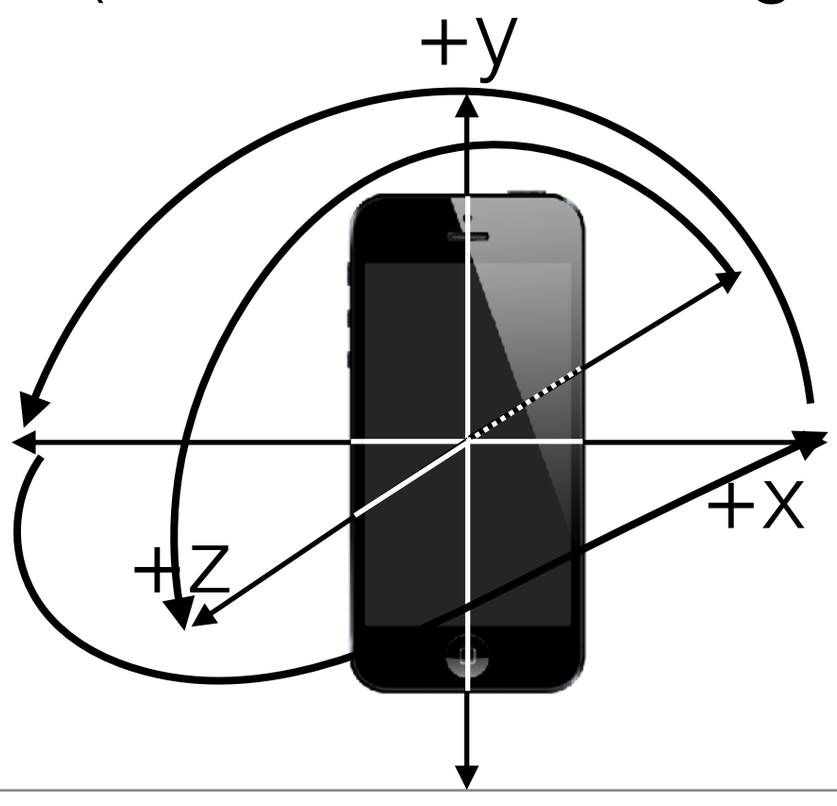


a cool example



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

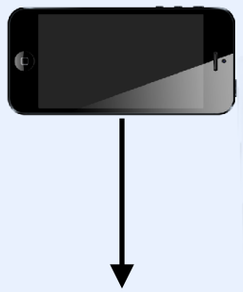
how often to push updates

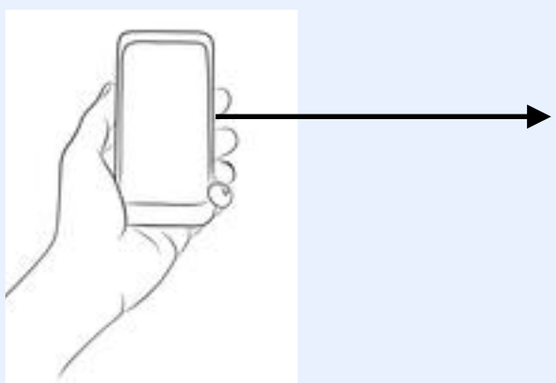
the data

summary

acceleration

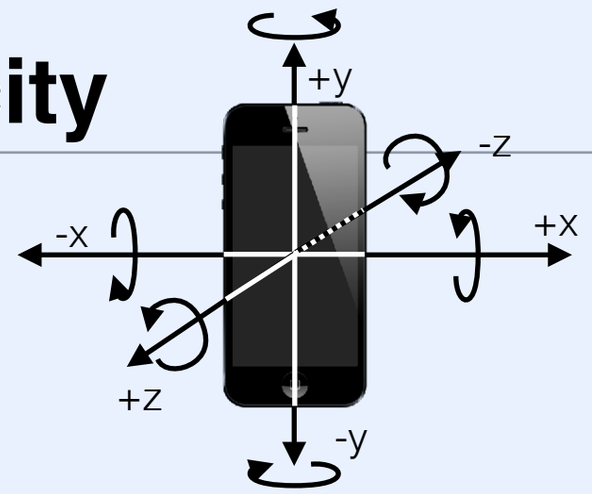
```
CMDeviceMotion *deviceMotion

deviceMotion.gravity
deviceMotion.userAcceleration

CMAcceleration gravity,
CMAcceleration userAcceleration

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

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





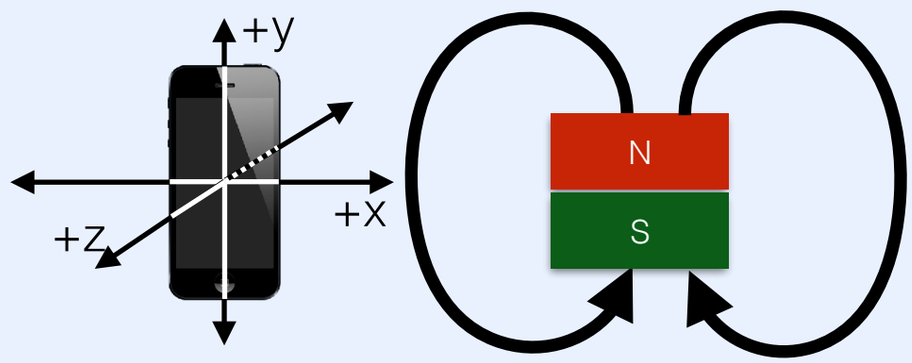
rotation velocity

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



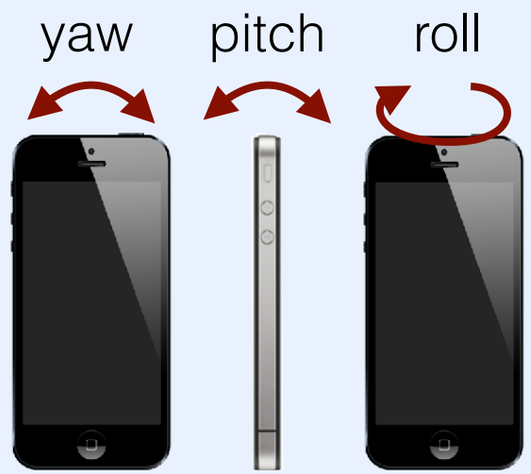
magnetic field

```
deviceMotion.magneticField
CMCalibratedMagneticField magneticField;

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

magneticField.accuracy
```



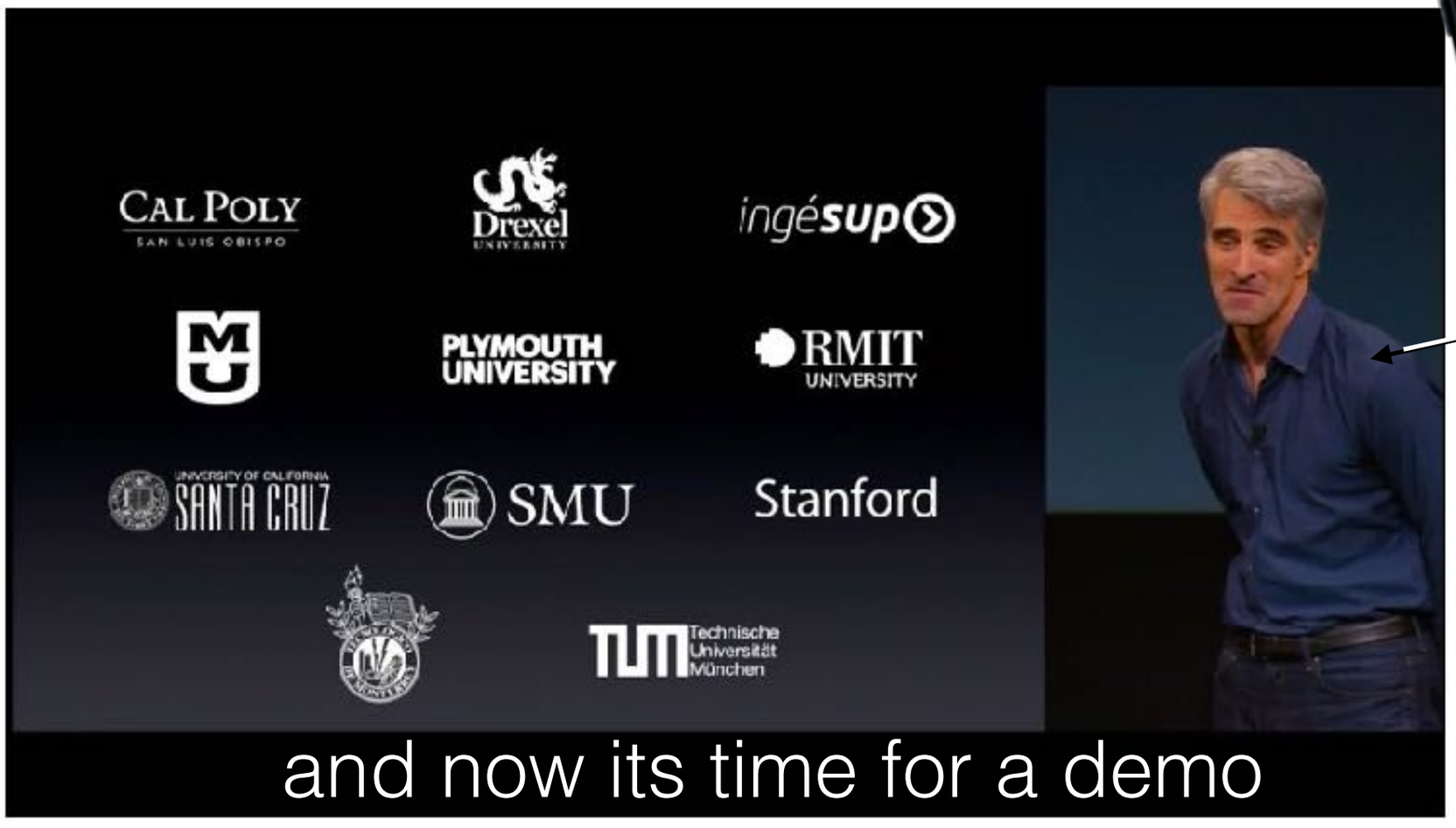
device position

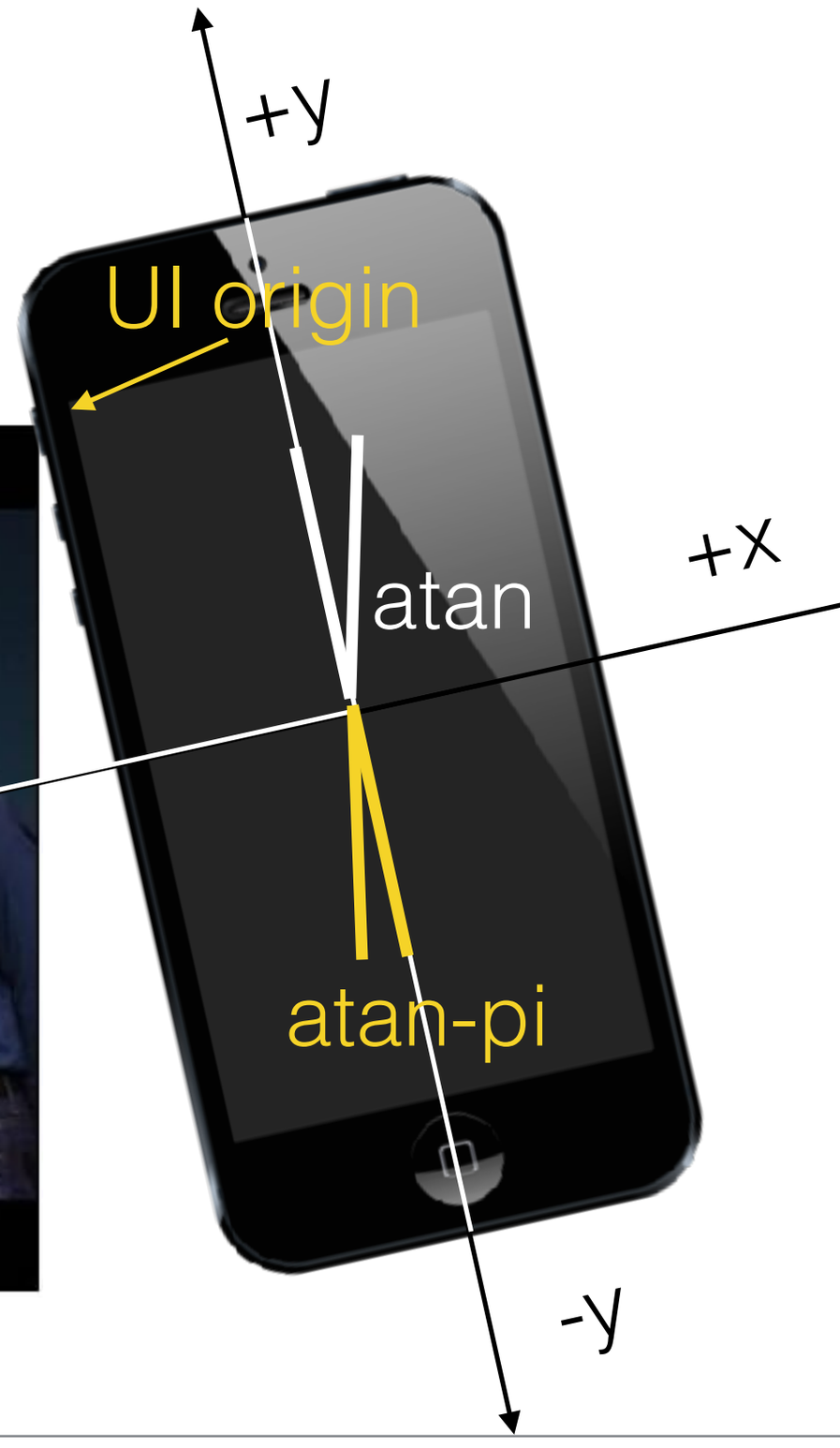
```
deviceMotion.attitude

CMAttitude* attitude

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



device motion demo

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

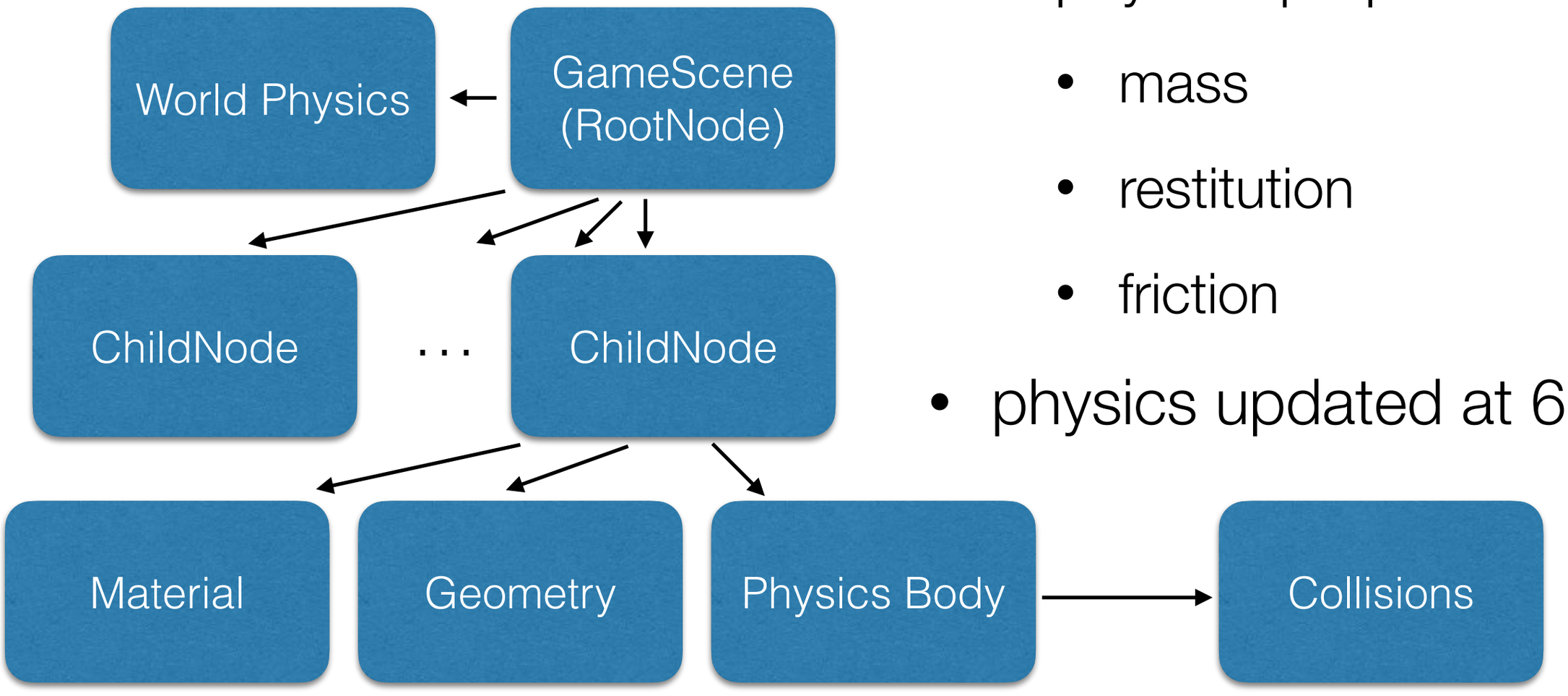




something more?

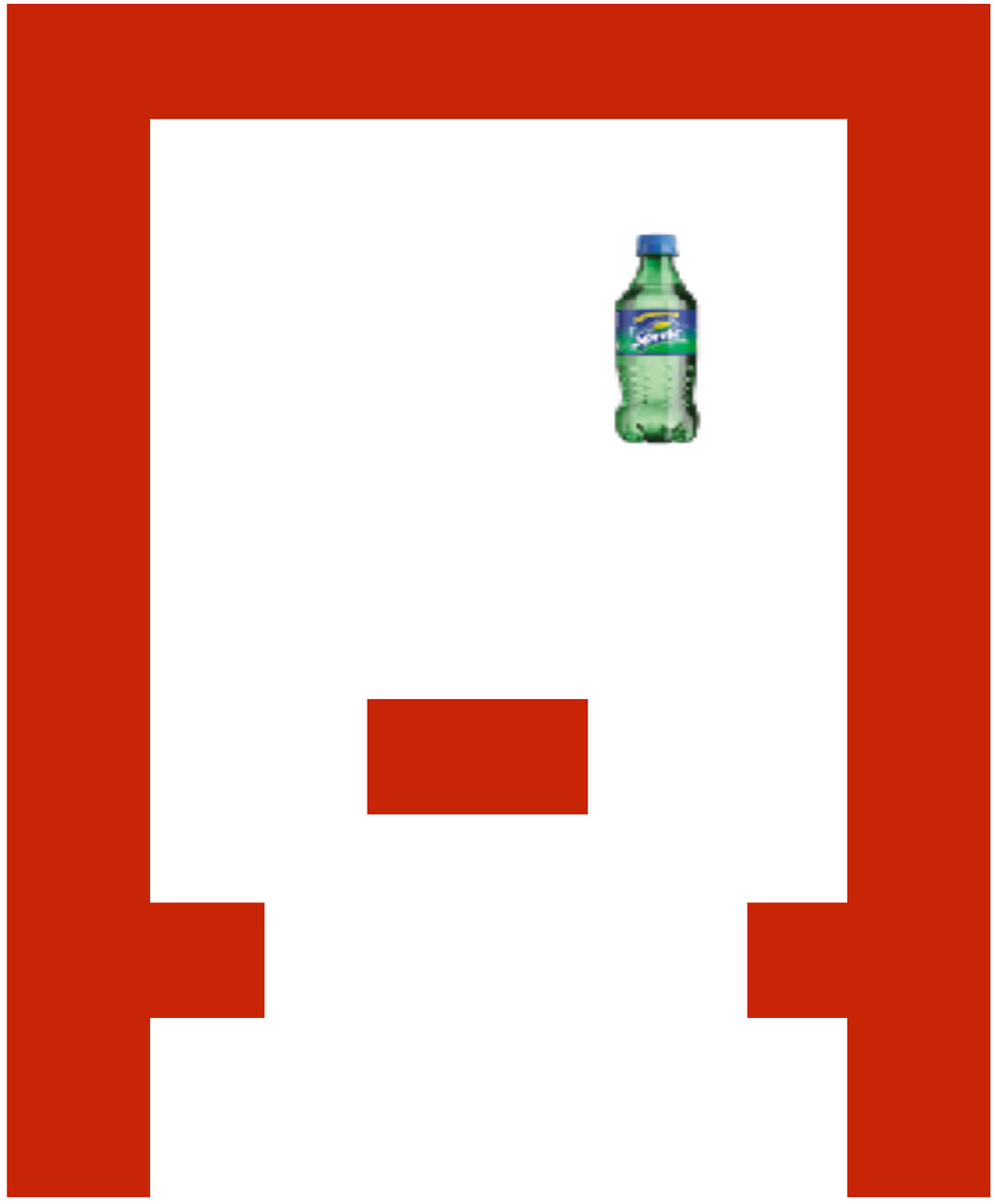
- 2D Physics Engine?
- Enter SpriteKit:
  - SK abbreviated
  - real time physics engine for game applications
  - …and 2D games in general
- how about a 3D physics engine?
  - Enter SceneKit

SpriteKit



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

SpriteKit



create “blocks”

create “sides/top”

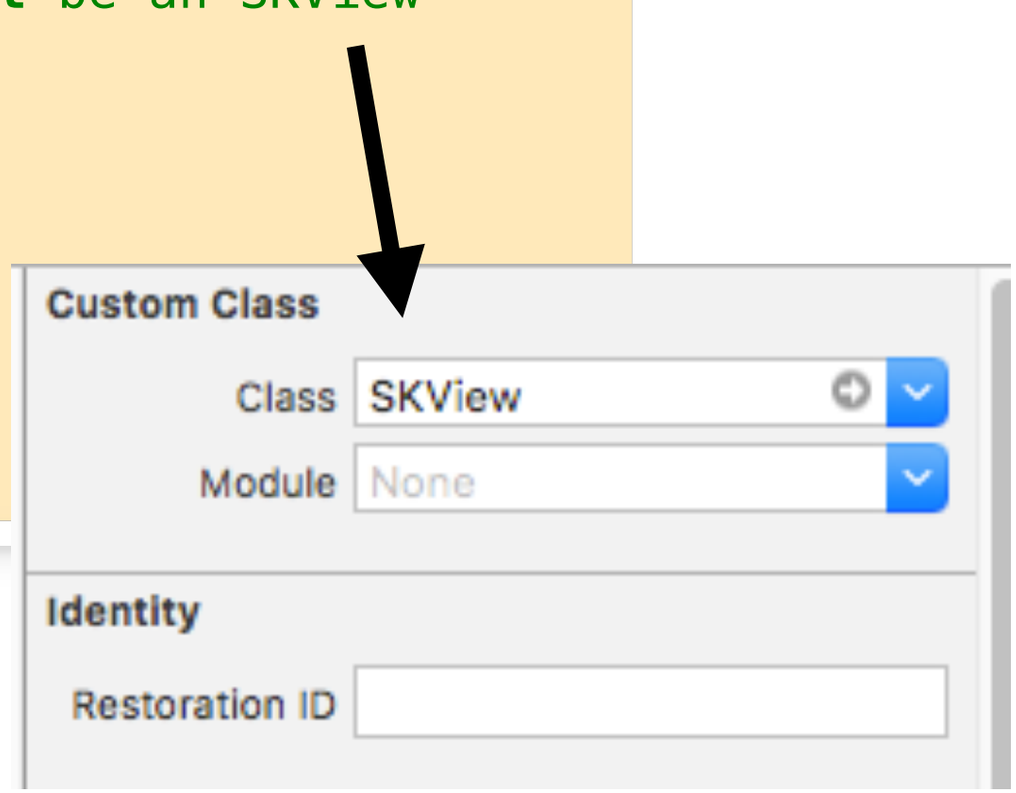
create “bouncy” sprite

make actual gravity == game gravity

user must move phone to keep sprite bouncing on target

setup view controller

```
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)
    }
}
```



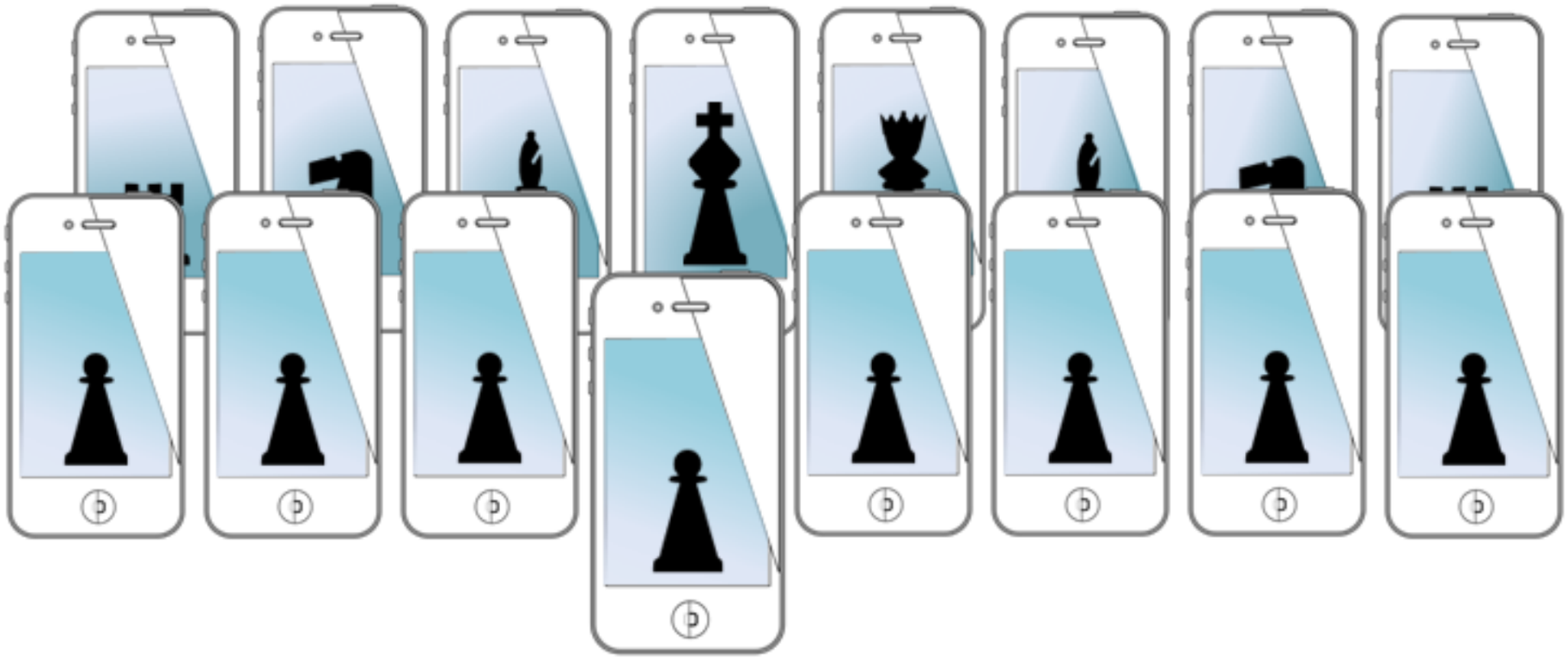
set gravity

```
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

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