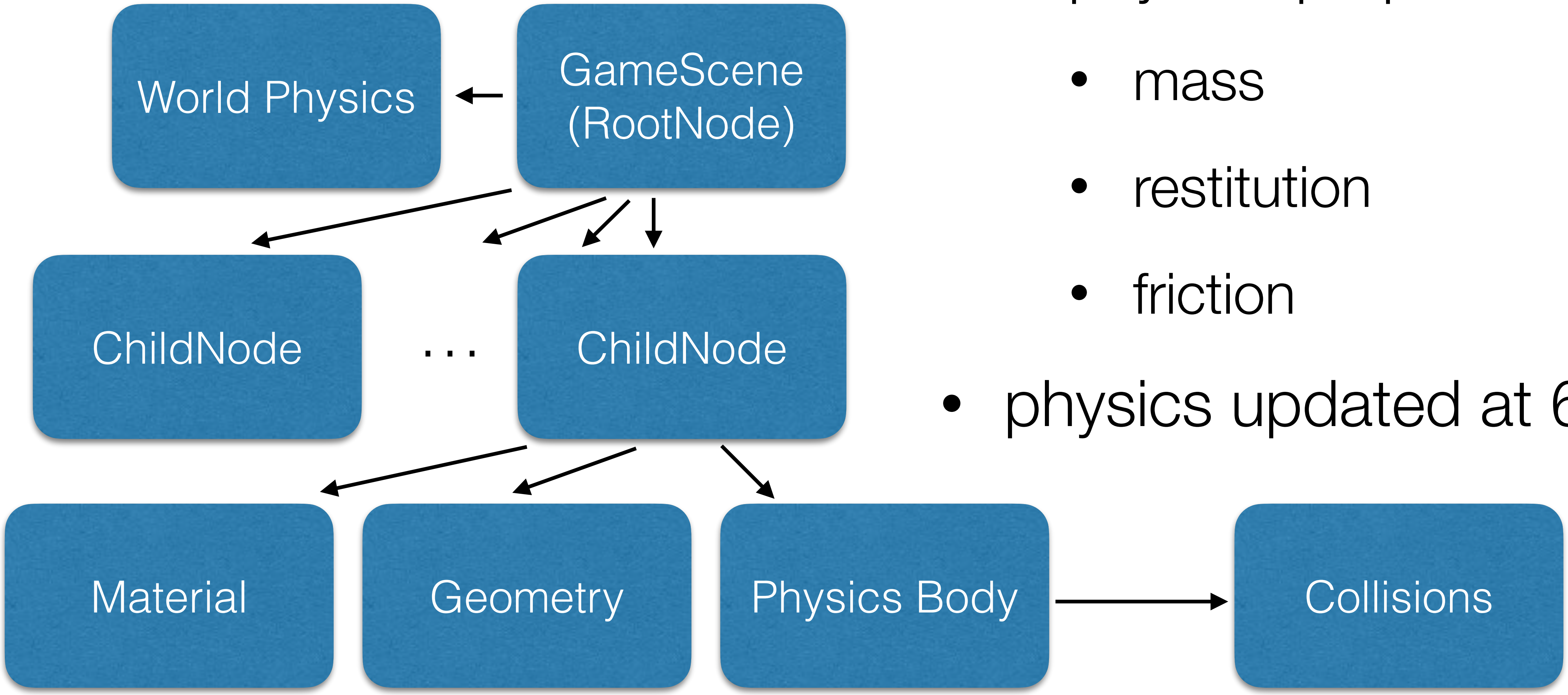
something more?

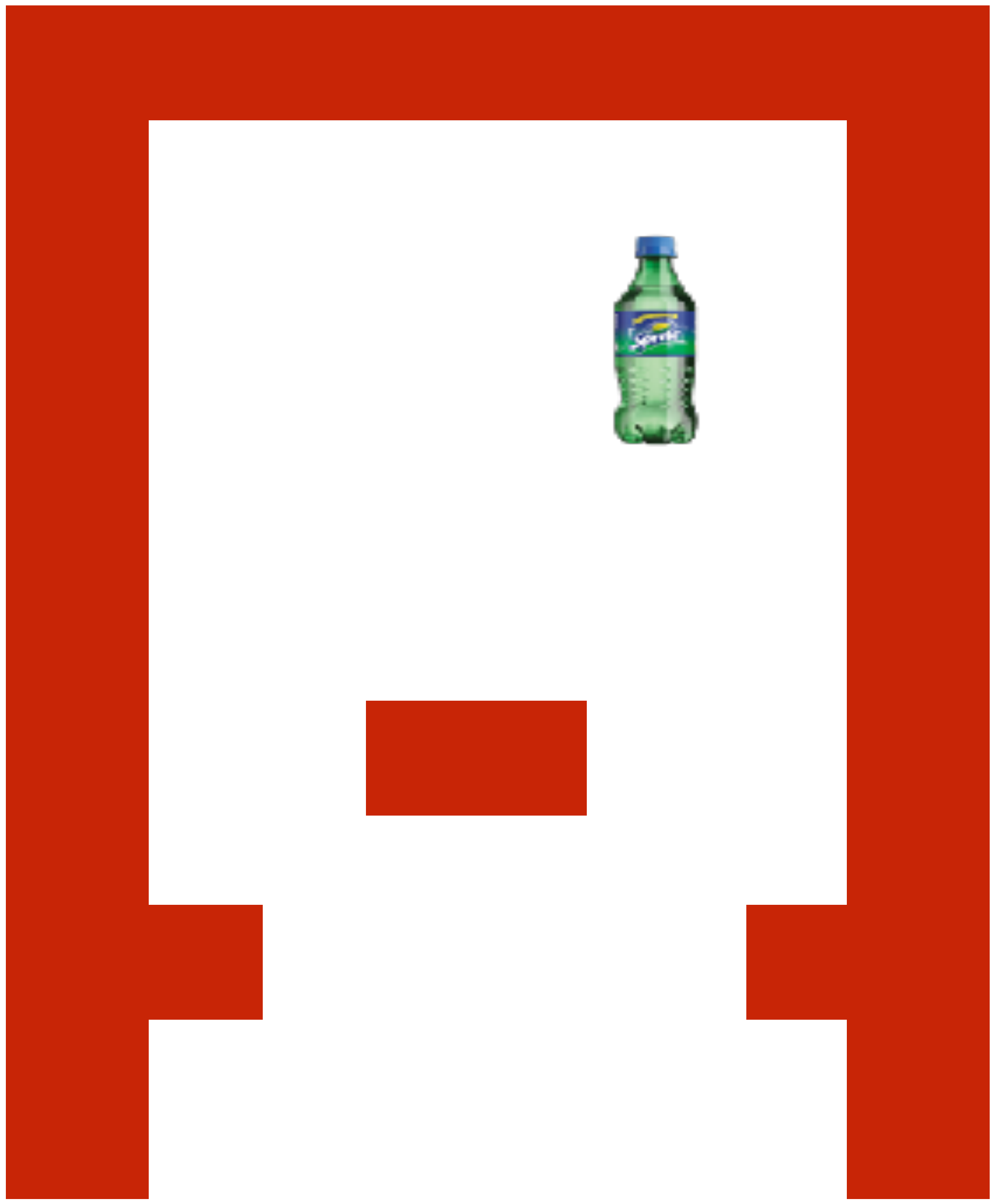
- 2D Physics Engine?
- Enter SpriteKit:
  - SK abbreviated
  - real time physics engine for game applications
  - …and 2D games in general
- how about a 3D physics engine?
  - Enter SceneKit

SpriteKit

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



SpriteKit



create "blocks"

create "sides/top"

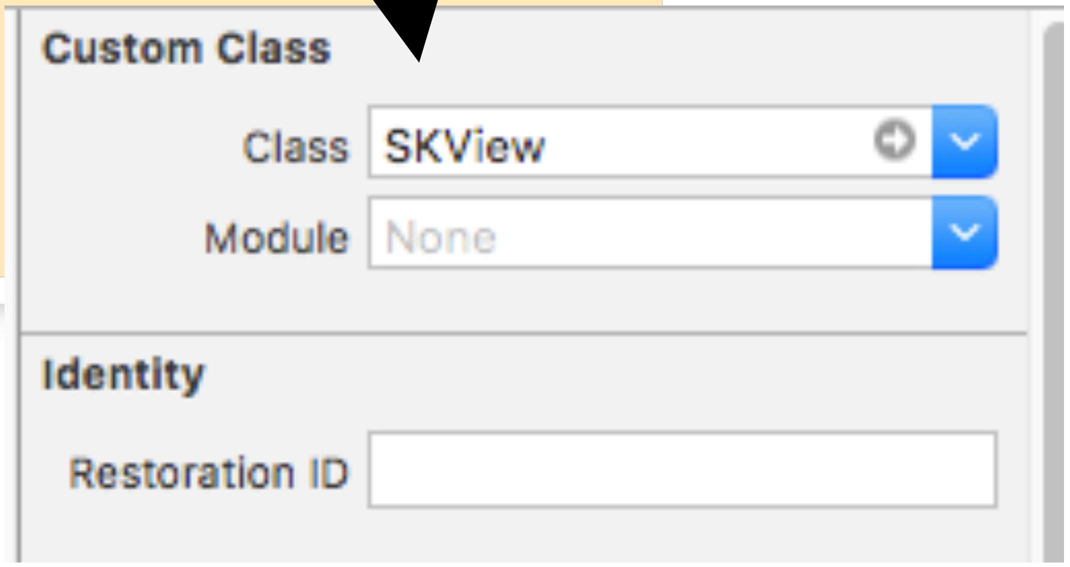
create "bouncy" sprite

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)
    }
}
```



Class: SKView
Module: None
Restoration ID:

set gravity

```
let motion = CMMotionManager()
func startMotionUpdates(){

     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

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

  spriteA.size = CGSize(width:…,height:…)

  let randNumber = random(min: CGFloat(0.1), max: CGFloat(0.9))
  spriteA.position = CGPoint(x: some_val * randNumber, y: … )

  spriteA.physicsBody = SKPhysicsBody(rectangleOf: spriteA.size)
  spriteA.physicsBody?.restitution = random(min: 1.0, max:1.5)
  spriteA.physicsBody?.isDynamic = true

  spriteA.physicsBody?.contactTestBitMask = 0x00000001
  spriteA.physicsBody?.collisionBitMask = 0x00000001
  spriteA.physicsBody?.categoryBitMask = 0x00000001
  self.addChild(spriteA)
}
```

add from image assets

setup size and position

interaction physics

collision and group

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.

collision

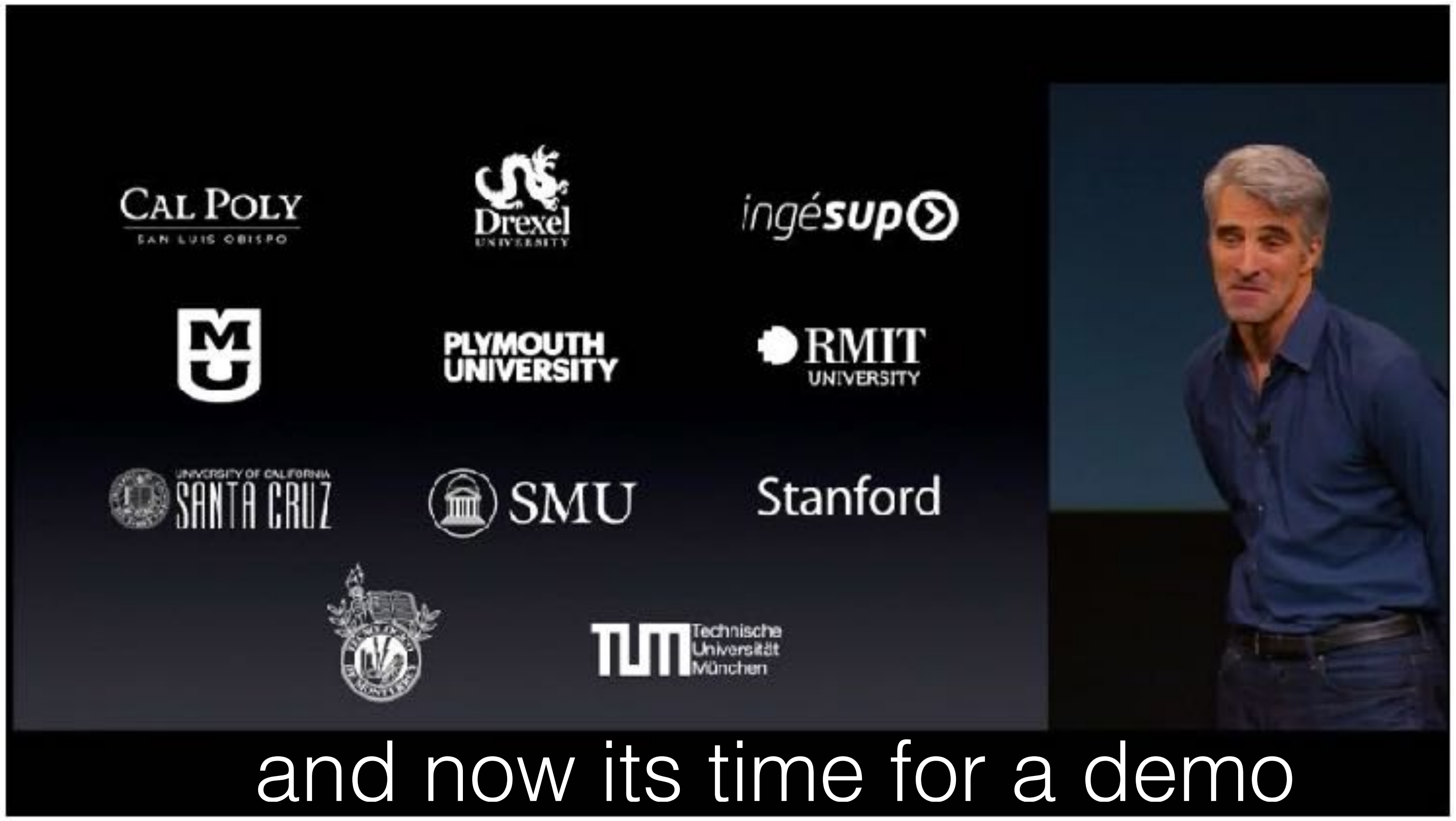
```
spriteA.physicsBody?.contactTestBitMask = 0x00000001
spriteA.physicsBody?.collisionBitMask = 0x00000001
spriteA.physicsBody?.categoryBitMask = 0x00000001
```

collision and group

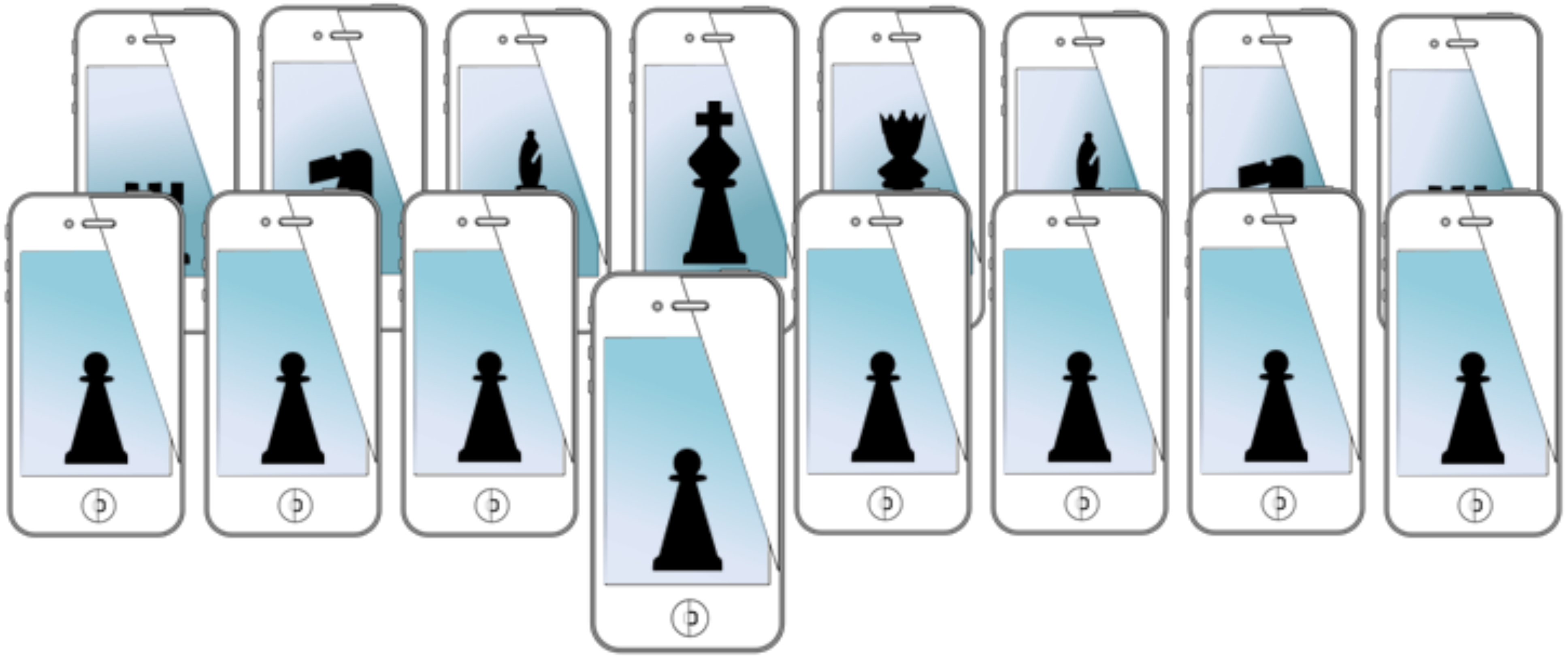
- **categoryBitMask**: A mask that defines which categories this physics body belongs to (***grouping***)
- **contactTestBitMask**: A mask that defines which categories of bodies cause intersection notifications with this physics body (***intersection***, ***could pass through***)
- **collisionBitMask**: A mask that defines which categories of physics bodies can ***collide*** with this physics body.

device motion demo 2

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



MOBILE SENSING LEARNING



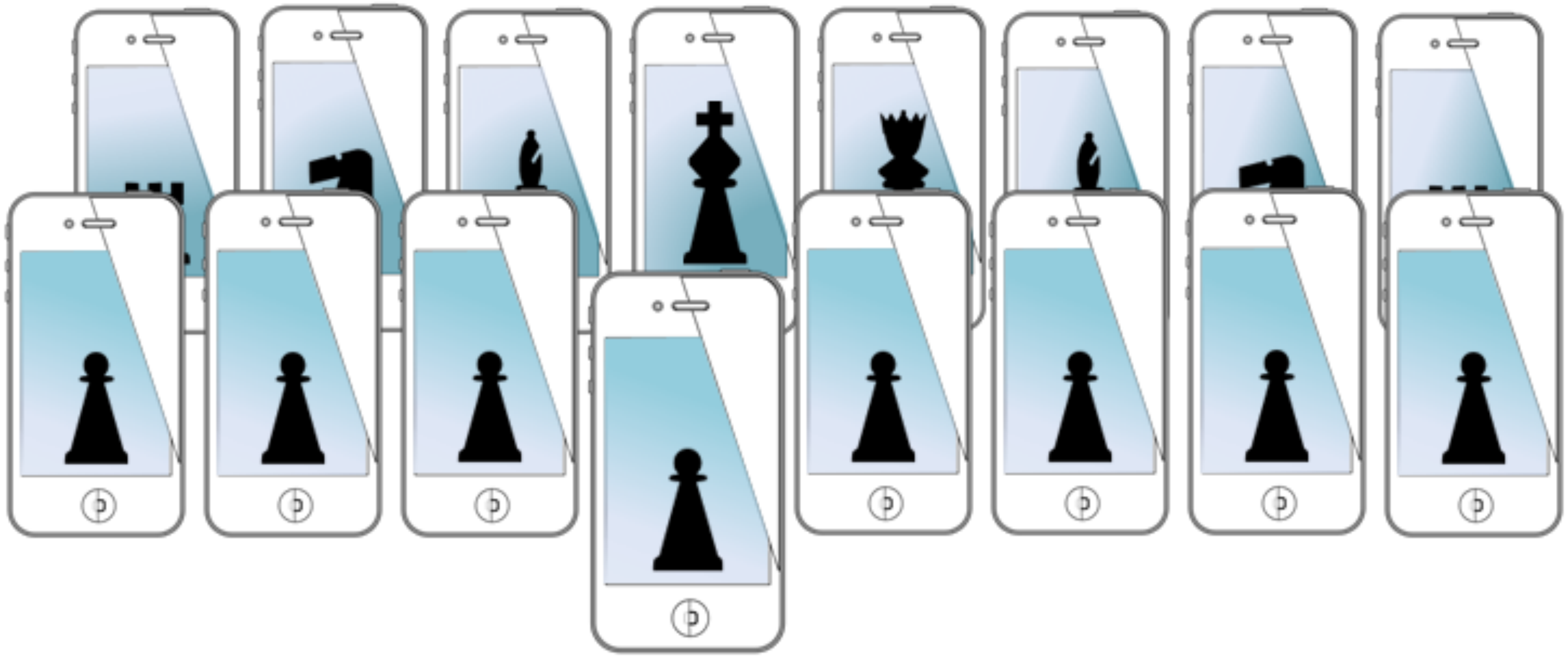
CS5323 & 7323

Mobile Sensing and Learning

activity, pedometers, and motion sensing

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

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

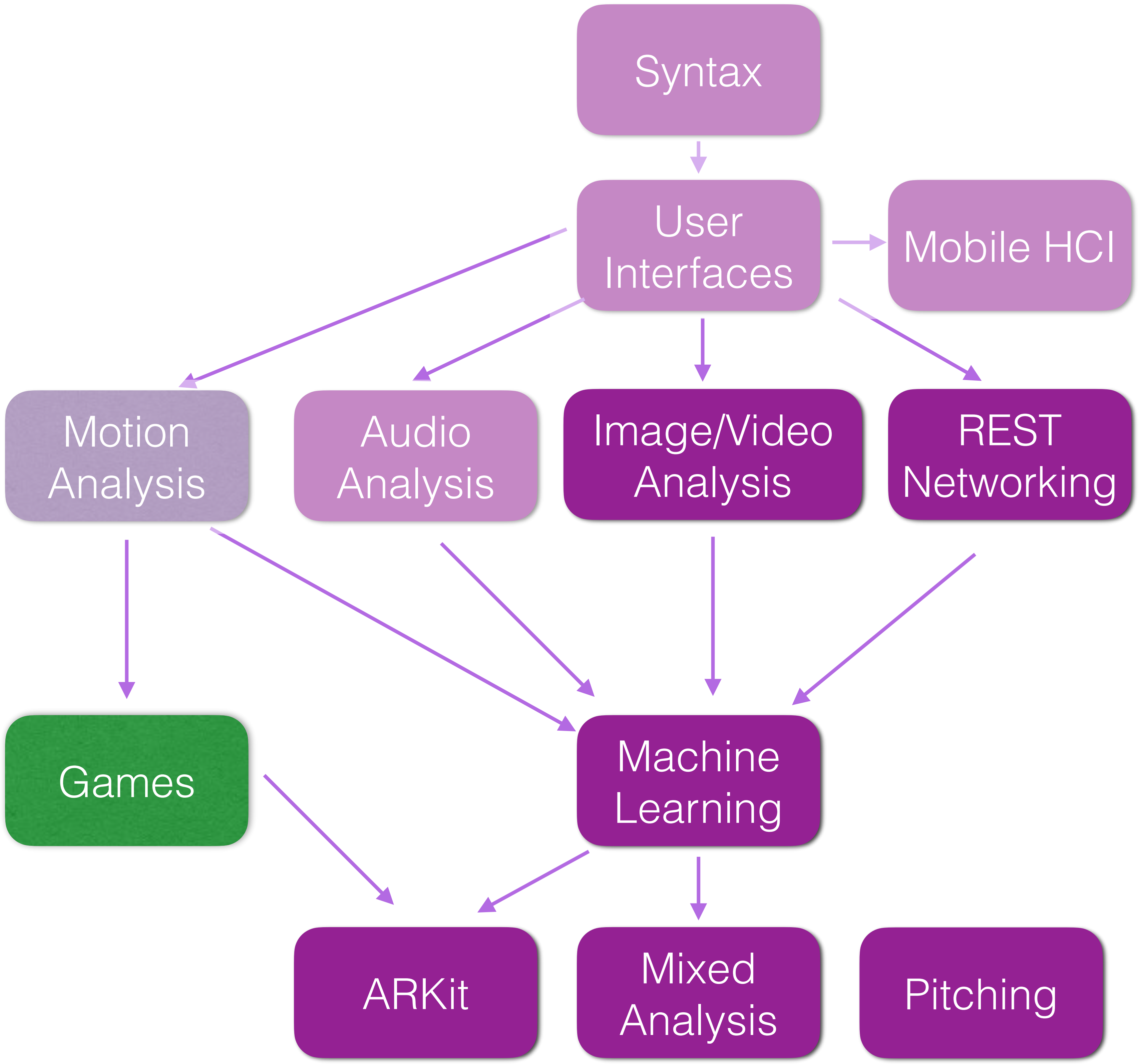
SceneKit and 3D Games

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

logistics and agenda

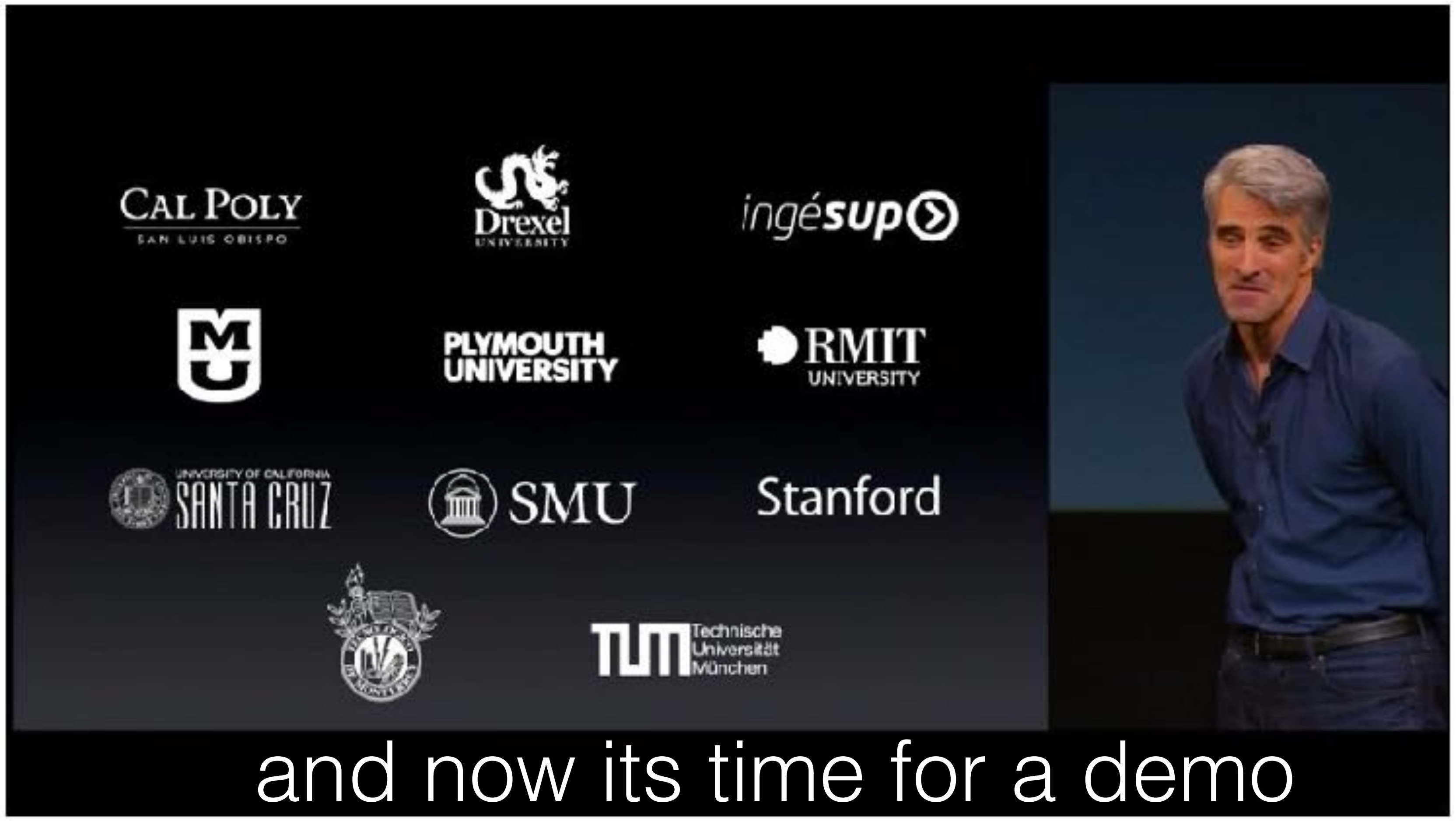
- Logistics:
  - grading update
- agenda:
  - SpriteKit Review
  - SceneKit

class overview



device motion game demo

- lemon lime bounce
- pre-made demo
- **Let's look at buttons in the game**



SceneKit: 3D scenes

- SceneKit allows you to create a 3D world and add physics, nodes, lighting, etc.
  - very powerful
- basic workflow:
  - setup world
  - add nodes

SpriteKit SceneKit

work flow in 3D scenes

SCNScene

rootNode

SCNNode

Name
Camera
Light
Geometry

childNodes[]

SCNGeometry

Name
Materials[]
Text
Shape (plane, box, sphere, cylinder…)

SCNMaterial

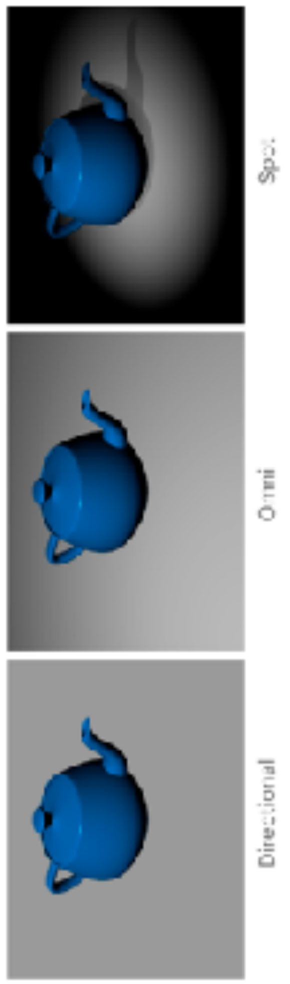
Name
Diffuse
Ambient
Specular
Reflective
…

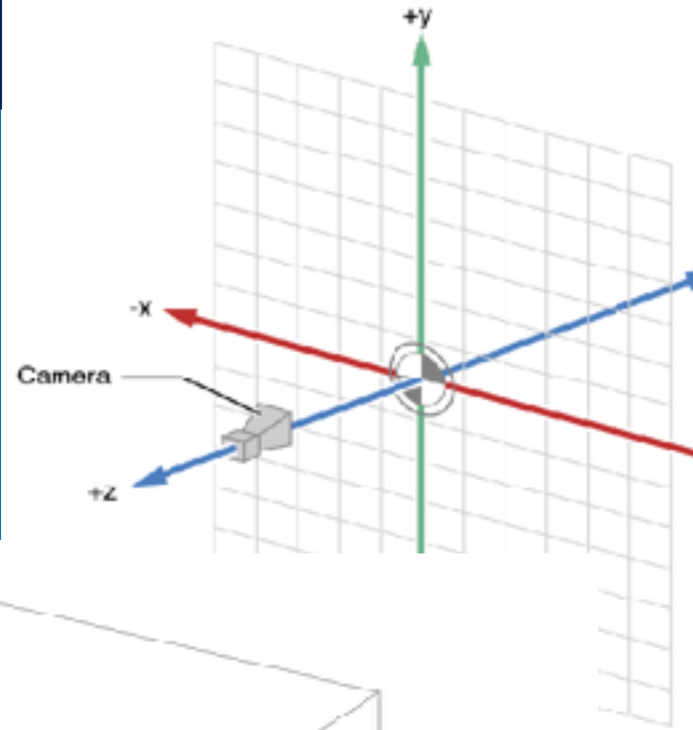
SCNLight

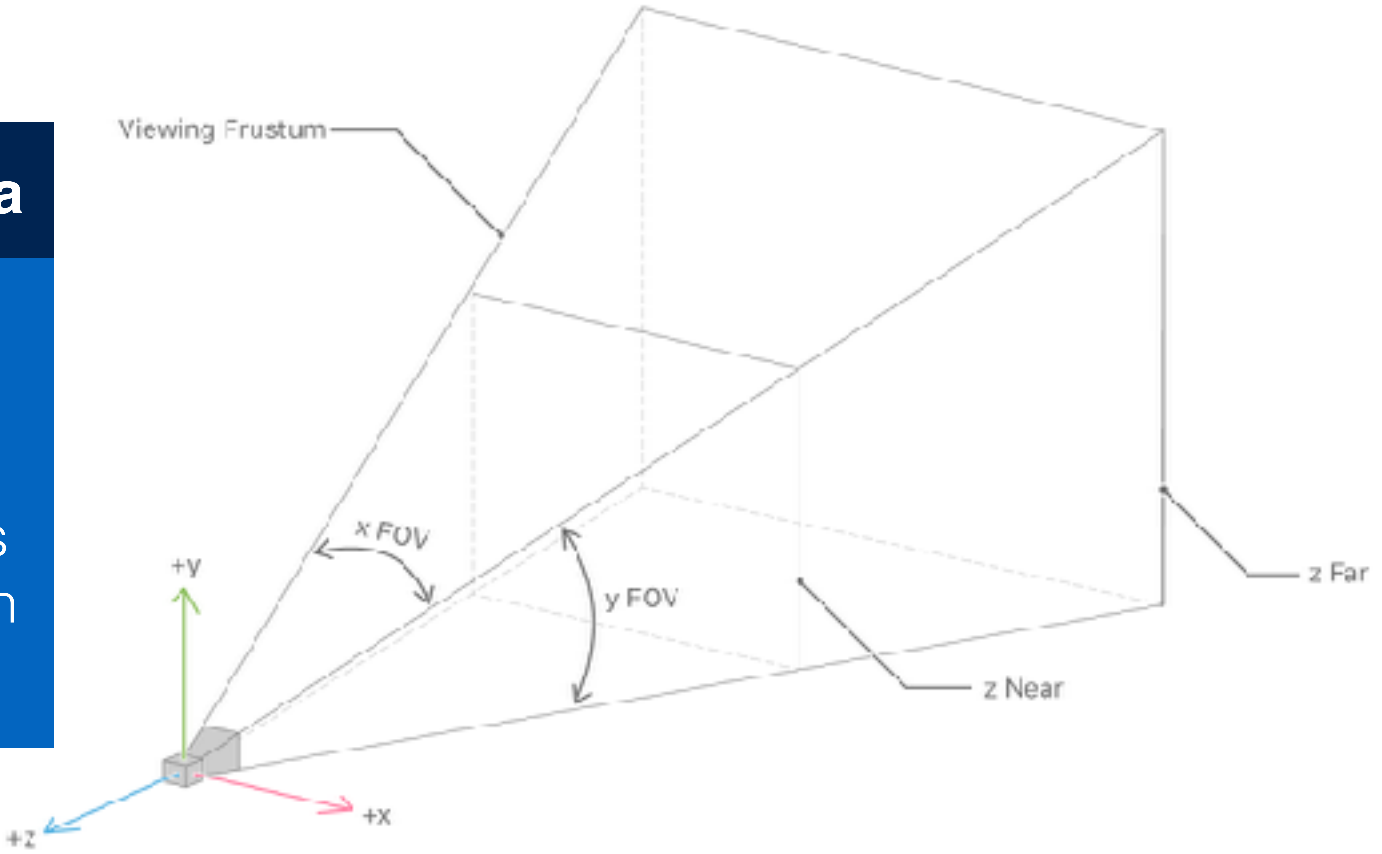
Type (ambient, omni, spot, directional)
Color
Temperature
Intensity
…

SCNCamera

Location
FoV
zNear / zFar
Euler Angles
Focal Length
…







SCNNode is the base for nearly everything in simulation env.

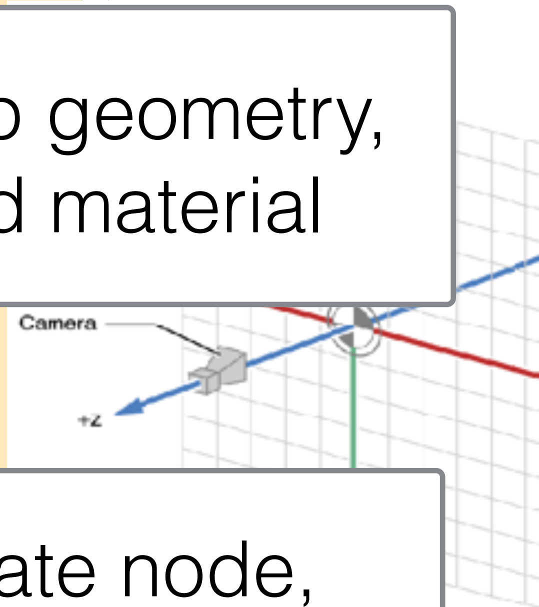
example: setting up a world

```swift
// Setup scene
scene = SCNScene()
scene.physicsWorld.speed = 1


// Setup camera position
cameraNode = SCNNode()
cameraNode.camera = SCNCamera()
cameraNode.position = SCNVector3(x: 0, y: 0, z: 30)


 let wall = SCNPlane(width: 10.0, height: 10.0)
 wall.firstMaterial?.doubleSided = true
 wall.firstMaterial?.diffuse.contents = UIColor.whiteColor()



// add the plane to the world as a static body (no dynamic physics)
wallNode = SCNNode()
wallNode.physicsBody = SCNPhysicsBody.staticBody()
wallNode.position = SCNVector3(x: 0.0, y: 0.0, z: -5)
wallNode.geometry = wall

 scene.rootNode.addChildNode(cameraNode)
 scene.rootNode.addChildNode(wallNode)


let view = self.view as SCNView
view.scene = scene
```

create empty scene

add camera

setup geometry, and material



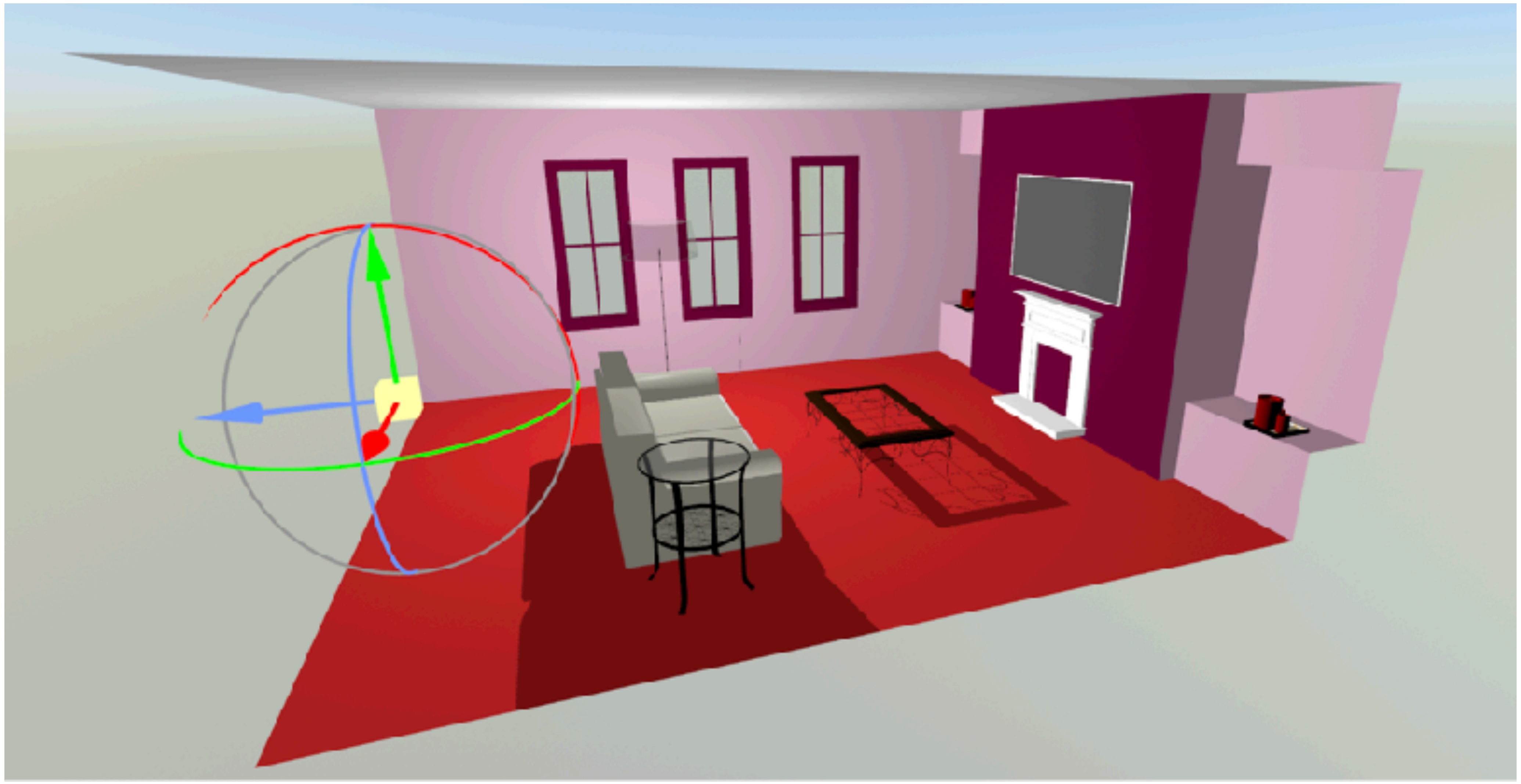
create node, set geometry

add nodes to scene

make this scene the world

making a scene

- many software allow export to .scn files (blender, sketchup, maya, etc.)
- many other exports can be imported by Xcode (like .dae file)
- once imported, Xcode allows manipulation of nodes



adding custom node to world

```
func addBall() {

    // add a sphere to the world
    let ballGeometry = SCNSphere(radius: 1.0)

    // make it have texture
     let ballMaterial = SCNMaterial()
     ballMaterial.diffuse.contents = UIImage(named: "textu
      // adjust physics to make it slightly highly bouncy
      let ball = SCNNode(geometry: ballGeometry)
      ball.geometry?.firstMaterial = ballMaterial;
      ball.position = SCNVector3(x: 0, y: 0, z: 0)

    ball.physicsBody = SCNPhysicsBody.dynamicBody()
    ball.physicsBody?.restitution = 2.5

    scene.rootNode.addChildNode(ball)
  }
```

make geometry

make material

make node

adjust physics

add to world

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.

world physics, motion

```
motionManager.startDeviceMotionUpdatesToQueue(
   OperationQueue.currentQueue())
        {  (deviceMotion, error) -> Void in

            let accel = deviceMotion.gravity
            self.scene.physicsWorld.gravity =
                   SCNVector3(x: accel.x, y: accel.y, z: accel.z)

        }
```

similar to SpriteKit
but in three dimensions!!

Physics in a Scene

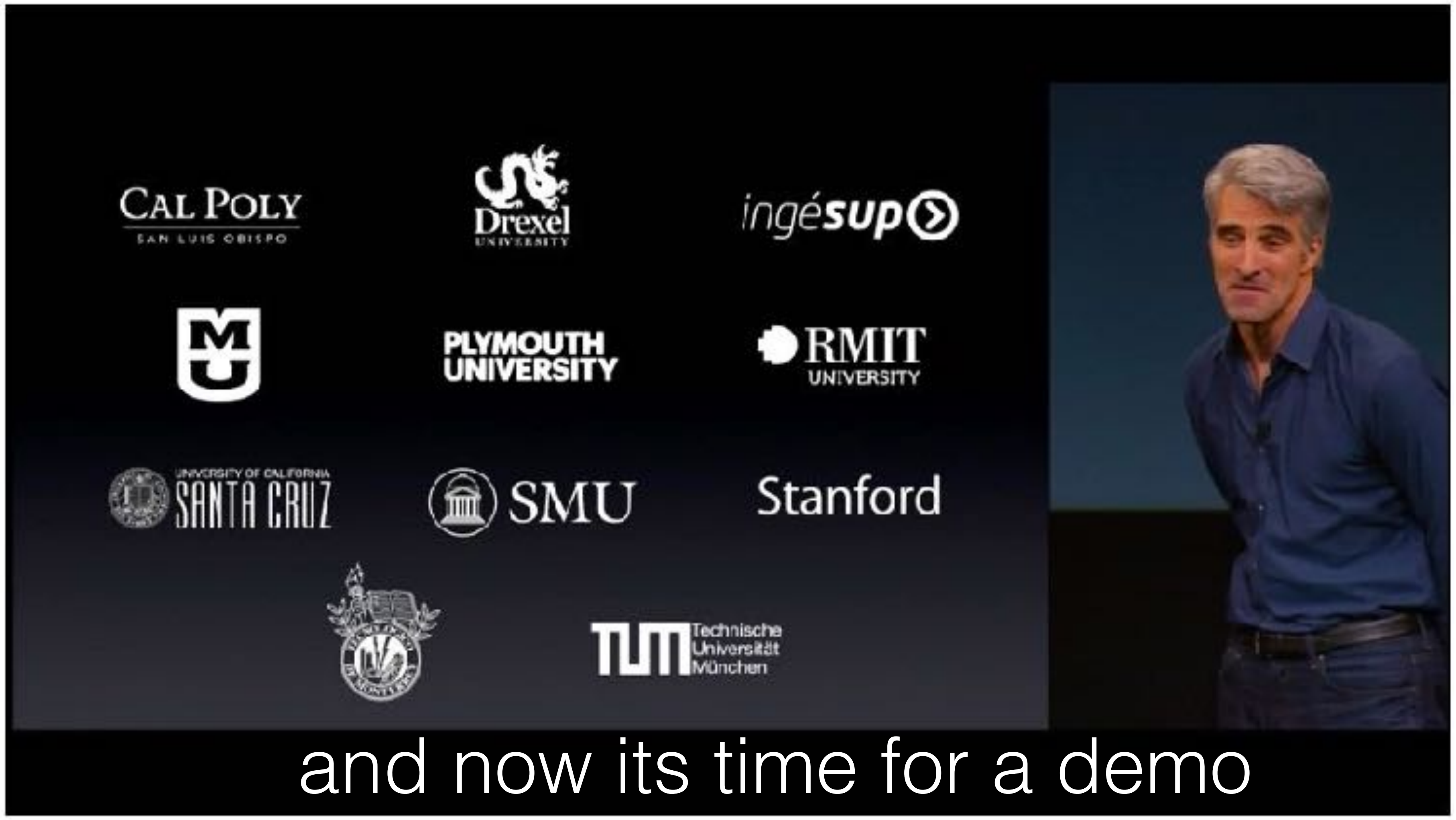
- class SCNPhysicsWorld
  - The global simulation of collisions, gravity, joints, and other physics effects in a scene.
- class SCNPhysicsField
  - An object that applies forces, such as gravitation, electromagnetism, and turbulence, to physics bodies within a certain area of effect.
- class SCNPhysicsBehavior
  - The abstract superclass for joints, vehicle simulations, and other high-level behaviors that incorporate multiple physics bodies.

When we move to **Augmented Reality**, SceneKit is the engine for adding **Virtual Elements** to the Actual World!

Directional
Omni
Spot

device motion demo 3

- SceneKit VR
    - intro to 3D
- hockey
    - formative demo


CAL POLY
SAN LUIS OBISPO
Drexel
UNIVERSITY
ingésup
PLYMOUTH UNIVERSITY
RMIT UNIVERSITY
UNIVERSITY OF CALIFORNIA SANTA CRUZ
SMU
Stanford
Technische Universität München
and now its time for a demo

… and the explanation of lab 3!

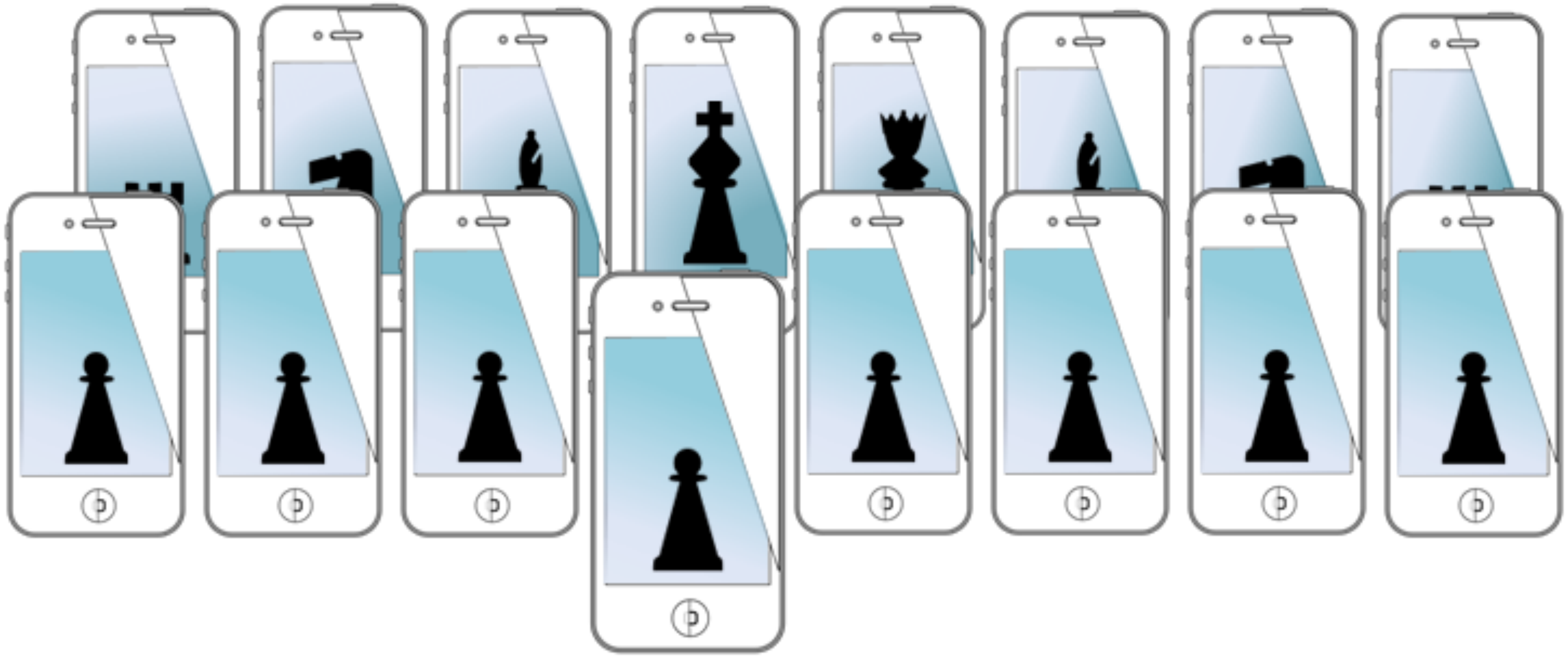
the end of motion…

- before moving on…
- assignment posted

for next time…

- Image processing!

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