Games...Woot!

iOS Game Engines

How does this iPhone stuff work, anyhow?
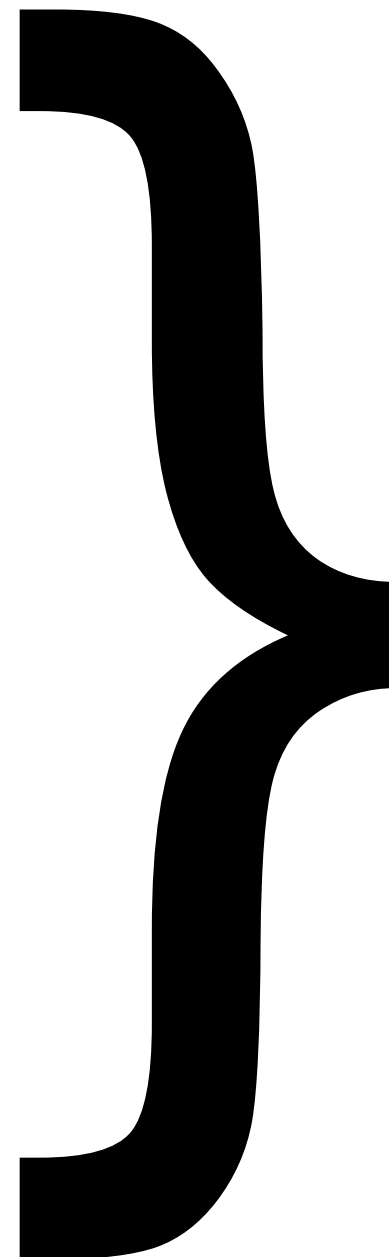
Review: Game Loop

Process Input

↓

Update Object States (position, etc.)

↓

Correct Object States (collisions, etc.)

↓

(Update Particles and other auxiliaries)

↓
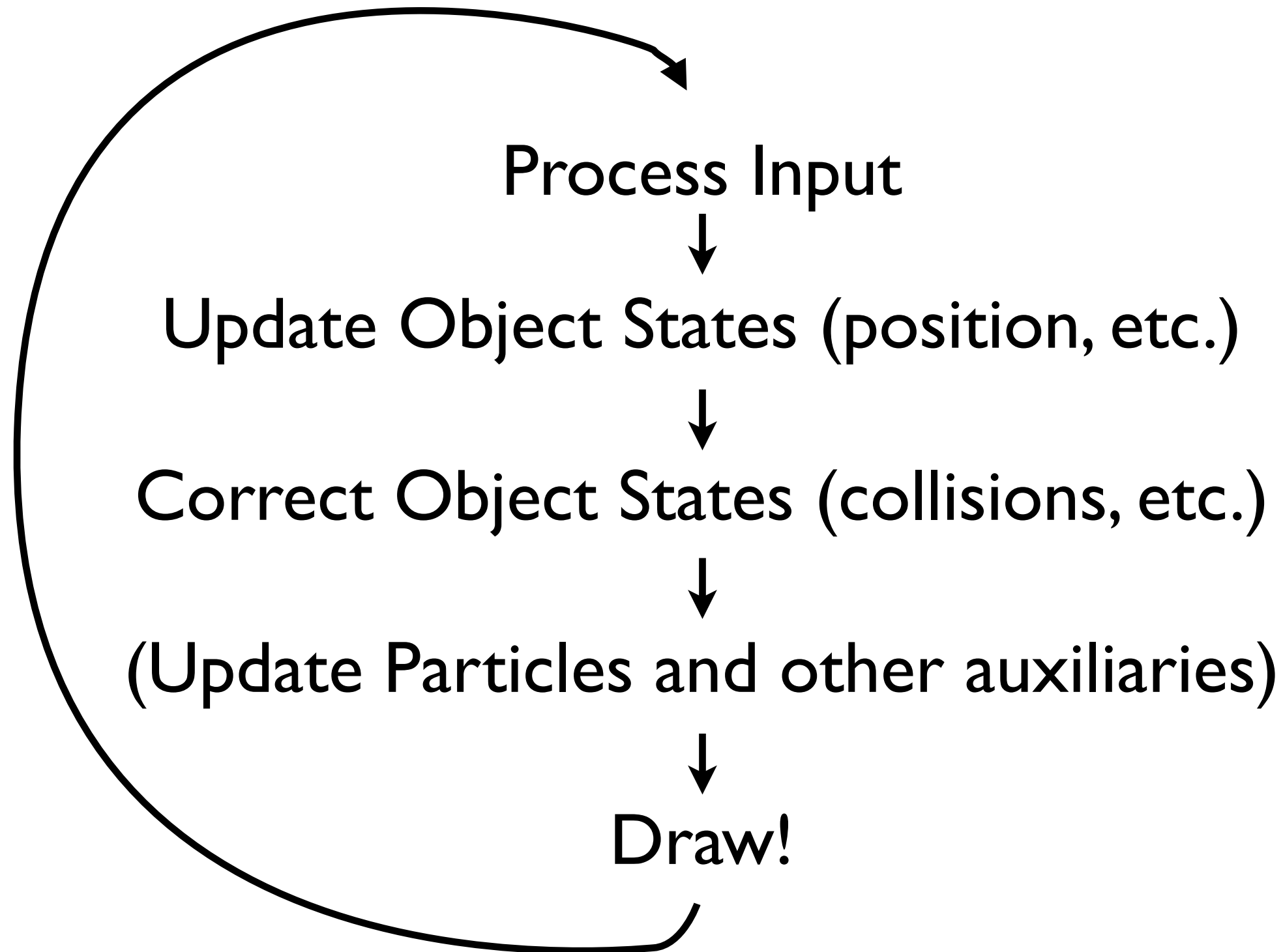
Draw!

Any Game Engine

Input Manager

Scene Controller

Collision Manager

AI Controller

Drawing Controller

Player Profile Manager

Audio Controller

...

} The Engine

Some iOS Basics

- Objective-C = C, C++, and Smalltalk

- Brackets are your new best friend, "[" and "]"

- Pretty much everything is an object...

- ...NSObject ("NeXTSTEP" Object)

- Function calls are called "messages"

Brackets? What?

object.performFunction(argument)

↓

[object performFunction:argument]

object.function(foo, bar)

↓

[object functionWithThis:foo withThat:bar]

UIKit? OpenGL?

UIKit (Apple)

- Easy to use

- Fully Documented

- Drag & drop

- *Slooooooooow*

OpenGL (Open Source)

- *FAST*

- C proficiency required

- No good starting point

- Very challenging
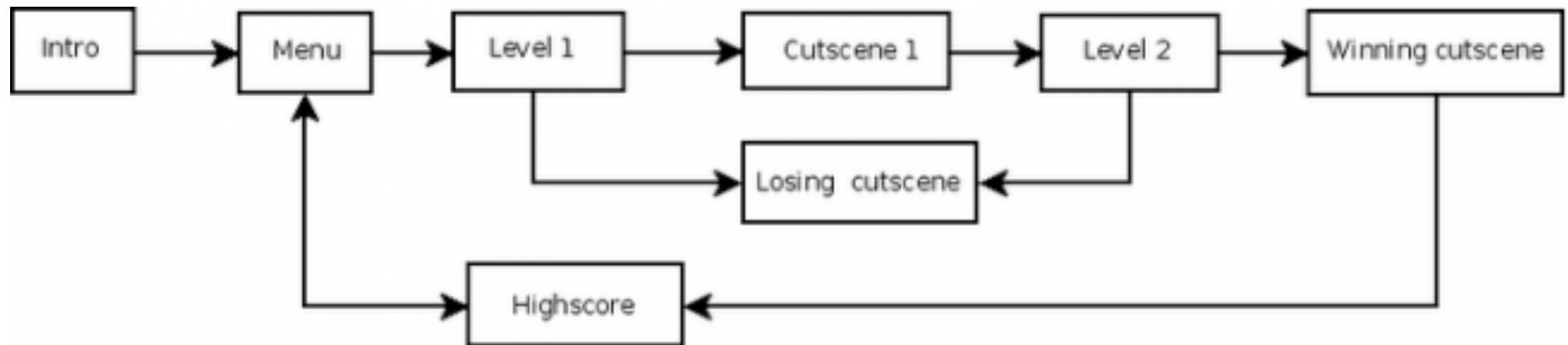
So what do I use then...?

cocos2d

- No reason to reinvent the wheel

- Open source, free to use engine

- Relatively easy to learn (all Objective-C)

- The "NSObject" of cocos2d is "CCNode"

- Did I mention free?

cocos2d: Scenes

- A "room" in a game (Menu, gameplay, etc.)

- Manages all your objects and has most logic

- Multiple scenes make up a full game


```
CCScene *scene = [CCScene node];
```
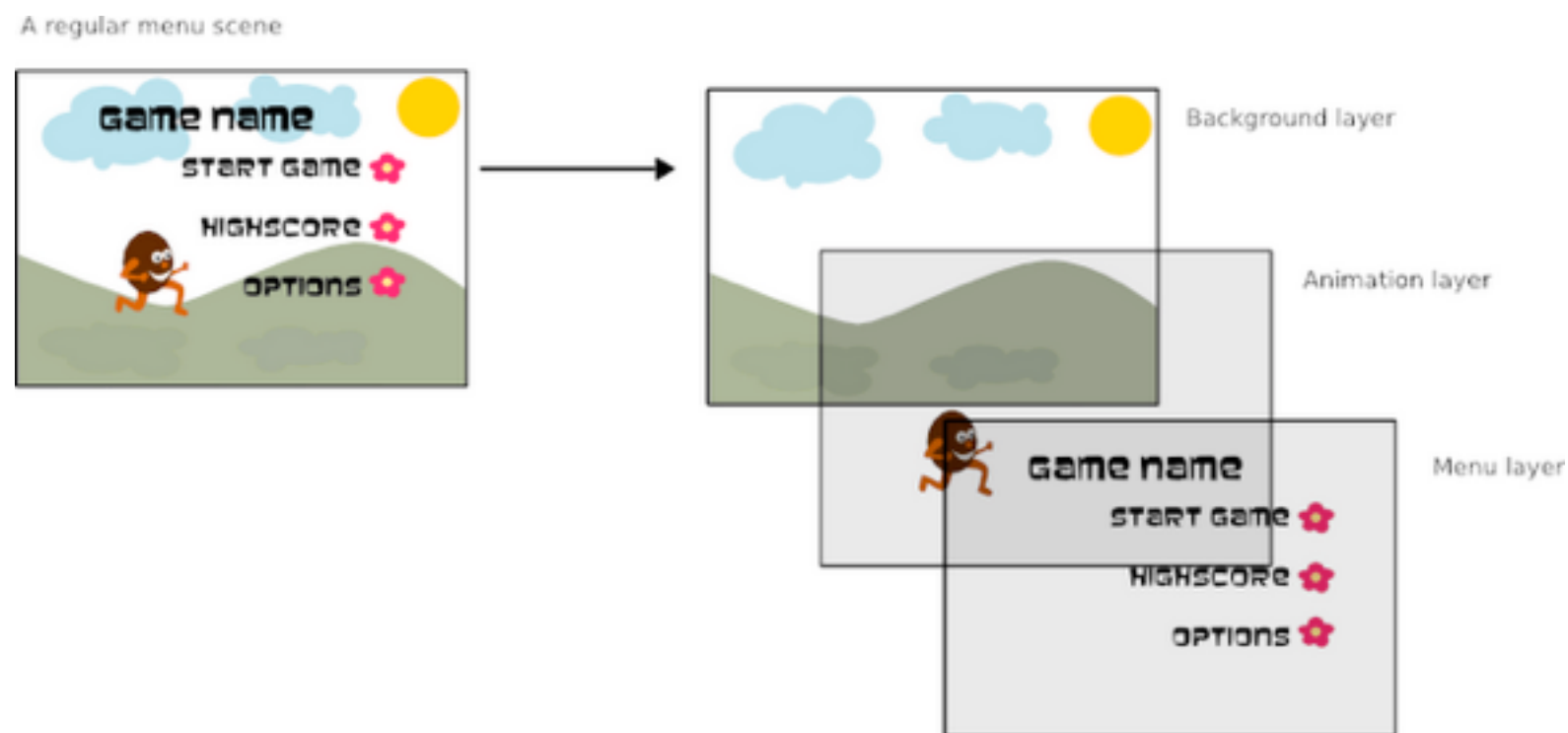
cocos2d: The Director

- A Singleton (there can only be ONE!)

- The outermost logic and management

- Handles transitions between scenes

```
CCDirector* director = [CCDirector sharedDirector];
```

cocos2d: Layers

- A rectangular drawing area, any size

- Layers can live within layers (always inside a scene)


```
CCLayer* layer = [CCLayer node];
```

cocos2d: Sprites

- The fun part! Finally some content!

- A transformable 2D image

- Sprites can live inside... other sprites?

- Always inside of a layer

```
CCSprite* sprite = [CCSprite spriteWithFile:@"sprite.png"];
```

cocos2d: Actions

- Easy way to make sprites move

  1. Create Action

  2. Make a sequence

  3. Apply it to a sprite

- Simple, but should be used with caution

```
CCMoveTo* move = [CCMoveTo actionWithDuration:1.0f position:ccp(200,400)];

CCSequence* seq = [CCSequence actions:move, nil];

[sprite runAction:seq];
```

Demo!