Moving Hoopy

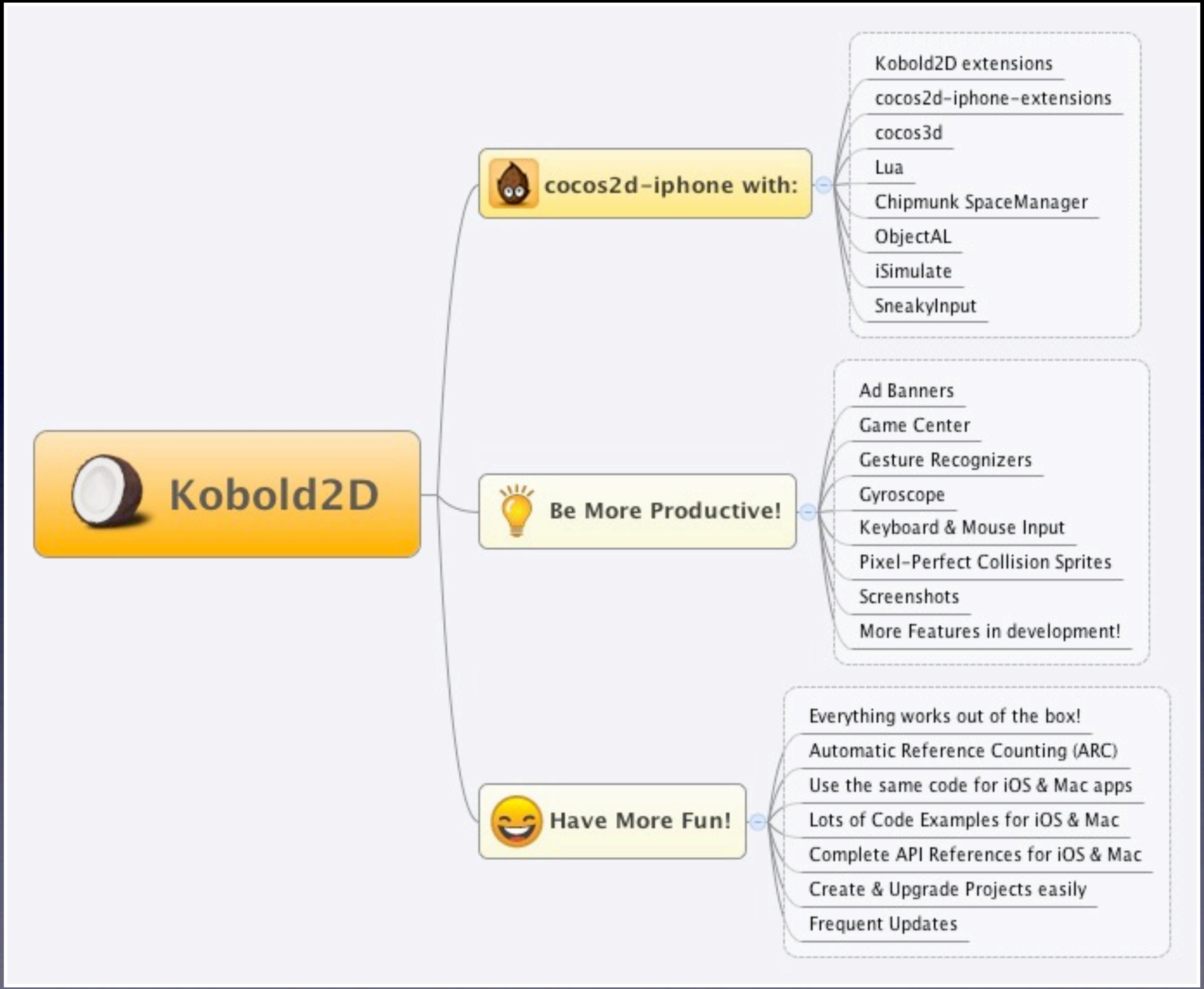
Jon DeJong
Principal Consultant, Object Partners

What We Are Going to Talk About

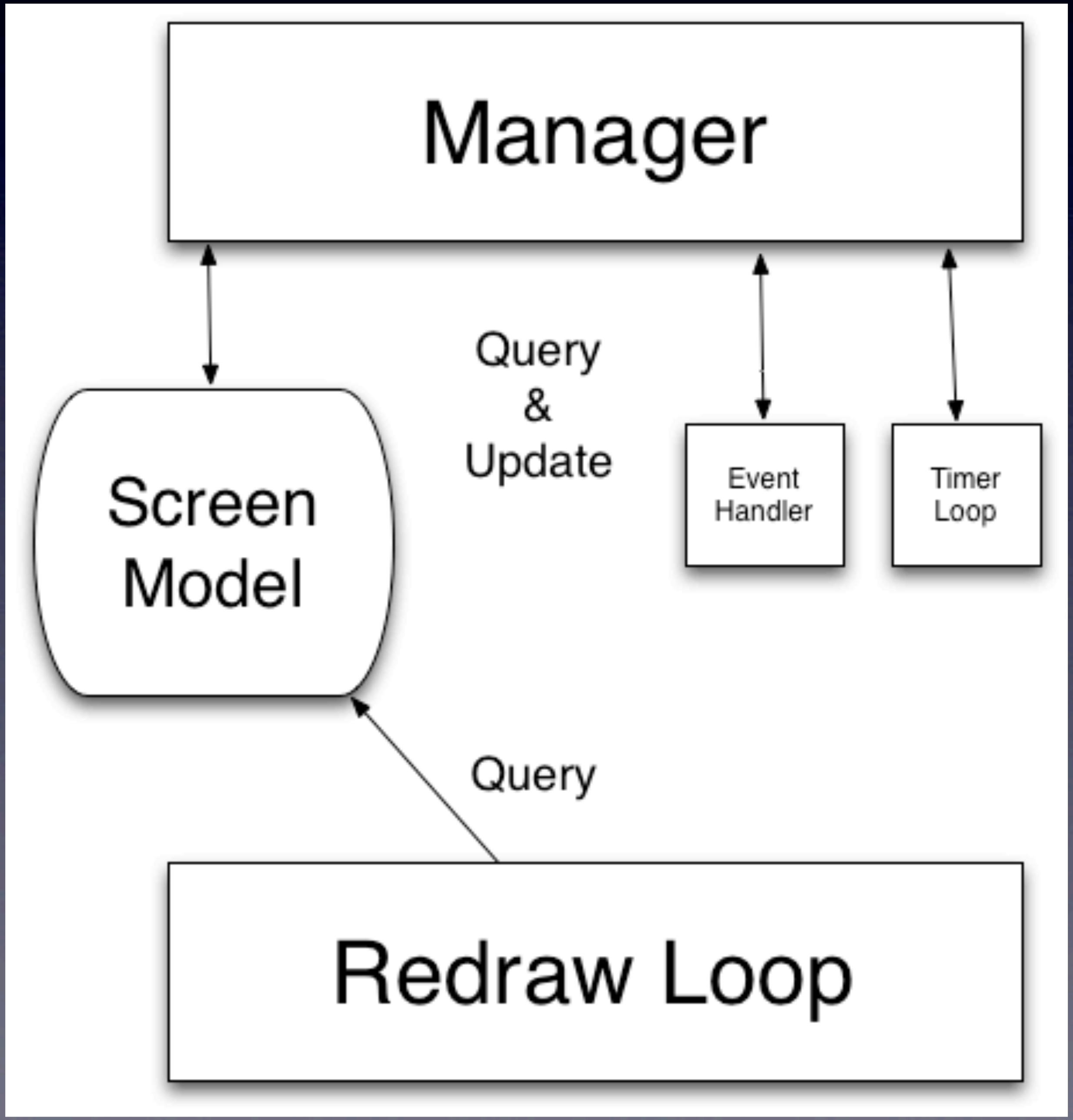
- Quick overview of game architecture
- Overview of Cocos2D architecture
- Basic concepts of Cocos2D
  - sprites, timers, sequences, touches, tiles, fonts
- **Put it all together into a project**
- Move the camera
- Add physics
- Show some helpful tools

Cocos2D

- 2D Game engine.
- Open source. Many versions.
- http://www.cocos2d-iphone.org/
- http://www.kobold2d.com/

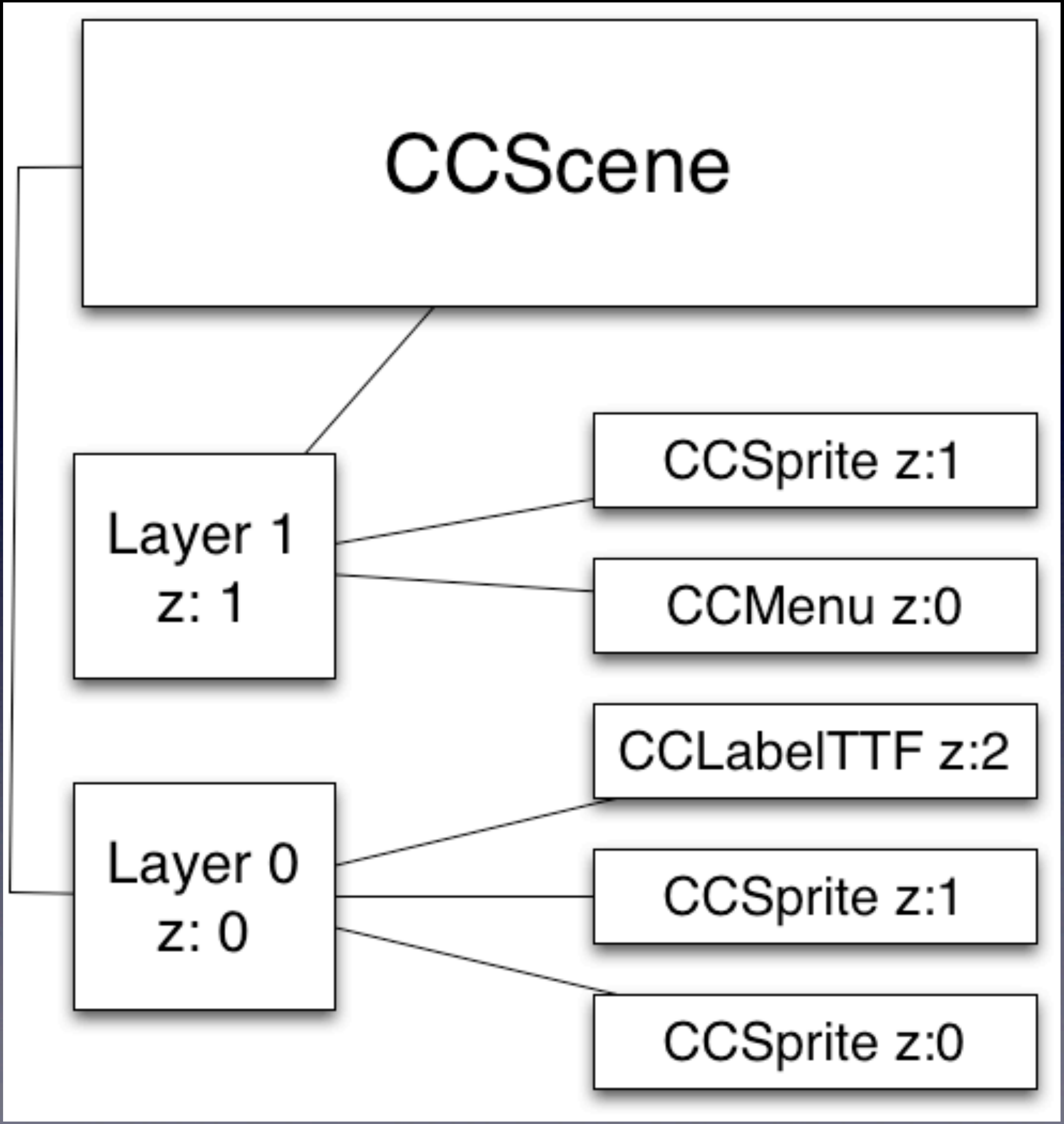

Kobold2D
cocos2d-iphone with:
Kobold2D extensions
cocos2d-iphone-extensions
cocos3d
Lua
Chipmunk SpaceManager
ObjectAL
iSimulate
SneakyInput
Be More Productive!
Ad Banners
Game Center
Gesture Recognizers
Gyroscope
Keyboard & Mouse Input
Pixel-Perfect Collision Sprites
Screenshots
More Features in development!
Have More Fun!
Everything works out of the box!
Automatic Reference Counting (ARC)
Use the same code for iOS & Mac apps
Lots of Code Examples for iOS & Mac
Complete API References for iOS & Mac
Create & Upgrade Projects easily
Frequent Updates

Game Architecture



Everything Is a Node

- Different types have different properties and purposes
- CCNode, CCScene, CCLayer, CCSprite


CCScene
Layer 1 z: 1
CCSprite z:1
CCMenu z:0
Layer 0 z: 0
CCLabelTTF z:2
CCSprite z:1
CCSprite z:0

Scene Stack

Scenes are loaded and unloaded via a stack

```
[[CCDirector sharedDirector] popScene];
[[CCDirector sharedDirector] pushScene: (CCScene*)scene];
[[CCDirector sharedDirector] runWithScene: (CCScene*)scene];
[[CCDirector sharedDirector] replaceScene: (CCScene*)scene];
```

CCSprite

- The fun one. Puts your bitmaps on the screen
- Has many modifiable properties to have fun with
    - Position
    - Anchor
    - Angle
    - RGBA

Timers

```
- (id)init
{
     self = [super init];
     if (self) {
      [self scheduleUpdate];
    }
}

-(void) update:(ccTime)delta
{
...
}
```

Sequences

- Animations
- Transitions

```
CCFiniteTimeAction* moveAction =
 [CCMoveTo actionWithDuration: 4 position: newPoint];

CCSequence* seq = [CCSequence actions:moveAction, nil]

[sprite runAction:seq];
```

Touches

```
-(id) init
{
    if( (self=[super init])){
    self.isTouchEnabled = YES;
}

-(void)ccTouchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
{
  for(UITouch *touch in touches) {
  }
}

-(void)ccTouchesEnded:(NSSet *)touches withEvent:(UIEvent *)event
{
  for(UITouch *touch in touches) {
  }
}
```

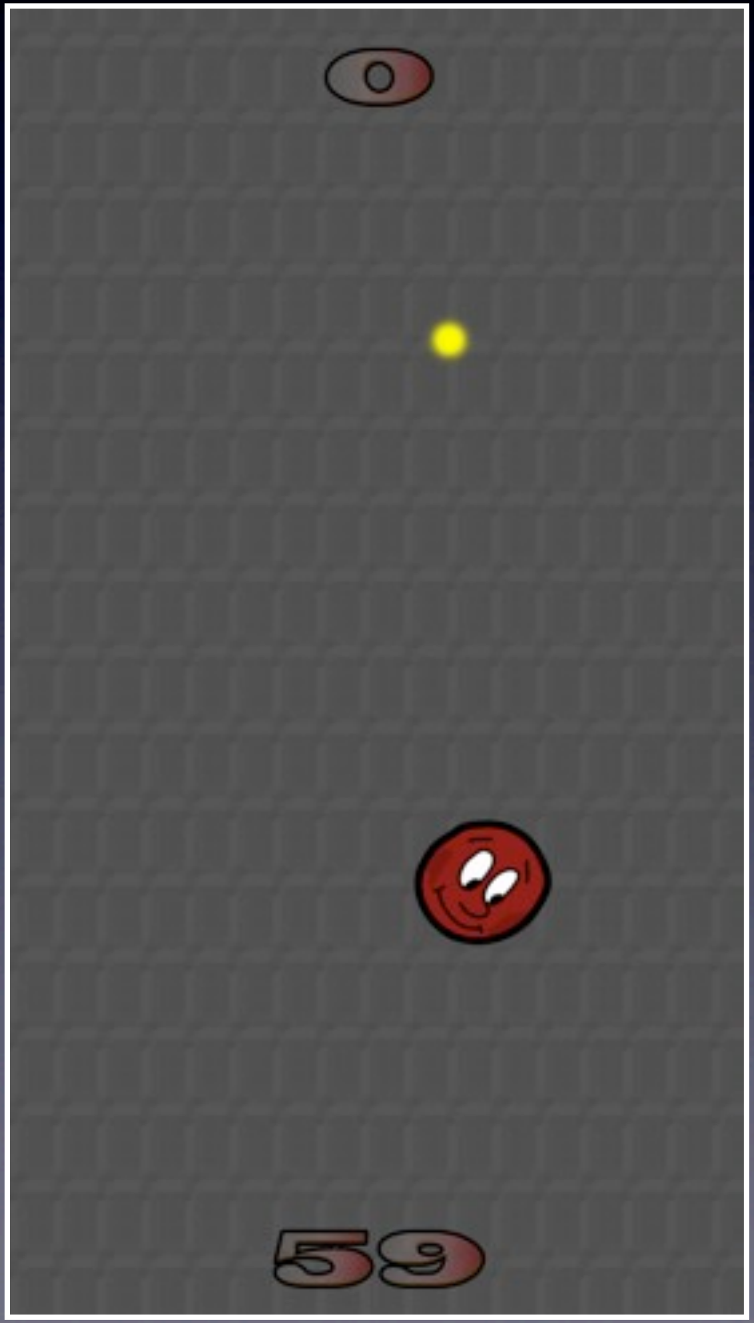
Touches (Kobold)

```
KKInput* input = [KKInput sharedInput];
if (input.anyTouchEndedThisFrame) {
  CCArray* touches = [KKInput sharedInput].touches;
  KKTouch* touch;
  CCARRAY_FOREACH(touches, touch) {
    if(touch && touch->isInvalid == NO) {
      CGPoint loc = [touch location];
      // DO SOMETHING WITH THIS LOCATION
    }
  }
}
```

Tiles Maps

```
<?xml version="1.0" encoding="UTF-8"?>
<map version="1.0" orientation="orthogonal" width="17" height="15" tilewidth="19"
tileheight="33">
 <tileset firstgid="1" name="hexabump" tilewidth="19" tileheight="33">
  <image source="hexabump.png" width="19" height="33"/>
 </tileset>
 <layer name="Tile Layer 1" width="17" height="15">
  <data encoding="base64" compression="zlib">
   eJxjZGBgYBzFo3gUjzgMAAIaAQA=
  </data>
 </layer>
</map>
```

Tile Map Result



Fonts

System Font

```
CCLabelTTF* ttf =
  [CCLabelTTF labelWithString:@"Text"
    fontName:@"Marker Felt" fontSize:30];
```

Bitmap Font

```
CCLabelBMFont *bm =
    [CCLabelBMFont labelWithString:labelString
      fntFile:@"font.fnt"];
```

Menus

```
CCMenuItemFont* loadNewSceneMenuItem =
  [CCMenuItemFont itemWithString:@"Go To Another Scene"
    target:self
    selector:@selector(handleGoToSecondScene)];

[loadNewSceneMenuItem setFontSize:20];
[loadNewSceneMenuItem setFontName:@"Marker Felt"];

CCMenu* newSceneMenu = [CCMenu loadNewSceneMenuItem, nil];
newSceneMenu = ccp(x, y);

[self addChild:goHomeMenu];
```

Show me the code!

Moving the camera

```
[self.camera setCenterX:x centerY:y centerZ:0];
[self.camera setEyeX:x eyeY:y eyeZ:[CCCamera getZEye]];
```

Show Me The Code!

Following Hoopy Ball

Physics

- Box2D
- Chipmunk

Box 2D

- Combined with Cocos2D it allows you to throw really mad birds at funny looking pigs
- Open source 2D physics engine written in C++

Box2D

- Create a physics world with properties
- Create bodies within that world
- Apply impulses to bodies
    - Don't move them, impulse them
    - Let the engine move them
    - If you really have to move them, you’re doing it wrong

Impulses

```
void ApplyForce(const b2Vec2& force, const b2Vec2& point);
void ApplyTorque(float32 torque);
void ApplyLinearImpulse(const b2Vec2& impulse, const b2Vec2& point);
void ApplyAngularImpulse(float32 impulse);
```

Box2D within Cocos2D

- Meters to Points for Sprite placement
- Inside of an update loop
  - Step the world
  - Query the bodies
    - Where are you?
    - What's your angle?
  - Move your sprites

Collision handling

- Physics world has one ContactListener
  - `BeginContact(b2Contact* contact)`
  - `EndContact(b2Contact* contact)`
- Each body has a void*
- See where I'm going with this?

Collision Handling

```
contactListener = new ContactListener();
world->SetContactListener(contactListener)

...

void ContactListener::BeginContact(b2Contact* contact)
{
   b2Body* bodyA = contact->GetFixtureA()->GetBody();
   b2Body* bodyB = contact->GetFixtureB()->GetBody();
   CollisionHandler* colA = (__bridge CollisionHandler*)bodyA->GetUserData();
   CollisionHandler* colB = (__bridge CollisionHandler*)bodyB->GetUserData();

   if(colA != nil) {
      [colA handleCollisionWith:colB];
   }
   if(colB != nil) {
      [colB handleCollisionWith:colA];
   }
}

void ContactListener::EndContact(b2Contact* contact)
{
}
```

Query and Move

```
world->Step(timeStep, velocityIterations, positionIterations);

for (b2Body* body = world->GetBodyList(); body != nil; body = body->GetNext()){
        CollisionHandler* handler = (__bridge CollisionHandler*)body->GetUserData();
          if(handler != NULL) {
                CCSprite* sprite = [handler sprite];
                if (sprite != NULL)
                {
                    // update the sprite's position to where their physics bodies are
                    sprite.position = [self toPixels:body->GetPosition()];
                    float angle = body->GetAngle();
                    sprite.rotation = CC_RADIANS_TO_DEGREES(angle) * -1;
                }
          }
   }
```

Hoopy

Be careful of your void pointers!

Because ARC won't

void* ARC Gotcha

```
-(void) createNewBody {

...

  b2BodyDef bodyDef;
  bodyDef.type = b2_staticBody;

  CollisionHandler* handler = [[CollisionHandler alloc]init];

  bodyDef.userData = (__bridge void*)handler;
  b2Body* body = world->CreateBody(&bodyDef);

...

}
```

void* ARC Gotcha

```
-(void) createNewBody {

...

  b2BodyDef bodyDef;
  bodyDef.type = b2_staticBody;

  CollisionHandler* handler = [[CollisionHandler alloc]init];

// trackingArray is an NSMutableArray referenced outside the method
  [trackingArray addObject:handler];

  bodyDef.userData = (__bridge void*)handler;
  b2Body* body = world->CreateBody(&bodyDef);


...

}
```

So Many Tools To Help You

- Glyph Designer
- TexturePacker
- PhysicsEditor
- Tiled

Other Stuff

- Audio
- Parallax Scrolling
- Sneaky Input
- Chipmunk
- Ads
- CocosBuilder