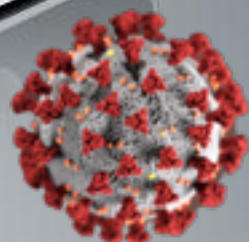


2007



2019



Programowanie dla iOS

Rozpoznawanie dotyku

Wielodotyk

Wielodotyky

- Ekran urządzenia iOS jest w stanie rozpoznać do 5 punktów dotyku

Wielodotyky

- Ekran urządzenia iOS jest w stanie rozpoznać do 5 punktów dotyku
- Zdarzenia są obsługiwane w kolejce

Wielodotyky

- Ekran urządzenia iOS jest w stanie rozpoznać do 5 punktów dotyku
- Zdarzenia są obsługiwane w kolejce
- Przekazywanie zdarzeń nie jest automatyczne

Metody obsługujące

Metody obsługujące

- touchesBegan
 - gdy palec dotknie ekranu
 - przekazuje informacje o każdym punkcie dotyku

Metody obsługujące

- touchesBegan
 - gdy palec dotknie ekranu
 - przekazuje informacje o każdym punkcie dotyku
- touchesMoved
 - gdy palec przesuwa się po ekranie

Metody obsługujące

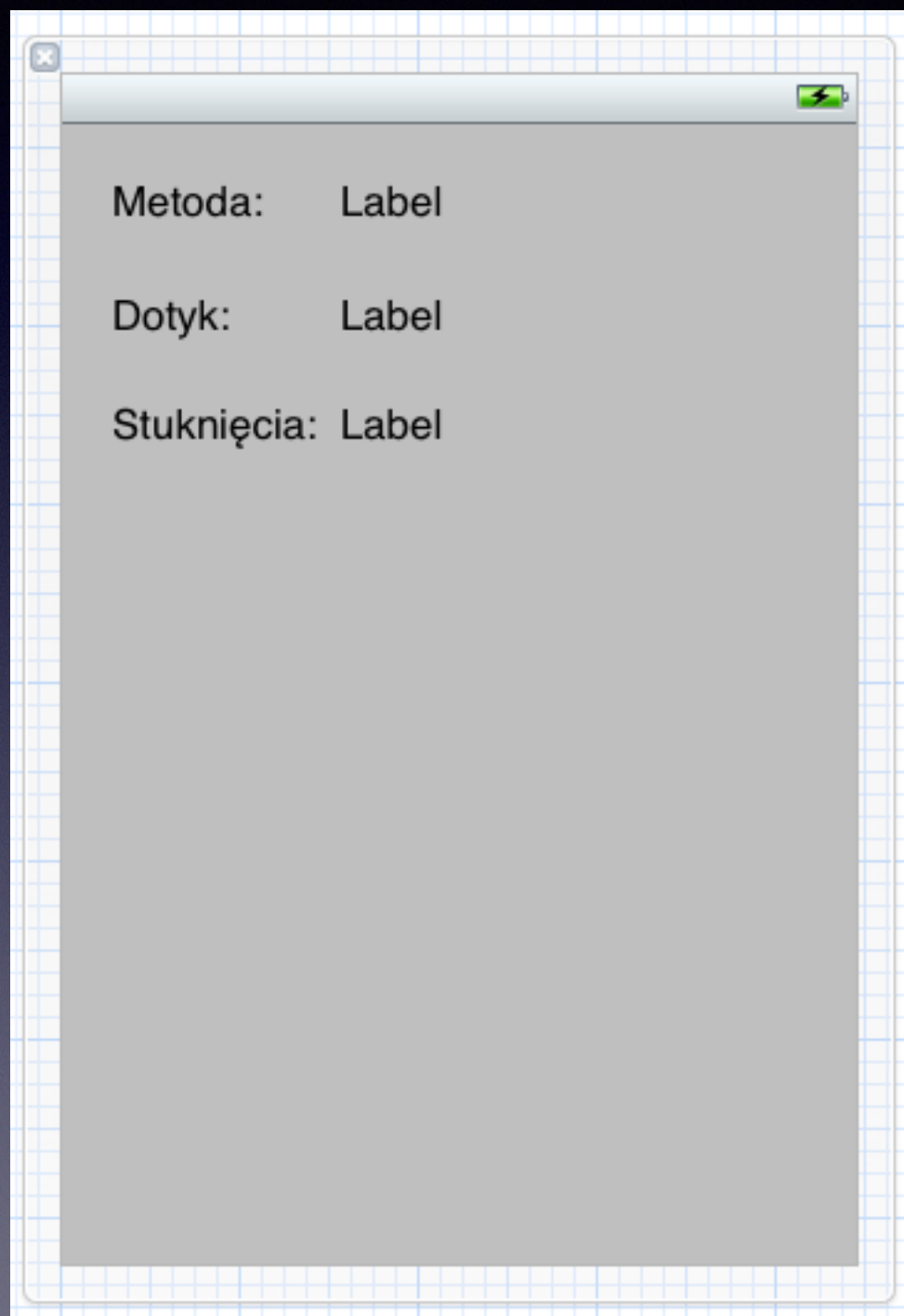
- touchesBegan
 - gdy palec dotknie ekranu
 - przekazuje informacje o każdym punkcie dotyku
- touchesMoved
 - gdy palec przesuwa się po ekranie
- touchesEnded
 - gdy palec się podniesie

Metody obsługujące

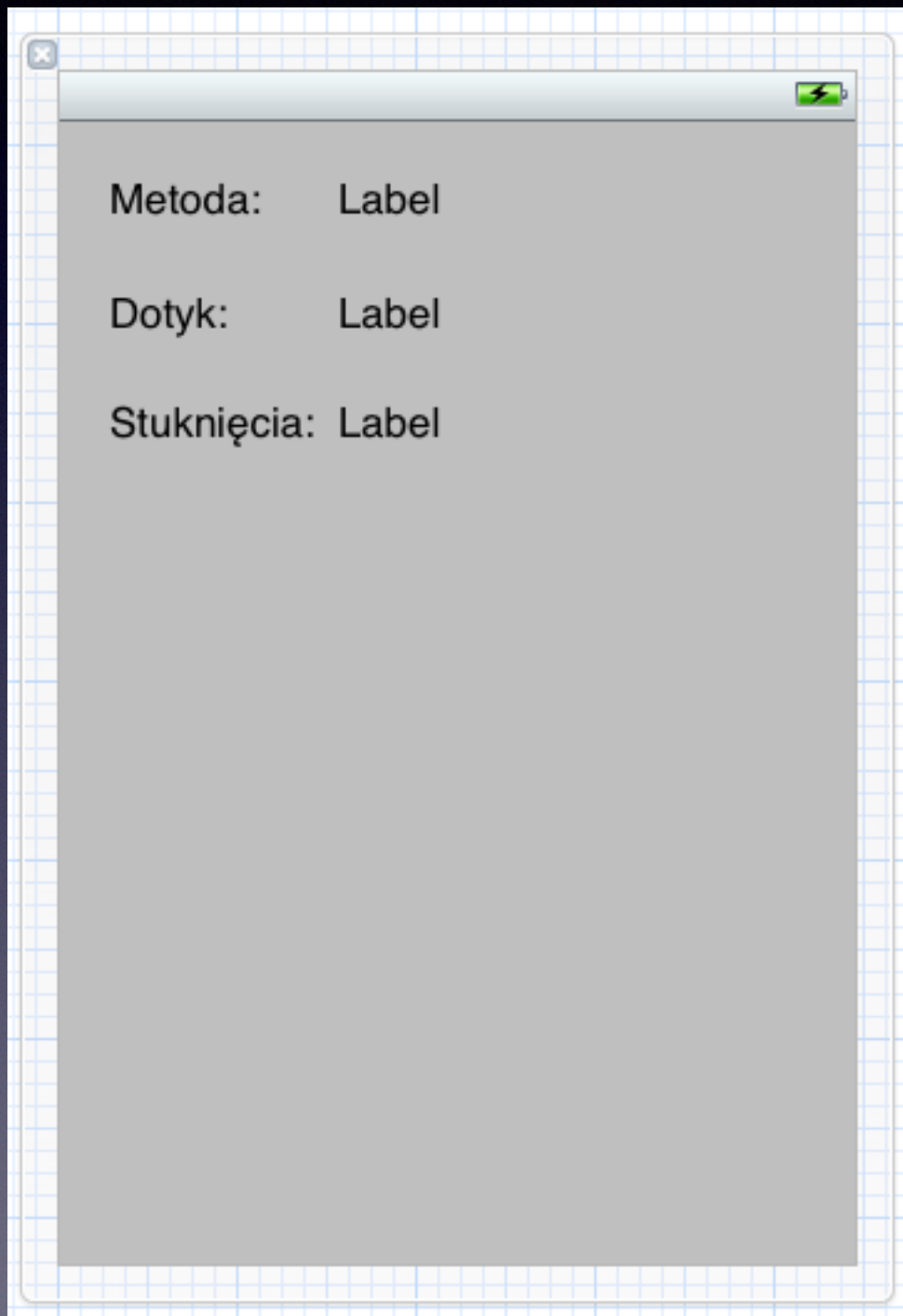
- touchesBegan
 - gdy palec dotknie ekranu
 - przekazuje informacje o każdym punkcie dotyku
- touchesMoved
 - gdy palec przesuwa się po ekranie
- touchesEnded
 - gdy palec się podniesie
- touchesCancelled
 - gdy nastąpi przerwanie, np. telefon

Przykład

Przykład



Przykład

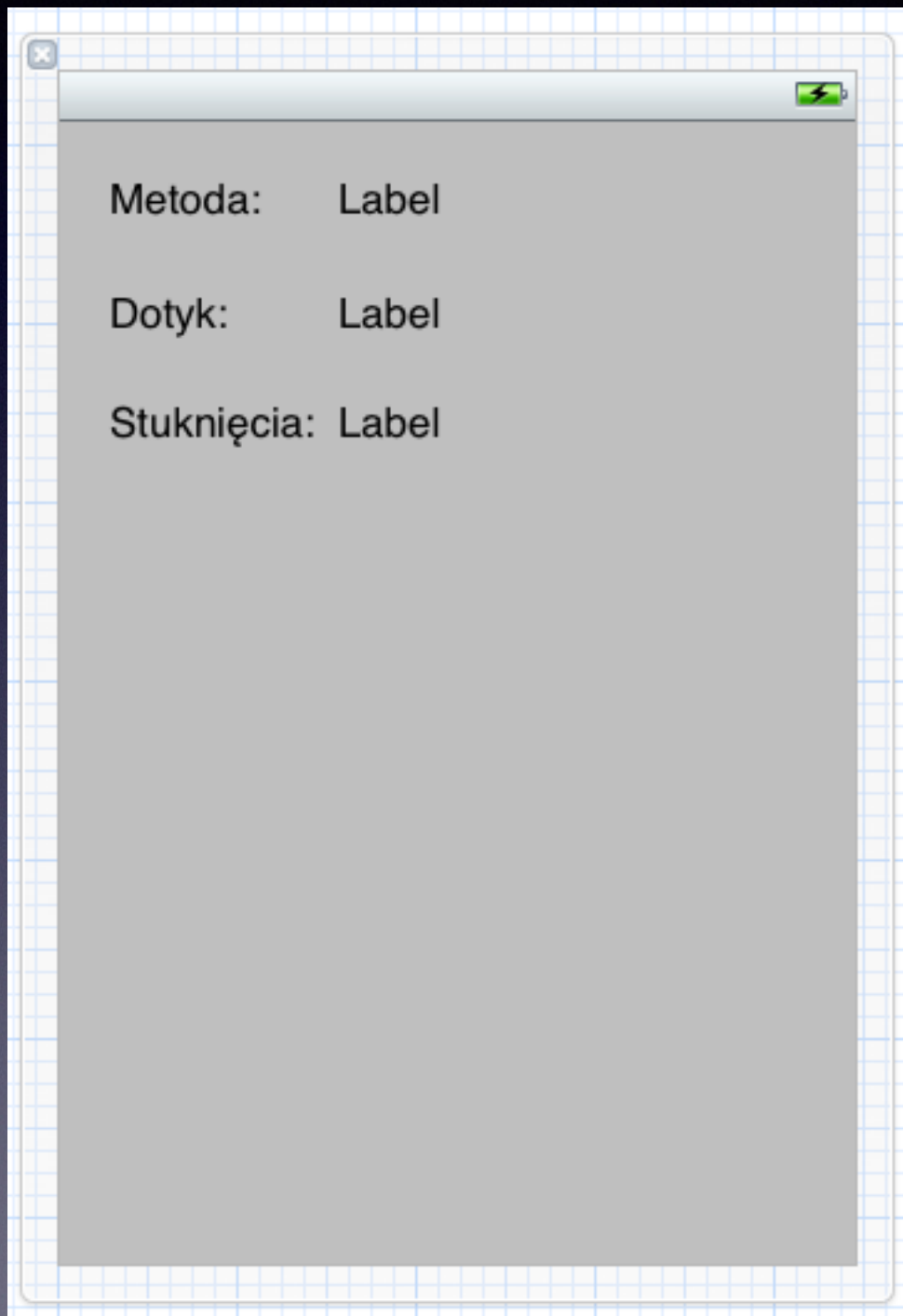


```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController {
    UILabel *stanDotyku;
    UILabel *stanMetody;
    UILabel *stanStukniecia;
}
@property (strong, nonatomic) IBOutlet UILabel *stanDotyku;
@property (strong, nonatomic) IBOutlet UILabel *stanMetody;
@property (strong, nonatomic) IBOutlet UILabel *stanStukniecia;

@end
```

Przykład



```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController {
    UILabel *stanDotyku;
    UILabel *stanMetody;
    UILabel *stanStukniecia;
}
@property (strong, nonatomic) IBOutlet UILabel *stanDotyku;
@property (strong, nonatomic) IBOutlet UILabel *stanMetody;
@property (strong, nonatomic) IBOutlet UILabel *stanStukniecia;

@end
```

```
#import "ViewController.h"

@implementation ViewController
@synthesize stanDotyku,stanMetody,stanStukniecia;
```

Przykład

Przykład

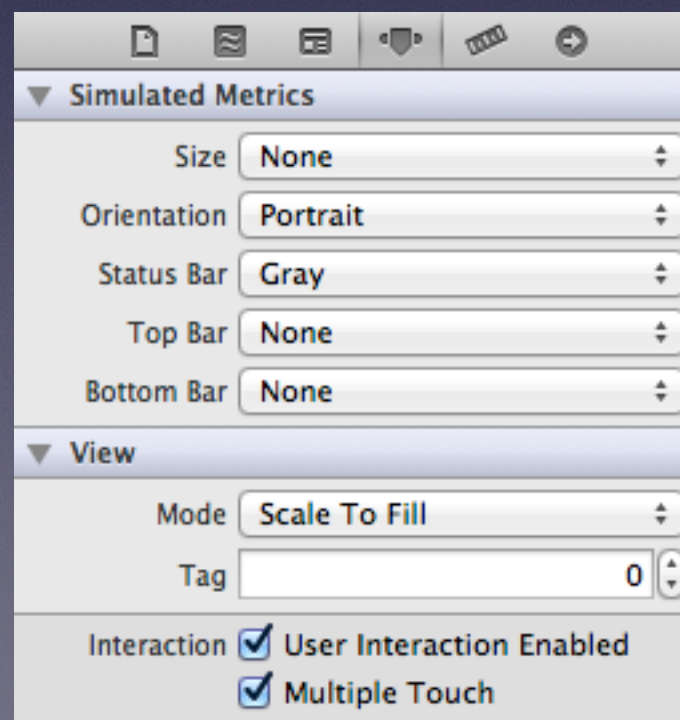
- Domyślnie widoki nie reagują na wielodotyk

Przykład

- Domyślnie widoki nie reagują na wielodotyk
- Zmieniamy opcję widoku

Przykład

- Domyślnie widoki nie reagują na wielodotyk
- Zmieniamy opcję widoku



Metoda touchesBegan

Metoda touchesBegan

```
- (void) touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event {
    NSUInteger touchCount = [touches count];
    NSUInteger tapCount = [[touches anyObject] tapCount];
    stanMetody.text = @"touchesBegan";
    stanDotyku.text = [NSString stringWithFormat:
        @"%d touches", touchCount];
    stanStukniecia.text = [NSString stringWithFormat:
        @"%d taps", tapCount];
}
```

touchesMoved

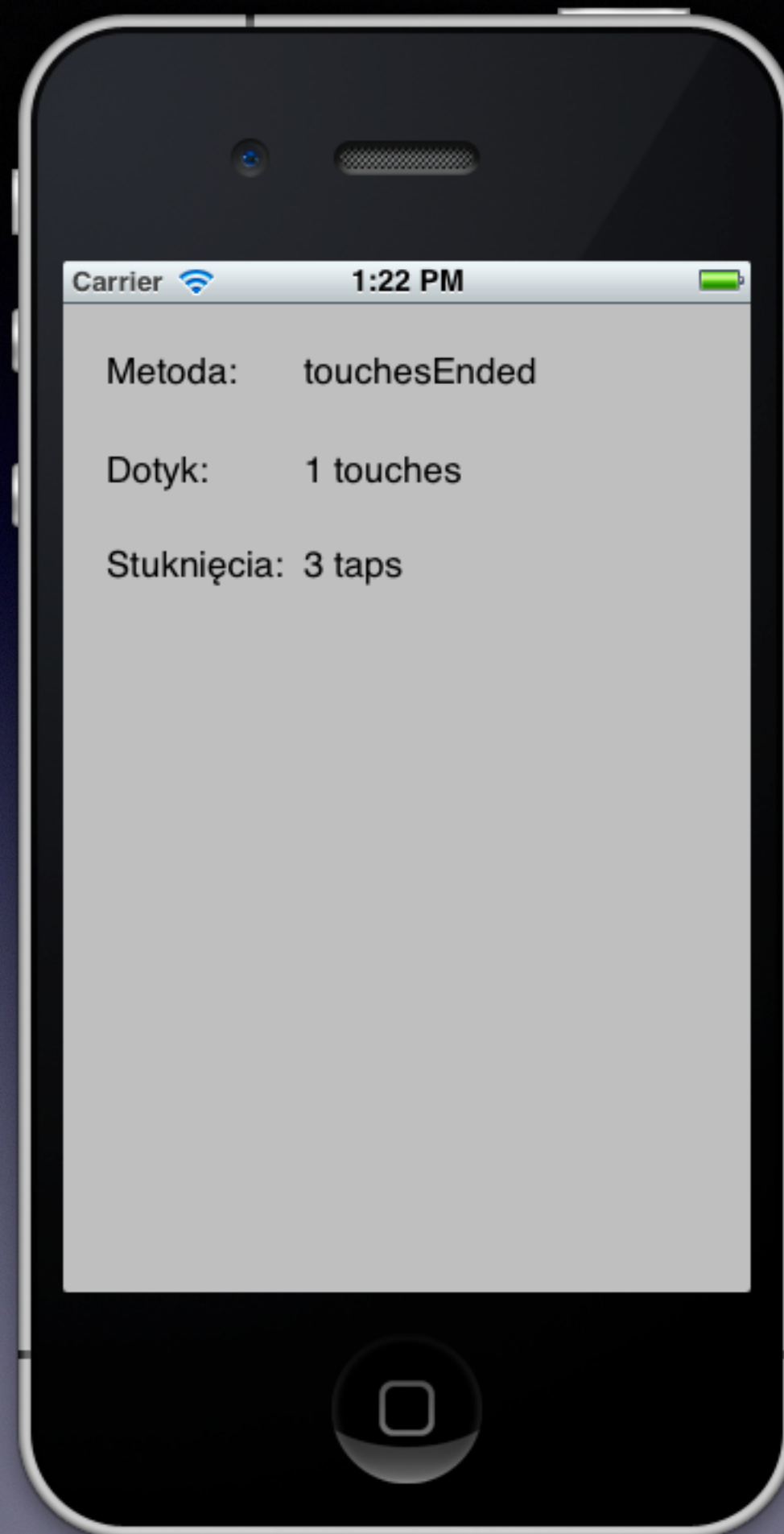
touchesMoved

```
- (void) touchesMoved:(NSSet *)touches
    withEvent:(UIEvent *)event {
    NSUInteger touchCount = [touches count];
    NSUInteger tapCount = [[touches anyObject] tapCount];
    stanMetody.text = @"touchesMoved";
    stanDotyku.text = [NSString stringWithFormat:
        @"%d touches", touchCount];
    stanStukniecia.text = [NSString stringWithFormat:
        @"%d taps", tapCount];
}
```

touchesEnded

touchesEnded

```
- (void) touchesEnded:(NSSet *)touches
    withEvent:(UIEvent *)event {
    NSUInteger touchCount = [touches count];
    NSUInteger tapCount = [[touches anyObject] tapCount];
    stanMetody.text = @"touchesEnded";
    stanDotyku.text = [NSString stringWithFormat:
        @"%d touches", touchCount];
    stanStukniecia.text = [NSString stringWithFormat:
        @"%d taps", tapCount];
}
```

Carrier



1:22 PM



Metoda: touchesEnded

Dotyk: 1 touches

Stuknięcia: 3 taps

Rozpoznawanie pozycji

Rozpoznawanie pozycji

```
@interface ViewController : UIViewController {
    UILabel *xCoord;
    UILabel *yCoord;
    CGPoint startPoint;
}
@property (strong, nonatomic) IBOutlet UILabel *xCoord;
@property (strong, nonatomic) IBOutlet UILabel *yCoord;
@property CGPoint startPoint;

@end
```

Rozpoznawanie pozycji

```
@interface ViewController : UIViewController {
    UILabel *xCoord;
    UILabel *yCoord;
    CGPoint startPoint;
}
@property (strong, nonatomic) IBOutlet UILabel *xCoord;
@property (strong, nonatomic) IBOutlet UILabel *yCoord;
@property CGPoint startPoint;

@end
```

```
@implementation ViewController
@synthesize xCoord, yCoord, startPoint;
```

Rozpoznawanie pozycji

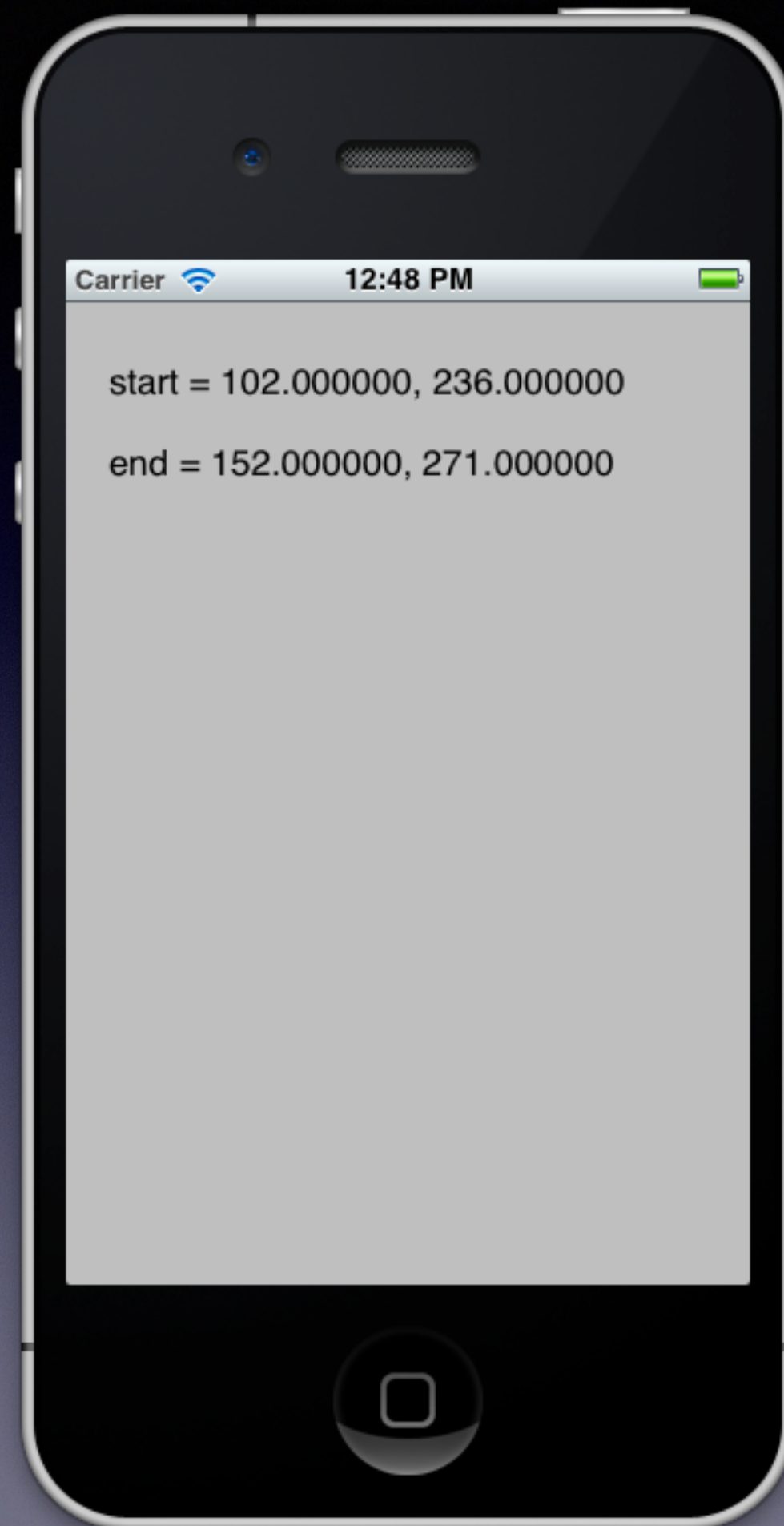
```
@interface ViewController : UIViewController {
    UILabel *xCoord;
    UILabel *yCoord;
    CGPoint startPoint;
}
@property (strong, nonatomic) IBOutlet UILabel *xCoord;
@property (strong, nonatomic) IBOutlet UILabel *yCoord;
@property CGPoint startPoint;

@end
```

```
@implementation ViewController
@synthesize xCoord, yCoord, startPoint;
```

```
- (void) touchesBegan:(NSSet *)touches
    withEvent:(UIEvent *)event {
    UITouch *theTouch = [touches anyObject];
    startPoint = [theTouch locationInView:self.view];
    CGFloat x = startPoint.x;
    CGFloat y = startPoint.y;
    xCoord.text = [NSString stringWithFormat:@"x = %f", x];
    yCoord.text = [NSString stringWithFormat:@"y = %f", y];
}

- (void) touchesEnded:(NSSet *)touches
    withEvent:(UIEvent *)event {
    UITouch *theTouch = [touches anyObject];
    CGPoint endPoint = [theTouch locationInView:self.view];
    xCoord.text = [NSString stringWithFormat:
        @"start = %f, %f", startPoint.x, startPoint.y];
    yCoord.text = [NSString stringWithFormat:
        @"end = %f, %f", endPoint.x, endPoint.y];
}
```

Gesty

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje
- `UITapGestureRecognizer`

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje
- `UITapGestureRecognizer`
- `UIPinchGestureRecognizer`

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje
- `UITapGestureRecognizer`
- `UIPinchGestureRecognizer`
- `UIPanGestureRecognizer`

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje
- `UITapGestureRecognizer`
- `UIPinchGestureRecognizer`
- `UIPanGestureRecognizer`
- `UISwipeGestureRecognizer`

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów UITapGestureRecognizer i jej specjalizacje
- UITapGestureRecognizer
- UIPinchGestureRecognizer
- UIPanGestureRecognizer
- UISwipeGestureRecognizer
- UIRotationGestureRecognizer

Gesty

- Od iOS 4 specjalna klasa do rozpoznawania gestów `UIGestureRecognizer` i jej specjalizacje
- `UITapGestureRecognizer`
- `UIPinchGestureRecognizer`
- `UIPanGestureRecognizer`
- `UISwipeGestureRecognizer`
- `UIRotationGestureRecognizer`
- `UILongPressGestureRecognizer`

Obsługa gestu

Obsługa gestu

- Definiujemy klasę do obsługi

Obsługa gestu

- Definiujemy klasę do obsługi

```
UITapGestureRecognizer *doubleTap =  
[[UITapGestureRecognizer alloc]  
 initWithTarget:self  
 action:@selector(tapDetected:)];  
doubleTap.numberOfTapsRequired = 2;  
[self.view addGestureRecognizer:doubleTap];
```

Obsługa gestu

- Definiujemy klasę do obsługi
- Przekazujemy jej referencję do metody, która ma być wywołana po wykryciu gestu

```
UITapGestureRecognizer *doubleTap =  
[[UITapGestureRecognizer alloc]  
 initWithTarget:self  
 action:@selector(tapDetected:)];  
doubleTap.numberOfTapsRequired = 2;  
[self.view addGestureRecognizer:doubleTap];
```

Obsługa gestu

- Definiujemy klasę do obsługi
- Przekazujemy jej referencję do metody, która ma być wywołana po wykryciu gestu

```
UITapGestureRecognizer *doubleTap =  
[[UITapGestureRecognizer alloc]  
 initWithTarget:self  
 action:@selector(tapDetected:)];  
doubleTap.numberOfTapsRequired = 2;  
[self.view addGestureRecognizer:doubleTap];
```

```
- (IBAction)tapDetected:(UITapGestureRecognizer *)sender {  
    // Code to respond to gesture here  
}
```

Obsługa gestu

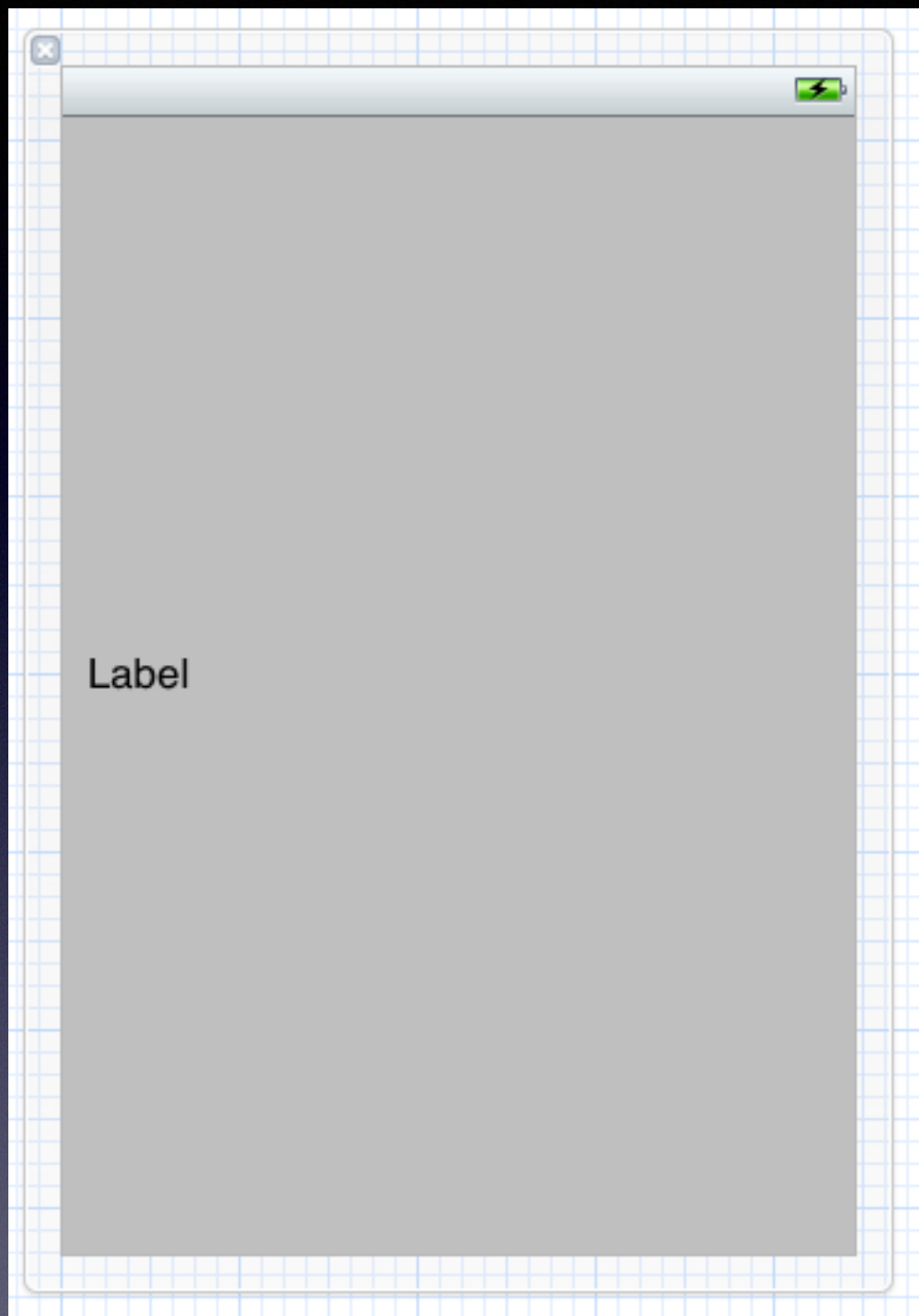
- Definiujemy klasę do obsługi
- Przekazujemy jej referencję do metody, która ma być wywołana po wykryciu gestu
- Niektóre gesty są pojedyncze (tap) inne ciągłe (swipe, pan)

```
UITapGestureRecognizer *doubleTap =  
[[UITapGestureRecognizer alloc]  
 initWithTarget:self  
 action:@selector(tapDetected:)];  
doubleTap.numberOfTapsRequired = 2;  
[self.view addGestureRecognizer:doubleTap];
```

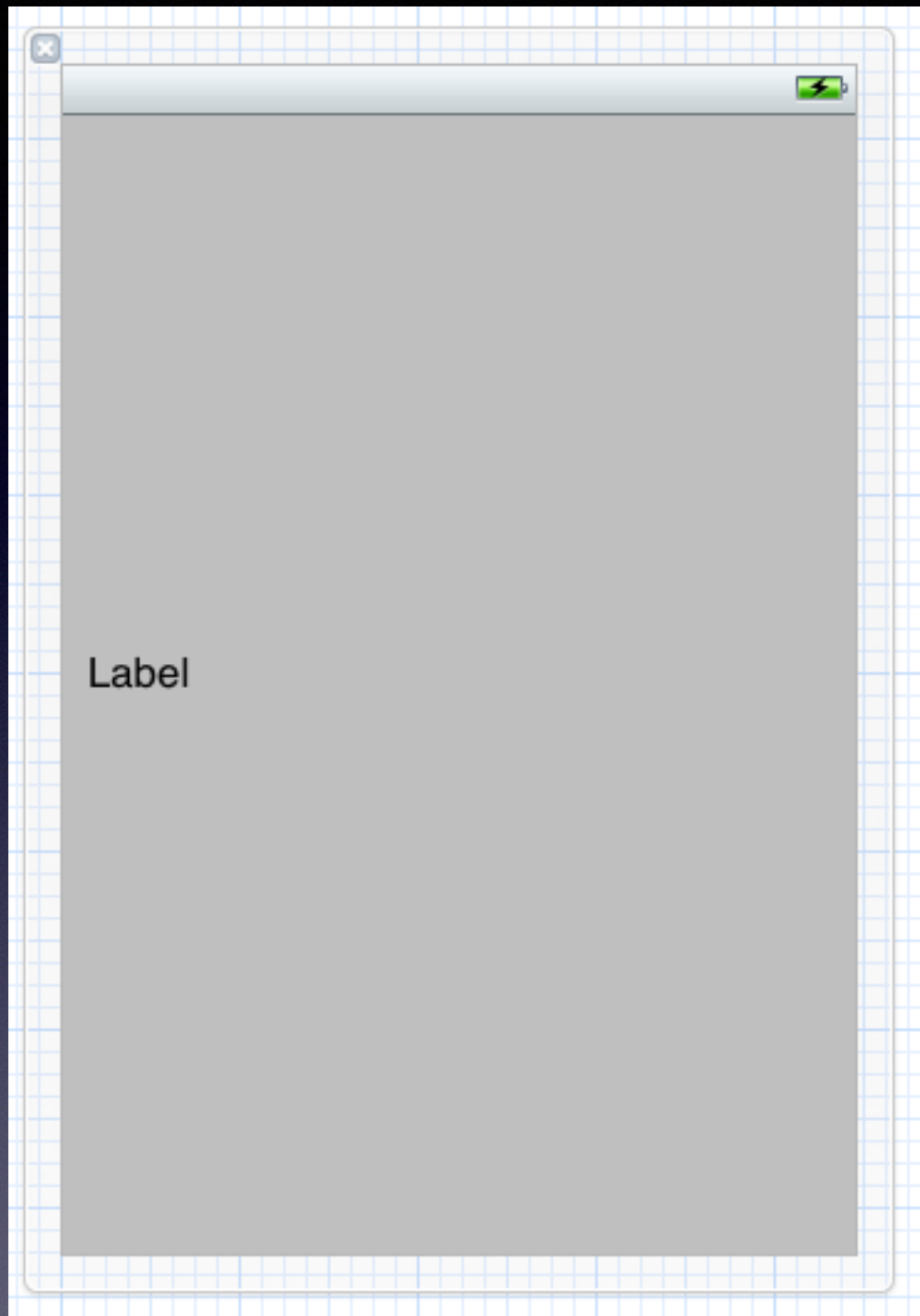
```
- (IBAction)tapDetected:(UITapGestureRecognizer *)sender {  
    // Code to respond to gesture here  
}
```

Przykład

Przykład



Przykład



```
#import <UIKit/UIKit.h>

@interface recognizerViewController : UIViewController {
    UILabel *statusLabel;
}
@property (strong, nonatomic) IBOutlet UILabel *statusLabel;

@end
```

Przykład



```
#import <UIKit/UIKit.h>

@interface recognizerViewController : UIViewController {
    UILabel *statusLabel;
}
@property (strong, nonatomic) IBOutlet UILabel *statusLabel;

@end
```

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    UITapGestureRecognizer *doubleTap =
    [[UITapGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(tapDetected:)];
    doubleTap.numberOfTapsRequired = 2;
    [self.view addGestureRecognizer:doubleTap];

    UIPinchGestureRecognizer *pinchRecognizer =
    [[UIPinchGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(pinchDetected:)];
    [self.view addGestureRecognizer:pinchRecognizer];

    UIRotationGestureRecognizer *rotationRecognizer =
    [[UIRotationGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(rotationDetected:)];
    [self.view addGestureRecognizer:rotationRecognizer];

    UISwipeGestureRecognizer *swipeRecognizer =
    [[UISwipeGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(swipeDetected:)];
    swipeRecognizer.direction = UISwipeGestureRecognizerDirectionRight;
    [self.view addGestureRecognizer:swipeRecognizer];

    UILongPressGestureRecognizer *longPressRecognizer =
    [[UILongPressGestureRecognizer alloc]
     initWithTarget:self
     action:@selector(longPressDetected:)];
    longPressRecognizer.minimumPressDuration = 3;
    longPressRecognizer.numberOfTouchesRequired = 1;
    [self.view addGestureRecognizer:longPressRecognizer];
}
```

Przykład

Przykład

```
- (IBAction)longPressDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Long Press";
}

- (IBAction)swipeDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Right Swipe";
}

- (IBAction)tapDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Double Tap";
}

- (IBAction)pinchDetected:(UIGestureRecognizer *)sender {

    CGFloat scale =
    [[UIPinchGestureRecognizer *)sender scale];
    CGFloat velocity =
    [[UIPinchGestureRecognizer *)sender velocity];

    NSString *resultString = [[NSString alloc] initWithFormat:
        @"Pinch - scale = %f, velocity = %f",
        scale, velocity];
    statusLabel.text = resultString;
}

- (IBAction)rotationDetected:(UIGestureRecognizer *)sender {
    CGFloat radians =
    [[UIRotationGestureRecognizer *)sender rotation];
    CGFloat velocity =
    [[UIRotationGestureRecognizer *)sender velocity];

    NSString *resultString = [[NSString alloc] initWithFormat:
        @"Rotation - Radians = %f, velocity = %f",
        radians, velocity];
    statusLabel.text = resultString;
}
```

Przykład

```
- (IBAction)longPressDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Long Press";
}

- (IBAction)swipeDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Right Swipe";
}

- (IBAction)tapDetected:(UIGestureRecognizer *)sender {
    statusLabel.text = @"Double Tap";
}

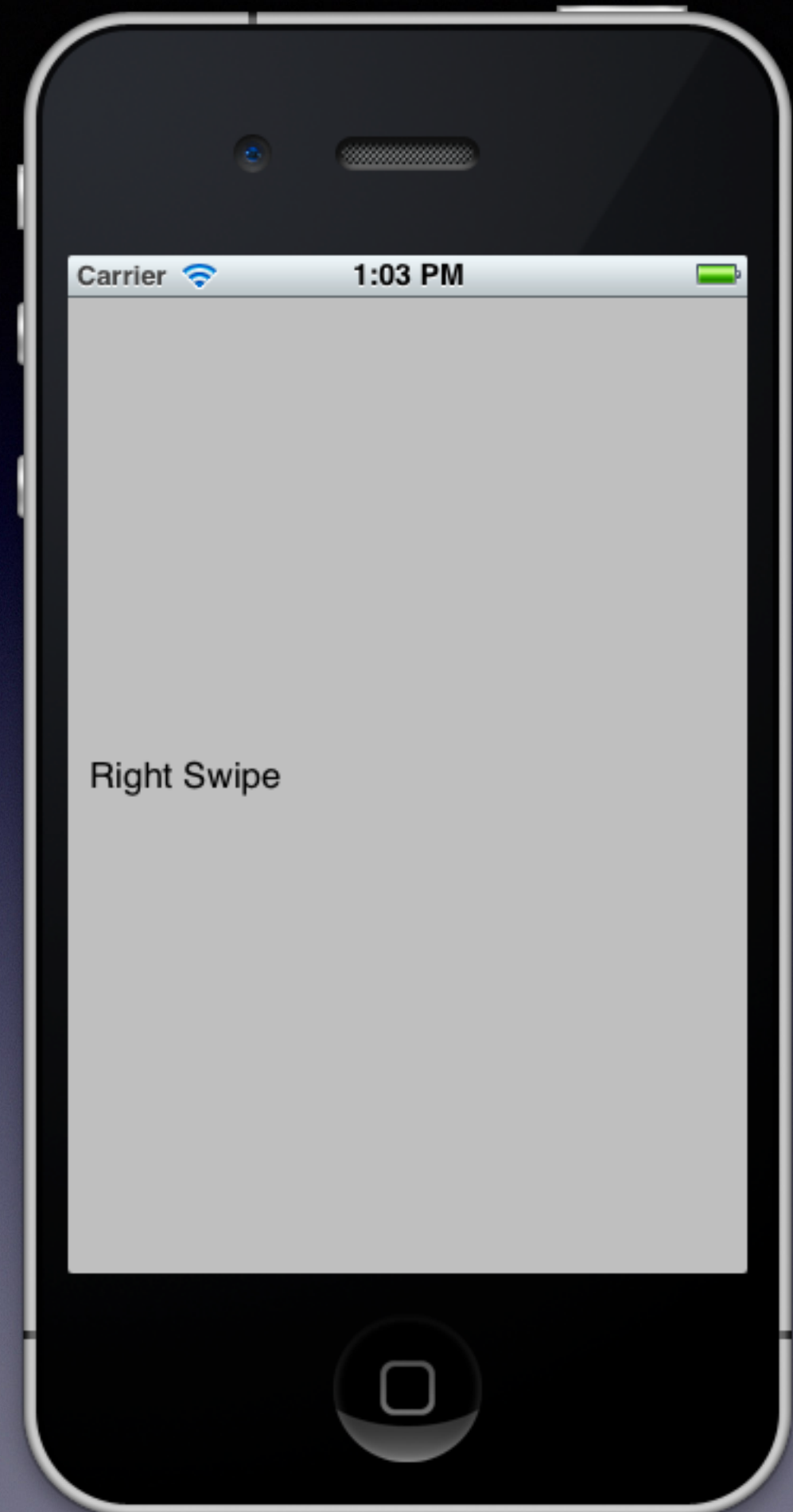
- (IBAction)pinchDetected:(UIGestureRecognizer *)sender {

    CGFloat scale =
    [[UIPinchGestureRecognizer *)sender scale];
    CGFloat velocity =
    [[UIPinchGestureRecognizer *)sender velocity];

    NSString *resultString = [[NSString alloc] initWithFormat:
        @"Pinch - scale = %f, velocity = %f",
        scale, velocity];
    statusLabel.text = resultString;
}

- (IBAction)rotationDetected:(UIGestureRecognizer *)sender {
    CGFloat radians =
    [[UIRotationGestureRecognizer *)sender rotation];
    CGFloat velocity =
    [[UIRotationGestureRecognizer *)sender velocity];

    NSString *resultString = [[NSString alloc] initWithFormat:
        @"Rotation - Radians = %f, velocity = %f",
        radians, velocity];
    statusLabel.text = resultString;
}
```



SpriteKit

SpriteKit

SpriteKit

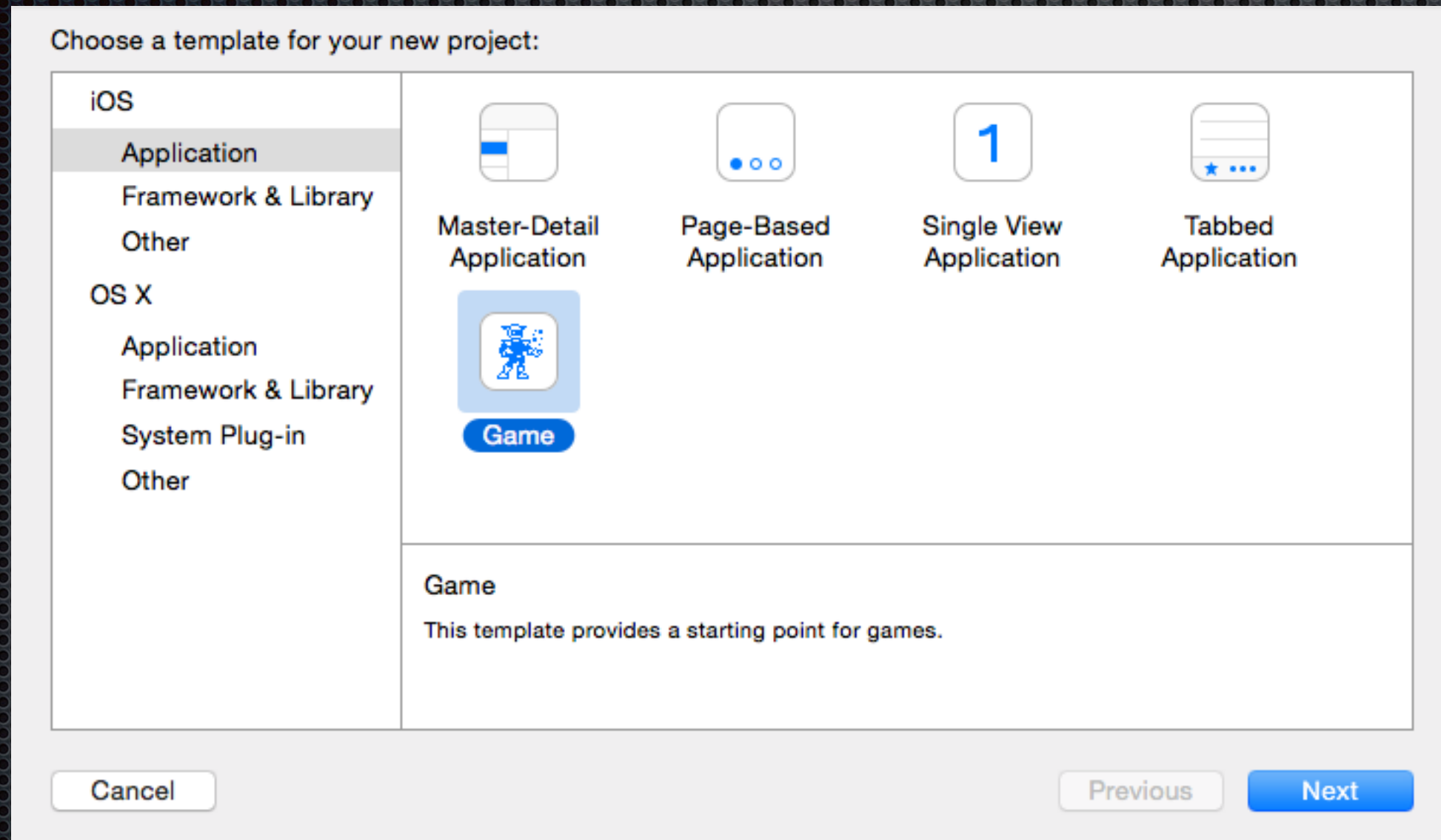
- Biblioteka do tworzenia gier wbudowana w SDK

SpriteKit

- ✦ Biblioteka do tworzenia gier wbudowana w SDK
- ✦ Wspiera grafikę 2D w oparciu o sprite'y

SpriteKit HelloWorld

SpriteKit HelloWorld

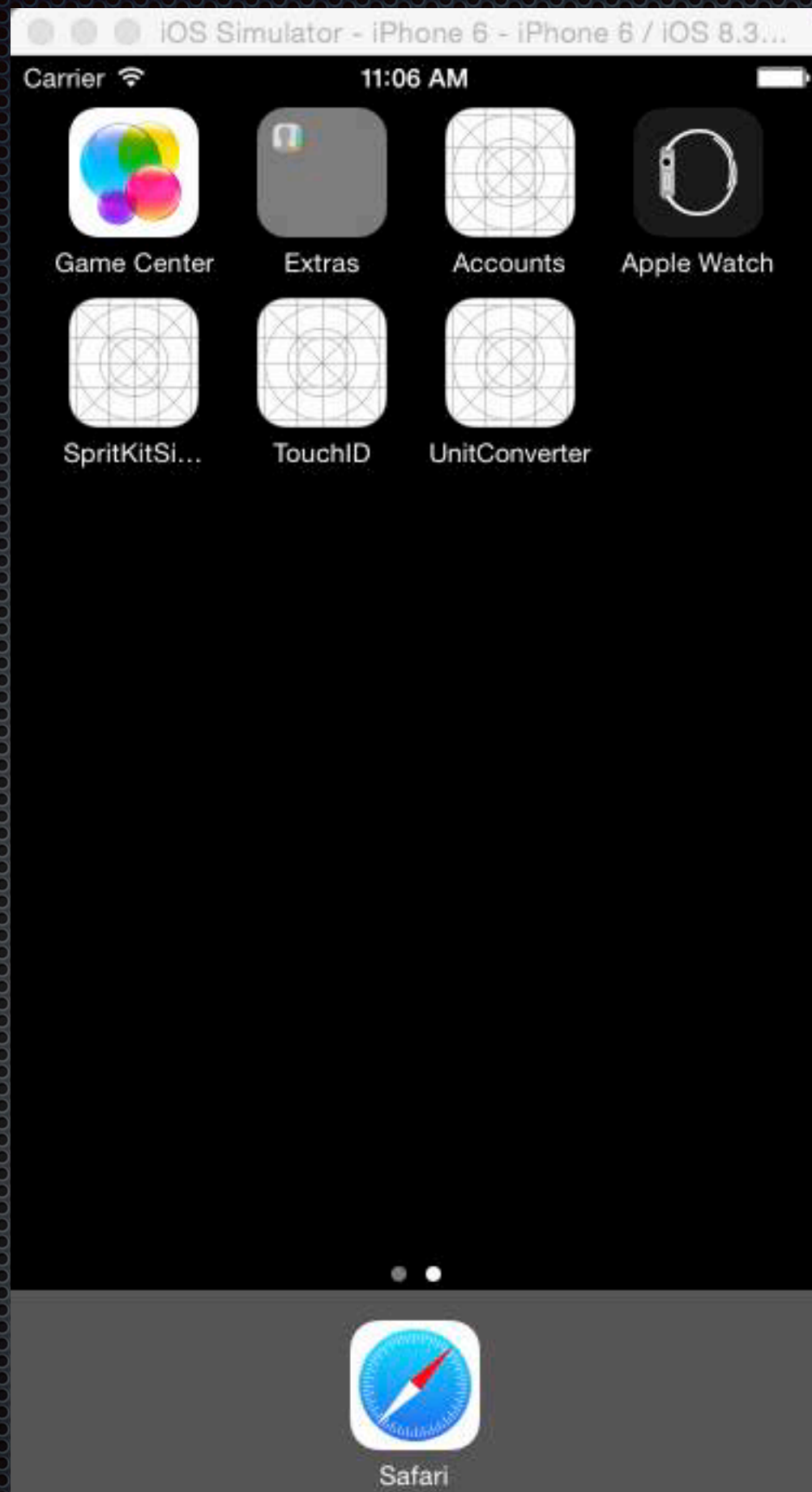


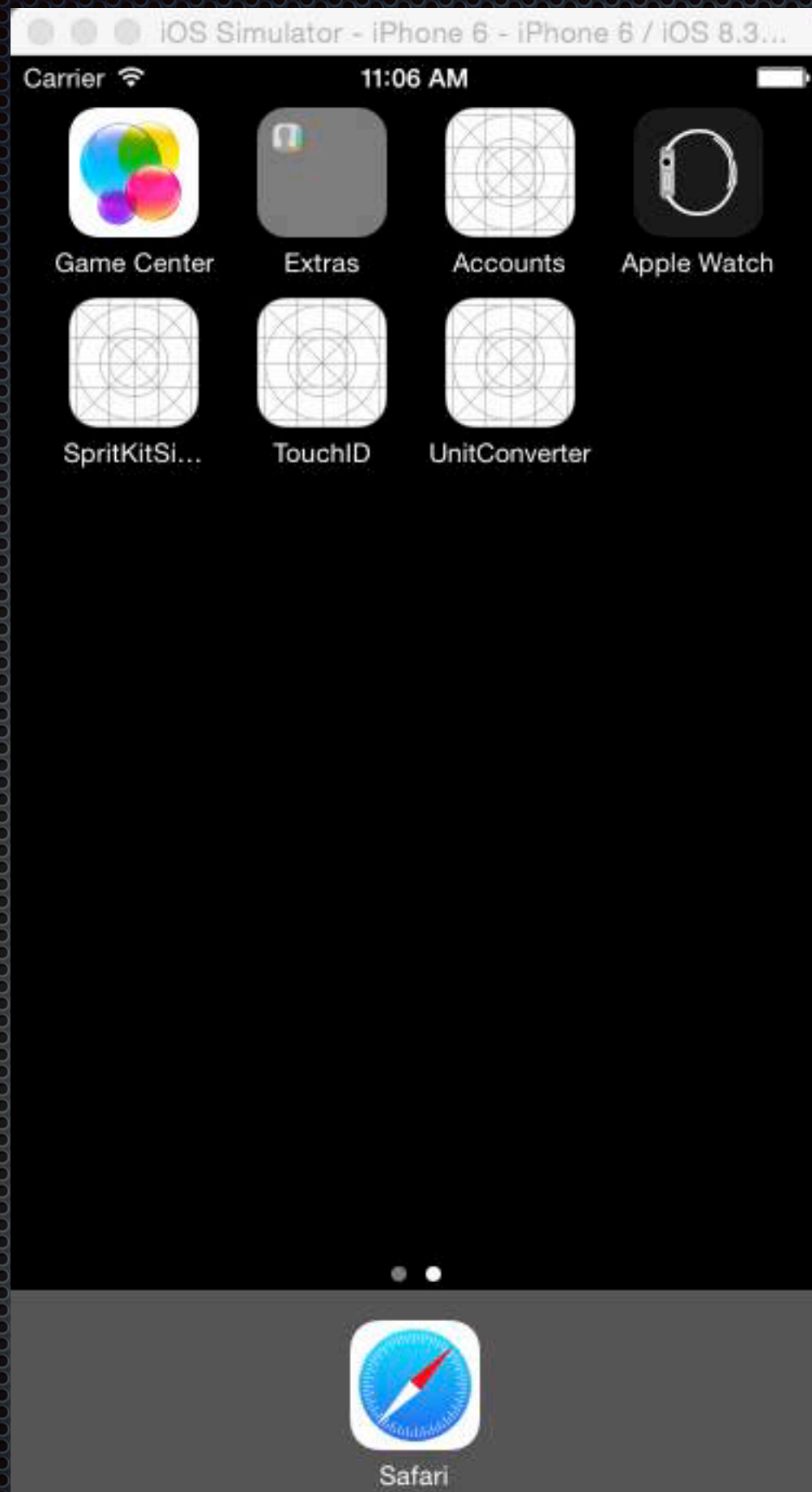
SpriteKit HelloWorld

SpriteKit HelloWorld

Choose options for your new project:

Product Name:	SpritKitSimpleGame
Organization Name:	Bartłomiej Prędko
Organization Identifier:	put
Bundle Identifier:	put.SpritKitSimpleGame
Language:	Swift
Game Technology:	SpriteKit
Devices:	iPhone

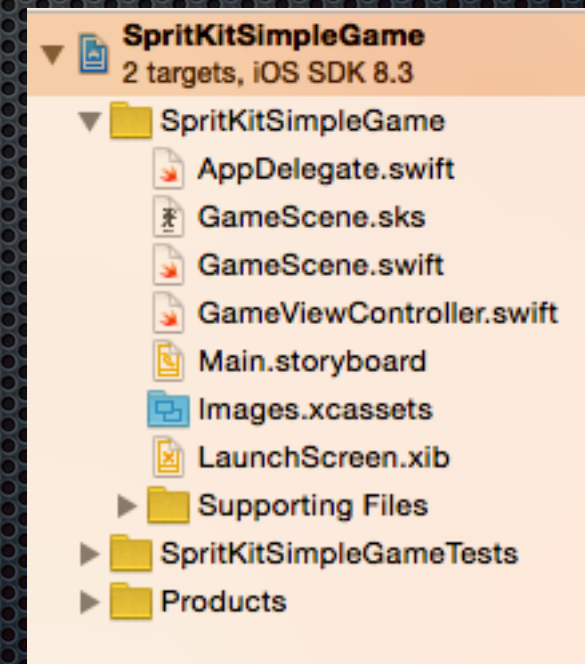




SpriteKit

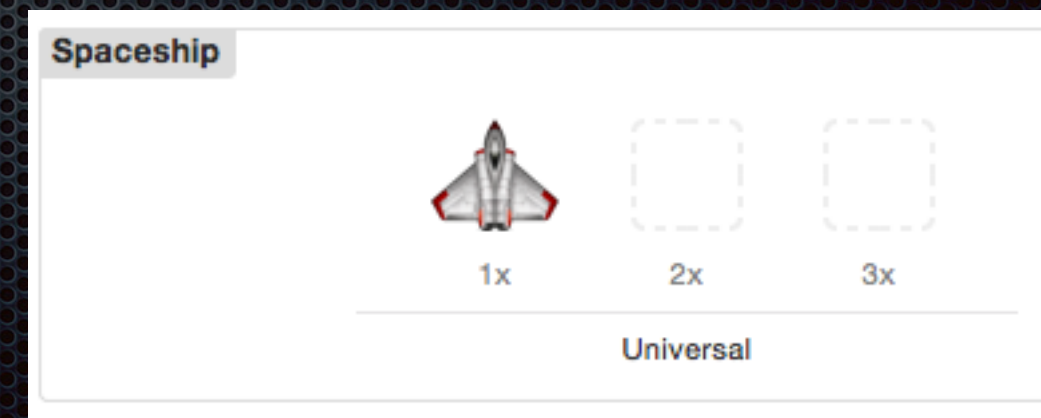
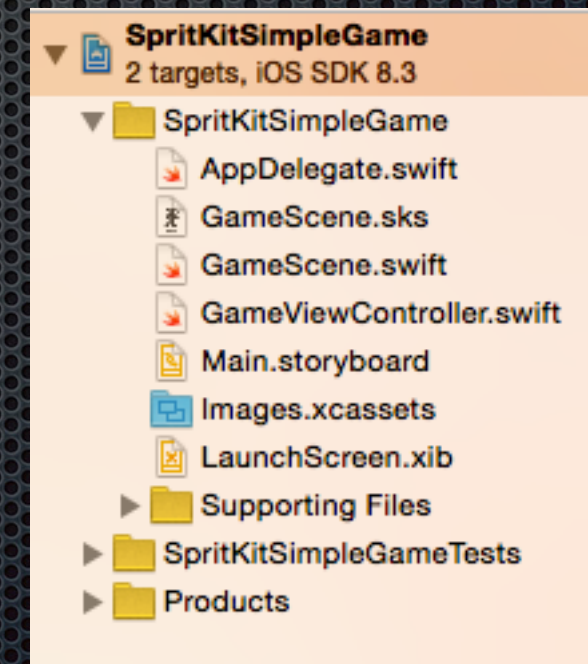
SpriteKit

- AppDelegate identyczny jak w innych szablonach
- Korzysta z mechanizmu scen (poziomy/ekrany)
 - scena ma swoją klasę GameScene.swift
 - i plik z projektem GameScene.sks
- kontroler klasy widoku ładujący scenę
GameViewController.swift
- scenorys z projektem UI
- sprite znajduje się w Images.xcassets



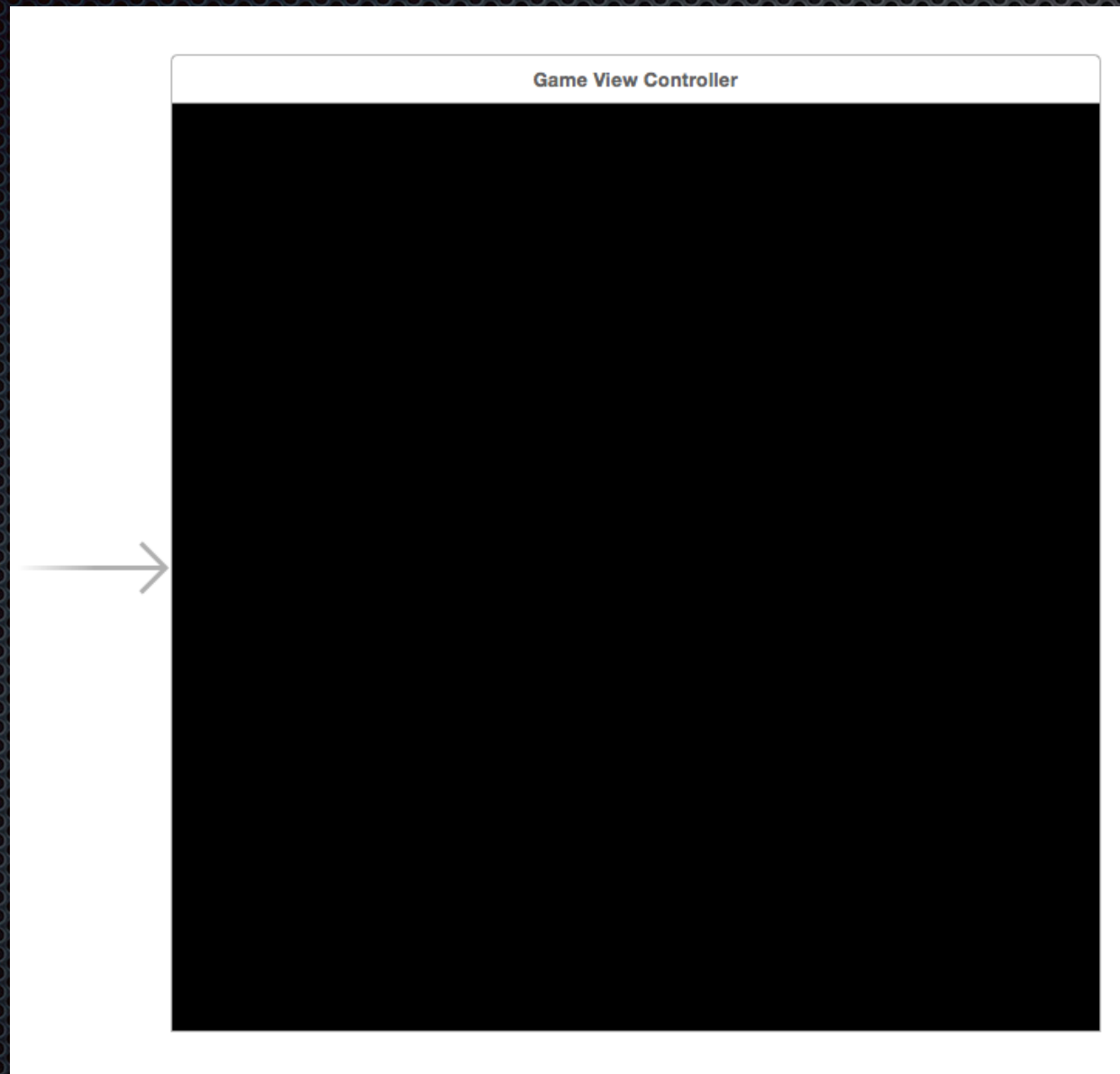
SpriteKit

- AppDelegate identyczny jak w innych szablonach
- Korzysta z mechanizmu scen (poziomy/ekrany)
 - scena ma swoją klasę GameScene.swift
 - i plik z projektem GameScene.sks
- kontroler klasy widoku ładujący scenę
GameViewController.swift
- scenorys z projektem UI
- sprite znajduje się w Images.xcassets



SpriteKit - scenorys

SpriteKit - scenorys

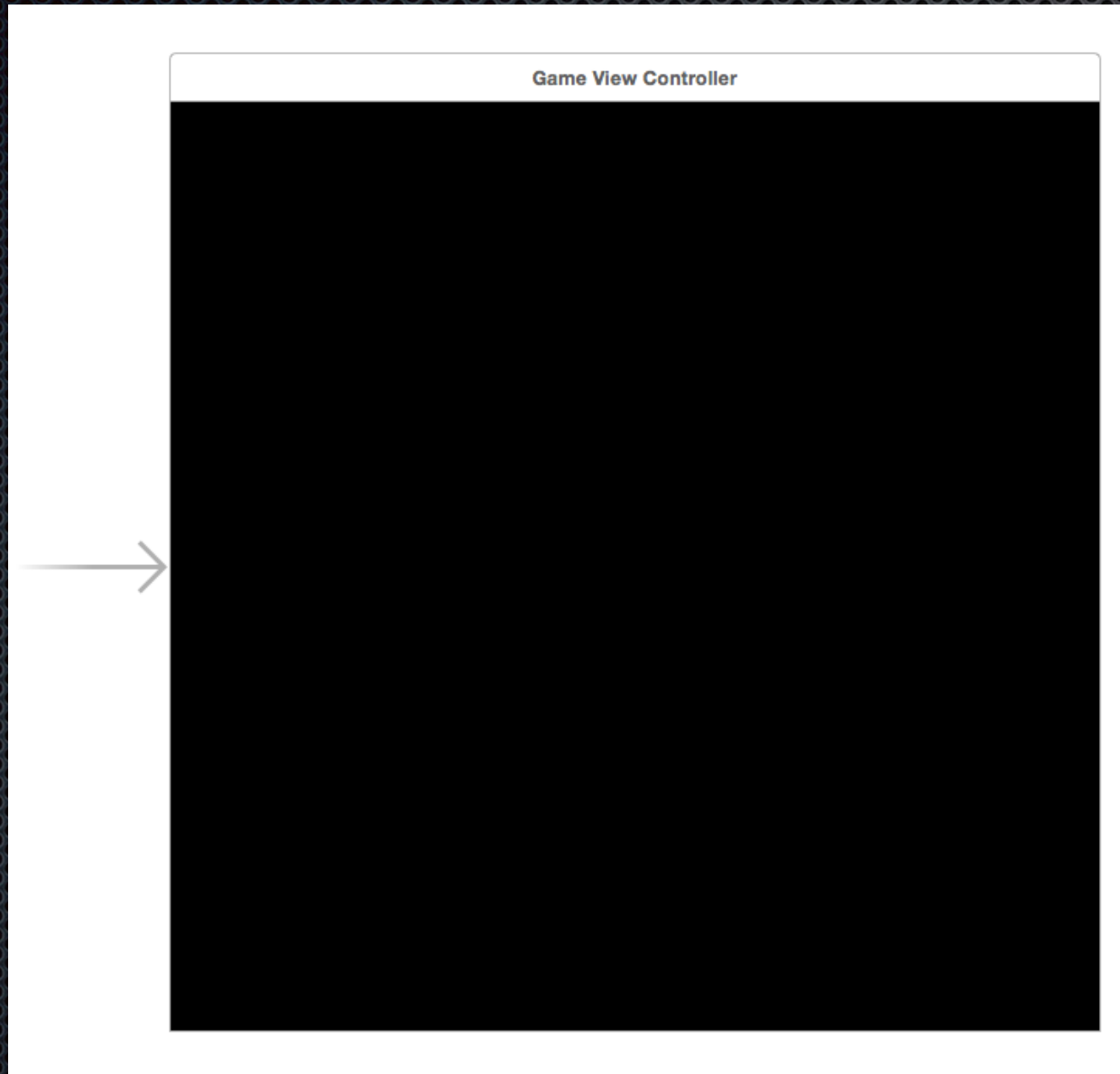


SpriteKit - scenorys

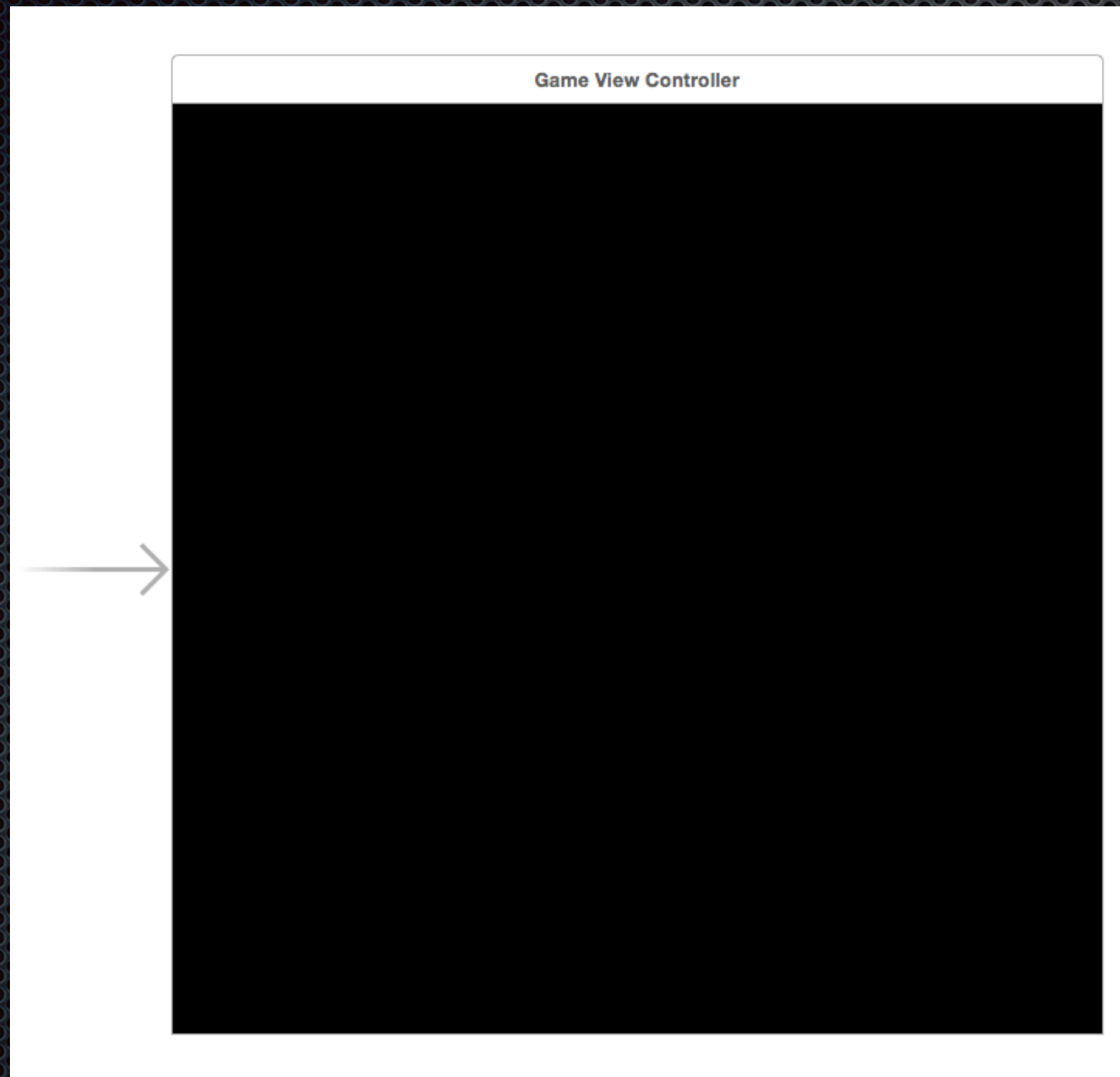
Custom Class

Class 

Module 



SpriteKit - scenorys



Custom Class

Class

Module

Simulated Metrics

Size

Orientation

Status Bar

Top Bar

Bottom Bar

View Controller

Title

☒ Is Initial View Controller

Layout ☒ Adjust Scroll View Insets

☐ Hide Bottom Bar on Push

☒ Resize View From NIB

☐ Use Full Screen (Deprecated)

Extend Edges ☒ Under Top Bars

☒ Under Bottom Bars

☐ Under Opaque Bars

Transition Style

Presentation

☐ Defines Context

☐ Provides Context

Content Size ☐ Use Preferred Explicit Size

Width Height

SpriteKit - kontroler widoku

SpriteKit - kontroler widoku

```
import UIKit
import SpriteKit

extension SKNode {
    class func unarchiveFromFile(file : String) -> SKNode? {
        if let path = NSBundle.mainBundle().pathForResource(file, ofType: "sks") {
            var sceneData = NSData(contentsOfFile: path, options: .DataReadingMappedIfSafe, error: nil)!
            var archiver = NSKeyedUnarchiver(forReadingWithData: sceneData)

            archiver.setClass(self.classForKeyedUnarchiver(), forClassName: "SKScene")
            let scene = archiver.decodeObjectForKey(NSKeyedArchiveRootObjectKey) as! GameScene
            archiver.finishDecoding()
            return scene
        } else {
            return nil
        }
    }
}

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        if let scene = GameScene.unarchiveFromFile("GameScene") as? GameScene {
            // Configure the view.
            let skView = self.view as! SKView
            skView.showsFPS = true
            skView.showsNodeCount = true

            /* Sprite Kit applies additional optimizations to improve rendering performance */
            skView.ignoresSiblingOrder = true

            /* Set the scale mode to scale to fit the window */
            scene.scaleMode = .AspectFill

            skView.presentScene(scene)
        }
    }
}
```

SpriteKit - kontroler widoku

SpriteKit - kontroler widoku

```
override func shouldAutorotate() -> Bool {
    return true
}

override func supportedInterfaceOrientations() -> Int {
    if UIDevice.currentDevice().userInterfaceIdiom == .Phone {
        return Int(UIInterfaceOrientationMask.AllButUpsideDown.rawValue)
    } else {
        return Int(UIInterfaceOrientationMask.All.rawValue)
    }
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Release any cached data, images, etc that aren't in use.
}

override func prefersStatusBarHidden() -> Bool {
    return true
}
}
```

SpriteKit - scena

SpriteKit - scena

```
import SpriteKit

class GameScene: SKScene {
    override func didMoveToView(view: SKView) {
        /* Setup your scene here */
        let myLabel = SKLabelNode(fontNamed:"Chalkduster")
        myLabel.text = "Hello, World!";
        myLabel.fontSize = 65;
        myLabel.position = CGPoint(x:CGRectGetMidX(self.frame), y:CGRectGetMidY(self.frame));

        self.addChild(myLabel)
    }

    override func touchesBegan(touches: Set<NSObject>, withEvent event: UIEvent) {
        /* Called when a touch begins */

        for touch in (touches as! Set<UITouch>) {
            let location = touch.locationInNode(self)

            let sprite = SKSpriteNode(imageNamed:"Spaceship")

            sprite.xScale = 0.5
            sprite.yScale = 0.5
            sprite.position = location

            let action = SKAction.rotateByAngle(CGFloat(M_PI), duration:1)

            sprite.runAction(SKAction.repeatActionForever(action))

            self.addChild(sprite)
        }
    }

    override func update(currentTime: CTimeInterval) {
        /* Called before each frame is rendered */
    }
}
```

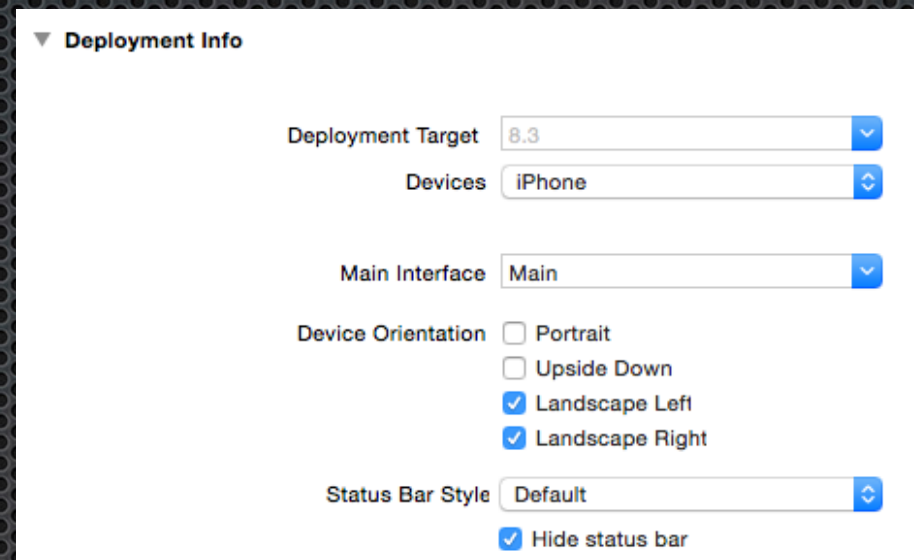
Przerabiamy szablon

Przerabiamy szablon

- Zmieniamy orientację ekranu na poziomą

Przerabiamy szablon

- ✦ Zmieniamy orientację ekranu na poziomą

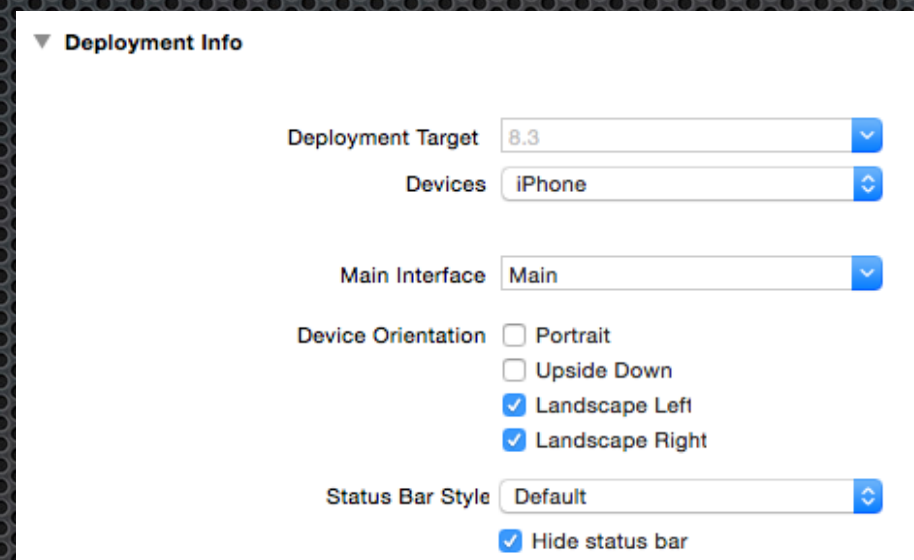


The screenshot shows the 'Deployment Info' panel in Xcode. It contains the following settings:

- Deployment Target:** 8.3
- Devices:** iPhone
- Main Interface:** Main
- Device Orientation:**
 - ☐ Portrait
 - ☐ Upside Down
 - ☒ Landscape Left
 - ☒ Landscape Right
- Status Bar Style:** Default
- ☒ Hide status bar

Przerabiamy szablon

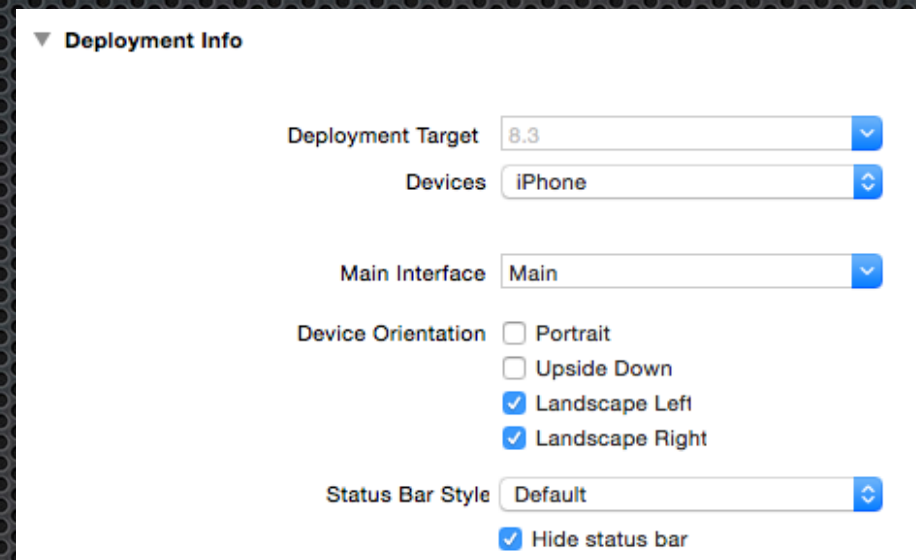
- ✦ Zmieniamy orientację ekranu na poziomą



- ✦ Wyrzucamy projekt sceny GameScene.sks

Przerabiamy szablon

- ✦ Zmieniamy orientację ekranu na poziomą



- ✦ Wyrzucamy projekt sceny GameScene.sks
- ✦ Zmieniamy GameViewController.swift

Kontroler widoku

Kontroler widoku

```
import UIKit
import SpriteKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        let scene = GameScene(size: view.bounds.size)
        let skView = view as! SKView
        skView.showsFPS = true
        skView.showsNodeCount = true
        skView.ignoresSiblingOrder = true
        scene.scaleMode = .ResizeFill
        skView.presentScene(scene)
    }

    override func prefersStatusBarHidden() -> Bool {
        return true
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Release any cached data, images, etc that aren't in use.
    }
}
```

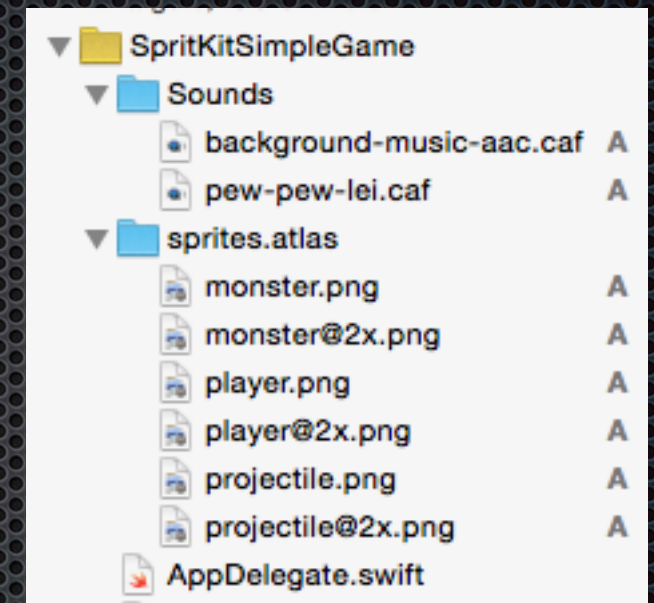
Sprite

Sprite

- Dodajemy zasoby do projektu

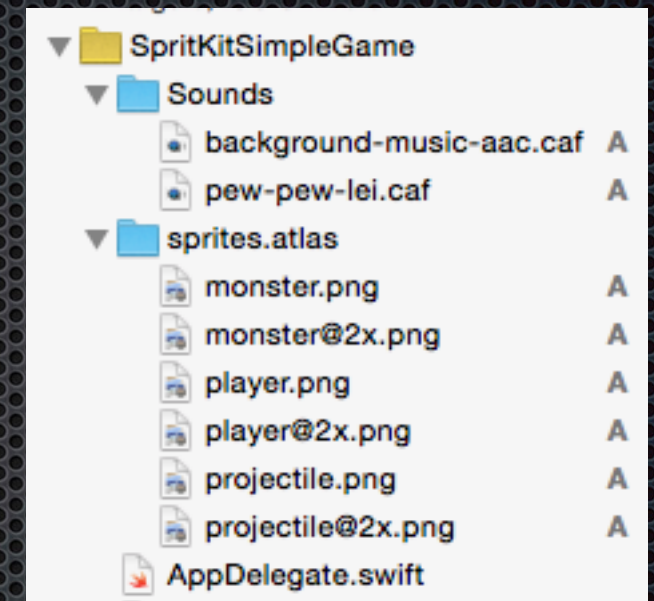
Sprite

- ✦ Dodajemy zasoby do projektu



Sprite

- ✦ Dodajemy zasoby do projektu



```
import SpriteKit

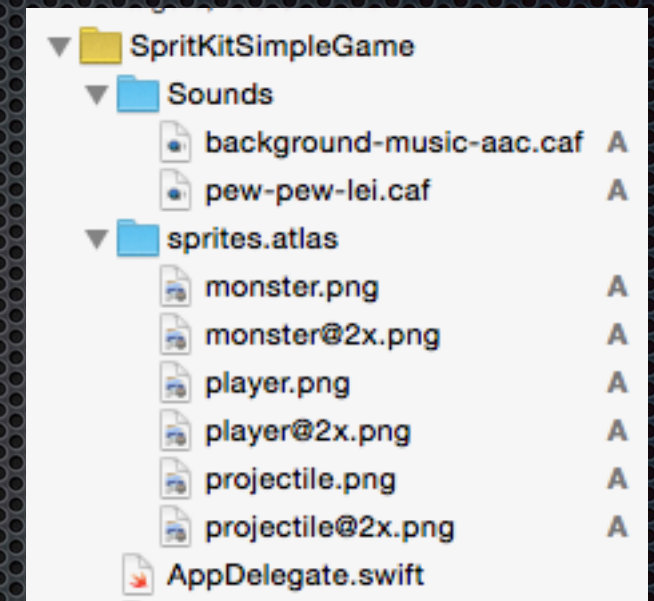
class GameScene: SKScene {

    // 1
    let player = SKSpriteNode(imageNamed: "player")

    override func didMoveToView(view: SKView) {
        // 2
        backgroundColor = SKColor.whiteColor()
        // 3
        player.position = CGPoint(x: size.width * 0.1, y: size.height * 0.5)
        // 4
        addChild(player)
    }
}
```

Sprite

- ✦ Dodajemy zasoby do projektu
- ✦ Edytujemy GameScene.swift



```
import SpriteKit

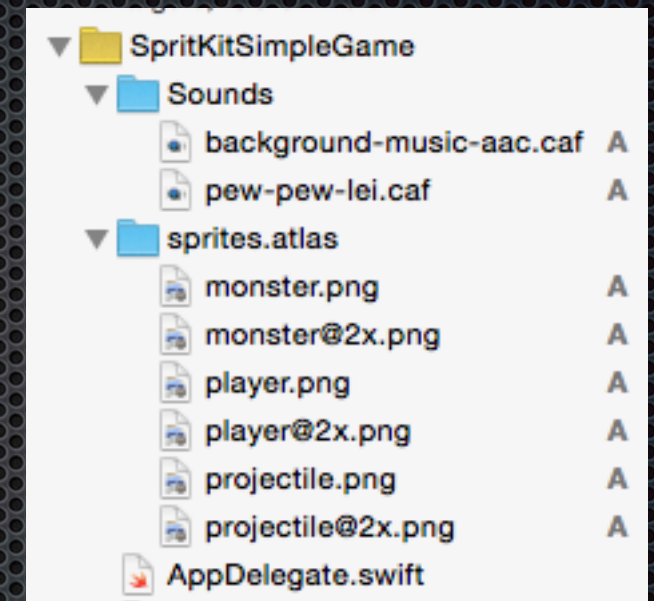
class GameScene: SKScene {

    // 1
    let player = SKSpriteNode(imageNamed: "player")

    override func didMoveToView(view: SKView) {
        // 2
        backgroundColor = SKColor.whiteColor()
        // 3
        player.position = CGPoint(x: size.width * 0.1, y: size.height * 0.5)
        // 4
        addChild(player)
    }
}
```

Sprite

- ✦ Dodajemy zasoby do projektu
- ✦ Edytujemy GameScene.swift
- ✦ Pokazujemy na ekranie naszego bohatera



```
import SpriteKit

class GameScene: SKScene {

    // 1
    let player = SKSpriteNode(imageNamed: "player")

    override func didMoveToView(view: SKView) {
        // 2
        backgroundColor = SKColor.whiteColor()
        // 3
        player.position = CGPoint(x: size.width * 0.1, y: size.height * 0.5)
        // 4
        addChild(player)
    }
}
```

Efekt

Efekt



Dodajemy przeciwników

Dodajemy przeciwników

- Dodajemy metody do GameScene.swift

Dodajemy przeciwników

- ✦ Dodajemy metody do GameScene.swift

```
func random() -> CGFloat {
    return CGFloat(Float(arc4random()) / 0xFFFFFFFF)
}

func random(#min: CGFloat, max: CGFloat) -> CGFloat {
    return random() * (max - min) + min
}

func addMonster() {
    // Create sprite
    let monster = SKSpriteNode(imageNamed: "monster")

    // Determine where to spawn the monster along the Y axis
    let actualY = random(min: monster.size.height/2, max: size.height - monster.size.height/2)

    // Position the monster slightly off-screen along the right edge,
    // and along a random position along the Y axis as calculated above
    monster.position = CGPoint(x: size.width + monster.size.width/2, y: actualY)

    // Add the monster to the scene
    addChild(monster)

    // Determine speed of the monster
    let actualDuration = random(min: CGFloat(2.0), max: CGFloat(4.0))

    // Create the actions
    let actionMove = SKAction.moveTo(CGPoint(x: -monster.size.width/2, y: actualY), duration:
        NSTimeInterval(actualDuration))
    let actionMoveDone = SKAction.removeFromParent()
    monster.runAction(SKAction.sequence([actionMove, actionMoveDone]))
}
```

Dodajemy przeciwników

Dodajemy przeciwników

- Modyfikujemy metodę didMoveToView

Dodajemy przeciwników

- ✦ Modyfikujemy metodę `didMoveToView`

```
override func didMoveToView(view: SKView) {  
    // 2  
    backgroundColor = SKColor.whiteColor()  
    // 3  
    player.position = CGPoint(x: size.width * 0.1, y: size.height * 0.5)  
    // 4  
    addChild(player)  
  
    runAction(SKAction.repeatActionForever(  
        SKAction.sequence([  
            SKAction.runBlock(addMonster),  
            SKAction.waitForDuration(1.0)  
        ])  
    ))  
}
```

Efekt



Efekt



Strzelanie

Strzelanie

- nasz bohater będzie strzelał w kierunku miejsca, w które stuknie gracz

Strzelanie

- nasz bohater będzie strzelał w kierunku miejsca, w które stuknie gracz
- potrzebujemy do tego trochę matematyki

Strzelanie

- nasz bohater będzie strzelał w kierunku miejsca, w które stuknie gracz
- potrzebujemy do tego trochę matematyki
- SpriteKit niestety nie udostępnia funkcji operujących na wektorach i macierzach

Operatory wektorowe

Operatory wektorowe

- ✦ Dodajemy je przed klasą GameScene

Operatory wektorowe

- ✦ Dodajemy je przed klasą GameScene

```
func + (left: CGPoint, right: CGPoint) -> CGPoint {
    return CGPoint(x: left.x + right.x, y: left.y + right.y)
}

func - (left: CGPoint, right: CGPoint) -> CGPoint {
    return CGPoint(x: left.x - right.x, y: left.y - right.y)
}

func * (point: CGPoint, scalar: CGFloat) -> CGPoint {
    return CGPoint(x: point.x * scalar, y: point.y * scalar)
}

func / (point: CGPoint, scalar: CGFloat) -> CGPoint {
    return CGPoint(x: point.x / scalar, y: point.y / scalar)
}

#if !(arch(x86_64) || arch(arm64))
func sqrt(a: CGFloat) -> CGFloat {
    return CGFloat(sqrtf(Float(a)))
}
#endif

extension CGPoint {
    func length() -> CGFloat {
        return sqrt(x*x + y*y)
    }

    func normalized() -> CGPoint {
        return self / length()
    }
}
```

Dotyk

Dotyk

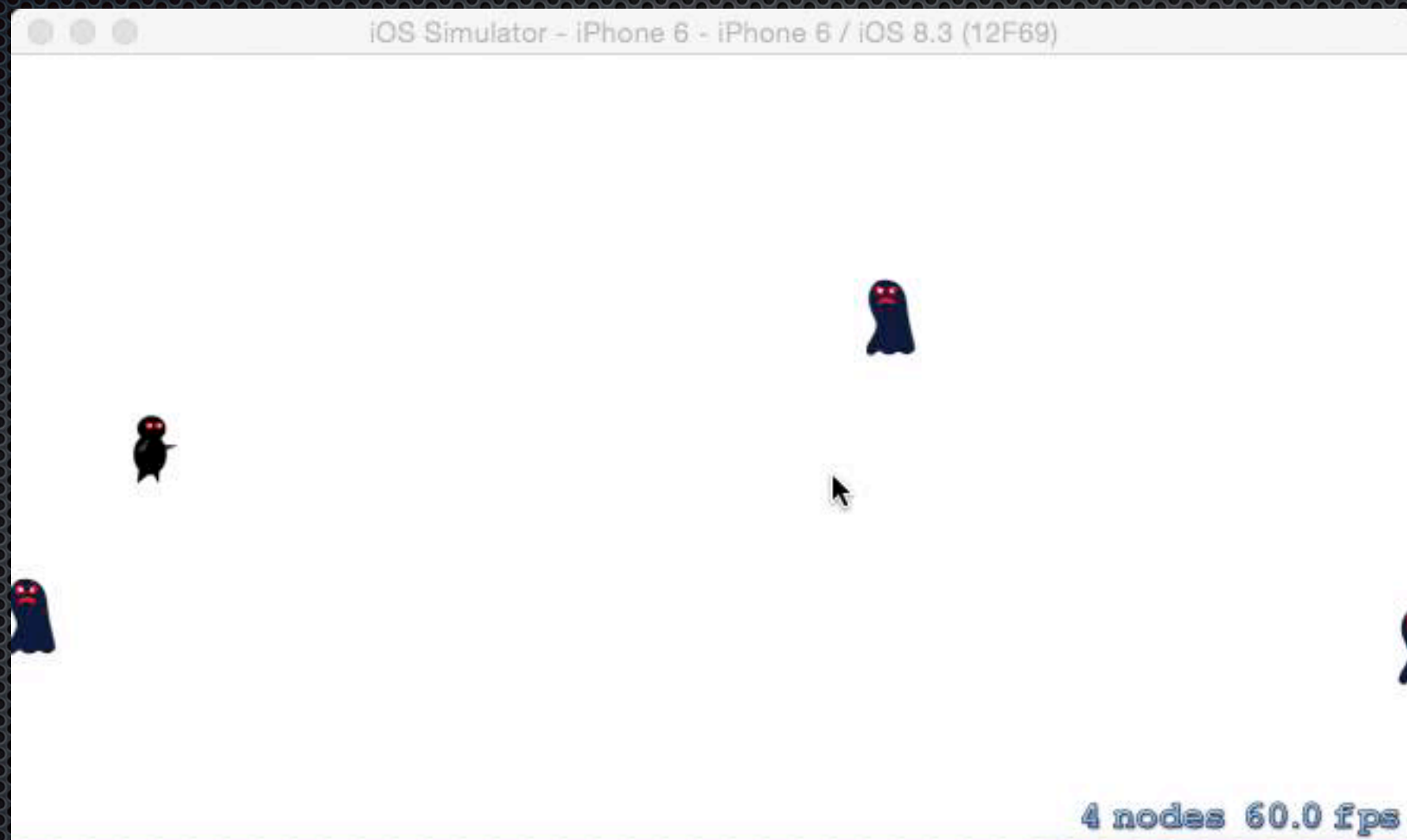
- ✦ Dodajemy obsługę dotyku w klasie GameScene

Dotyk

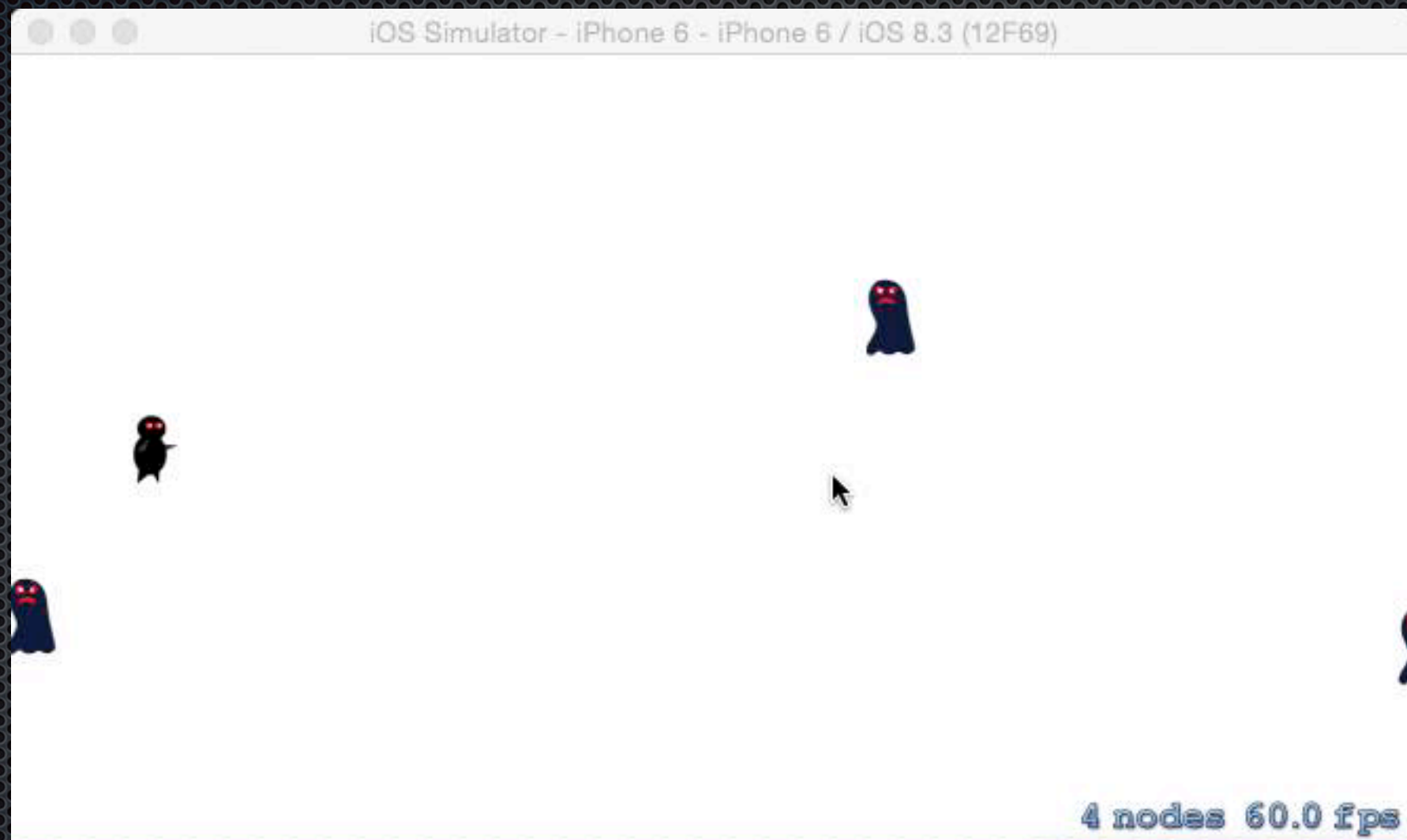
- ✦ Dodajemy obsługę dotyku w klasie GameScene

```
override func touchesEnded(touches: Set<NSObject>, withEvent event: UIEvent) {  
  
    // 1 - Choose one of the touches to work with  
    let touch = touches.first as! UITouch  
    let touchLocation = touch.locationInNode(self)  
  
    // 2 - Set up initial location of projectile  
    let projectile = SKSpriteNode(imageNamed: "projectile")  
    projectile.position = player.position  
  
    // 3 - Determine offset of location to projectile  
    let offset = touchLocation - projectile.position  
  
    // 4 - Bail out if you are shooting down or backwards  
    if (offset.x < 0) { return }  
  
    // 5 - OK to add now - you've double checked position  
    addChild(projectile)  
  
    // 6 - Get the direction of where to shoot  
    let direction = offset.normalized()  
  
    // 7 - Make it shoot far enough to be guaranteed off screen  
    let shootAmount = direction * 1000  
  
    // 8 - Add the shoot amount to the current position  
    let realDest = shootAmount + projectile.position  
  
    // 9 - Create the actions  
    let actionMove = SKAction.moveTo(realDest, duration: 2.0)  
    let actionMoveDone = SKAction.removeFromParent()  
    projectile.runAction(SKAction.sequence([actionMove, actionMoveDone]))  
}
```

Efekt



Efekt



Wykrywanie kolizji

Wykrywanie kolizji

- SpriteKit zawiera obsługę fizyki, która znacząco ułatwi wykrywanie kolizji

Wykrywanie kolizji

- SpriteKit zawiera obsługę fizyki, która znacząco ułatwi wykrywanie kolizji
- sprite'y mogą należeć do jednej z 32 kategorii

Wykrywanie kolizji

- SpriteKit zawiera obsługę fizyki, która znacząco ułatwi wykrywanie kolizji
- sprite'y mogą należeć do jednej z 32 kategorii
- zdefiniujemy sobie strukturę

Wykrywanie kolizji

- ✦ SpriteKit zawiera obsługę fizyki, która znacząco ułatwi wykrywanie kolizji
- ✦ sprite'y mogą należeć do jednej z 32 kategorii
- ✦ zdefiniujemy sobie strukturę

```
struct PhysicsCategory {  
    static let None      : UInt32 = 0  
    static let All       : UInt32 = UInt32.max  
    static let Monster   : UInt32 = 0b1      // 1  
    static let Projectile: UInt32 = 0b10     // 2  
}
```

Wykrywanie kolizji

Wykrywanie kolizji

- nasz klasa będzie informowana o zdarzeniach fizycznych, gdy będzie realizować protokół SKPhysicsContactDelegate

Wykrywanie kolizji

- nasza klasa będzie informowana o zdarzeniach fizycznych, gdy będzie realizować protokół SKPhysicsContactDelegate

```
class GameScene: SKScene, SKPhysicsContactDelegate {
```

Wykrywanie kolizji

- nasza klasa będzie informowana o zdarzeniach fizycznych, gdy będzie realizować protokół SKPhysicsContactDelegate

```
class GameScene: SKScene, SKPhysicsContactDelegate {
```

- w metodzie didMoveToView dodajemy

Wykrywanie kolizji

- nasza klasa będzie informowana o zdarzeniach fizycznych, gdy będzie realizować protokół SKPhysicsContactDelegate

```
class GameScene: SKScene, SKPhysicsContactDelegate {
```

- w metodzie didMoveToView dodajemy

```
physicsWorld.gravity = CGVectorMake(0, 0)  
physicsWorld.contactDelegate = self
```

Wykrywanie kolizji

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora
 1. przypisujemy potworowi jego pole

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora
 - 1.przypisujemy potworowi jego pole
 - 2.potwór będzie poruszany przez nas, a nie przez fizykę

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora
 - 1.przypisujemy potworowi jego pole
 - 2.potwór będzie poruszany przez nas, a nie przez fizykę
 - 3.ustawiamy kategorię sprite'a

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora
 - 1.przypisujemy potworowi jego pole
 - 2.potwór będzie poruszany przez nas, a nie przez fizykę
 - 3.ustawiamy kategorię sprite'a
 - 4.ustawiamy kategorię sprite'ów, dla których ma być wykrywana kolizja

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

- ✦ w metodzie addMonster dopisujemy kod po utworzeniu potwora
 - 1.przypisujemy potworowi jego pole
 - 2.potwór będzie poruszany przez nas, a nie przez fizykę
 - 3.ustawiamy kategorię sprite'a
 - 4.ustawiamy kategorię sprite'ów, dla których ma być wykrywana kolizja
 - 5.tu by można ustawić np. odbijanie się obiektów

```
monster.physicsBody = SKPhysicsBody(rectangleOfSize: monster.size) // 1
monster.physicsBody?.dynamic = true // 2
monster.physicsBody?.categoryBitMask = PhysicsCategory.Monster // 3
monster.physicsBody?.contactTestBitMask = PhysicsCategory.Projectile // 4
monster.physicsBody?.collisionBitMask = PhysicsCategory.None // 5
```

Wykrywanie kolizji

Wykrywanie kolizji

- Podobny kod dopisujemy w touchesEnded dla pocisków

Wykrywanie kolizji

- Podobny kod dopisujemy w touchesEnded dla pocisków

```
projectile.physicsBody = SKPhysicsBody(circleOfRadius: projectile.size.width/2)
projectile.physicsBody?.dynamic = true
projectile.physicsBody?.categoryBitMask = PhysicsCategory.Projectile
projectile.physicsBody?.contactTestBitMask = PhysicsCategory.Monster
projectile.physicsBody?.collisionBitMask = PhysicsCategory.None
projectile.physicsBody?.usesPreciseCollisionDetection = true
```

Wykrywanie kolizji

- Podobny kod dopisujemy w touchesEnded dla pocisków

```
projectile.physicsBody = SKPhysicsBody(circleOfRadius: projectile.size.width/2)
projectile.physicsBody?.dynamic = true
projectile.physicsBody?.categoryBitMask = PhysicsCategory.Projectile
projectile.physicsBody?.contactTestBitMask = PhysicsCategory.Monster
projectile.physicsBody?.collisionBitMask = PhysicsCategory.None
projectile.physicsBody?.usesPreciseCollisionDetection = true
```

- Dodajemy metodę obsługującą kolizję

Wykrywanie kolizji

- Podobny kod dopisujemy w touchesEnded dla pocisków

```
projectile.physicsBody = SKPhysicsBody(circleOfRadius: projectile.size.width/2)
projectile.physicsBody?.dynamic = true
projectile.physicsBody?.categoryBitMask = PhysicsCategory.Projectile
projectile.physicsBody?.contactTestBitMask = PhysicsCategory.Monster
projectile.physicsBody?.collisionBitMask = PhysicsCategory.None
projectile.physicsBody?.usesPreciseCollisionDetection = true
```

- Dodajemy metodę obsługującą kolizję

```
func projectileDidCollideWithMonster(projectile:SKSpriteNode, monster:SKSpriteNode) {
    println("Hit")
    projectile.removeFromParent()
    monster.removeFromParent()
}
```

Wykrywanie kolizji

Wykrywanie kolizji

- Dodajemy metodę wymaganą przez protokół

Wykrywanie kolizji

- ✦ Dodajemy metodę wymaganą przez protokół

```
func didBeginContact(contact: SKPhysicsContact) {  
  
    // 1  
    var firstBody: SKPhysicsBody  
    var secondBody: SKPhysicsBody  
    if contact.bodyA.categoryBitMask < contact.bodyB.categoryBitMask {  
        firstBody = contact.bodyA  
        secondBody = contact.bodyB  
    } else {  
        firstBody = contact.bodyB  
        secondBody = contact.bodyA  
    }  
  
    // 2  
    if ((firstBody.categoryBitMask & PhysicsCategory.Monster != 0) &&  
        (secondBody.categoryBitMask & PhysicsCategory.Projectile != 0)) {  
        projectileDidCollideWithMonster(firstBody.node as! SKSpriteNode, monster: secondBody.  
            node as! SKSpriteNode)  
    }  
  
}
```

Efekt



Efekt



Dźwięk

Dźwięk

- ✦ SpriteKit nie ma dedykowanego mechanizmu do obsługi dźwięku

Dźwięk

- ✦ SpriteKit nie ma dedykowanego mechanizmu do obsługi dźwięku
- ✦ Dźwięk można odtwarzać korzystając z normalnego API

Dźwięk

Dźwięk

- Przed klasą GameScene dodajemy kod

Dźwięk

- ✦ Przed klasą GameScene dodajemy kod

```
import AVFoundation

var backgroundMediaPlayer: AVAudioPlayer!

func playBackgroundMusic(filename: String) {
    let url = NSBundle.mainBundle().URLForResource(
        filename, withExtension: nil)
    if (url == nil) {
        println("Could not find file: \(filename)")
        return
    }

    var error: NSError? = nil
    backgroundMediaPlayer =
        AVAudioPlayer(contentsOfURL: url, error: &error)
    if backgroundMediaPlayer == nil {
        println("Could not create audio player: \(error!)")
        return
    }

    backgroundMediaPlayer.numberOfLoops = -1
    backgroundMediaPlayer.prepareToPlay()
    backgroundMediaPlayer.play()
}
```

Dźwięk

Dźwięk

- ✦ Na początku metody `didMoveToView` dodajemy

Dźwięk

- ✦ Na początku metody `didMoveToView` dodajemy

```
playBackgroundMusic("background-music-aac.caf")
```

Dźwięk

- ✦ Na początku metody `didMoveToView` dodajemy

```
playBackgroundMusic("background-music-aac.caf")
```

- ✦ a na początku `touchesEnded` to

Dźwięk

- ✦ Na początku metody `didMoveToView` dodajemy

```
playBackgroundMusic("background-music-aac.caf")
```

- ✦ a na początku `touchesEnded` to

```
runAction(SKAction.playSoundFileNamed("pew-pew-lei.caf", waitForCompletion: false))
```

Efekt



Efekt



Ekran GameOver

Ekran GameOver

- ✦ Dodajemy nowy plik swift GameOverScene.swift

Ekran GameOver

- ✦ Dodajemy nowy plik swift GameOverScene.swift
- ✦ Edytujemy kod

Ekran GameOver

- ✦ Dodajemy nowy plik swift GameOverScene.swift
- ✦ Edytujemy kod

```
import Foundation
import SpriteKit

class GameOverScene: SKScene {

    init(size: CGSize, won: Bool) {

        super.init(size: size)
        // 1
        backgroundColor = SKColor.whiteColor()
        // 2
        var message = won ? "You Won!" : "You Lose :["
        // 3
        let label = SKLabelNode(fontNamed: "Chalkduster")
        label.text = message
        label.fontSize = 40
        label.fontColor = SKColor.blackColor()
        label.position = CGPoint(x: size.width/2, y: size.height/2)
        addChild(label)
        // 4
        runAction(SKAction.sequence([
            SKAction.waitForDuration(3.0),
            SKAction.runBlock() {
                // 5
                let reveal = SKTransition.flipHorizontalWithDuration(0.5)
                let scene = GameScene(size: size)
                self.view?.presentScene(scene, transition: reveal)
            }
        ]))
    }
    // 6
    required init(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}
```

Warunek końca

Warunek końca

- Gdy potwór nas ominie przegrywamy

Warunek końca

- Gdy potwór nas ominie przegrywamy
- Zamieniamy ostatnią linię w metodzie addMonster na

Warunek końca

- ✦ Gdy potwór nas ominie przegrywamy
- ✦ Zamieniamy ostatnią linię w metodzie addMonster na

```
//monster.runAction(SKAction.sequence([actionMove, actionMoveDone]))  
let loseAction = SKAction.runBlock() {  
    let reveal = SKTransition.flipHorizontalWithDuration(0.5)  
    let gameOverScene = GameOverScene(size: self.size, won: false)  
    self.view?.presentScene(gameOverScene, transition: reveal)  
}  
monster.runAction(SKAction.sequence([actionMove, loseAction, actionMoveDone]))
```

Efekt



Efekt



Wygrana

Wygrana

- Wygrywamy, gdy zabijemy 30 potworów

Wygrana

- ✦ Wygrywamy, gdy zabijemy 30 potworów
- ✦ Dodajemy pole w klasie GameScen

Wygrana

- ✦ Wygrywamy, gdy zabijemy 30 potworów
- ✦ Dodajemy pole w klasie GameScen

```
var monstersDestroyed = 0
```

Wygrana

- ✦ Wygrywamy, gdy zabijemy 30 potworów
- ✦ Dodajemy pole w klasie GameScen

```
var monstersDestroyed = 0
```

- ✦ modyfikujemy metodę didColideWithMonster

Wygrana

- ✦ Wygrywamy, gdy zabijemy 30 potworów
- ✦ Dodajemy pole w klasie GameScen

```
var monstersDestroyed = 0
```

- ✦ modyfikujemy metodę didCollideWithMonster

```
func projectileDidCollideWithMonster(projectile:SKSpriteNode, monster:SKSpriteNode) {  
    println("Hit")  
    projectile.removeFromParent()  
    monster.removeFromParent()  
    monstersDestroyed++  
    if (monstersDestroyed > 30) {  
        let reveal = SKTransition.flipHorizontalWithDuration(0.5)  
        let gameOverScene = GameOverScene(size: self.size, won: true)  
        self.view?.presentScene(gameOverScene, transition: reveal)  
    }  
}
```

Efekt



Efekt



SceneKit

SceneKit

SceneKit

- podobna biblioteka do SpriteKit tyle, że do grafiki 3D

SceneKit

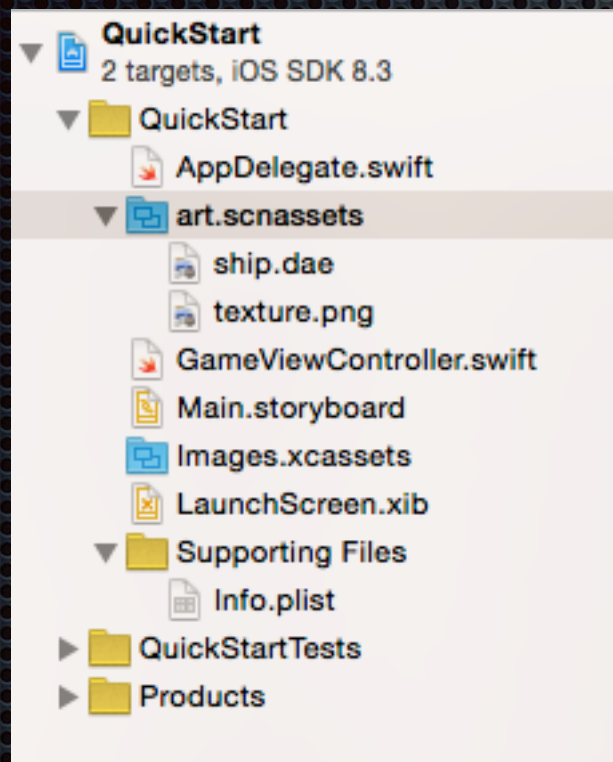
- podobna biblioteka do SpriteKit tyle, że do grafiki 3D

Choose options for your new project:

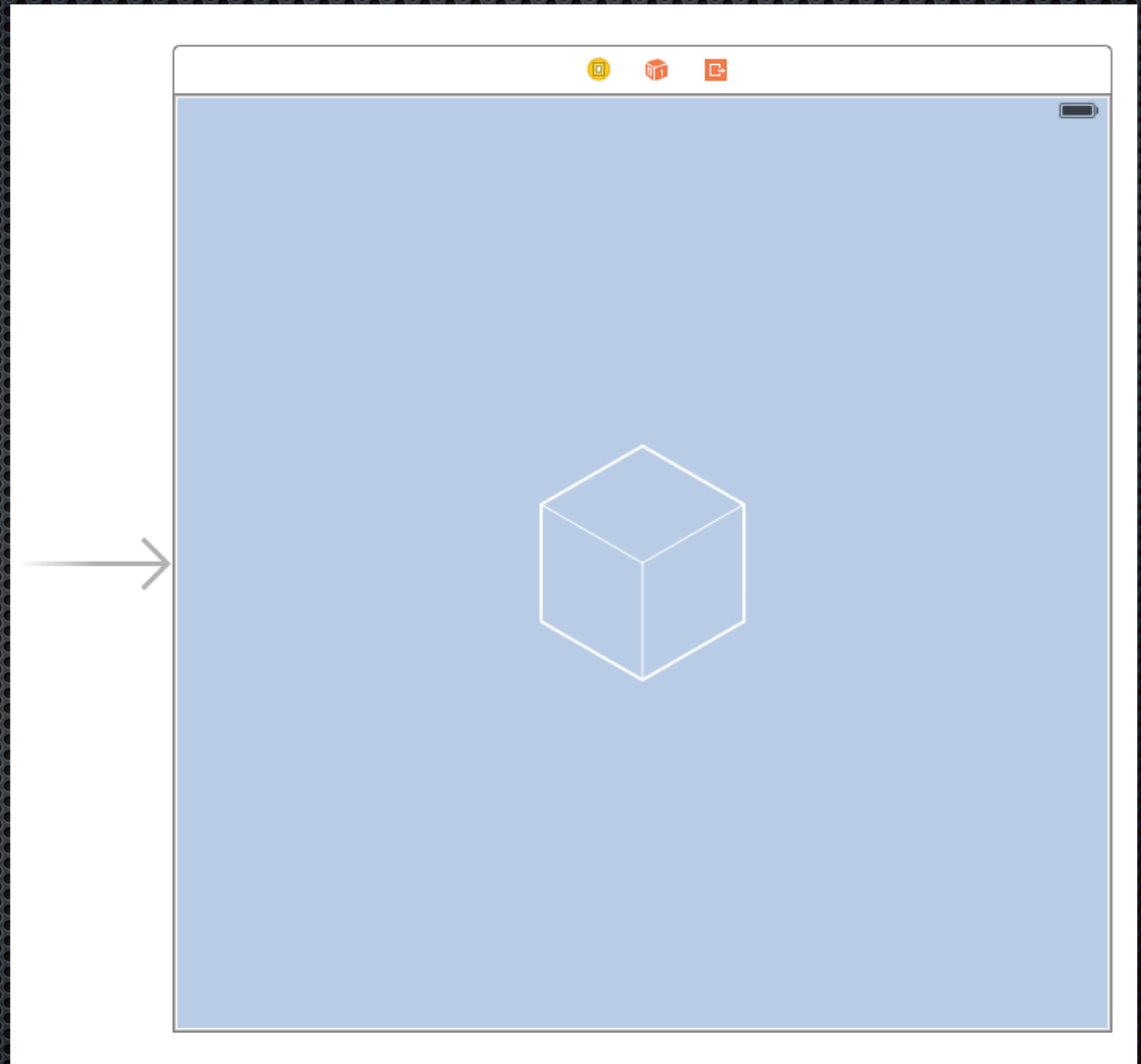
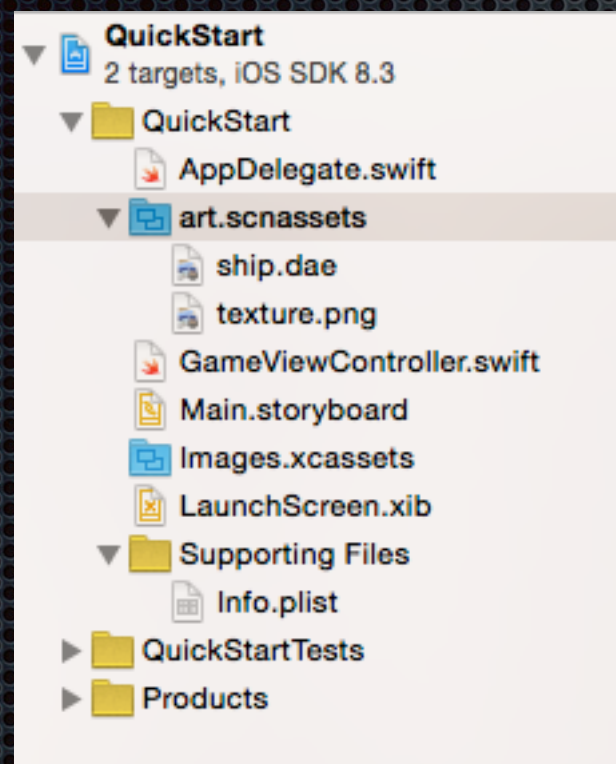
Product Name:	QuickStart
Organization Name:	Bartłomiej Prędko
Organization Identifier:	com.put
Bundle Identifier:	com.put.QuickStart
Language:	Swift
Game Technology:	SceneKit
Devices:	Universal

SceneKit

SceneKit



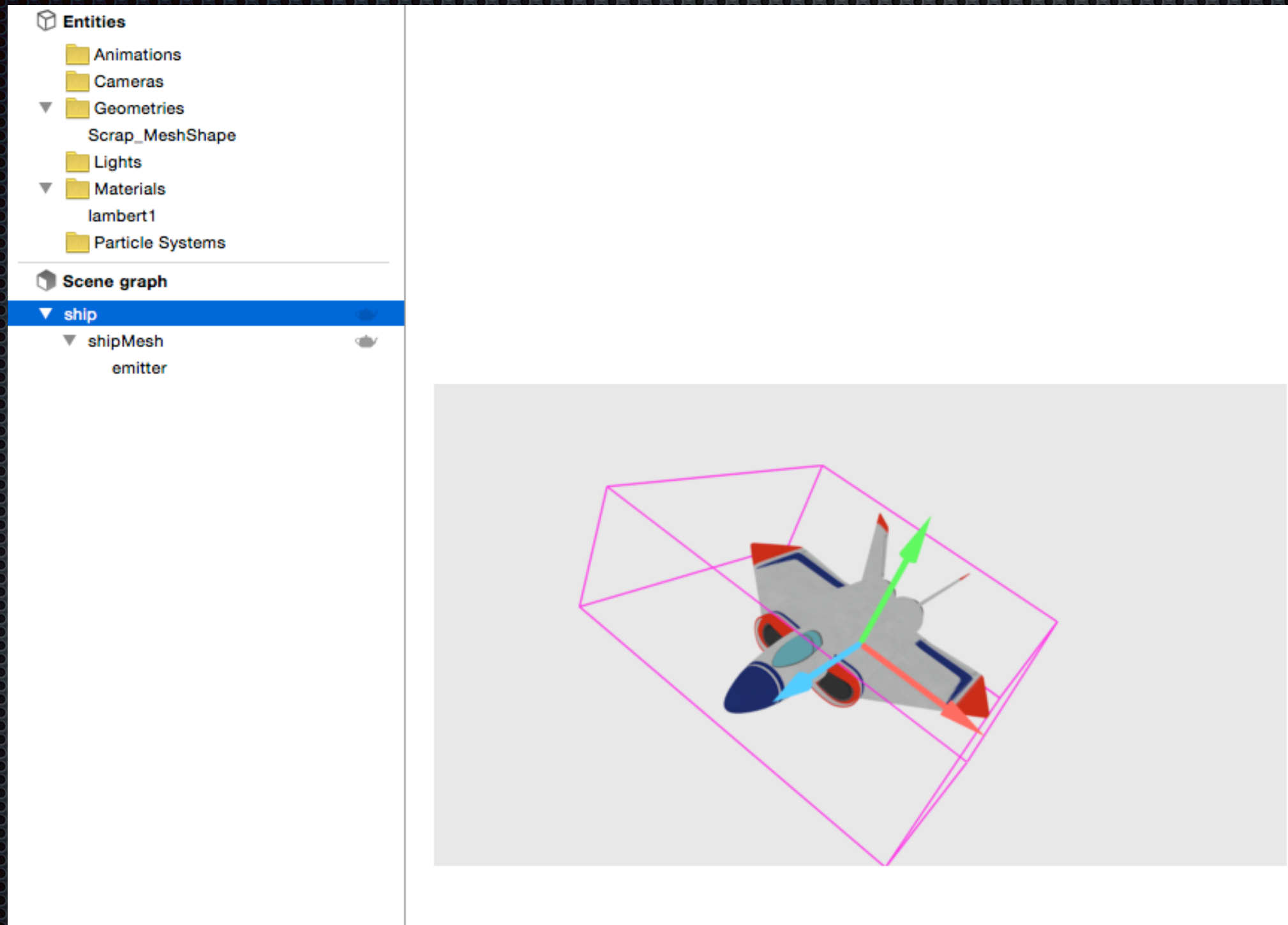
SceneKit



Scenorys - obiekt GameViewController

SceneKit

SceneKit



SceneKit - kontroler widoku

SceneKit - kontroler widoku

```
import UIKit
import QuartzCore
import SceneKit

class GameViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // create a new scene
        let scene = SCNScene(named: "art.scnassets/ship.dae")!

        // create and add a camera to the scene
        let cameraNode = SCNNode()
        cameraNode.camera = SCNCamera()
        scene.rootNode.addChildNode(cameraNode)

        // place the camera
        cameraNode.position = SCNVector3(x: 0, y: 0, z: 15)

        // create and add a light to the scene
        let lightNode = SCNNode()
        lightNode.light = SCNLight()
        lightNode.light!.type = SCNLightTypeOmni
        lightNode.position = SCNVector3(x: 0, y: 10, z: 10)
        scene.rootNode.addChildNode(lightNode)

        // create and add an ambient light to the scene
        let ambientLightNode = SCNNode()
        ambientLightNode.light = SCNLight()
        ambientLightNode.light!.type = SCNLightTypeAmbient
        ambientLightNode.light!.color = UIColor.darkGrayColor()
        scene.rootNode.addChildNode(ambientLightNode)

        // retrieve the ship node
        let ship = scene.rootNode.childNodeWithName("ship", recursively: true)!

        // animate the 3d object
        ship.runAction(SCNAction.repeatActionForever(SCNAction.rotateByX(0, y: 2, z: 0, duration: 1)))
    }
}
```

SceneKit - kontroler widoku

SceneKit - kontroler widoku

```
// retrieve the SCNView
let scnView = self.view as! SCNView

// set the scene to the view
scnView.scene = scene

// allows the user to manipulate the camera
scnView.allowsCameraControl = true

// show statistics such as fps and timing information
scnView.showsStatistics = true

// configure the view
scnView.backgroundColor = UIColor.blackColor()

// add a tap gesture recognizer
let tapGesture = UITapGestureRecognizer(target: self, action: "handleTap:")
var gestureRecognizers = [AnyObject]()
gestureRecognizers.append(tapGesture)
if let existingGestureRecognizers = scnView.gestureRecognizers {
    gestureRecognizers.extend(existingGestureRecognizers)
}
scnView.gestureRecognizers = gestureRecognizers
}
```

SceneKit - kontroler widoku

SceneKit - kontroler widoku

```
func handleTap(gestureRecognizer: UIGestureRecognizer) {  
    // retrieve the SCNView  
    let scnView = self.view as! SCNView  
  
    // check what nodes are tapped  
    let p = gestureRecognizer.locationInView(scnView)  
    if let hitResults = scnView.hitTest(p, options: nil) {  
        // check that we clicked on at least one object  
        if hitResults.count > 0 {  
            // retrieved the first clicked object  
            let result: AnyObject! = hitResults[0]  
  
            // get its material  
            let material = result.node!.geometry!.firstMaterial!  
  
            // highlight it  
            SCNTransaction.begin()  
            SCNTransaction.setAnimationDuration(0.5)  
  
            // on completion - unhighlight  
            SCNTransaction.setCompletionBlock {  
                SCNTransaction.begin()  
                SCNTransaction.setAnimationDuration(0.5)  
  
                material.emission.contents = UIColor.blackColor()  
  
                SCNTransaction.commit()  
            }  
  
            material.emission.contents = UIColor.redColor()  
  
            SCNTransaction.commit()  
        }  
    }  
}
```

Przykładowa aplikacja

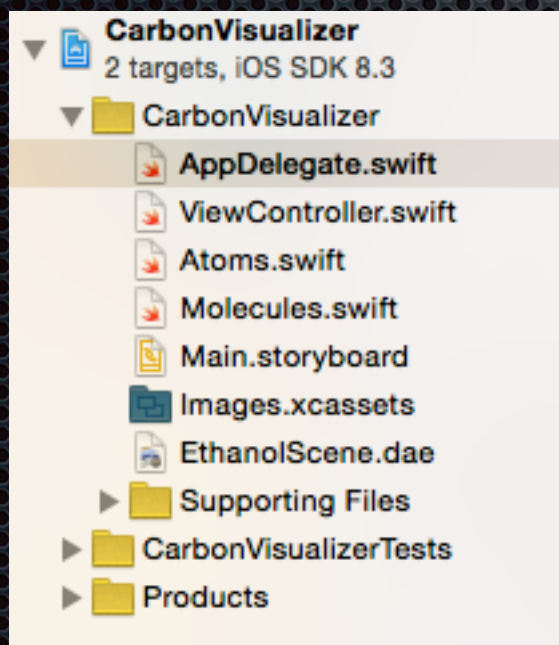
Przykładowa aplikacja

Przykładowa aplikacja

- program do wizualizacji cząsteczek w 3D

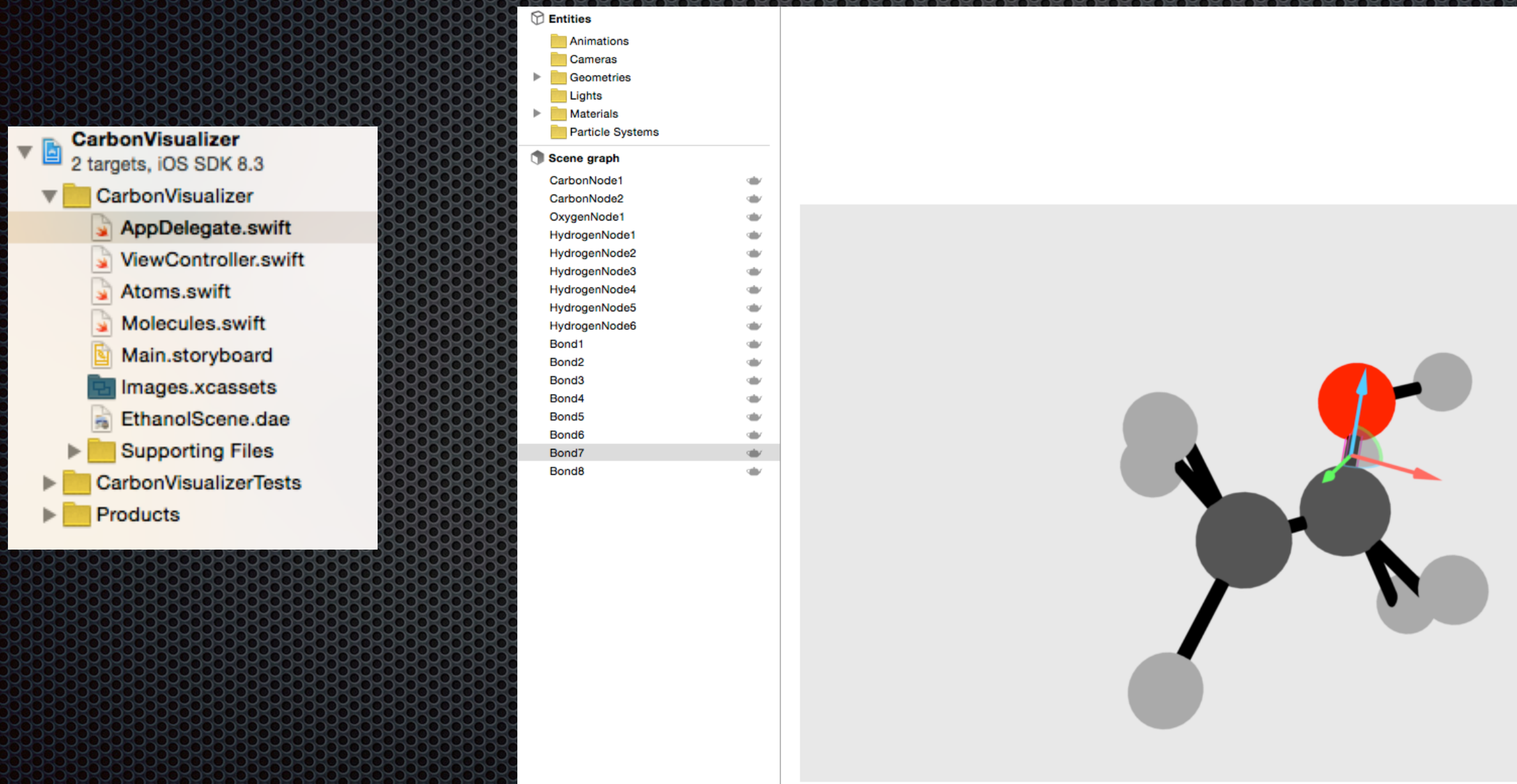
Przykładowa aplikacja

- ✦ program do wizualizacji cząsteczek w 3D



Przykładowa aplikacja

- ✦ program do wizualizacji cząsteczek w 3D



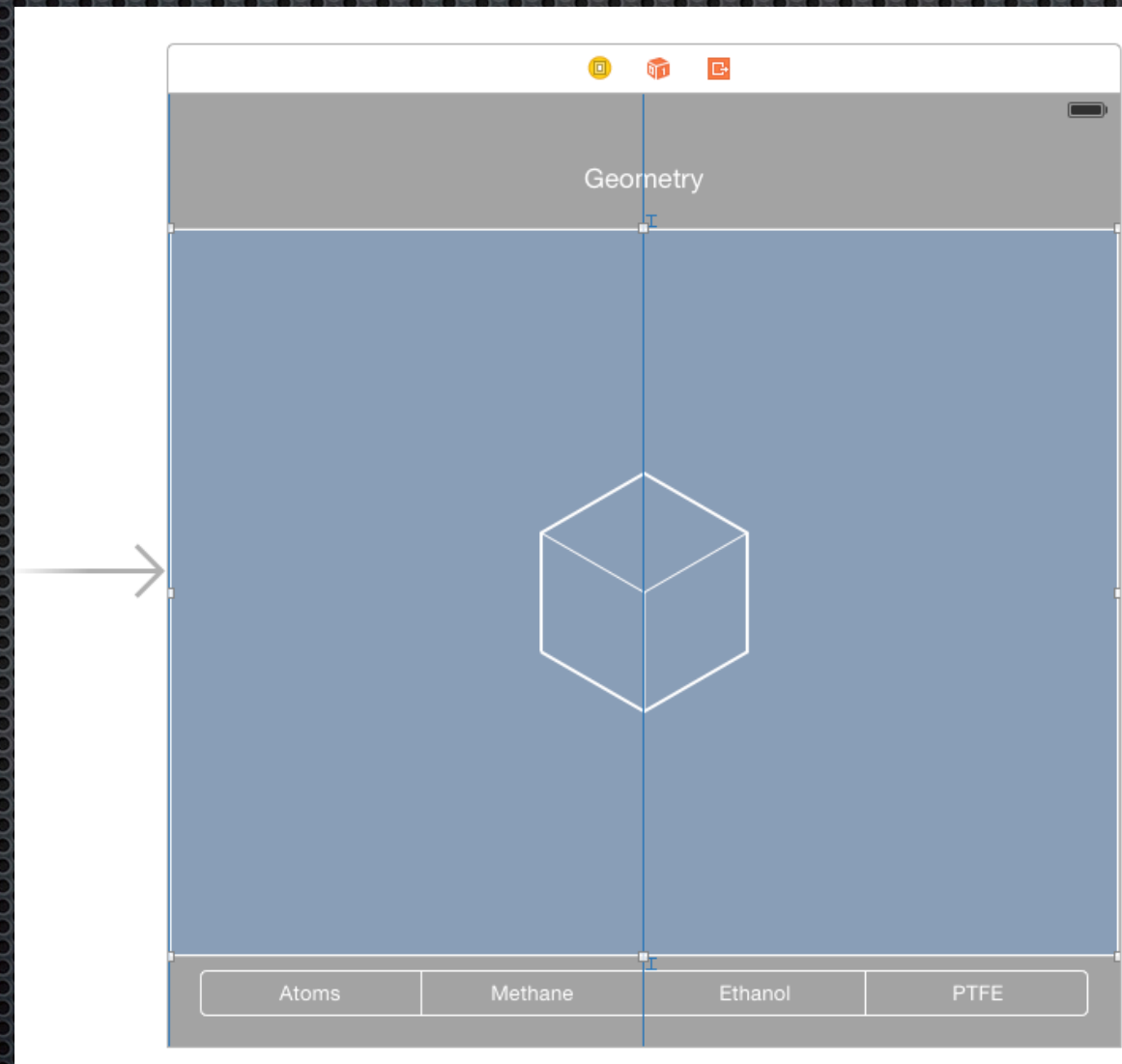
Scenorys

Scenorys

- Na widok składa się etykieta, pole wyboru i SceneKit View

Scenorys

- Na widok składa się etykieta, pole wyboru i SceneKit View



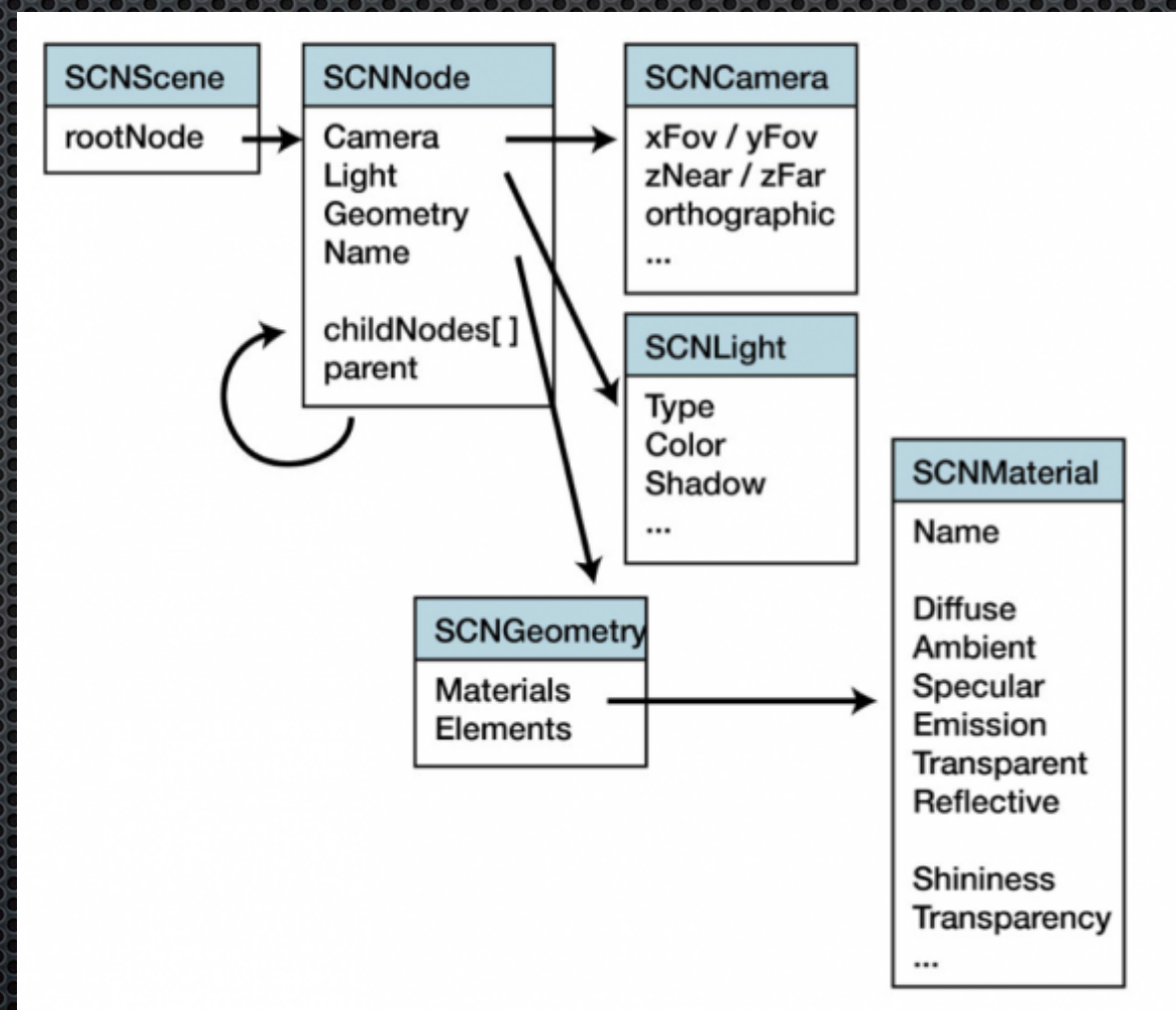
SceneKit

SceneKit

- operuje na hierarchicznej strukturze węzłów

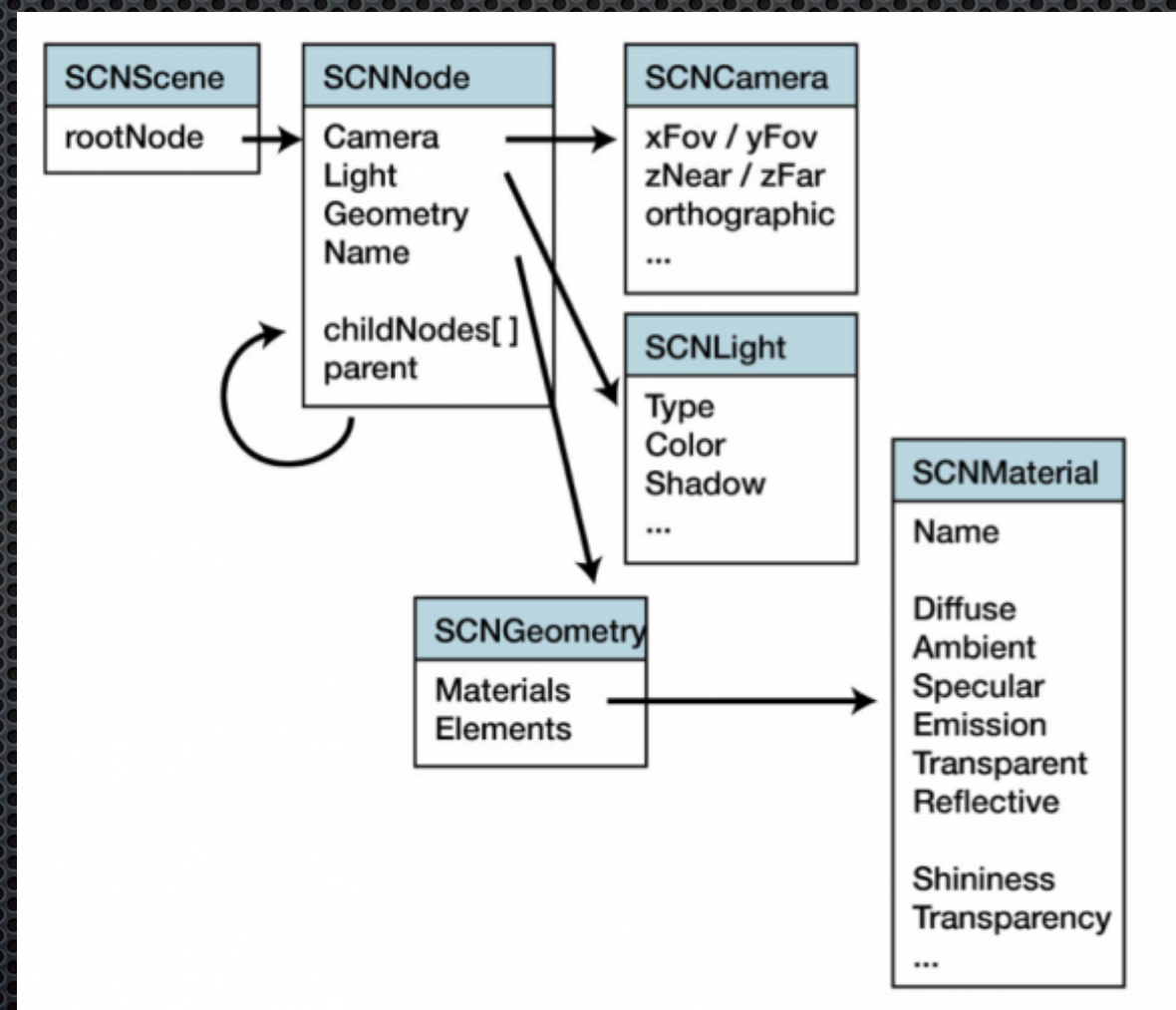
SceneKit

- operuje na hierarchicznej strukturze węzłów



SceneKit

- ✦ operuje na hierarchicznej strukturze węzłów
- ✦ każda transformacja dotyczy węzła i jego dzieci



Przykładowa aplikacja

Przykładowa aplikacja

- Początkowy ViewController

Przykładowa aplikacja

✦ Początkowy ViewController

```
import UIKit
import SceneKit

class ViewController: UIViewController {
    // UI
    @IBOutlet weak var geometryLabel: UILabel!
    @IBOutlet weak var sceneView: SCNView!

    // MARK: Lifecycle
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)
    }

    // MARK: IBActions
    @IBAction func segmentValueChanged(sender: UISegmentedControl) {
    }

    // MARK: Style
    override func preferredStatusBarStyle() -> UIStatusBarStyle {
        return UIStatusBarStyle.LightContent
    }

    // MARK: Transition
    override func viewWillTransitionToSize(size: CGSize, withTransitionCoordinator coordinator: UIViewControllerTransitionCoordinator) {
        super.viewWillTransitionToSize(size, withTransitionCoordinator: coordinator)
        sceneView.stop(nil)
        sceneView.play(nil)
    }
}
```




Tworzenie sceny

Tworzenie sceny

- dodajemy metodę do klasy kontrolera

Tworzenie sceny

- ✦ dodajemy metodę do klasy kontrolera
- ✦ i wywołujemy ją w viewDidLoadAppear

Tworzenie sceny

- ✦ dodajemy metodę do klasy kontrolera

```
// MARK: Scene
func sceneSetup() {
    // 1
    let scene = SCNScene()

    // 2
    let boxGeometry = SCNBox(width: 10.0, height: 10.0, length: 10.0, chamferRadius: 1.0)
    let boxNode = SCNNode(geometry: boxGeometry)
    scene.rootNode.addChildNode(boxNode)

    // 3
    sceneView.scene = scene
}
```

- ✦ i wywołujemy ją w viewDidLoadAppear

Tworzenie sceny

- ✦ dodajemy metodę do klasy kontrolera

```
// MARK: Scene
func sceneSetup() {
    // 1
    let scene = SCNScene()

    // 2
    let boxGeometry = SCNBox(width: 10.0, height: 10.0, length: 10.0, chamferRadius: 1.0)
    let boxNode = SCNNode(geometry: boxGeometry)
    scene.rootNode.addChildNode(boxNode)

    // 3
    sceneView.scene = scene
}
```

- ✦ i wywołujemy ją w viewDidLoad

```
override func viewDidLoad(animated: Bool) {
    super.viewDidLoad(animated)
    sceneSetup()
}
```



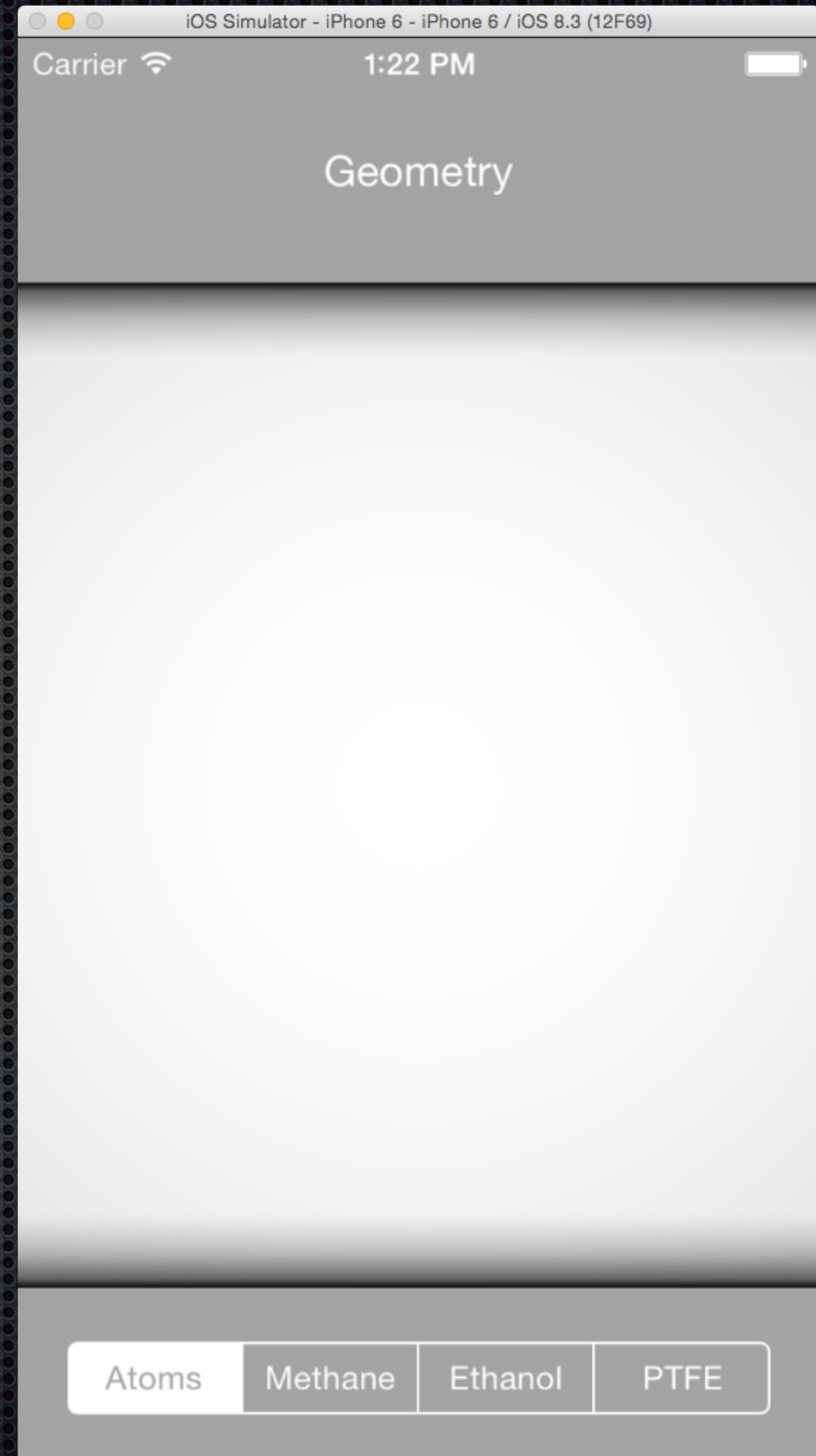

Dodamy oświetlenie

Dodamy oświetlenie

```
// 3|  
sceneView.scene = scene  
sceneView.autoenablesDefaultLighting = true
```

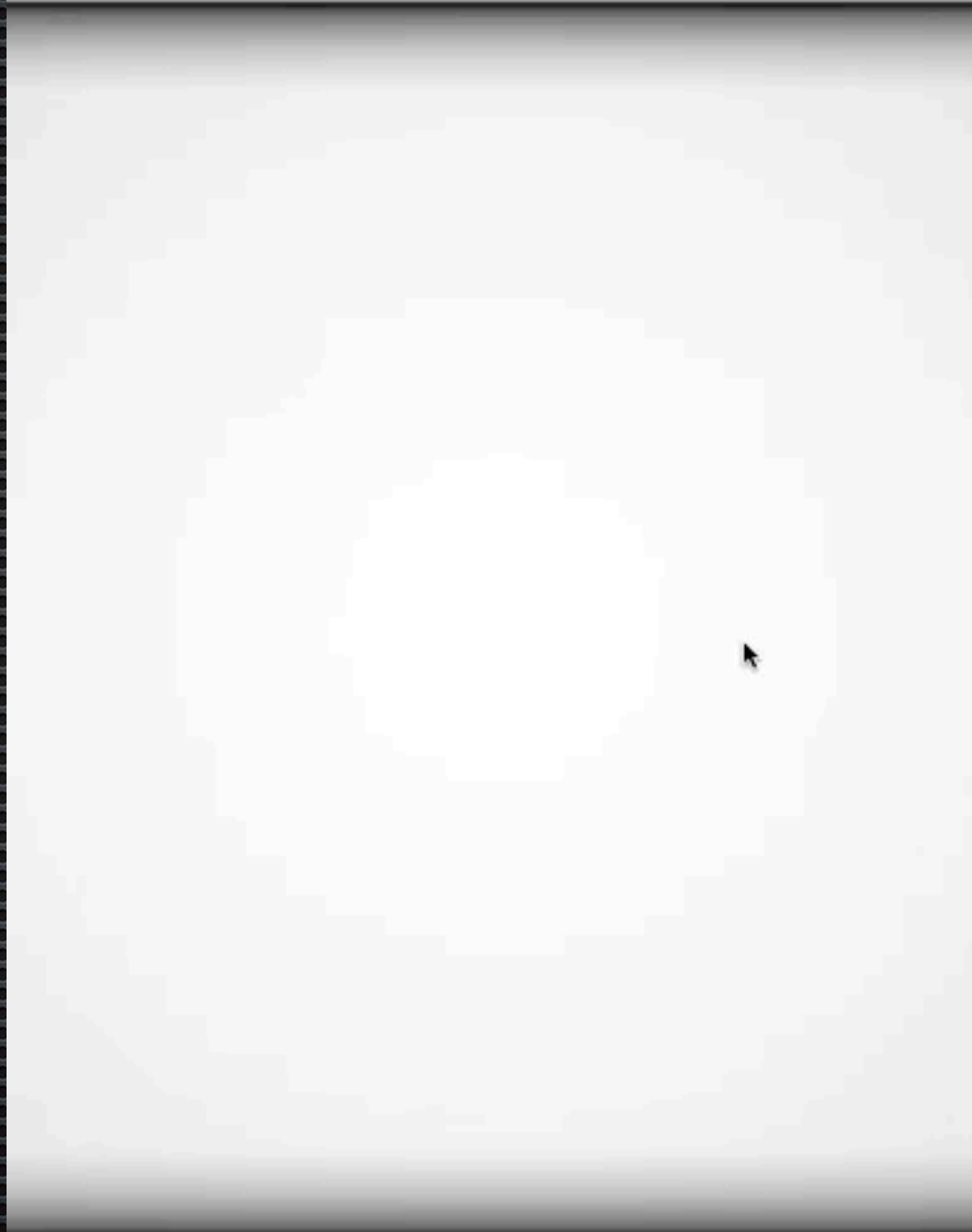
Dodamy oświetlenie

```
// 3|  
sceneView.scene = scene  
sceneView.autoenablesDefaultLighting = true
```





Geometry



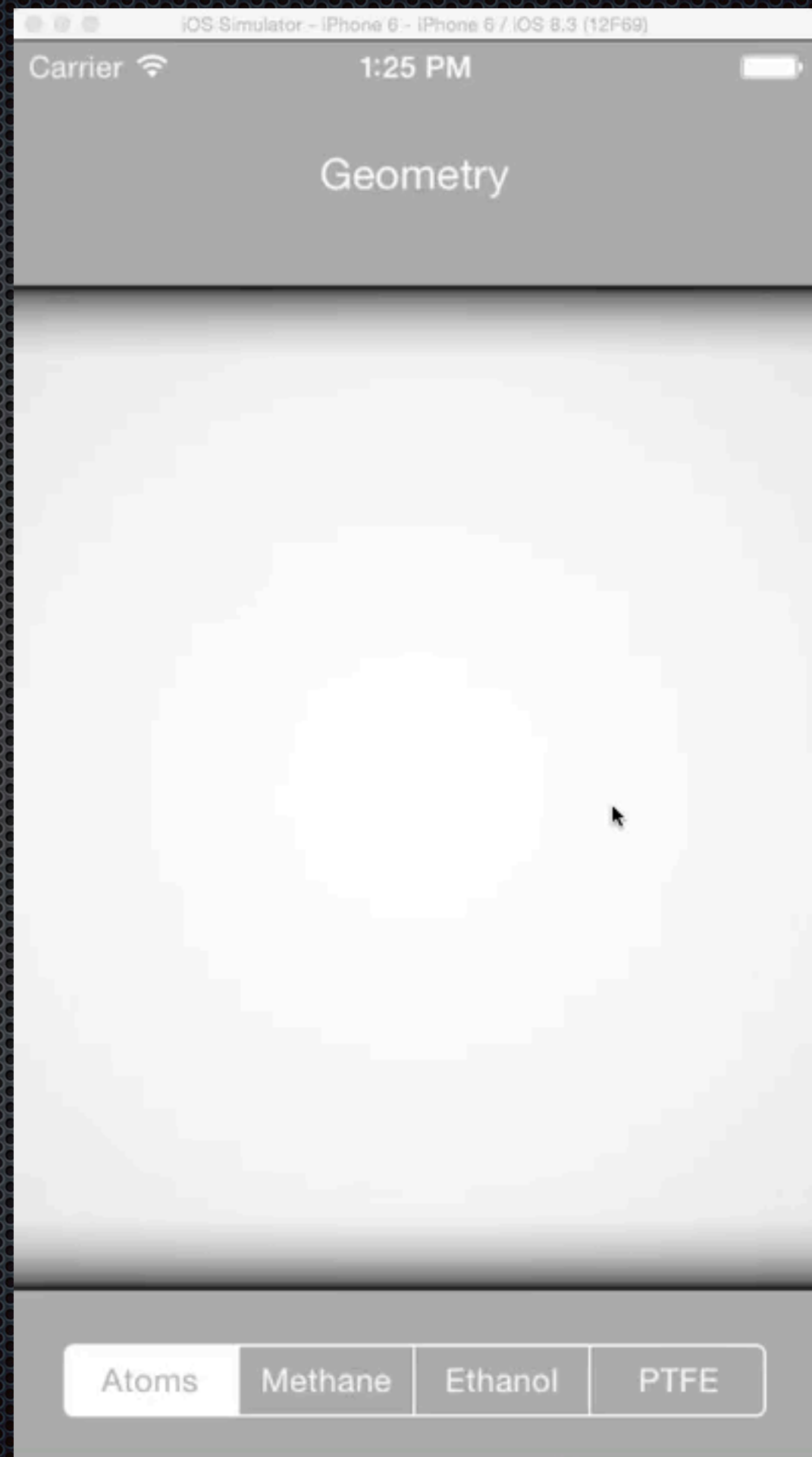
Atoms

Methane

Ethanol

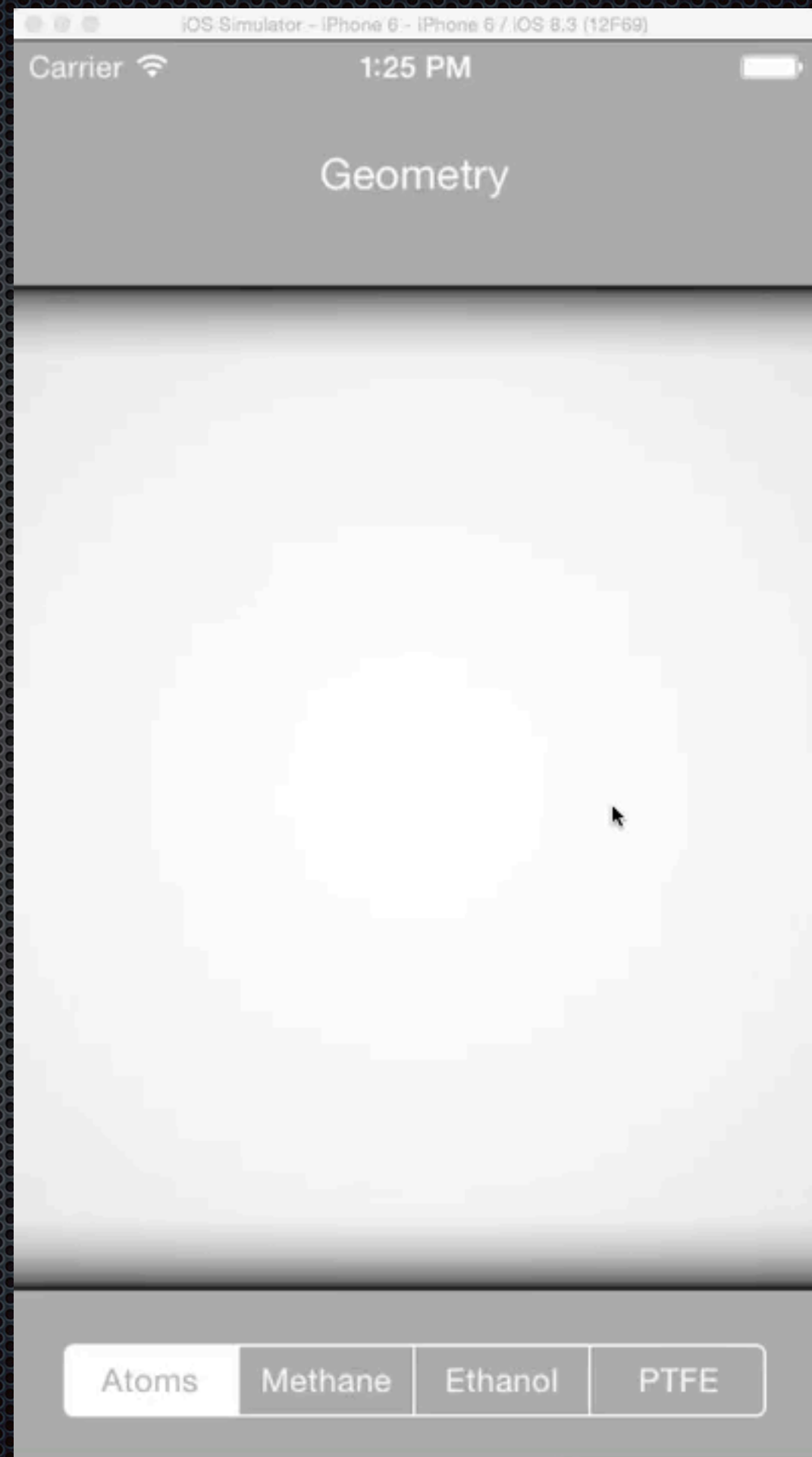
PTFE

Gdy dodamy jeszcze



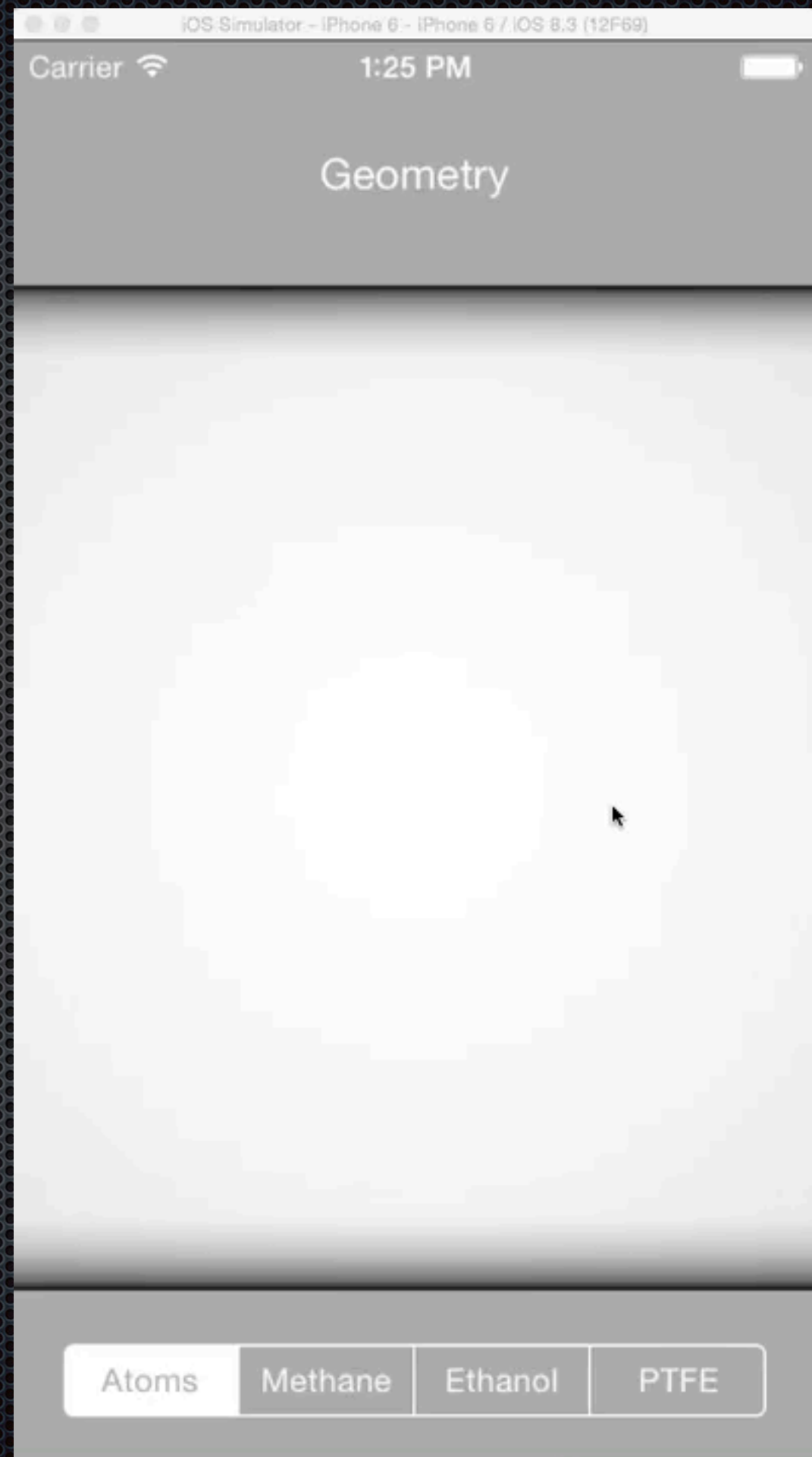
Gdy dodamy jeszcze

```
sceneView.autoenablesDefaultLighting  
sceneView.allowsCameraControl = true
```



Gdy dodamy jeszcze

```
sceneView.autoenablesDefaultLighting  
sceneView.allowsCameraControl = true
```



Dodamy dodatkowe oświetlenie

Dodamy dodatkowe oświetlenie

```
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```

Dodamy dodatkowe oświetlenie

```
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```



Dodamy dodatkowe oświetlenie

```
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```

```
let omniLightNode = SCNNode()
omniLightNode.light = SCNLight()
omniLightNode.light!.type = SCNLightTypeOmni
omniLightNode.light!.color = UIColor(white: 0.75, alpha: 1.0)
omniLightNode.position = SCNVector3Make(0, 50, 50)
scene.rootNode.addChildNode(omniLightNode)
```

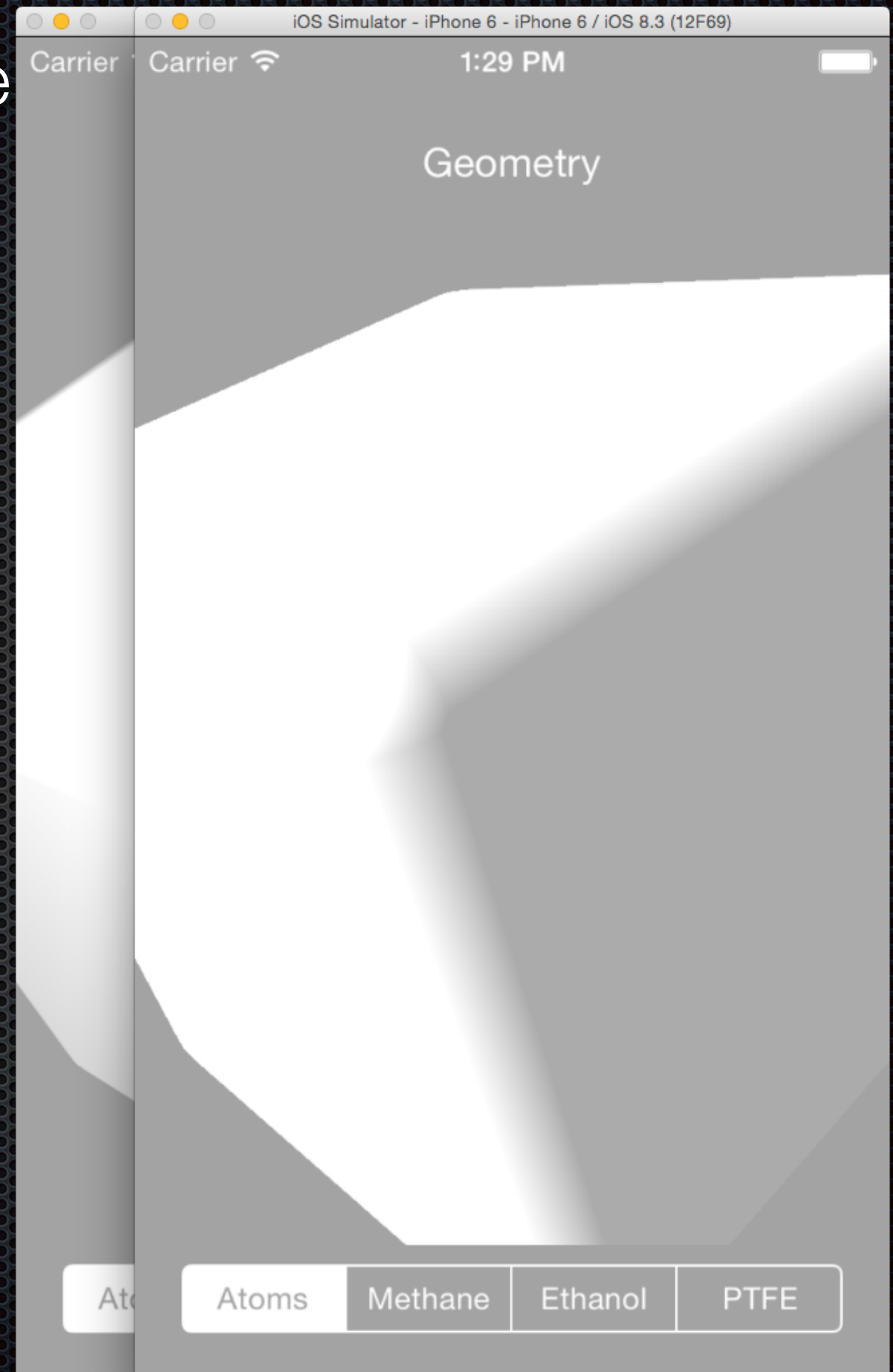


Dodamy dodatkowe oświetlenie

```
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```

```
let omniLightNode = SCNNode()
omniLightNode.light = SCNLight()
omniLightNode.light!.type = SCNLightTypeOmni
omniLightNode.light!.color = UIColor(white: 0.75, alpha: 1.0)
omniLightNode.position = SCNVector3Make(0, 50, 50)
scene.rootNode.addChildNode(omniLightNode)
```



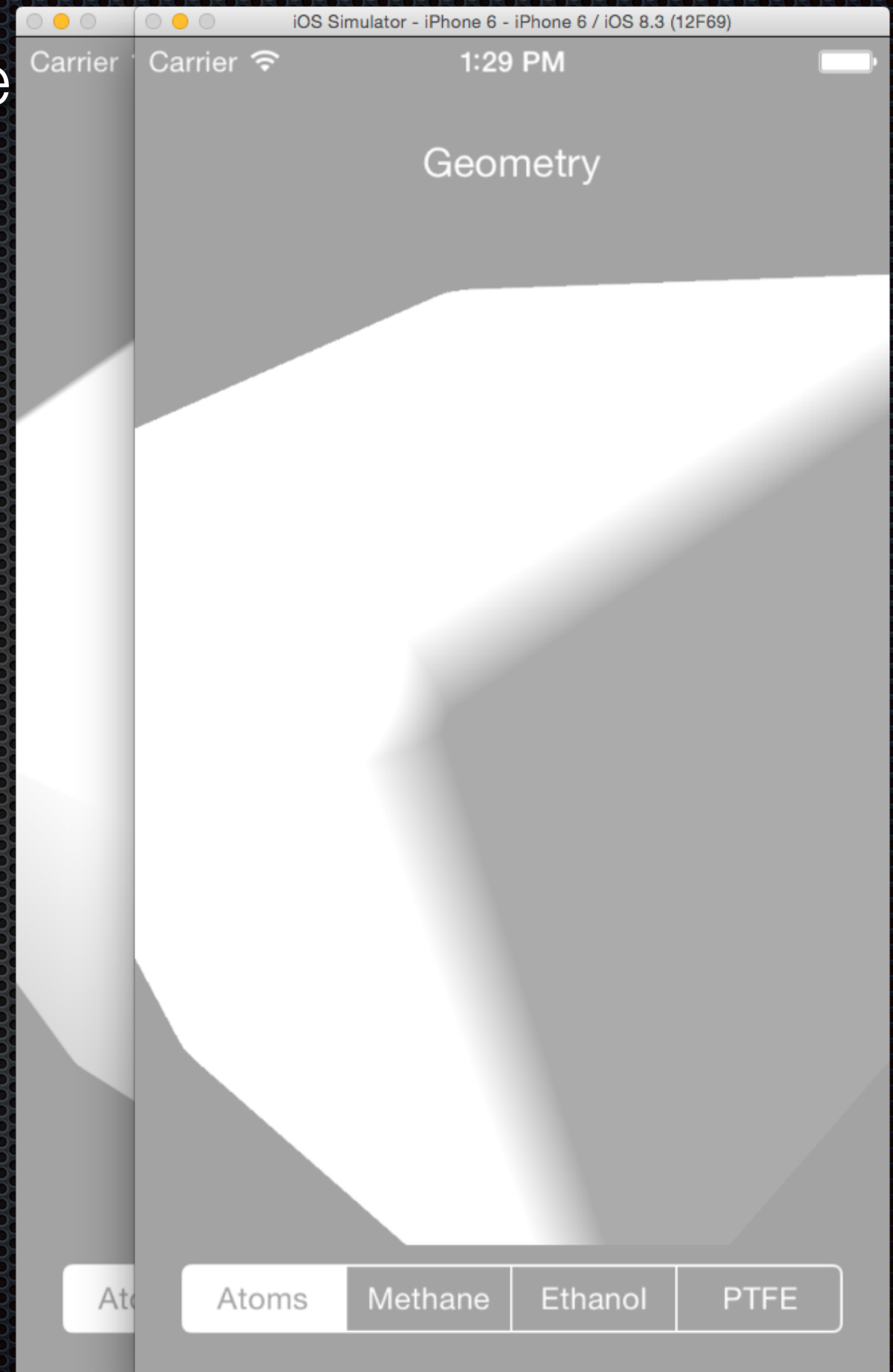
Dodamy dodatkowe oświetlenie

```
// 2
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```

```
let omniLightNode = SCNNode()
omniLightNode.light = SCNLight()
omniLightNode.light!.type = SCNLightTypeOmni
omniLightNode.light!.color = UIColor(white: 0.75, alpha: 1.0)
omniLightNode.position = SCNVector3Make(0, 50, 50)
scene.rootNode.addChildNode(omniLightNode)
```

```
// 3
sceneView.scene = scene
//sceneView.autoenablesDefaultLighting = true
sceneView.allowsCameraControl = true
```



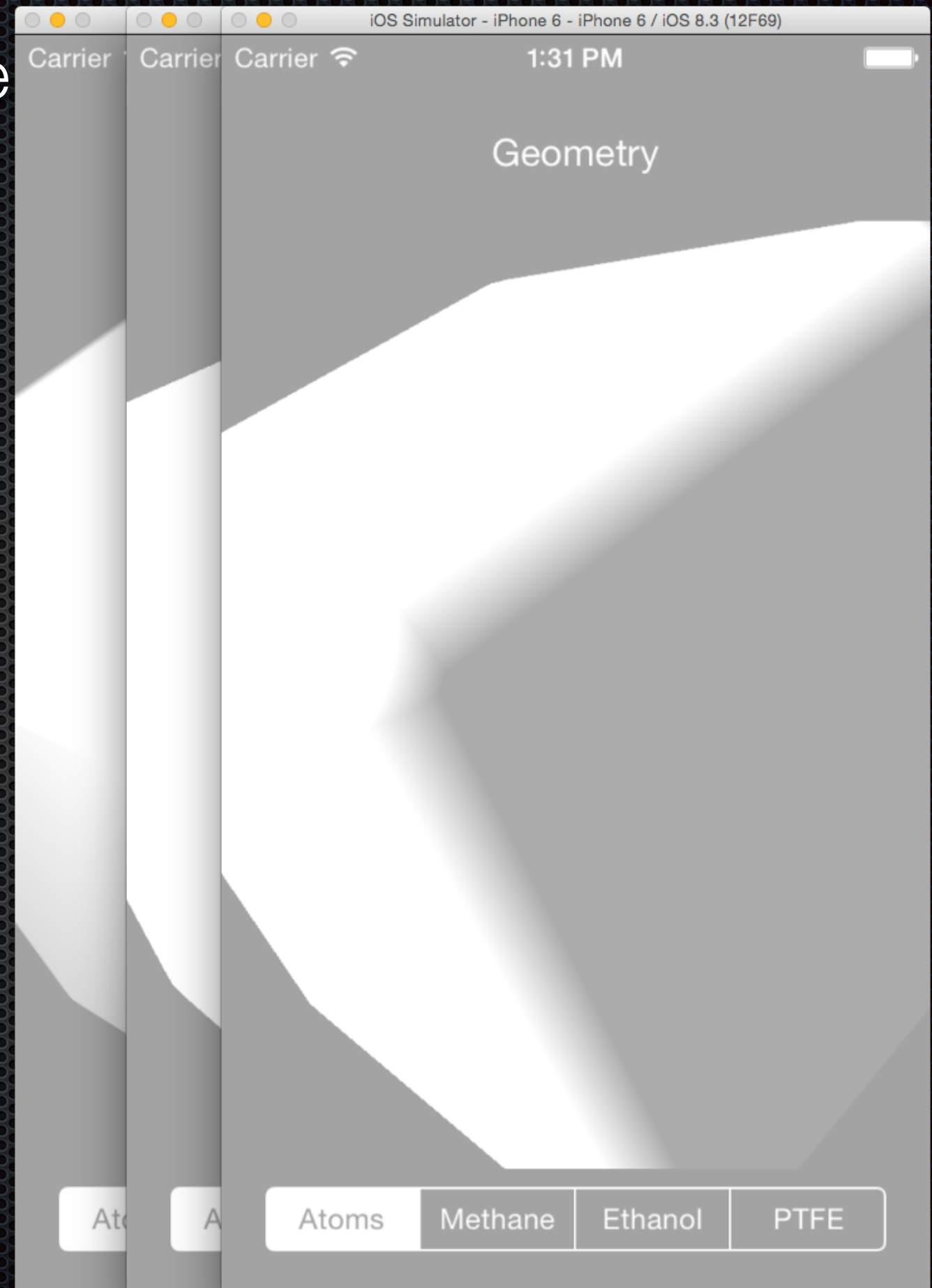
Dodamy dodatkowe oświetlenie

```
let scene = SCNScene()

let ambientLightNode = SCNNode()
ambientLightNode.light = SCNLight()
ambientLightNode.light!.type = SCNLightTypeAmbient
ambientLightNode.light!.color = UIColor(white: 0.67, alpha: 1.0)
scene.rootNode.addChildNode(ambientLightNode)
```

```
let omniLightNode = SCNNode()
omniLightNode.light = SCNLight()
omniLightNode.light!.type = SCNLightTypeOmni
omniLightNode.light!.color = UIColor(white: 0.75, alpha: 1.0)
omniLightNode.position = SCNVector3Make(0, 50, 50)
scene.rootNode.addChildNode(omniLightNode)
```

```
// 3
sceneView.scene = scene
//sceneView.autoenablesDefaultLighting = true
sceneView.allowsCameraControl = true
```



Ustawiamy kamerę

Ustawiamy kamerę

```
let cameraNode = SCNNode()  
cameraNode.camera = SCNCamera()  
cameraNode.position = SCNVector3Make(0, 0, 25)  
scene.rootNode.addChildNode(cameraNode)
```

Ustawiamy kamerę

```
let cameraNode = SCNNode()  
cameraNode.camera = SCNCamera()  
cameraNode.position = SCNVector3Make(0, 0, 25)  
scene.rootNode.addChildNode(cameraNode)
```



Ręczne sterowanie kamerą

Ręczne sterowanie kamerą

- Usuwamy sterowanie automatyczne

Ręczne sterowanie kamerą

- ✦ Usuwamy sterowanie automatyczne

```
//sceneView.allowsCameraControl = true
```

Ręczne sterowanie kamerą

- ✦ Usuwamy sterowanie automatyczne

```
//sceneView.allowsCameraControl = true
```

- ✦ Dodajemy 2 pola w klasie kontrolera

Ręczne sterowanie kamerą

- ✦ Usuwamy sterowanie automatyczne

```
//sceneView.allowsCameraControl = true
```

- ✦ Dodajemy 2 pola w klasie kontrolera

```
// Geometry  
var geometryNode: SCNNode = SCNNode()  
  
// Gestures  
var currentAngle: Float = 0.0
```

Ręczne sterowanie kamerą

- ✦ Usuwamy sterowanie automatyczne

```
//sceneView.allowsCameraControl = true
```

- ✦ Dodajemy 2 pola w klasie kontrolera

```
// Geometry  
var geometryNode: SCNNode = SCNNode()  
  
// Gestures  
var currentAngle: Float = 0.0
```

- ✦ Dodajemy nową metodę

Ręczne sterowanie kamerą

- ✦ Usuwamy sterowanie automatyczne

```
//sceneView.allowsCameraControl = true
```

- ✦ Dodajemy 2 pola w klasie kontrolera

```
// Geometry  
var geometryNode: SCNNode = SCNNode()  
  
// Gestures  
var currentAngle: Float = 0.0
```

- ✦ Dodajemy nową metodę

```
func panGesture(sender: UIPanGestureRecognizer) {  
    let translation = sender.translationInView(sender.view!)  
    var newAngle = (Float)(translation.x)*(Float)(M_PI)/180.0  
    newAngle += currentAngle  
  
    geometryNode.transform = SCNMatrix4MakeRotation(newAngle, 0, 1, 0)  
  
    if(sender.state == UIGestureRecognizerState.Ended) {  
        currentAngle = newAngle  
    }  
}
```

Ręczne sterowanie kamerą

Ręczne sterowanie kamerą

- Dopisujemy kod do tworzenia sceny

Ręczne sterowanie kamerą

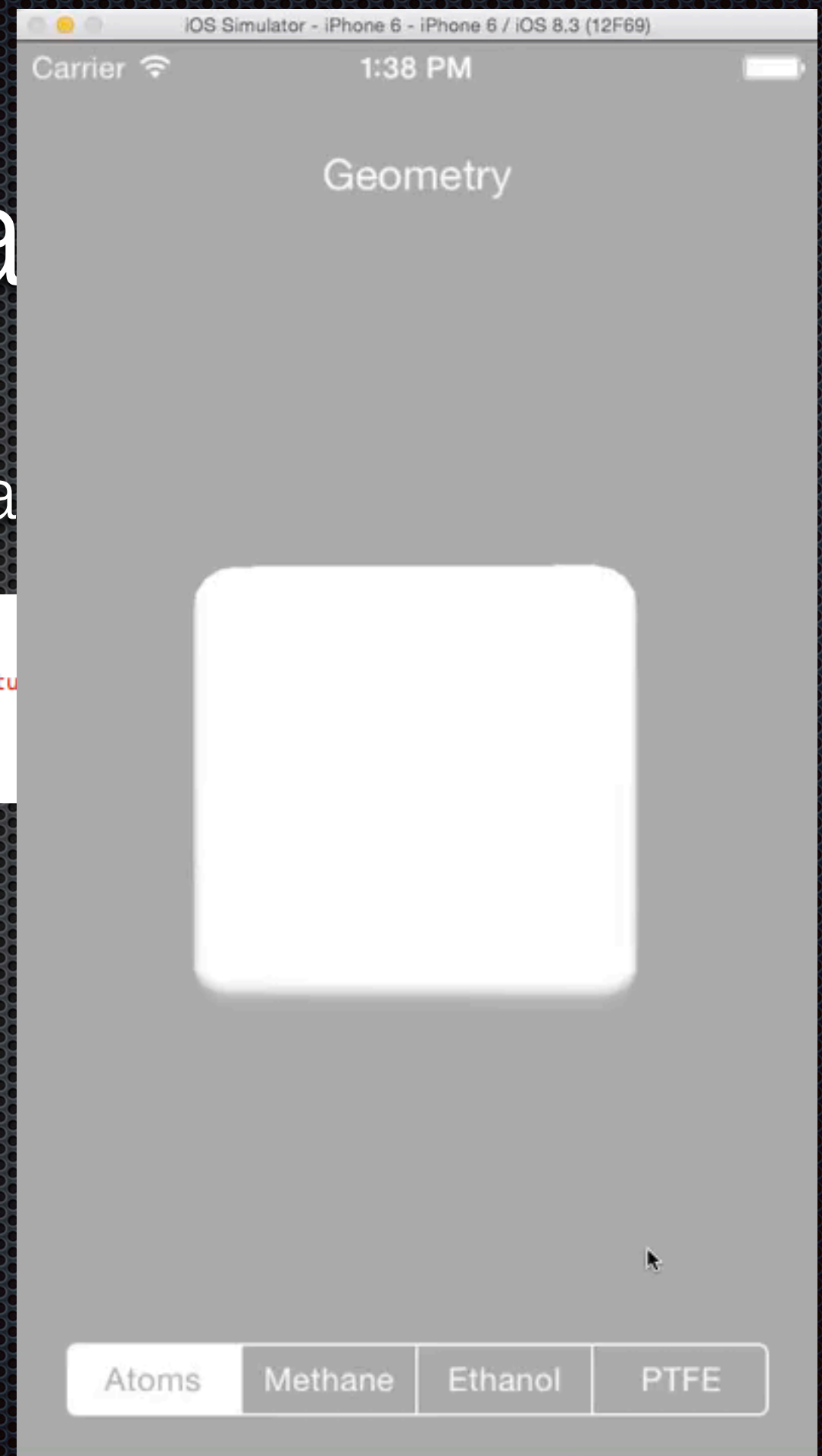
- ✦ Dopisujemy kod do tworzenia sceny

```
geometryNode = boxNode  
  
let panRecognizer = UIPanGestureRecognizer(target: self, action: "panGesture:")  
sceneView.addGestureRecognizer(panRecognizer)  
  
// 3  
sceneView.scene = scene
```

Ręczne sterowanie

- ✦ Dopisujemy kod do tworzenia

```
geometryNode = boxNode  
  
let panRecognizer = UIPanGestureRecognizer(target: self, action: "panGesture")  
sceneView.addGestureRecognizer(panRecognizer)  
  
// 3  
sceneView.scene = scene
```



Tworzymy atomy

Tworzymy atomy

- Tworzymy nową klasę Atoms.swift

Tworzymy atomy

- ✦ Tworzymy nową klasę Atoms.swift

```
import Foundation
import SceneKit

class Atoms {

    class func carbonAtom() -> SCNGeometry {
        // 1
        let carbonAtom = SCNSphere(radius: 1.70)

        // 2
        carbonAtom.firstMaterial!.diffuse.contents = UIColor.darkGrayColor()

        // 3
        carbonAtom.firstMaterial!.specular.contents = UIColor.whiteColor()

        // 4
        return carbonAtom
    }
}
```

Tworzymy atomy

Tworzymy atomy

- i kolejne atomy

Tworzymy atomy

- ✦ i kolejne atomy

```
class func hydrogenAtom() -> SCNGeometry {
    let hydrogenAtom = SCNSphere(radius: 1.20)
    hydrogenAtom.firstMaterial!.diffuse.contents = UIColor.lightGrayColor()
    hydrogenAtom.firstMaterial!.specular.contents = UIColor.whiteColor()
    return hydrogenAtom
}

class func oxygenAtom() -> SCNGeometry {
    let oxygenAtom = SCNSphere(radius: 1.52)
    oxygenAtom.firstMaterial!.diffuse.contents = UIColor.redColor()
    oxygenAtom.firstMaterial!.specular.contents = UIColor.whiteColor()
    return oxygenAtom
}

class func fluorineAtom() -> SCNGeometry {
    let fluorineAtom = SCNSphere(radius: 1.47)
    fluorineAtom.firstMaterial!.diffuse.contents = UIColor.yellowColor()
    fluorineAtom.firstMaterial!.specular.contents = UIColor.whiteColor()
    return fluorineAtom
}
```

Tworzymy atomy

Tworzymy atomy

- ✦ dodajemy kolejną metodę

Tworzymy atomy

- ✦ dodajemy kolejną metodę

```
class func allAtoms() -> SCNNode {  
  let atomsNode = SCNNode()  
  
  let carbonNode = SCNNode(geometry: carbonAtom())  
  carbonNode.position = SCNVector3Make(-6, 0, 0)  
  atomsNode.addChildNode(carbonNode)  
  
  let hydrogenNode = SCNNode(geometry: hydrogenAtom())  
  hydrogenNode.position = SCNVector3Make(-2, 0, 0)  
  atomsNode.addChildNode(hydrogenNode)  
  
  let oxygenNode = SCNNode(geometry: oxygenAtom())  
  oxygenNode.position = SCNVector3Make(+2, 0, 0)  
  atomsNode.addChildNode(oxygenNode)  
  
  let fluorineNode = SCNNode(geometry: fluorineAtom())  
  fluorineNode.position = SCNVector3Make(+6, 0, 0)  
  atomsNode.addChildNode(fluorineNode)  
  
  return atomsNode  
}
```

Kontroler widoku

Kontroler widoku

- ✦ usuwamy pudełko

Kontroler widoku

- ✦ usuwamy pudełko
- ✦ dodajemy tworzenie atomów

Kontroler widoku

- ✦ usuwamy pudełko
- ✦ dodajemy tworzenie atomów

```
override fun viewDidAppear(animated: Bool) {  
    super.viewDidAppear(animated)  
    sceneSetup()  
  
    geometryLabel.text = "Atoms\n"  
    geometryNode = Atoms.allAtoms()  
    sceneView.scene!.rootNode.addChildNode(geometryNode)  
}
```

Tworzymy cząsteczki

Tworzymy cząsteczki

- Dodajemy klasę Molecules.swift

Tworzymy cząsteczki

- ✦ Dodajemy klasę Molecules.swift
- ✦ Dopisujemy metody tworzące cząsteczki

Tworzymy cząsteczki

- ✦ Dodajemy klasę Molecules.swift
- ✦ Dopisujemy metody tworzące cząsteczki

```
import Foundation
import SceneKit

class Molecules {

    class func methaneMolecule() -> SCNNode {
        var methaneMolecule = SCNNode()
        return methaneMolecule
    }

    class func ethanolMolecule() -> SCNNode {
        var ethanolMolecule = SCNNode()
        return ethanolMolecule
    }

    class func ptfeMolecule() -> SCNNode {
        var ptfeMolecule = SCNNode()
        return ptfeMolecule
    }

}
```

Tworzymy cząsteczki

Tworzymy cząsteczki

- Dopisujemy akcję reagującą na zmianę cząsteczki

Tworzymy cząsteczki

- ✦ Dopisujemy akcję reagującą na zmianę cząsteczki

```
// MARK: IBActions
@IBAction func segmentValueChanged(sender: UISegmentedControl) {
    // 1
    geometryNode.removeFromParentNode()
    currentAngle = 0.0

    // 2
    switch sender.selectedSegmentIndex {
    case 0:
        geometryLabel.text = "Atoms\n"
        geometryNode = Atoms.allAtoms()
    case 1:
        geometryLabel.text = "Methane\n(Natural Gas)"
        geometryNode = Molecules.methaneMolecule()
    case 2:
        geometryLabel.text = "Ethanol\n(Alcohol)"
        geometryNode = Molecules.ethanolMolecule()
    case 3:
        geometryLabel.text = "Polytetrafluoroethylene\n(Teflon)"
        geometryNode = Molecules.ptfeMolecule()
    default:
        break
    }

    // 3
    sceneView.scene!.rootNode.addChildNode(geometryNode)
}
```

Tworzymy cząsteczki

Tworzymy cząsteczki

- W Molecules dodajemy metodę pomocniczą

Tworzymy cząsteczki

- ✦ W Molecules dodajemy metodę pomocniczą

```
class func nodeWithAtom(atom: SCNGeometry, molecule: SCNNode, position: SCNVector3) -> SCNNode {  
    let node = SCNNode(geometry: atom)  
    node.position = position  
    molecule.addChildNode(node)  
    return node  
}
```

Tworzymy cząsteczki

- ✦ W Molecules dodajemy metodę pomocniczą

```
class func nodeWithAtom(atom: SCNGeometry, molecule: SCNNode, position: SCNVector3) -> SCNNode {  
    let node = SCNNode(geometry: atom)  
    node.position = position  
    molecule.addChildNode(node)  
    return node  
}
```

- ✦ I tworzymy prawdziwy metan CH₄

Tworzymy cząsteczki

- ✦ W Molecules dodajemy metodę pomocniczą

```
class func nodeWithAtom(atom: SCNGeometry, molecule: SCNNode, position: SCNVector3) -> SCNNode {  
    let node = SCNNode(geometry: atom)  
    node.position = position  
    molecule.addChildNode(node)  
    return node  
}
```

- ✦ I tworzymy prawdziwy metan CH₄

```
class func methaneMolecule() -> SCNNode {  
    var methaneMolecule = SCNNode()  
  
    // 1 Carbon  
    let carbonNode1 = nodeWithAtom(Atoms.carbonAtom(), molecule: methaneMolecule, position: SCNVector3Make(0, 0, 0))  
  
    // 4 Hydrogen  
    let hydrogenNode1 = nodeWithAtom(Atoms.hydrogenAtom(), molecule: methaneMolecule, position: SCNVector3Make(-4, 0, 0))  
    let hydrogenNode2 = nodeWithAtom(Atoms.hydrogenAtom(), molecule: methaneMolecule, position: SCNVector3Make(+4, 0, 0))  
    let hydrogenNode3 = nodeWithAtom(Atoms.hydrogenAtom(), molecule: methaneMolecule, position: SCNVector3Make(0, -4, 0))  
    let hydrogenNode4 = nodeWithAtom(Atoms.hydrogenAtom(), molecule: methaneMolecule, position: SCNVector3Make(0, +4, 0))  
  
    return methaneMolecule  
}
```



Atoms



Atoms

Methane

Ethanol

PTFE

Carrier

1:54 PM



Atoms



Atoms

Methane

Ethanol

PTFE

SpriteKit i SceneKit wg strony
<http://www.raywenderlich.com>

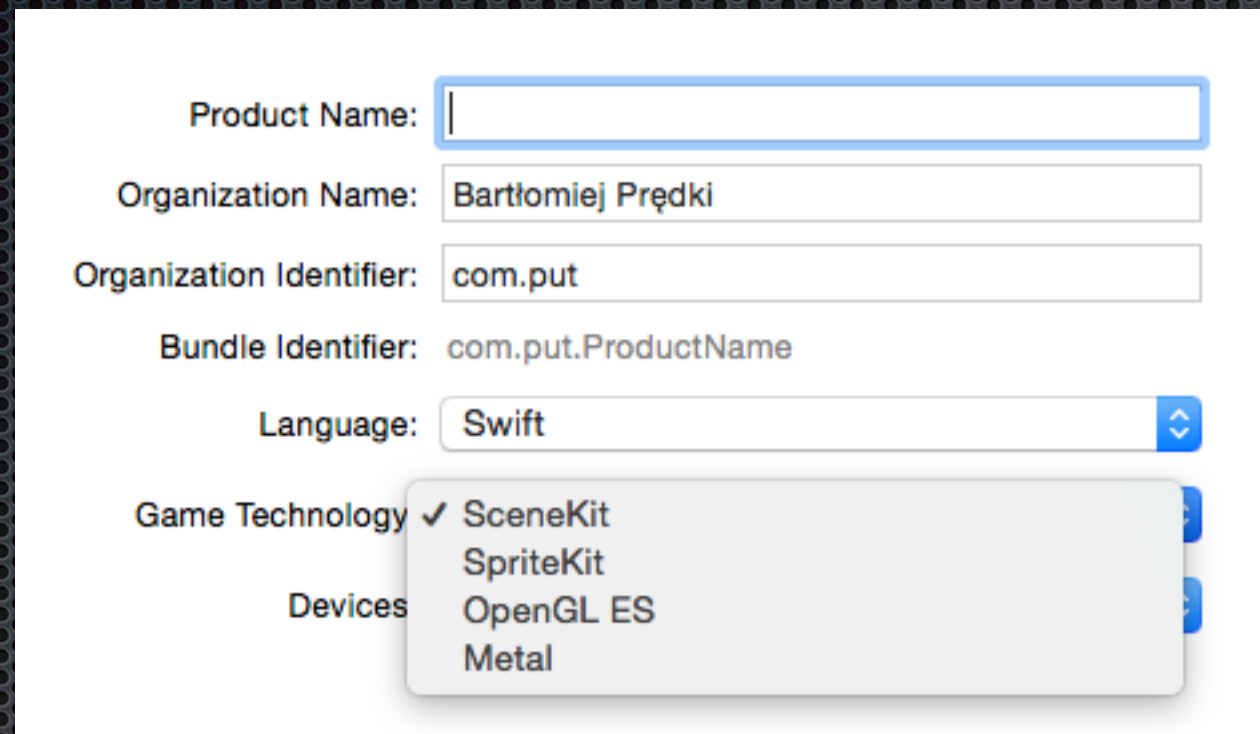
Inne technologie

- ✦ OpenGL
- ✦ Metal

Inne technologie

- ✦ OpenGL

- ✦ Metal



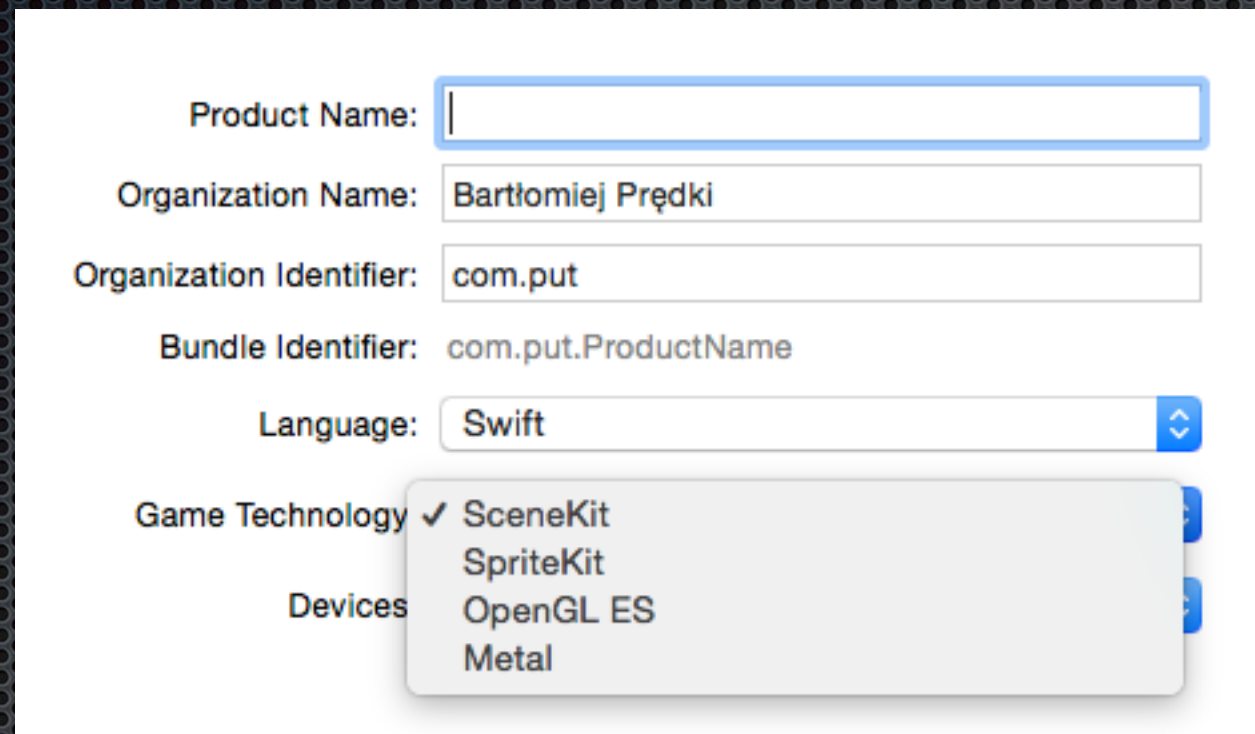
A screenshot of the Xcode project settings interface. The 'Product Name' field is empty. The 'Organization Name' field contains 'Bartłomiej Prędko'. The 'Organization Identifier' field contains 'com.put'. The 'Bundle Identifier' field contains 'com.put.ProductName'. The 'Language' dropdown menu is set to 'Swift'. The 'Game Technology' dropdown menu is open, showing a list of options: 'SceneKit' (selected with a checkmark), 'SpriteKit', 'OpenGL ES', and 'Metal'. The 'Devices' section is partially visible below the dropdown menu.

Product Name:	
Organization Name:	Bartłomiej Prędko
Organization Identifier:	com.put
Bundle Identifier:	com.put.ProductName
Language:	Swift
Game Technology:	SceneKit SpriteKit OpenGL ES Metal
Devices:	

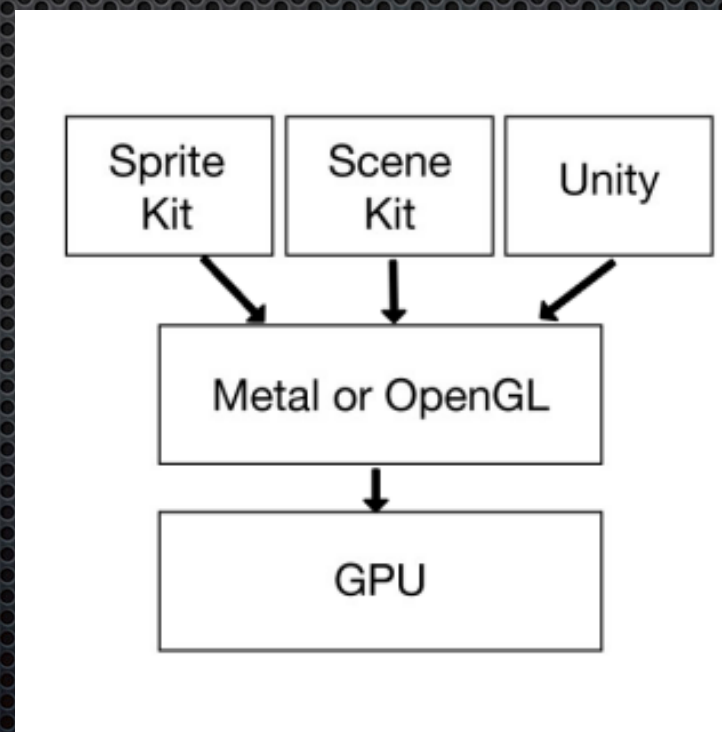
Inne technologie

- ✦ OpenGL

- ✦ Metal



A screenshot of the Xcode project settings window, specifically the 'Game Technology' section. The 'Product Name' field is empty. The 'Organization Name' is 'Bartłomiej Prędko'. The 'Organization Identifier' is 'com.put'. The 'Bundle Identifier' is 'com.put.ProductName'. The 'Language' is set to 'Swift'. The 'Game Technology' dropdown menu is open, showing options: 'SceneKit' (selected with a checkmark), 'SpriteKit', 'OpenGL ES', and 'Metal'. The 'Devices' section is partially visible below.



Do zobaczenia

速