# iPhone Game Programming for Kids (Sprite Kit).

## What is Sprite Kit?

- **Sprite kit is an awesome framework built into XCode 5 and 6. Apple at last has seen the Game Revolution they have made with the touch devices, and they made Sprite Kit which again is a Game Revolution with it comes sprite support instead of UIImageView's you would use, support for cool special effects like filters, videos, and masking, an integrated physics library and even particle effects like explosions and much much more.**

## What we will be achieving!

**We are making a simple 2D game where you swipe right or left and the baddies disappear and the more baddies you defeat the faster it gets.**
**You swipe right and left if you swipe right the right baddy disappears then if you swipe left the left baddie disappears. The speed increases by 0.2 each time.**

## Why choose Sprite Kit?

**It is built into XCode and iOS 7 and above. Built in physics, particle effects and no need for download it just comes in built. Even has a visual editor which comes with the project when you make it. A Visual particle editor where you visually see your particle update as you change the values, and Texture Atlases. It saves time completely by simplifying hard tasks that you would normally do in Unity and others. Looks like Apple has proved them selves again.**

## What makes Sprite Kit different from the rest?

**If you didn't know there are heaps of frameworks to use like Corona SDK, Cocos 2d, Unity and millions of others just dedicated to games and they have their own positives.**

**If you are a beginner and only want to get into App Store and not Android Store, the Sprite Kit is for you but if you are looking at a bigger audience then you look at Unity or something like that, one project and you have got it in all the app stores.**

**3D support, yeah sorry guys no 3D games with Sprite Kit you'r going in the wrong direction if you want to make a 3D game leave this tutorial and and go to this tutorial - [http://www.raywenderlich.com/25205/beginning-unity-3d-for-ios-part-13](http://www.raywenderlich.com/25205/beginning-unity-3d-for-ios-part-13)**

**Thats the one you need.**

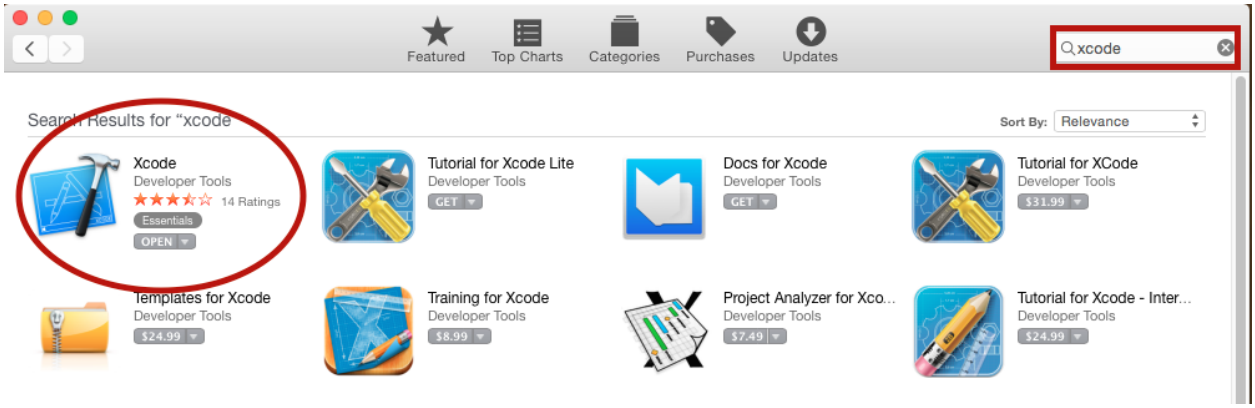
**Custom OpenGL code is not supported in Sprite Kit… Yet. :( So sorry for that guys you would be going towards Unity and Cocos2d-x but like I said al frameworks have their own positives and negatives. But if you just want to make a game then follow me.**

# Getting Started!

**-What do I need?**
**You will need a Mac, any Mac is alright , Macbook pro, iMac, Macbook Air. You will also need XCode, don't worry that will be covered here.**
**Download the Graphics used in this game!**
**Link - (https://github.com/AjayVenkat/Monsters)**
**The graphics used in the game are from the website OpenGameArt.org**
**Game difficulty level - Beginners**

# Installing XCode 6 (XCode 5 is alright).

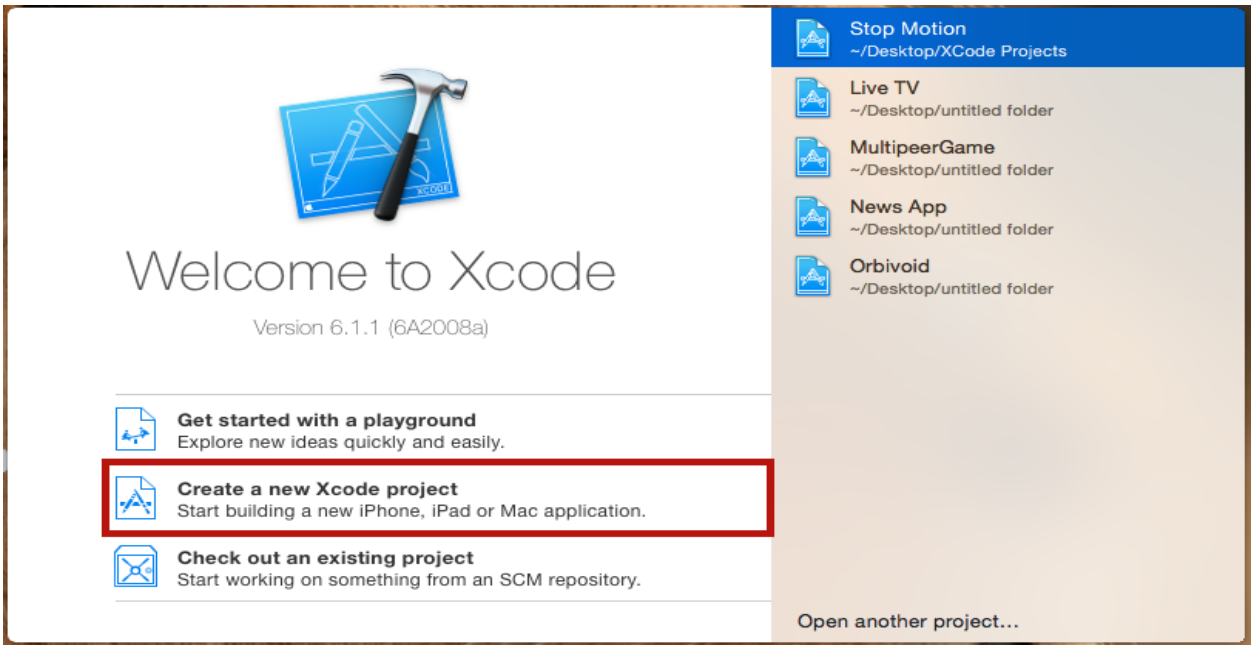
**If you don't have XCode or if you have under XCode 5 then download XCode 5 like this. If you have App Store on your Mac, then open it up and type XCode up top in the search bar. Then click install under XCode. (It says open on mine because I have it already). Now you have to wait…**




**You can also get XCode from the internet. By searching for it on Google.**

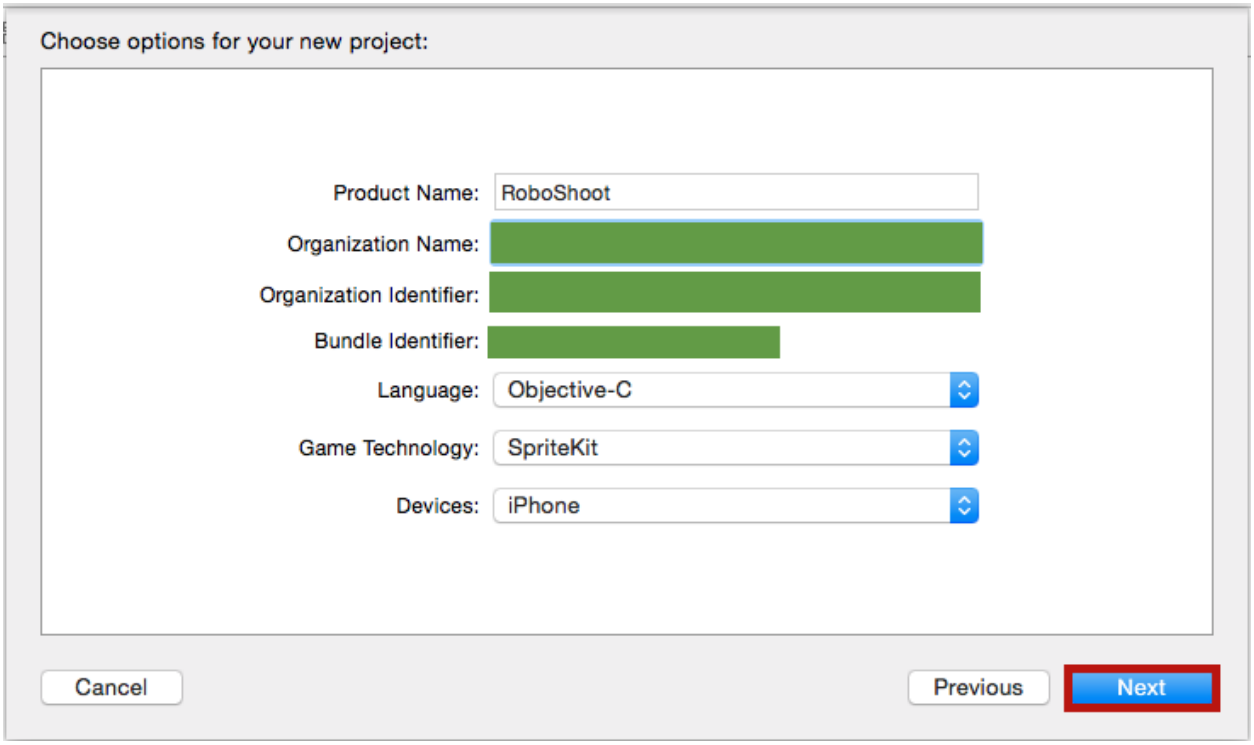
# Lets start. Create our project!

**Now that you have XCode installed open it up and create a new XCode project.**




**XCode will open up.**
**Now when the XCode window opens in the slide out menu click Application under iOS and then click on Game, on XCode 5 it will say something else but it will be the same Icon, click on the game and then click Next.**

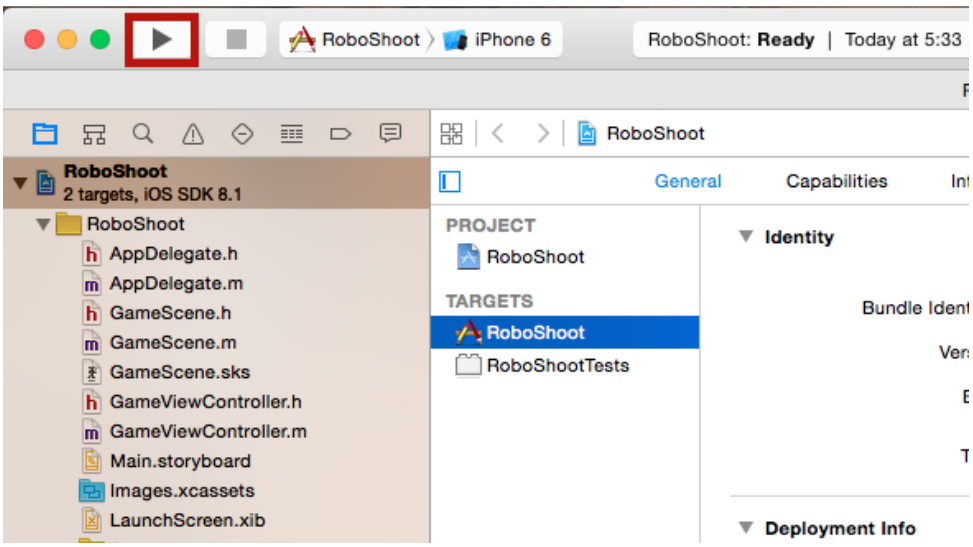
**Enter RoboShoot for the Product Name, iPhone for Devices, your Organization Name (Company Name), Bundle Identifier if you have not paid 99 dollars for iOS devoloper program  yet don't worry (leave that empty). If you did pay then enter your bundle identifier you made. Make your language Objective-C, Game Technology Sprite Kit. I have blocked out my information. Then after you click next save it to somehwere on**




**your drive.**

## Looking at the default project given by Sprite Kit.

**Before we jump into the project we are going to build the project , Sprite Kit comes with a default project and we are going to view that, click on the Play button above with iPhone 6 as the device.**

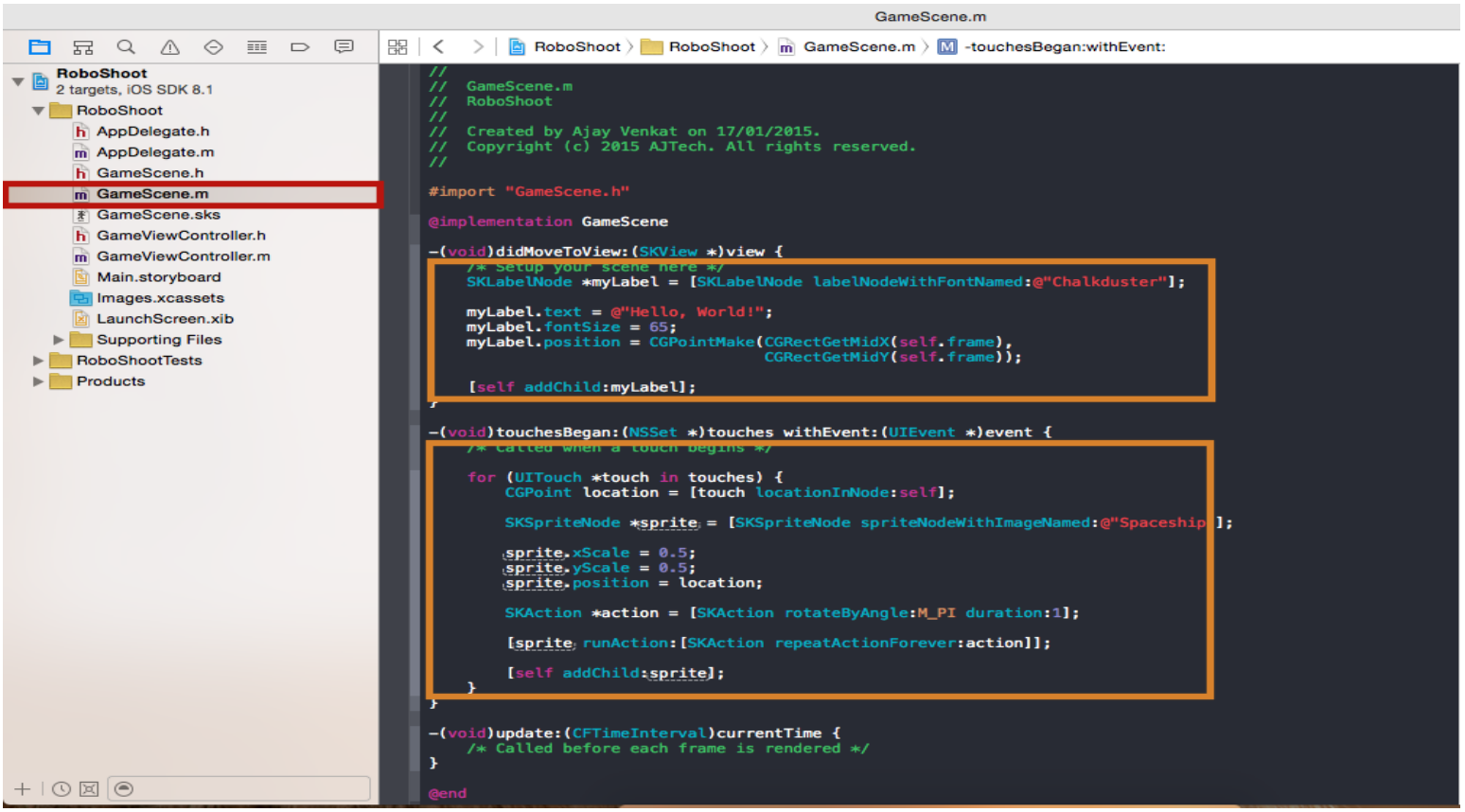


It should look like this, now try tapping anywhere on the screen. A rocket appears. The label is a SKLabelNode, when you click on the screen the rocket that appears is a SKSpriteNode.


This is created because Apple has created some default code for you already but it is easy to take out, which we are going to.

We are going to head over to GameScene.m and delete everything inside

```
-(void)didMoveToView:(SKView *)view {
//Delete everything here
}
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
 //Delete everything here
}
```

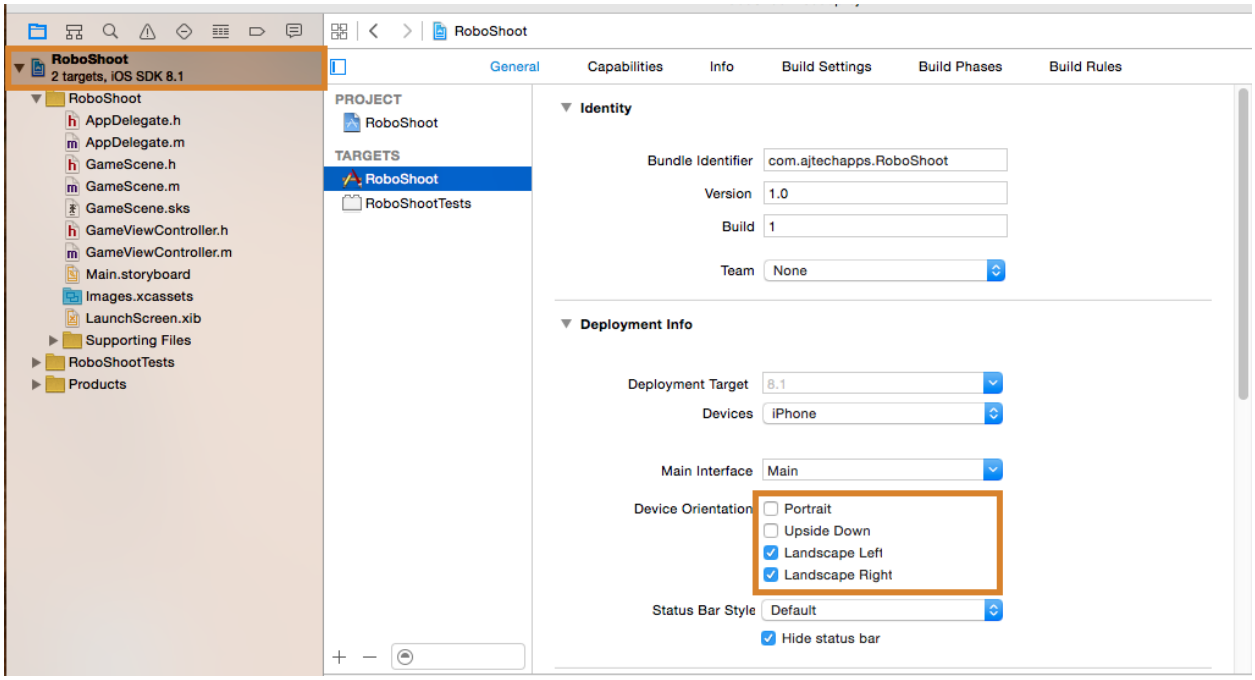


# Making the application landscape.

**There are some bugs with Sprite Kit when we make the App landscape, you can do this experiment yourself, get the x frame size and y frame size and programmatically enter it in to NSLog. If you don't know how to do that do this -**

```
-(void)didMoveToView:(SKView *)view {

    int x = self.frame.size.width;
    int y = self.frame.size.height;

    NSLog(@"X : %d Y : %d", x, y);

}
```

**First run in portrait note the X and Y down , then run it again but this time in landscape by changing Device orientation only to landscape modes.**




**Now when you run your project look in your log again has it changed? No it hasn't Sprite Kit thinks you are still running in portrait mode that means if you set a sprite to a landscape position then it won't show.**

# Fixing the bug!

**Its rather a easy fix, all you have to do to fix it is go over to your GameViewController.m.**

**Do you see this code?  -**

```
- (void)viewDidLoad
{
    [super viewDidLoad];
// Configure the view.
    SKView * skView = (SKView *)self.view;
    skView.showsFPS = YES;
    skView.showsNodeCount = YES;
    /* Sprite Kit applies additional optimizations to improve rendering performance */
    skView.ignoresSiblingOrder = YES;
    // Create and configure the scene.
    GameScene *scene = [GameScene unarchiveFromFile:@"GameScene"];
    scene.scaleMode = SKSceneScaleModeAspectFill;
    // Present the scene.
    [skView presentScene:scene];
}
```

**Delete that code and then add this code in its place.**

```
- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
 // Configure the view.
    SKView * skView = (SKView *)self.view;
    skView.showsFPS = YES;
    skView.showsNodeCount = YES;
    /* Sprite Kit applies additional optimizations to improve rendering performance */
    skView.ignoresSiblingOrder = YES;
     // Create and configure the scene.
    GameScene *scene = [GameScene unarchiveFromFile:@"GameScene"];
    scene.scaleMode = SKSceneScaleModeAspectFill;
       // Present the scene.
    [skView presentScene:scene];
}
```

**What is viewDidLayoutSubviews?**
**When the bounds change for a view controller's view, the view adjusts the positions of its subviews and then the system calls this method. However, this method being called *does not* indicate that the individual layouts of the view's subviews have been adjusted. Each subview is responsible for adjusting its own layout.**

**What i did was change the viewDidLoad to a viewDidLayoutSubviews**
**Make sure you do that step its very important. Now you have a landscape app.**

# Getting our hero ready.

**Go to our GameScene.m, and under #import "GameScene.h" Add a SKSpriteNode. Like so -**

**SKSpriteNode *hero;**

```
#import "GameScene.h"
SKSpriteNode *hero;
```

**If you hadn't already downloaded the graphics do so now.**
**https://github.com/AjayVenkat/Monsters**
**Once again these graphics are not mine they are from OpenGameArt.org.**

**Now go to your GameScene.h file and add this code -**

**-(void)spawnHero;**

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

@interface GameScene : SKScene
-(void)spawnHero;
@end
```

**SKSpriteNode is just a sprite which we are going to add a image to.**

# Time to spawn our hero.

**Ok now navigate to our GameScene.m and inside the didMoveToView method add this single line of code -**

**[self spawnHero];**

**All I am doing here is making our method start.**

Now when the view loads up it will call that method now we have to make it.

In your GameScene.m code add this code -

```
-(void)spawnHero {
//Our spawn code will go here
}
```

Delete the //Our spawn code will go here and replace with the following code I will explain what I am doing after the code.

```
-(void)spawnHero {
    self.backgroundColor = [SKColor grayColor];

    hero = [SKSpriteNode spriteNodeWithImageNamed:@"Hero2"];
    hero.position = CGPointMake(self.frame.size.width/2, self.frame.size.height/2);
    hero.size = CGSizeMake(92, 128);
    hero.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:92];
    hero.physicsBody.affectedByGravity = NO;

    [self addChild:hero];



}
```

First I set the background color of the view to a grayColor.

Second I change the image of the sprite to the Hero2 file which you don't need to write the extension because it is png format.

Third I set the hero's position to the middle of the screen, then I change the size of the hero.

Fourth I add a physics body.

Fifth I make the sprite not affected to gravity. We will focus more on the physics when we get to collision.

Easy enough right?

Ok we have the sprite ready. Now we need to create a floor.

# Creating the floor so the hero can land when jumps.

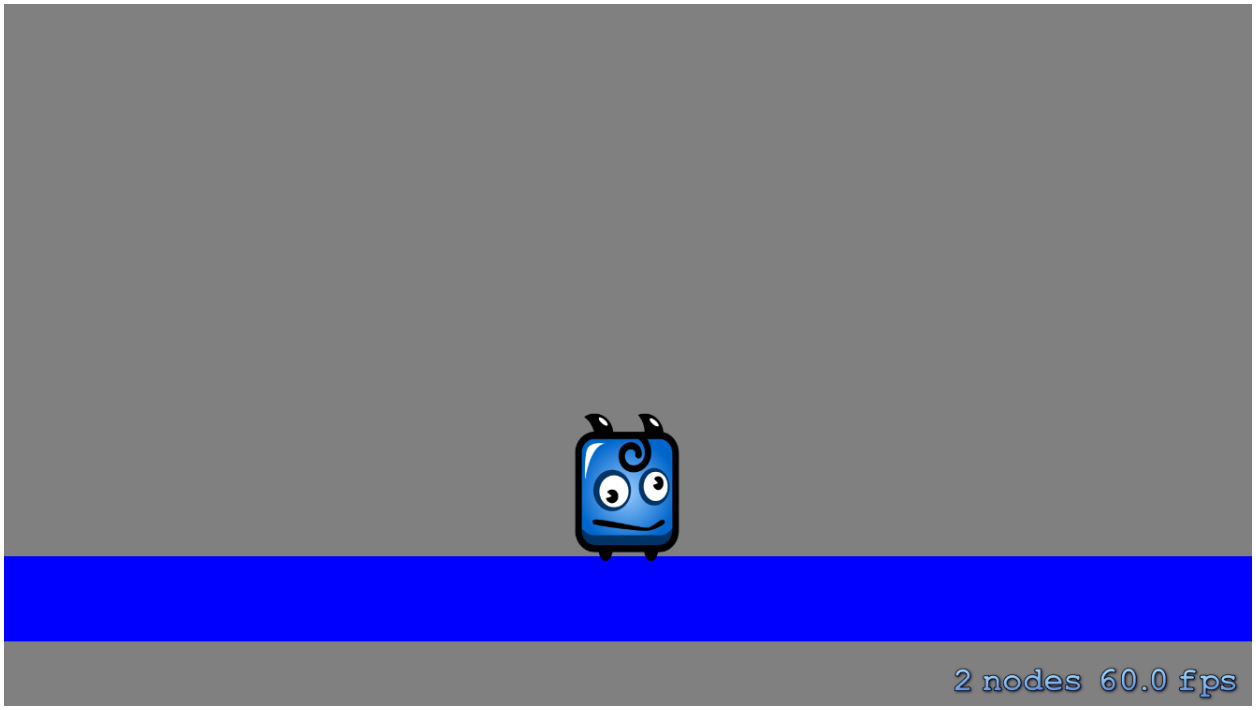
Ok lets create the floor, lets add the floor in the spawnHero method as well because that gets called only once.
First add a SKSpriteNode under the #import "GameScene.h" like this -

```
SKSpriteNode *landingSpot;
```

Ok now for adding it. I dont really have graphics for this because I want it to be blue. Delete the whole -(void)spawnSprite and replace with this.

```
-(void)spawnHero {
    self.backgroundColor = [SKColor grayColor];

    hero = [SKSpriteNode spriteNodeWithImageNamed:@"Hero2"];
    hero.position = CGPointMake(self.frame.size.width/2, self.frame.size.height/2);
    hero.size = CGSizeMake(92, 128);
    hero.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
    hero.physicsBody.affectedByGravity = YES;
    hero.physicsBody.mass = 0;

    [self addChild:hero];


    landingSpot = [SKSpriteNode node];
    landingSpot.position = CGPointMake(self.frame.size.width/2, self.frame.size.height/2
- 200);
    landingSpot.size = CGSizeMake(self.frame.size.width, 70);
    CGSize size = CGSizeMake(self.frame.size.width, 70);

    landingSpot.physicsBody = [SKPhysicsBody bodyWithRectangleOfSize:size];
    landingSpot.physicsBody.affectedByGravity = NO;
    landingSpot.physicsBody.mass = 100000000;

    landingSpot.physicsBody.resting = YES;

    landingSpot.color = [SKColor blueColor];

    [self addChild:landingSpot];  }
```

## Running the application so far!




**You can kind of see before when the application start the sprite drops from the middle of the screen to the floor. If you want you can explain with the code in spawnHero , change the mass of the floor and watch it sink, thats the reason I kept the mass of the floor to 1000 because then it would be too light and it would drop out of the screen.**

## Getting our monsters ready.

**Now lets get on with the monsters , same drill we are going to create the void again.
Go to your GameScene.h and under your last -(void)spawnHero;
add -**

```
-(void)spawnEnemyLeft;
-(void)spawnEnemyRight;
```

**Now go back to GameScene.m and like usual under #import "GameScene.h" add -**

```
SKSpriteNode *enemyR;
SKSpriteNode *enemyL;
```

## Time to spawn our enemy.

**Like before inside the -(void)didMoveToView {}, add this one line of code -**

```
[self spawnEnemyRight];
[self spawnEnemyLeft];
```

**Now in your GameScene.m add this code under -(void)spawnHero {} ,**

```
-(void)spawnEnemyRight {
   enemyR = [SKSpriteNode spriteNodeWithImageNamed:@"Monster3"];
   enemyR.position = CGPointMake(0 + 50, self.frame.size.height/2);
   enemyR.size = CGSizeMake(92, 128);
   enemyR.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
   enemyR.physicsBody.affectedByGravity = YES;
   enemyR.physicsBody.mass = 0;
   enemyR.name = @"enemyR";
[self addChild:enemyR];
}

-(void)spawnEnemyLeft {
enemyL = [SKSpriteNode spriteNodeWithImageNamed:@"Monster2"];
   enemyL.position = CGPointMake(self.frame.size.width - 50, self.frame.size.height/2);
   enemyL.size = CGSizeMake(92, 128);
   enemyL.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
enemyL.physicsBody.affectedByGravity = YES;
enemyL.physicsBody.mass = 0;
       enemyR.name = @"enemyL";

   [self addChild:enemyL];

}
```

**The only different thing we did here is we added a name for the sprites. But we don't need them anyway.**

# Making our hero jump!

**In our GameScene.m add this code if you already have touchesBegan delete it -**

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {

   CGPoint jumpPoint = CGPointMake(hero.position.x+0, hero.position.y + 250);

   SKAction *jump = [SKAction moveTo:jumpPoint duration:0.3];

   [hero runAction:jump];
}
```

**First in that code we create a point in the screen called a CGPoint then we make the point with CGPointMake we are making the point to be 250 higher than the hero already**

**is, then we make a SKAction and we use the moveTo and then the CGPoint we made earlier and the duration 0.3 so it smoothens the transition and then we run the action. The physics take cares of the rest.**

## Oh no! Now he can double jump! We don't want that to happen.

**Just for the collision between the user and the wall I am using a cheap collision method but for the enemies and the user I will use the proper collision method. I am using the bad collision method because we want to save time.**

**In our update method -**

```
-(void)update:(CFTimeInterval)currentTime {
    /* Called before each frame is rendered */



}
```

**We are going to add some code here.**

**But before that, we are going to add a BOOL and of course we are going to add it under our #import "GameScene.h",**

```
BOOL isTouching;
```

**Ok so now we have created that we can go back to our update, and add this code -**

```
-(void)update:(CFTimeInterval)currentTime {
    /* Called before each frame is rendered */

    if (CGRectIntersectsRect(hero.frame, landingSpot.frame)) {

        isTouching = YES;

    }

    else if (!CGRectIntersectsRect(hero.frame, landingSpot.frame)) {

        isTouching = NO;

    }
    }
```

**Now we have that out of the way one more thing we need to change our touchesBegan , delete your touchesBegan code and replace with this -**

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
if (isTouching == YES) {

   CGPoint jumpPoint = CGPointMake(hero.position.x+0, hero.position.y + 250);
SKAction *jump = [SKAction moveTo:jumpPoint duration:0.3];
 [hero runAction:jump];

 }
  else if (isTouching == NO) {

      }
   }
```

**Now we need the enemies to make some movement!!**

## Moving the enemies!

**We will move them when the touches Begin.**

```
-(void)spawnEnemyRight {
   enemyR = [SKSpriteNode spriteNodeWithImageNamed:@"Monster3"];
   enemyR.position = CGPointMake(0 + 50, self.frame.size.height/2 - 150);
   enemyR.size = CGSizeMake(92, 128);
   enemyR.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
   enemyR.physicsBody.affectedByGravity = YES;
   enemyR.physicsBody.mass = 0;
   enemyR.name = @"enemyR";

   [self addChild:enemyR];
 CGPoint movePointR = CGPointMake(hero.position.x, enemyR.position.y);

   SKAction *jumpR = [SKAction moveTo:movePointR duration:3];

   [enemyR runAction:jumpR];
}

-(void)spawnEnemyLeft {

   enemyL = [SKSpriteNode spriteNodeWithImageNamed:@"Monster2"];
   enemyL.position = CGPointMake(self.frame.size.width - 50, self.frame.size.height/2 -
150);
   enemyL.size = CGSizeMake(92, 128);
   enemyL.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
enemyL.physicsBody.affectedByGravity = YES;
enemyL.physicsBody.mass = 0;
       enemyR.name = @"enemyL";

   [self addChild:enemyL];

   CGPoint movePointL = CGPointMake(hero.position.x, enemyL.position.y);

   SKAction *jumpL = [SKAction moveTo:movePointL duration:3];

   [enemyL runAction:jumpL];

      }
```

**We don’t want collision between the 2 monsters only between the monsters and the user but we are not up to the collision part yet , now we have to make sure the user can tap the monster to delete it.**

# Make sure your on track!

**Just incase you have lost any code I will give whole GameScene.m code.**

```
#import "GameScene.h"
SKSpriteNode *hero;
SKSpriteNode *landingSpot;
SKSpriteNode *enemyR;
SKSpriteNode *enemyL;
NSMutableArray *enemyArray;
BOOL isTouching;
@implementation GameScene

-(void)didMoveToView:(SKView *)view {

    [self spawnHero];
    [self spawnEnemyRight];
    [self spawnEnemyLeft];


}

-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event {
 if (isTouching == YES) {



        [self startR];
        [self startL];

    CGPoint jumpPoint = CGPointMake(hero.position.x+0, hero.position.y + 250);

    SKAction *jump = [SKAction moveTo:jumpPoint duration:0.5];

    [hero runAction:jump];

}

    else if (isTouching == NO) {

        // Do Nothing!
    }

-(void)update:(CFTimeInterval)currentTime {
    /* Called before each frame is rendered
    if (CGRectIntersectsRect(hero.frame, landingSpot.frame)) {
```

```
        isTouching = YES;

    }

    else if (!CGRectIntersectsRect(hero.frame, landingSpot.frame)) {

        isTouching = NO;

    }


}

-(void)spawnHero {
    self.backgroundColor = [SKColor grayColor];

    hero = [SKSpriteNode spriteNodeWithImageNamed:@"Hero2"];
    hero.position = CGPointMake(self.frame.size.width/2, self.frame.size.height/2);
    hero.size = CGSizeMake(92, 128);
    hero.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
    hero.physicsBody.affectedByGravity = YES;
    hero.physicsBody.mass = 0;

    [self addChild:hero];


    landingSpot = [SKSpriteNode node];
    landingSpot.position = CGPointMake(self.frame.size.width/2, self.frame.size.height/2 - 200);
    landingSpot.size = CGSizeMake(self.frame.size.width, 70);
    CGSize size = CGSizeMake(self.frame.size.width, 70);

    landingSpot.physicsBody = [SKPhysicsBody bodyWithRectangleOfSize:size];
    landingSpot.physicsBody.affectedByGravity = NO;
    landingSpot.physicsBody.mass = 100000000;

    landingSpot.physicsBody.resting = YES;

    landingSpot.color = [SKColor blueColor];

    [self addChild:landingSpot];




}

-(void)spawnEnemyRight {
    enemyR = [SKSpriteNode spriteNodeWithImageNamed:@"Monster3"];
    enemyR.position = CGPointMake(0 + 50, self.frame.size.height/2 - 150);
    enemyR.size = CGSizeMake(92, 128);
    enemyR.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
    enemyR.physicsBody.affectedByGravity = YES;
    enemyR.physicsBody.mass = 0;
    enemyR.name = @"enemyR";

    [self addChild:enemyR];
```

```
    CGPoint movePointR = CGPointMake(hero.position.x, enemyR.position.y);

    SKAction *jumpR = [SKAction moveTo:movePointR duration:3];

    [enemyR runAction:jumpR];




}

-(void)spawnEnemyLeft {

    enemyL = [SKSpriteNode spriteNodeWithImageNamed:@"Monster2"];
    enemyL.position = CGPointMake(self.frame.size.width - 50, self.frame.size.height/2 - 150);
    enemyL.size = CGSizeMake(92, 128);
    enemyL.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
enemyL.physicsBody.affectedByGravity = YES;
enemyL.physicsBody.mass = 0;
        enemyR.name = @"enemyL";

    [self addChild:enemyL];

    CGPoint movePointL = CGPointMake(hero.position.x, enemyL.position.y);

    SKAction *jumpL = [SKAction moveTo:movePointL duration:3];

    [enemyL runAction:jumpL];




}
```

**Hopefully you got all that code so far. If you do lets keep on going.**

# Defeating the enemies!

**Just a little idea that I had instead of tapping on them to remove them, what if we swipe on right and left side. If you swipe right you remove the right enemy. Thats fun.**
**After this we do our collision.**

**In your GameScene.h add this code under the others -**

```
-(void)slideToRightWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer;

-(void)slideToLeftWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer;
```

**Go back to GameScene.m and in the didMoveToView add this code -**

```
 UISwipeGestureRecognizer *swipeLeft = [[UISwipeGestureRecognizer alloc]
initWithTarget:self action:@selector(slideToLeftWithGestureRecognizer:)];
    swipeLeft.direction = UISwipeGestureRecognizerDirectionLeft;

    UISwipeGestureRecognizer *swipeRight = [[UISwipeGestureRecognizer alloc]
initWithTarget:self action:@selector(slideToRightWithGestureRecognizer:)];
    swipeRight.direction = UISwipeGestureRecognizerDirectionRight;

 [[self view] addGestureRecognizer: swipeLeft];
    [[self view] addGestureRecognizer:swipeRight];
```

## Adding the Gestures -

```
-(void)slideToLeftWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer{

        [enemyL removeFromParent];




   [self spawnEnemyLeft];

}

-(void)slideToRightWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer{
       [enemyR removeFromParent];



   [self spawnEnemyRight];

}
```

**Lets add some difficulty. like increasing the speed each time you swipe.**

# Increasing the Speed!

**Ok for this we will need to add a float first.**
**So like always after the #import "GameScene.h" add this code -**

```
CGFloat speed;
```

**Now in your didMoveToView write this code -**

```
speed = 10.0;
```

**Now again this time copy all this code -**

```
-(void)spawnEnemyRight {
    enemyR = [SKSpriteNode spriteNodeWithImageNamed:@"Monster3"];
    enemyR.position = CGPointMake(0 + 50, self.frame.size.height/2 - 180);
    enemyR.size = CGSizeMake(92, 128);
    enemyR.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
    enemyR.physicsBody.affectedByGravity = YES;
    enemyR.physicsBody.mass = 0;
    enemyR.name = @"enemyR";
[self addChild:enemyR];
    enemyR.alpha = 0.0;

    SKAction *R = [SKAction fadeAlphaTo:1.0 duration:0.5];
    [enemyR runAction:R];


   CGPoint movePointR = CGPointMake(hero.position.x, enemyR.position.y);

    SKAction *jumpR = [SKAction moveTo:movePointR duration:speed];

    [enemyR runAction:jumpR];
}

-(void)spawnEnemyLeft {

    enemyL = [SKSpriteNode spriteNodeWithImageNamed:@"Monster2"];
    enemyL.position = CGPointMake(self.frame.size.width - 50, self.frame.size.height/2 -
180);
    enemyL.size = CGSizeMake(92, 128);
    enemyL.physicsBody = [SKPhysicsBody bodyWithCircleOfRadius:0.1];
enemyL.physicsBody.affectedByGravity = YES;
enemyL.physicsBody.mass = 0;
        enemyR.name = @"enemyL";
  [self addChild:enemyL];
enemyL.alpha = 0.0;
SKAction *R = [SKAction fadeAlphaTo:1.0 duration:0.5];
    [enemyL runAction:R];
 CGPoint movePointL = CGPointMake(hero.position.x, enemyL.position.y);
  SKAction *jumpL = [SKAction moveTo:movePointL duration:speed];
   [enemyL runAction:jumpL];  }
```

**Now our enemies are under the control of our float. Now we need to change it. Now replace slideToLeft and slideToRight methods with this ,**

```
-(void)slideToLeftWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer{

        [enemyL removeFromParent];
speed = speed - 0.2;


 [self spawnEnemyLeft];

}
-(void)slideToRightWithGestureRecognizer:(UISwipeGestureRecognizer
*)gestureRecognizer{

          [enemyR removeFromParent];
speed = speed - 0.2;
 [self spawnEnemyRight];

}
```

# Collision Detection!

**Again put these 2 under our very favourite #import “GameScene.h”**

```
static const int heroHitCategory = 1;
static const int enemyHitCategory = 2;
```

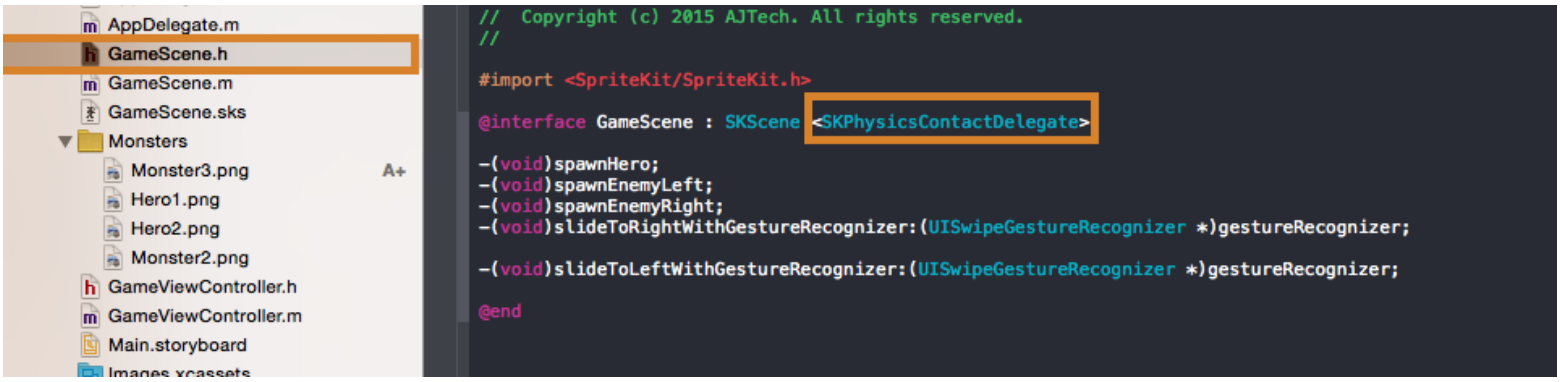
**Now go to our didMoveToView and add this code inside it -**

```
self.physicsWorld.contactDelegate = self;

    hero.physicsBody.categoryBitMask = heroHitCategory;
    hero.physicsBody.contactTestBitMask = enemyHitCategory;
    hero.physicsBody.collisionBitMask =  enemyHitCategory;
    enemyR.physicsBody.categoryBitMask = enemyHitCategory;
    enemyR.physicsBody.contactTestBitMask = heroHitCategory;
    enemyR.physicsBody.collisionBitMask =  heroHitCategory;
    enemyL.physicsBody.categoryBitMask = enemyHitCategory;
    enemyL.physicsBody.contactTestBitMask = enemyHitCategory;
    enemyL.physicsBody.collisionBitMask =  enemyHitCategory;
```

**Now we have to go to our GameScene.h file and add**

```
<SKPhysicsContactDelegate>
```

**Like shown here -**

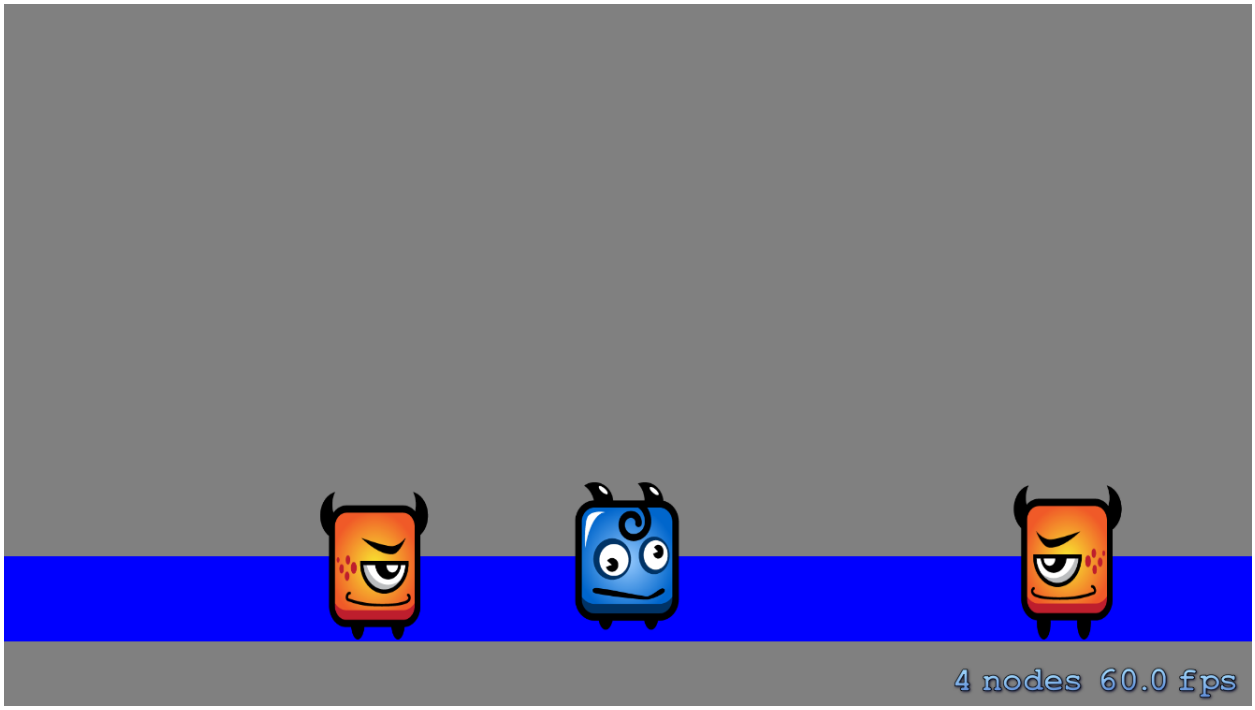



**Then it will work. One last thing to add. Go to our GameScene.m file.**
**Add this method into the .m file -**

```
-(void)didBeginContact:(SKPhysicsContact *)contact
{
    SKPhysicsBody *firstBody, *secondBody;

    firstBody = contact.bodyA;
    secondBody = contact.bodyB;

    if(firstBody.categoryBitMask == enemyHitCategory || secondBody.categoryBitMask
== enemyHitCategory)
    {


        NSLog(@"You died");

        //setup your methods and other things here

    }
}
```

**You see where it says NSLog(@”You died”);**
**Thats where you add your code of what you want to happen when the user dies. But for now it will show you in the log.**

**Final results -**



# Where to go from here?

**That is all for this tutorial , hope you had fun making this app with me. Lots of things you can still do with this. Or even use this as your learning material to use things you have learnt in this as a resource for your projects even add some more graphics to this game and make it look fabulous.**

**Here is the full download for the project : https://github.com/AjayVenkat/RoboShoot**

**If you want to learn way more here is a link by Ray Wenderlich : http://www.raywenderlich.com/store/ios-games-by-tutorials**

**Have fun with this guys and bye for now!**

**BLANK PAGE!**

**BLANK PAGE!**