2017

The Defend – A Collision Based Detection iOS Game

Amith Kumar Aellanki
Grand Valley State University

The Defend - A Collision Based Detection iOS game

By
Amith Kumar Aellanki
April, 2017

The Defend - A Collision Based Detection iOS game

By
Amith Kumar Aellanki

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University
April, 2017

Your Professor **Date**

Table of Contents

Abstract

The overall concept of **The Defend** is to save the Puppies which are in danger from the Puppy monster. The Dog is the hero of the game and the important thing to remember while playing the game is that whenever there is a collision between the dog and puppies, then a chain is formed behind the dog with puppies. Another rule is that whenever there is collision between the dog and puppy monster, there is a decrease of two puppies from the chain. The most challenging part of this development was to detect the collisions, interpreting the User's touch location on the screen, and implementing various actions to the sprites like sound, enabling movement and most importantly providing the animation.

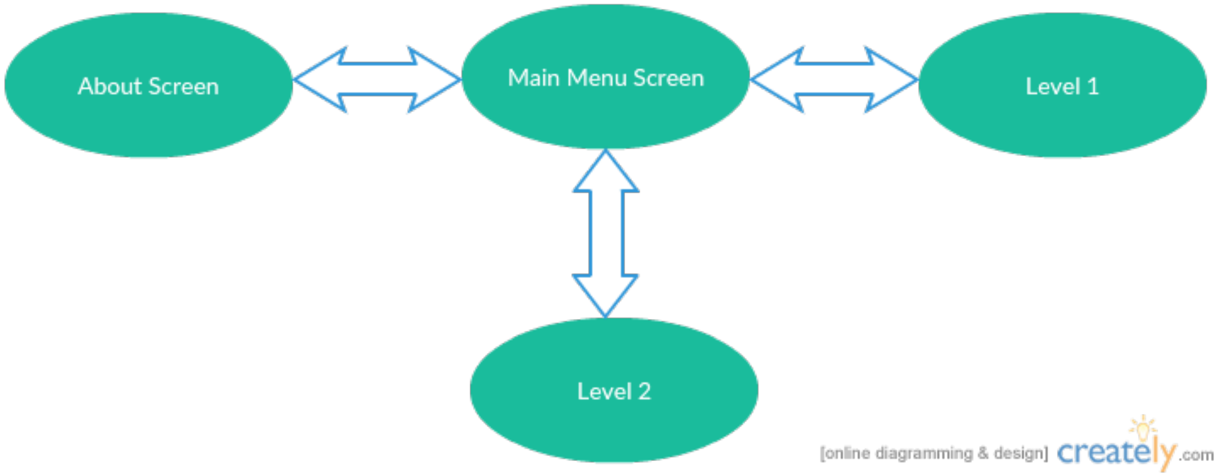
The Defend is a two dimensional Universal mobile game developed for all IOS platforms. The implementation of this game is done using the **SpriteKit** framework. To develop a two dimensional IOS game **SpriteKit** is one of the best available frameworks. The language used for this development is **SWIFT**. The collisions are detected whenever there is intersection between the frames of different sprites in the game.

As this game is a Universal game, the design of this game is done in such a way that the images are compatible and adjusted automatically with in the screen on any device it is installed. This was implemented with the help of some Auto Layout constraints. This game has been successfully submitted to the Apple App store and has been approved.

Introduction

The Defend is a two dimensional IOS game. This game is developed using **SpriteKit**. An overall perspective of this game is that it is a collision detection based game. There are three main characters in the game. A Dog, a Puppy, and a Puppy monster. The idea of this game is to save the puppies from the puppy monster. The Dog is the hero of the game and the control of its movement is assigned to the user playing the game. The movement of the dog is directed by the user tapping locations on the screen. The user has to save the puppies by colliding the dog with the puppies and also by avoiding collision with the puppy monster.

There are two basic rules for this game. Whenever there is a collision between the dog and puppies, then a chain is formed behind the dog with the puppies. Another rule is that whenever there is a collision between the dog and puppy monster the chain is sliced by two puppies [decrease in two puppies] and also there is a decrease in total lives of the user by one. The user can track the total number of lives remaining and the number of puppies collected with the help of labels present at the bottom of the screen. The figure below gives a basic idea of the workflow of the game. I wanted to learn game development in IOS. This was the main reason behind the development of this game.



Program Requirements

This game is developed using Sprite Kit. The programming language used is SWIFT. The IDE [Integrated Development Environment] used is XCode. This game can be developed with the help of the mentioned requirements and a computer that runs OSX is required.

As this is a two dimensional IOS game, Sprite Kit is the best suitable framework that can be used. **SpriteKit** is Apple's framework for making two Dimensional games. It has **sprite** support, support for cool special effects — like videos, filters, and masking. It includes an integrated physics library as well as other features. If you are new to game development, this is a great place to start.

Design

I have divided the development of this game into four phases. I haven't used game scenes in developing this game. I have created programmatically all the required game scenes accordingly. There are four scenes in the game. They are Mainmenu, Level 1, Level 2, and About scenes. Additionally, there are two more scenes which only come into existence when a player wins or loses the game.

- **Mainmenu Scene:** This Scene is useful to navigate to other screens like Level 1, Level 2, and About scenes.
- **Level 1 Scene:** This Scene is the first level of the game.
- **Level 2 Scene:** This Scene is the second level of the game.
- **About Scene:** This Scene gives the information related to developing this game.

1. Phase one involved adding the sprites [background and other characters to the screen] and also defining the process to calculate the device frame size.
2. Phase two involved providing the movement for the sprites and also defining the actions to the sprite.

3. Phase three involved writing the methods to detect collision between the sprites accordingly.
4. Phase four involved adding the labels, sound, and creating the About scene.

Designing the Scenes:

Whenever the game is loaded, the Mainmenu scene is the first scene that pops up or is loaded. Initially the buttons are created and these buttons are used to redirect to another game scene whenever the user taps the button. Identifying the user tapped location can be achieved using the function touches began.

Implementation:

// Touches began function.

```
func scene1() {

    let myscene = GameScene(size: size)
    myscene.scaleMode = scaleMode
    let reveal = SKTransition.fade(withDuration: 1.5)
    view?.presentScene(myscene, transition: reveal)

  }

override func touchesBegan(_ touches: Set<UITouch>,
                               with event: UIEvent?) {

    let touch = touches.first
    let touchlocation = touch!.location(in: self)
    if button.contains(touchlocation) {

        scene1()

        run(touchSound)

    }
```

Buttons are created as nodes with the help of SKSpriteNode. Whenever the user taps this node, different functionality is implemented accordingly.

Syntax Example:

```
let button  = SKSpriteNode( imageNamed: "level1")
```

A Scene can be implemented programmatically by creating a custom sub class of SKScene.

Syntax Example:

```
class Mainmenu: SKScene {

}
```

When implementing a custom subclass of SKScene, the function didMove() is the necessary method or function that has to be implemented.

Syntax Example:

```
override func didMove(to view: SKView) {

        let background  = SKSpriteNode(imageNamed: "menu")
        background.zPosition = -1
        background.position = CGPoint(x: size.width/2, y: size.height/2)

        addChild(background)

        createButton()
   }
```

Whatever we see on the screen is initialized with the help of the didmove function. The sprites are added and then initialized with the help of the addChild() function. The above can also be done with other methodologies.

There are lot of important features that are very useful during the development of the project. One of them is that we can decrease or increase the size the original sprite which we want use in the project. It can be achieved with the help of setScale() function.

Syntax Example:

```
enemy.setScale(0.7)
```

This game involves detecting the collisions between the sprites. It is detected when there is any intersection between frames of different sprites. The following code was used to detect the collisions.

Syntax Example:

```
func checkCollisions() {
        var hitdogs: [SKSpriteNode] = []
        enumerateChildNodes(withName: "dog") { node, _ in
```

```
            let dog = node as! SKSpriteNode
            if dog.frame.intersects(self.puppy.frame) {
                hitdogs.append(dog)
            }
        }
        for dog in hitdogs {
            puppyHit(dog: dog)
        }

        if invincible {
            return
        }

        var hitEnemies: [SKSpriteNode] = []
        enumerateChildNodes(withName: "enemy") { node, _ in
            let enemy = node as! SKSpriteNode
            if node.frame.insetBy(dx: 20, dy: 20).intersects(
                self.puppy.frame) {
                hitEnemies.append(enemy)
            }
        }
        for enemy in hitEnemies {
            puppyHit(enemy: enemy)
        }
    }
```

There is a formation of line when there is a collision between the Dog and the puppy. The following code was used in achieving the above criteria.

Syntax Example:

```
func moveTrain() {
        var trainCount = 0

        var targetPosition = puppy.position
enumerateChildNodes(withName: "train") { node, stop in
            trainCount += 1
            if !node.hasActions() {
                let actionDuration = 0.3
                let offset = targetPosition - node.position
                let direction = offset.normalized()
                let amountToMovePerSec = direction * self.dogMovePointsPerSec
                let amountToMove = amountToMovePerSec * CGFloat(actionDuration)
                let moveAction = SKAction.moveBy(x: amountToMove.x, y:
amountToMove.y, duration: actionDuration)
```

```
            node.run(moveAction)
         }
         targetPosition = node.position
      }
      if trainCount >= 10 && !gameOver {
         gameOver = true
         print("You win!")

          backgroundMusicPlayer.stop()

       let gameOverScene = GameOverScene(size: size, won: true)
         gameOverScene.scaleMode = scaleMode

         let reveal = SKTransition.flipHorizontal(withDuration: 0.5)

         view?.presentScene(gameOverScene, transition: reveal)
      }
      dogsLabel.text = "Puppie: \(trainCount)"
   }
```

The game sounds are played with help of SKActions. The following code is used to perform the Action.

Syntax Example;

```
let dogCollisionSound: SKAction = SKAction.playSoundFileNamed(
      "puppybark.mp3", waitForCompletion: false)

run(dogCollisionSound)
```

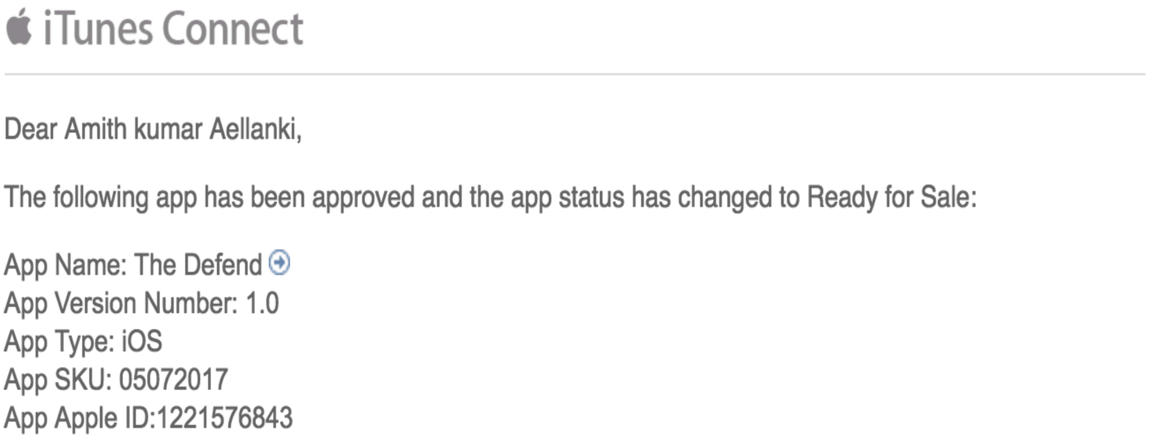
There is automatic scrolling of the screen from left to right when the game is played. The following code is used to achieve this functionality.

Syntax Example:

```
func moveCamera() {
      let backgroundVelocity =
         CGPoint(x: cameraMovePointsPerSec, y: 0)
      let amountToMove = backgroundVelocity * CGFloat(dt)
      cameraNode.position += amountToMove
      enumerateChildNodes(withName: "background") { node, _ in
         let background = node as! SKSpriteNode
         if background.position.x + background.size.width <
            self.cameraRect.origin.x {
            background.position = CGPoint(
            x: background.position.x + background.size.width*2,
            y: background.position.y)
    }}}
```

Results, Evaluation, and Reflection

I have submitted this game in the Apple app store and it has been approved.



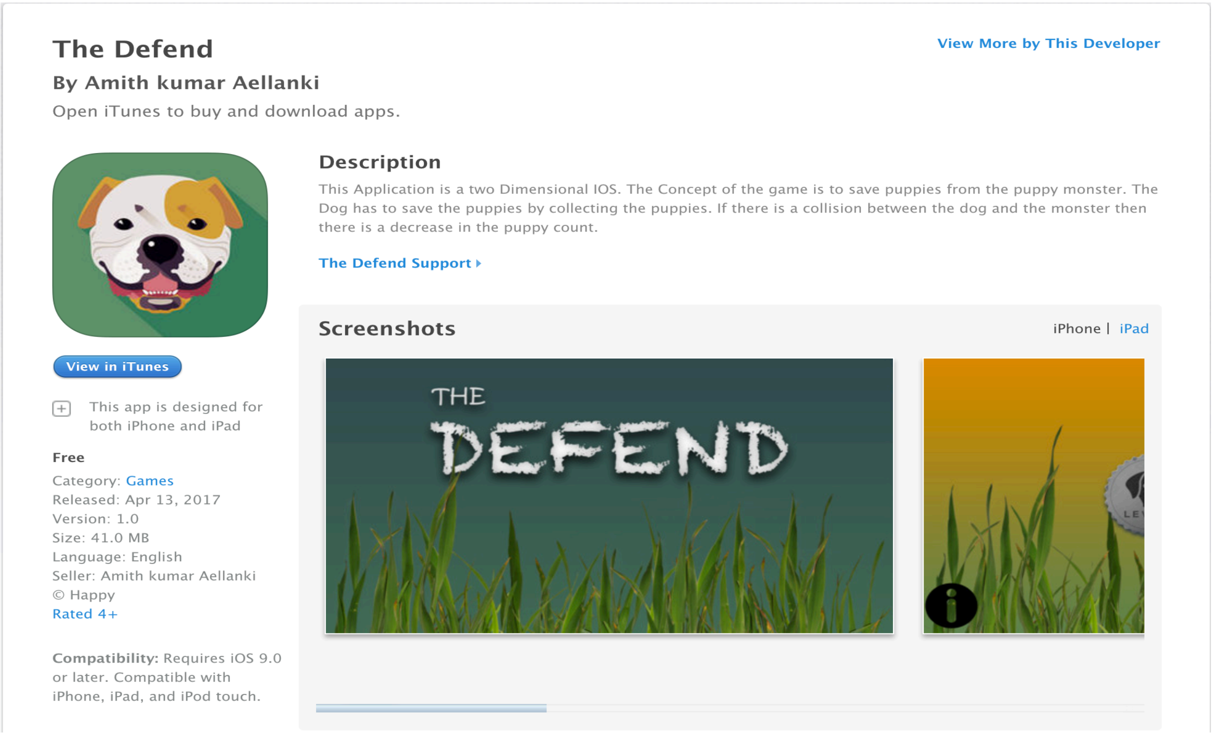
iTunes Connect

Dear Amith kumar Aellanki,

The following app has been approved and the app status has changed to Ready for Sale:

App Name: The Defend
App Version Number: 1.0
App Type: iOS
App SKU: 05072017
App Apple ID:1221576843

ITunes preview:



The Defend

View More by This Developer

By Amith kumar Aellanki

Open iTunes to buy and download apps.

Description

This Application is a two Dimensional IOS. The Concept of the game is to save puppies from the puppy monster. The Dog has to save the puppies by collecting the puppies. If there is a collision between the dog and the monster then there is a decrease in the puppy count.

The Defend Support ▸

Screenshots

iPhone | iPad

This app is designed for both iPhone and iPad

Free

Category: Games
Released: Apr 13, 2017
Version: 1.0
Size: 41.0 MB
Language: English
Seller: Amith kumar Aellanki
© Happy
Rated 4+

Compatibility: Requires iOS 9.0 or later. Compatible with iPhone, iPad, and iPod touch.

Apple App store Link: https://appsto.re/us/II8Zib.i

Conclusions and Future Work

This game is a basic two dimensional IOS game developed using SpriteKit framework. When it comes to the implementation of this game, as this is a universal IOS game, I had to define an initializer that would calculate the size of the screen and its frame size including width and height. The next important step is defining all the Sprites and providing the movement and actions to them. After providing the physical movement, methods were written to detect the collisions between the sprites and updating the user scores with the help of the labels. Finally, various sounds for the game has been implemented for various actions and for tapping various button in the game.

I would like to implement the functionality of motion sensing to the game; that is, playing the game with the help of physical movement of the device in which game is played. I would also like to host this game in an online Database to increase the number of levels in the game and implementing the multi-player functionality.

Bibliography

1. Apple – Sprite Kit, API reference from https://developer.apple.com/reference/spritekit/
2. Ray, SpriteKit tutorials for beginners, October 11 2016 from https://www.raywenderlich.com/145318/spritekit-swift-3-tutorial-beginners
3. Stack Overflow from http://stackoverflow.com/