Pong Game Project

Επιβλέπων:Δασυγένης Μηνάς
Φοιτητής:Τερζή Αναστασία
Ιούνιος 2018,Κοζάνη
Τμήμα Μηχανικών πληροφορικής και τηλεπικοινωνιών
Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών
http://arch.icte.uowm.gr/

Πλάνο Παρουσίασης

- Στο πρώτο μέρος γίνεται η παρουσίαση της εφαρμογής. Θα παρατηρήσουμε την εμφάνιση της σε διάφορα μεγέθη οθόνης και θα δούμε σκηνές από την εκτέλεση της.
- Στο δεύτερο μέρος θα μελετήσουμε τον σχεδιασμό και κώδικα πίσω από την εφαρμογή.
- Στο τρίτο μέρος έχουμε τα συμπεράσματα από την διαδικασία.

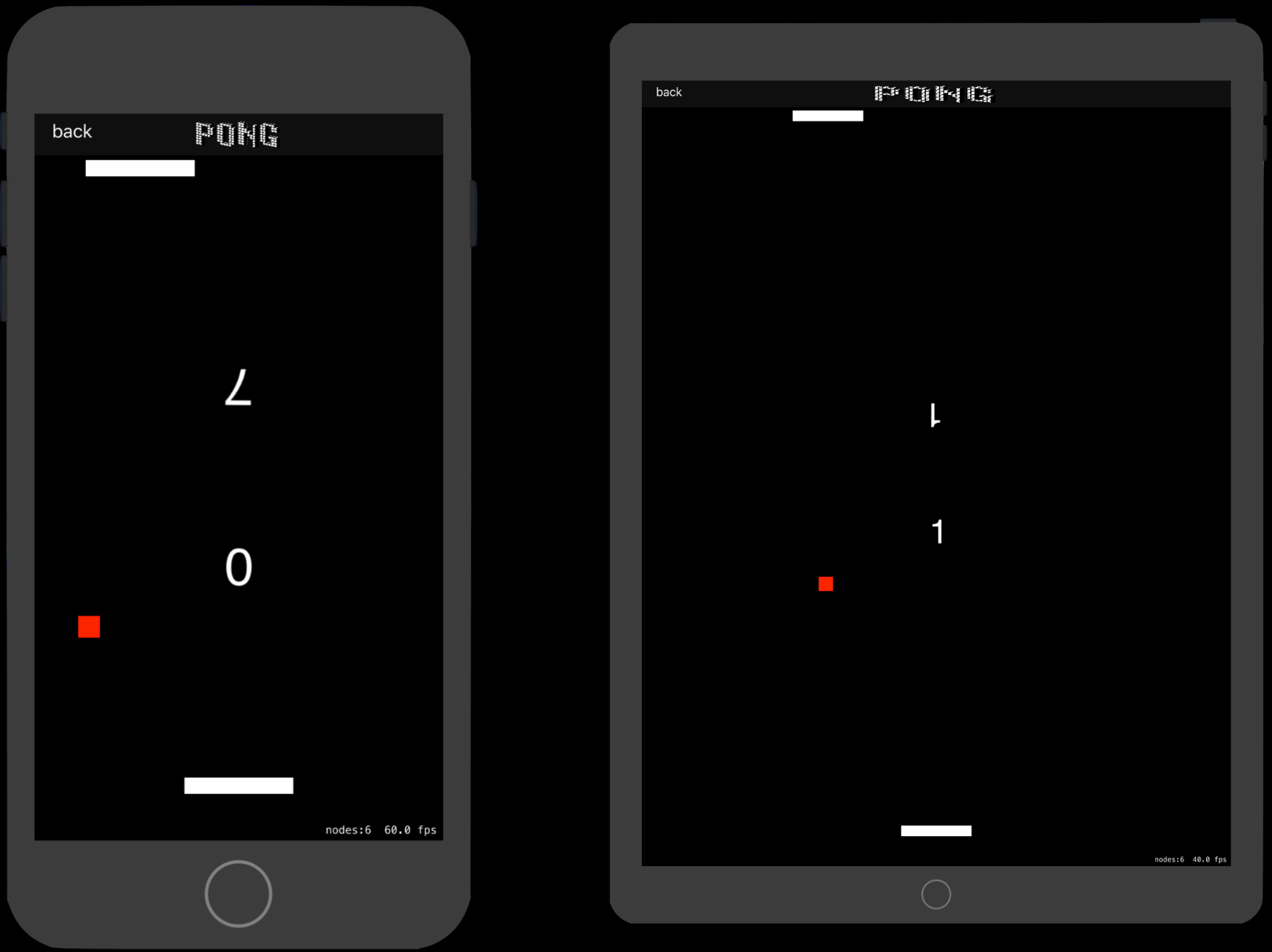
PONG GAME

Στοιχεία Εμφάνισης

App icon εφαρμογής

Εφαρμόγη σε αρχική οθόνη συσκευής

Αρχικό menu εφαρμογής



Εικόνες από την εκτέλεση

Εμφάνιση τελικής οθόνης

Απόκτηση Εφαρμογής

Απόκτηση Εφαρμογής

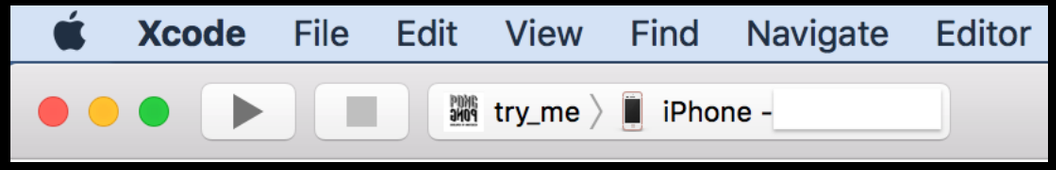
Η εφαρμογή δεν διατίθεται στο AppStore για την εγκατάσταση της ο χρήστης θα χρειαστεί να ακολουθήσει τα βήματα:

- Να κατεβάσει το αρχείο .ipa

try_me.ipa

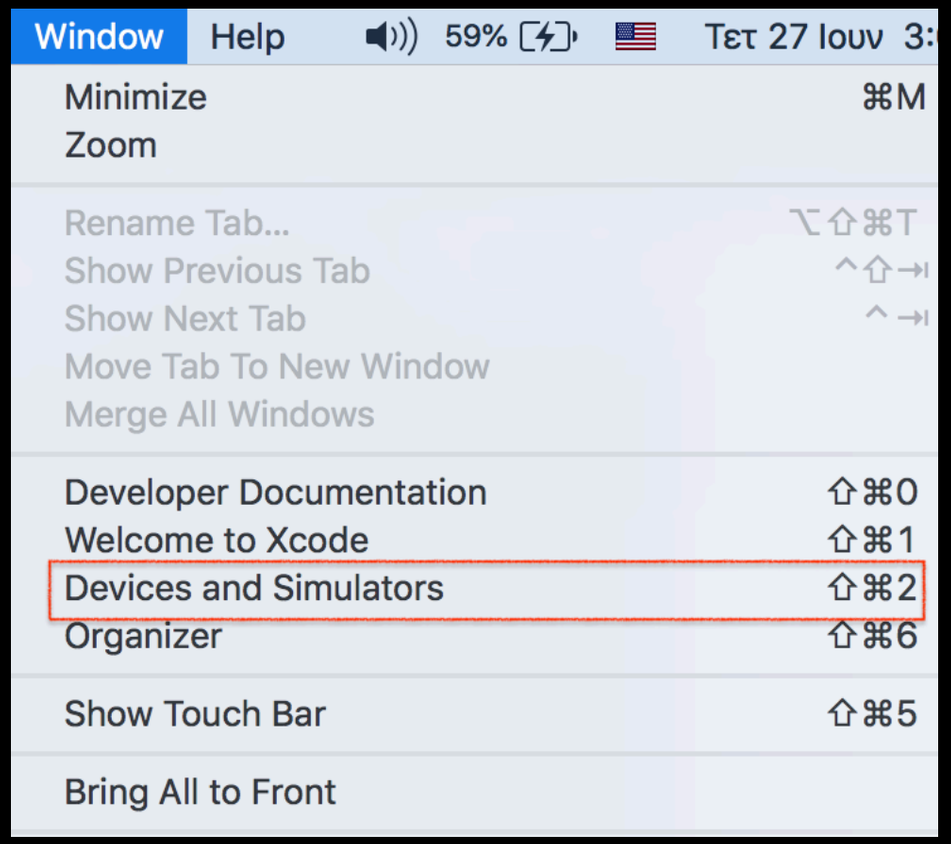
Απόκτηση Εφαρμογής

- Να συνδέσει τη συσκευή που επιθυμεί να εγκαταστήσει την εφαρμογή και να ανοίξει το Xcode



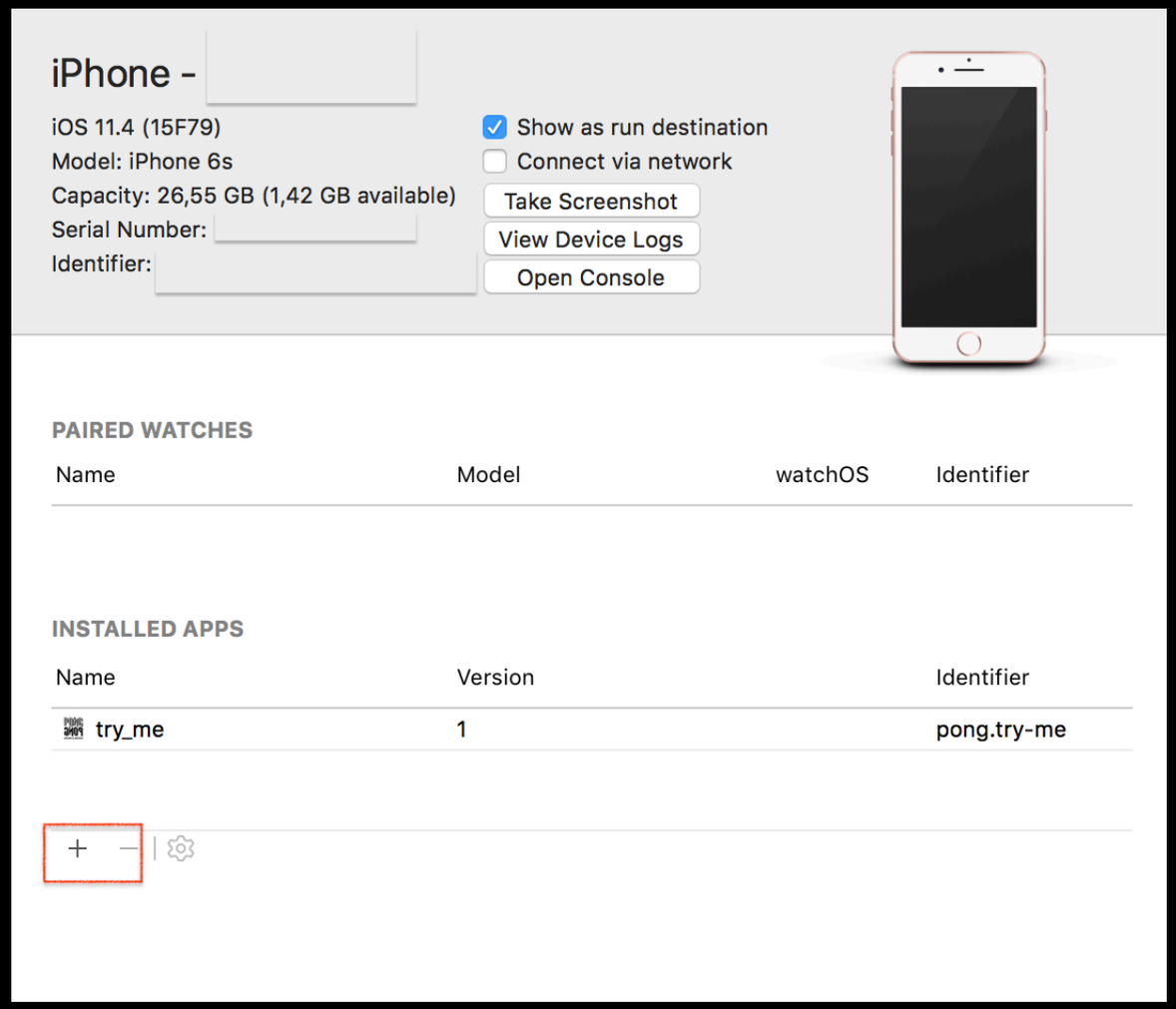
Απόκτηση Εφαρμογής

- Μπάρα εργαλείων Παράθυρο->Συσκευές και Προσομοιωτές. Θα ανοίξει παράθυρο με τα χαρακτηριστικά της συσκευής



Απόκτηση Εφαρμογής

- Στην περιοχή εγκατεστημένες εφαρμογές επιλέγει το +. Στο παράθυρο που θα ανοίξει επιλέγει το αρχείο .ipa και ολοκληρώνεται η εγκατάσταση.



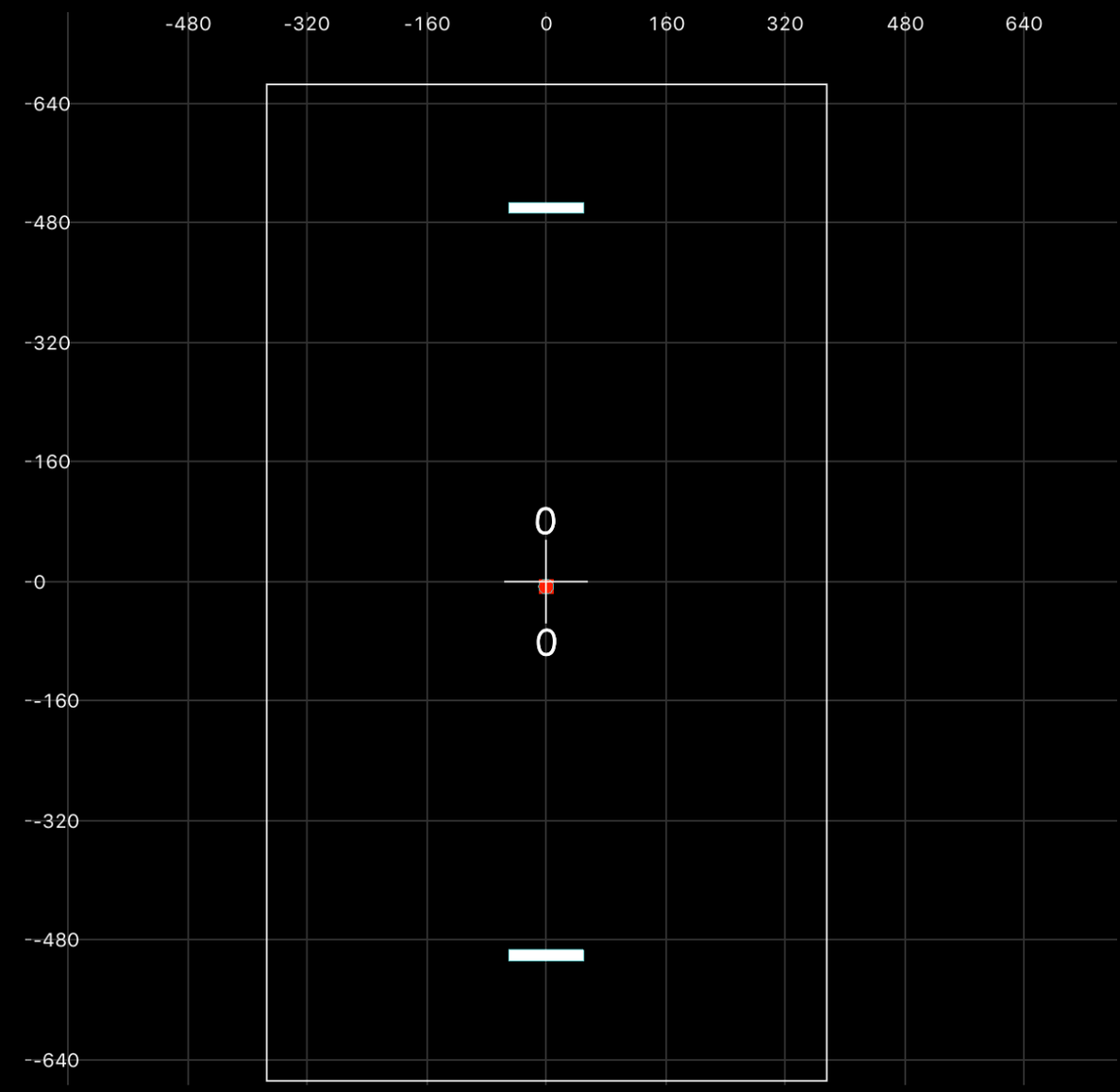
Σχεδιασμός

Σχεδιασμός

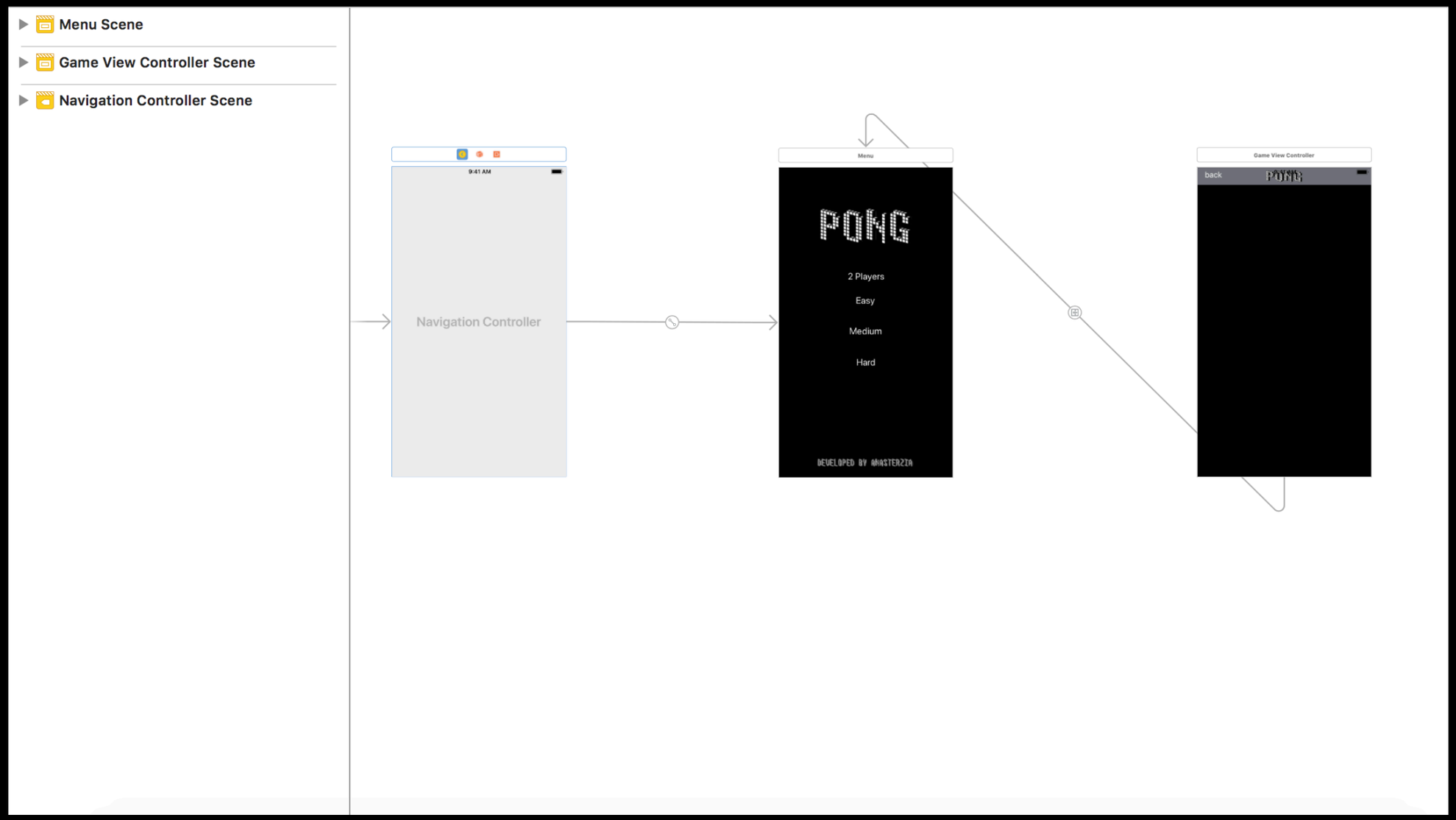
Βλέπουμε την GameScene.sks. Δημιουργήσαμε πέντε αντικείμενα τις ρακέτες , τη μπάλα και τις ταμπέλες για το σκορ.

Αποτελεί η βασική περιοχή σχεδιασμού του παιχνιδιού.

Τα αντικείμενα που περιέχονται είναι αρχικά στατικά και οι συμμπεριφορές τους τροποποιούνται από τον προγραμματιστή



Σχεδιασμός



Στο storyboard καθορίζουμε την σειρά των σκηνών. Τις σχέσεις που τις συνδέουν όπως και κάποιες συμπεριφορές αντικειμένων.
Συγκεκριμένα παρατηρούμε ότι η οθόνη ελεγχου μας οδηγεί στην οθόνη menu και η βασική οθόνη παιχνιδιού με το κουμπί back στο menu

Μελέτη Κώδικα

Menu

```swift
//  menu.swift
//  try_me
//
//  Created by Anastasia T on 20/6/18.
//  Copyright © 2018 Anastassia Terzi. All rights reserved.
//
//menu scene

import Foundation
import UIKit

//cases of game
enum gametype {
  case easy
  case medium
  case hard
  case player2
}

class Menu : UIViewController{

//connection between buttons and actions
  @IBAction func Player2(_ sender: AnyObject) {
    move_Game(game: .player2)}

  @IBAction func Easy(_ sender: AnyObject) {
    move_Game(game: .easy)}

  @IBAction func medium(_ sender: AnyObject) {
    move_Game(game: .medium)}

  @IBAction func Hard(_ sender: AnyObject) {
    move_Game(game: .hard)}

//setting the game type and transfer the player to the right scene
    func move_Game(game : gametype) {
      let gameVC = self.storyboard?.instantiateViewController(withIdentifier: "gameVC") as! GameViewController
      currentGameType = game
    self.navigationController?.pushViewController(gameVC, animated: true)
  }

}
```

Σε αυτό το αρχείο καθορίζουμε την συμπεριφορά του αρχικού καταλόγου και των κουμπιών.

Οι κύκλοι που παρεμβάλλονται της αρίθμησης αποτελούν την απευθείας σύνδεση των αντικειμένων από το Storyboard με το αρχείο menu.swift

Menu

```
//  menu.swift
//  try_me
//
//  Created by Anastasia T on 20/6/18.
//  Copyright © 2018 Anastassia Terzi. All rights reserved.
//
//menu scene

import Foundation
import UIKit

//cases of game
enum gametype {
  case easy
  case medium
  case hard
  case player2
}

class Menu : UIViewController{

//connection between buttons and actions
  @IBAction func Player2(_ sender: AnyObject) {
    move_Game(game: .player2)}

  @IBAction func Easy(_ sender: AnyObject) {
    move_Game(game: .easy)}

  @IBAction func medium(_ sender: AnyObject) {
    move_Game(game: .medium)}

  @IBAction func Hard(_ sender: AnyObject) {
    move_Game(game: .hard)}

//setting the game type and transfer the player to the right scene
    func move_Game(game : gametype) {
      let gameVC = self.storyboard?.instantiateViewController(withIdentifier: "gameVC") as! GameViewController
      currentGameType = game
    self.navigationController?.pushViewController(gameVC, animated: true)
  }

}
```

Με μια απαρίθμηση(enumeration) καταγράφουμε τις τέσσερις διαφορετικές επιλογές. Αυτές αντιστοιχούν στους διαφορετικούς τύπους παιχνιδιού.

Η συνάρτηση move_Game δέχεται ως όρισμα την επιλογή του χρήστη. Έπειτα σε αυτή δηλώνουμε μια σταθερά με το όνομα gameVC. Εκεί αποθηκεύουμε την μορφή οθόνης που αντιστοιχεί στο γνώρισμα gameVc. Στο Storyboard έχεουμε ήδη ονομάσει το stoyboard ID της βασικής οθόνης gameVc.

Menu

```swift
//  menu.swift
//  try_me
//
//  Created by Anastasia T on 20/6/18.
//  Copyright © 2018 Anastassia Terzi. All rights reserved.
//
//menu scene

import Foundation
import UIKit

//cases of game
enum gametype {
  case easy
  case medium
  case hard
  case player2
}

class Menu : UIViewController{

//connection between buttons and actions
  @IBAction func Player2(_ sender: AnyObject) {
    move_Game(game: .player2)}

  @IBAction func Easy(_ sender: AnyObject) {
    move_Game(game: .easy)}

  @IBAction func medium(_ sender: AnyObject) {
    move_Game(game: .medium)}

  @IBAction func Hard(_ sender: AnyObject) {
    move_Game(game: .hard)}

//setting the game type and transfer the player to the right scene
    func move_Game(game : gametype) {
      let gameVC = self.storyboard?.instantiateViewController(withIdentifier: "gameVC") as! GameViewController
      currentGameType = game
    self.navigationController?.pushViewController(gameVC, animated: true)
  }

}
```

Η currentGame είναι μία μεταβλήτη, η αρχική ανέθεση τιμής γίνεται στο αρχείο GameViewController.swift

```swift
var currentGameType = gametype.medium
```

Αυτή περιέχει την κατηγορία του τρέχοντος παιχνιδιού.
Τέλος η move_Game βάζει την τρέχουσα οθόνη στο σωρό και προβάλει την οθόνη που έχει μέσα η σταθερά gameVC.

Menu

Ας διευκρινήσουμε τη σημασία των τριών επιπέδων δυσκολίας.
Τα επίπεδα δε διαφέρουν ώς προς την ταχύτητα της μπάλας αλλά προς την ταχύτητα απόκρισης της ρακέτας.
Το χαρακτηριστικό duration αφορά στην διάρκεια αποκρισης κίνησης αλλά είναι αντιληπτό ως ταχύτητα ρακέτας. Όσο μικρότερη τιμή έχει αυτή η αντίδραση τόσο μεγαλύτερη ταχύτητα έχει η ρακέτα.
Για το χρήστη η τιμή είναι σε κάθε επίπεδο 0.2. Για τον υπολογιστή αλλάζει αυτή η ταχύτητα ανάλογα με το επίπεδο.
Σε επόμενες διαφάνειες υπάρχει το σημείο στο οποίο γίνοται αυτές οι αλλαγές.

Game over Scene

```
7  import SpriteKit
8
9  class GameOverScene: SKScene {
10
11   //adding logo image to scene
12
13    let logo = SKSpriteNode(imageNamed: "ponglogo")
14
15   init(size: CGSize, won:Bool) {
16     super.init(size: size)
17
18     //setting background
19     backgroundColor = SKColor.black
20
21     //adding message to scene using label
22     let message = won ? "Player2:You Won!" : "Player1:You Won!"
23
24     //setting font , font size and position for the label
25     let label = SKLabelNode(fontNamed: "Comic Sans")
26     label.text = message
27     label.fontSize = 40
28     label.fontColor = SKColor.white
29     label.position = CGPoint(x: size.width/2, y: size.height/2)
30     addChild(label)
31
32     //setting position for the logo image
33      logo.position = CGPoint(x: size.width/2, y: size.height/2+100)
34     addChild(logo)
35
36   }
```

Η τελική αυτή σκηνή αποτελείται από μια εικόνα και μια ταμπέλα με μήνυμα στον νικητή.

Η σταθερά logo περιέχει την εικόνα που θα χρησιμοποιήσουμε.

Οι εικόνες εμπεριέχονται σε έναν φάκελο που δημιουργείται με το project

Τοποθετούμε την εικόνα στο κέντρο της εκάστοτε οθόνης.

Game over Scene

```
7  import SpriteKit
8
9  class GameOverScene: SKScene {
10
11   //adding logo image to scene
12
13    let logo = SKSpriteNode(imageNamed: "ponglogo")
14
15   init(size: CGSize, won:Bool) {
16     super.init(size: size)
17
18     //setting background
19     backgroundColor = SKColor.black
20
21     //adding message to scene using label
22     let message = won ? "Player2:You Won!" : "Player1:You Won!"
23
24     //setting font , font size and position for the label
25     let label = SKLabelNode(fontNamed: "Comic Sans")
26     label.text = message
27     label.fontSize = 40
28     label.fontColor = SKColor.white
29     label.position = CGPoint(x: size.width/2, y: size.height/2)
30     addChild(label)
31
32     //setting position for the logo image
33      logo.position = CGPoint(x: size.width/2, y: size.height/2+100)
34     addChild(logo)
35
36   }
```

Θέτουμε το χρώμα υποβάθρου μαύρο.

Η σταθερά message παίρνει τιμή ανάλογα με αυτή που επιστρέφει η παράμετρος won.

Σε μια ταμπέλα δίνουμε όλες τις ιδιότητες που θέλουμε να έχει το μήνυμα, χρώμα, μέγεθος, γραμματοσειρά και θέση. Τέλος σε αυτή τη ταμπέλα τοποθετούμε το μήνυμα μας.

Game Scene

```swift
import SpriteKit
import GameplayKit

class GameScene: SKScene {
  //variable declaration
  var ball = SKSpriteNode()
  var enemy = SKSpriteNode()
  var friend = SKSpriteNode()
  var score = [Int]()
  var top_lab = SKLabelNode()
  var bottom_lab = SKLabelNode()

  override func didMove(to view: SKView) {

    ball = self.childNode(withName: "ball") as! SKSpriteNode

    //positioning paddles
    enemy = self.childNode(withName: "enemy") as! SKSpriteNode
    enemy.position.y = (self.frame.height / 2) - 50

    friend = self.childNode(withName: "friend") as! SKSpriteNode
    friend.position.y = (-self.frame.height / 2) + 50

    //positioning score labels
    top_lab = self.childNode(withName: "top_label") as! SKLabelNode
    bottom_lab = self.childNode(withName: "bottom_label") as! SKLabelNode

    //setting physics body of borders
    let border = SKPhysicsBody(edgeLoopFrom: self.frame)
    border.friction = 0
    border.restitution = 1
    self.physicsBody = border

    //starting game
     startGame()

    }
```

Σε αυτό το αρχείο καθορίζουμε την συμπεριφορά της βασικής σκηνής παιχνιδιού και των αντικειμένων της.

Δηλώνουμε μεταβλητές στις οποίες αντιστοιχούμε αντικείμενα που έχουμε σχεδιάσει. Επιπλέον για τις δύο ρακέτες friend και enemy ορίζουμε τη θέση τους σε σχέση με το μέγεθος της οθόνης. Με τον τρόπο αυτό για κάθε μέγεθος συσκευής οι ρακέτες μας θα έχουν αρχική θέση στο ίδιο και αντιδιαμετρικό σημείο.

Game Scene

```
10  import SpriteKit
11  import GameplayKit
12
13  class GameScene: SKScene {
14    //variable declaration
15    var ball = SKSpriteNode()
16    var enemy = SKSpriteNode()
17    var friend = SKSpriteNode()
18    var score = [Int]()
19    var top_lab = SKLabelNode()
20   var bottom_lab = SKLabelNode()
21
22    override func didMove(to view: SKView) {
23
24      ball = self.childNode(withName: "ball") as! SKSpriteNode
25
26      //positioning paddles
27      enemy = self.childNode(withName: "enemy") as! SKSpriteNode
28      enemy.position.y = (self.frame.height / 2) - 50
29
30      friend = self.childNode(withName: "friend") as! SKSpriteNode
31      friend.position.y = (-self.frame.height / 2)  + 50
32
33      //positioning score labels
34      top_lab = self.childNode(withName: "top_label") as! SKLabelNode
35      bottom_lab = self.childNode(withName: "bottom_label") as! SKLabelNode
36
37      //setting physics body of borders
38      let border = SKPhysicsBody(edgeLoopFrom: self.frame)
39      border.friction = 0
40      border.restitution = 1
41      self.physicsBody = border
42
43      //starting game
44       startGame()
45
46     }
```

Έπειτα δημιουργούμε ένα φυσικό σώμα για το σύνορο της οθόνης. Αυτό είναι απαραίτητο να είναι λείο και να προσφέρει αντίσταση για την αναπήδηση της μπάλας.

Game Scene

```
47  //setting game
48  func startGame(){
49
50    score = [ 0, 0 ]
51    top_lab.text = "\(score[1])"
52    bottom_lab.text = "\(score[0])"
53    ball.physicsBody?.applyImpulse(CGVector(dx: 20, dy: 20))
54
55  }
56
57  func add_score(player_Won : SKSpriteNode){
58
59    //positioning ball in the center
60    ball.position = CGPoint( x: 0, y: 0 )
61    ball.physicsBody?.velocity = CGVector (dx: 0, dy: 0)
62
63    //adding score to player2 or friend
64    if player_Won == friend {
65      score[0] += 1
66      ball.physicsBody?.applyImpulse(CGVector(dx: 20, dy: 20))
67    }
68    //adding score to player1 or enemy
69    else if player_Won == enemy{
70      score[1] += 1
71      ball.physicsBody?.applyImpulse(CGVector(dx: -20, dy: -20))
72
73    }
74  top_lab.text = "\(score[1])"
75  bottom_lab.text = "\(score[0])"
76
77    //case of ending the game
78    //changing to winning scene for player2
79    if score[0] == 10{
80      let gameOverScene = GameOverScene(size: self.size, won: true)
81      self.view?.presentScene(gameOverScene)
82    }
83     //changing to winning scene for player1
```

Η συνάρτηση startGame αρχικοποιεί το σκορ των παικτών στη μηδενική τιμή και την ταχύτητα της μπαλάς.

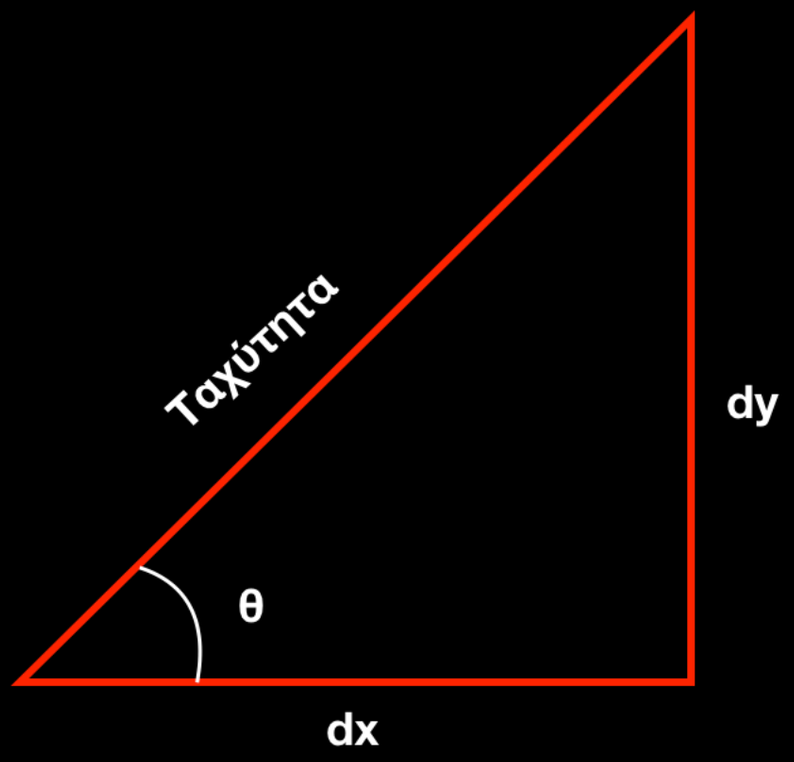
Η add_score αυξάνει και εμφανίζει το σκορ των παικτών. Σε κάθε πόντο η μπάλα επανατοποθετείται στο κέντρο και αποκτά κατεύθυνση προς τον ηττημένο.

Game Scene

```
53      ball.physicsBody?.applyImpulse(CGVector(dx: 20, dy: 20))
54
55    }
56
57    func add_score(player_Won : SKSpriteNode){
58
59      //positioning ball in the center
60      ball.position = CGPoint( x: 0, y: 0 )
61      ball.physicsBody?.velocity = CGVector (dx: 0, dy: 0)
62
63      //adding score to player2 or friend
64      if player_Won == friend {
65        score[0] += 1
66        ball.physicsBody?.applyImpulse(CGVector(dx: 20, dy: 20))
67      }
68      //adding score to player1 or enemy
69      else if player_Won == enemy{
70        score[1] += 1
71        ball.physicsBody?.applyImpulse(CGVector(dx: -20, dy: -20))
72
73      }
74     top_lab.text = "\(score[1])"
75     bottom_lab.text = "\(score[0])"
76
77      //case of ending the game
78      //changing to winning scene for player2
79      if score[0] == 10{
80        let gameOverScene = GameOverScene(size: self.size, won: true)
81        self.view?.presentScene(gameOverScene)
82      }
83       //changing to winning scene for player1
84      if score[1] == 10{
85        let gameOverScene = GameOverScene(size: self.size, won: false)
86        self.view?.presentScene(gameOverScene)
87      }
88    }
```

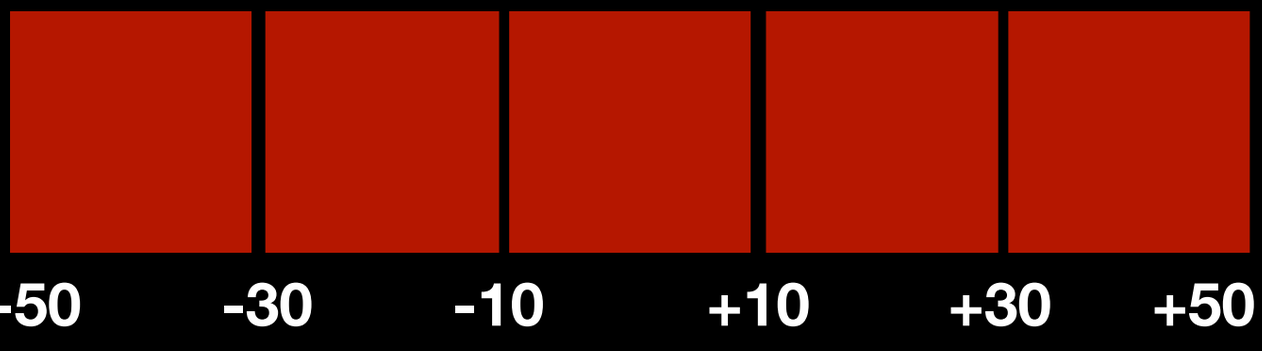
Μόλις κάποιος παίκτης φτάσει τους 10 πόντους η add_score χρησιμοποιεί μια σταθερά για να καλέσει την αντίστοιχη game over scene. Με won true ή false ικανοποίει την συνθήκη της τελικής σκηνής και γίνεται η ανακατεύθυνση.

Game Scene



Πριν μελετήσουμε την επόμενη συνάρτηση είναι σημαντικό να εξηγήσουμε την κίνηση της μπάλας.
Ορίζουμε τα διανύσματα dx και dy , που είναι η κάθετες ενός τριγώνου. Η ταχύτητα κίνησης της μπάλας υπολογίζεται από πυθαγόρειο θεώρημα και αντιστοιχεί στο μήκος υποτείνουσας.
Τέλος η γωνία θ αποτελεί την γωνία αναπήδησης.
Αν θέλουμε συγκεκριμένη γωνία ανάκλασης πολλαπλασιάζουμε dx και dy με το συνημίτονο και το ημίτονο της γωνίας αντίστοιχα.

Game Scene



Για να αλλάξουμε τη γωνία ανάκλασης της μπάλας, χωρίσαμε τις ρακέτες σε 5 περιοχές. Οι περιοχές χαρακτηρίζονται από την απόστασή τους από το κέντρο της ρακέτας.
Η επαφή της μπάλας σε κάποια περιοχή αλλάζει τα διανύσματα dx, dy και κατα συνέπεια την ταχύτητα και την κατεύθυνση της μπάλας

Game Scene

```
func change(posi: SKSpriteNode){

 //ball is in the same position as player2 or friend
 if posi==friend{

  //ball is in position left to the center of paddle
 if ball.position.x < friend.position.x && ball.position.x >= friend.position.x - 50{
  if ball.position.x <= friend.position.x - 30{
    //115
    ball.physicsBody?.applyImpulse(CGVector(dx: 12 * (-0.9), dy: 20 * 0.42))}
  if ball.position.x > friend.position.x - 30 && ball.position.x <= friend.position.x - 10{
    //120
    ball.physicsBody?.applyImpulse(CGVector(dx: -10, dy: 10 * 0.86))  }
  if ball.position.x > friend.position.x - 10 &&  ball.position.x <= friend.position.x + 10 {
    //45
    ball.physicsBody?.applyImpulse(CGVector(dx: 20 * 0.707 , dy: 20 * 0.707))  }
 }
  //ball is in position right to the center of paddle
 else if ball.position.x > friend.position.x && ball.position.x <= friend.position.x + 50{
   if ball.position.x >= friend.position.x + 30{
    //62
    ball.physicsBody?.applyImpulse(CGVector(dx: 5 * (0.46), dy: 10 * 0.88))}
  if ball.position.x > friend.position.x + 10 &&  ball.position.x < friend.position.x + 30 {
    //30
    ball.physicsBody?.applyImpulse(CGVector(dx: 5 * 0.86, dy: 10))}
 }
}
```

Η συνάρτηση change είναι υπεύθυνη για την αλλαγή κατεύθυνσης της μπάλας. Δέχεται όρισμα το όνομα της ρακέτας.

Στην περίπτωση της κάτω ρακέτας, εξετάζουμε σε ποιό σημείο έγινε η επαφή. Αν η μπάλα χτύπησε σε σημείο αριστερά του κέντρου, ανάλογα με την περιοχή θα έχουμε γωνία ανάκλασης 155 ή 120 μοιρών. Στο κέντρο η γωνία ανάκλασης είναι 45 μοίρες. Δεξιά του κέντρου έχουμε γωνίες 62 και 30 μοιρών.

Game Scene

```
func change(posi: SKSpriteNode){

 //ball is in the same position as player2 or friend
 if posi==friend{

  //ball is in position left to the center of paddle
 if ball.position.x < friend.position.x && ball.position.x >= friend.position.x - 50{
   if ball.position.x <= friend.position.x - 30{
     //115
     ball.physicsBody?.applyImpulse(CGVector(dx: 12 * (-0.9), dy: 20 * 0.42))}
   if ball.position.x > friend.position.x - 30 && ball.position.x <= friend.position.x - 10{
     //120
     ball.physicsBody?.applyImpulse(CGVector(dx: -10, dy: 10 * 0.86))  }
  if ball.position.x > friend.position.x - 10 &&  ball.position.x <= friend.position.x + 10 {
     //45
     ball.physicsBody?.applyImpulse(CGVector(dx: 20 * 0.707 , dy: 20 * 0.707))  }
 }
  //ball is in position right to the center of paddle
 else if ball.position.x > friend.position.x && ball.position.x <= friend.position.x + 50{
    if  ball.position.x >= friend.position.x + 30{
     //62
     ball.physicsBody?.applyImpulse(CGVector(dx: 5 * (0.46), dy: 10 * 0.88))}
   if ball.position.x > friend.position.x + 10 &&  ball.position.x < friend.position.x + 30 {
     //30
     ball.physicsBody?.applyImpulse(CGVector(dx: 5 * 0.86, dy: 10))}
 }
}
```

Η επιλογή των μοιρών είναι τέτοια ώστε η μπάλα να έχει κατεύθυνση προς το κοντινότερο τοίχωμα.

Επιπλέον η διαφοροποίηση των γωνιών έγινε για να ακολουθεί η μπάλα διαφορετική διαδρομή.

Game Scene

```
//ball is in the same position as player1 or enemy
if posi == enemy{

//ball is in position left to the center of paddle
if ball.position.x < enemy.position.x && ball.position.x >= enemy.position.x - 50{
  if ball.position.x <= enemy.position.x - 30 {
    //65
   ball.physicsBody?.applyImpulse(CGVector(dx: -20 * 0.42, dy: -20 * 0.89 ))}
  if ball.position.x > enemy.position.x - 30 && ball.position.x <= enemy.position.x - 10{
    //30
    ball.physicsBody?.applyImpulse(CGVector(dx: -10, dy: -20 * 0.86))}
  if ball.position.x > enemy.position.x - 10 &&  ball.position.x <= enemy.position.x + 10 {
    //45
    ball.physicsBody?.applyImpulse(CGVector(dx: -20 * 0.707 , dy: -20 * 0.707))}
 }
   //ball is in position right to the center of paddle
 else if ball.position.x > enemy.position.x && ball.position.x < enemy.position.x + 50{
  if  ball.position.x >= enemy.position.x + 30{
    //118
    ball.physicsBody?.applyImpulse(CGVector(dx: 5 * (0.46), dy: -10 * 0.88))}
  if ball.position.x > enemy.position.x + 10 &&  ball.position.x < enemy.position.x + 30 {
    //120
    ball.physicsBody?.applyImpulse(CGVector(dx: 5 * 0.86, dy: -10))}
  }
 }
}
```

Στην περίπτωση της άνω ρακέτας, εξετάζουμε πάλι το σημείο επαφής.

Οι περιοχές αριστερά του κέντρου έχουν γωνίες ανάκλασης 65 και 30 μοιρών. Στο κέντρο και εδώ έχουμε γωνία 45 μοίρες. Τέλος δεξιά του κέντρου οι γωνίες είναι 118 και 120 μοίρες

Game Scene

```
143    //activate action
144    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
145      for touch in touches {
146        let location = touch.location(in: self)
147        //case for two players
148        if currentGameType == .player2{
149          if location.y > 0 {
150            enemy.run(SKAction.moveTo(x: location.x, duration: 0.2))}
151          if location.y < 0{
152            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
153        }
154          //case for playing against cpu
155          else {
156           friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
157      }
158    }
159
160    //manage touch movement
161    override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
162      for touch in touches {
163
164        let location = touch.location(in: self)
165         //case for two players and moving the paddles
166        if currentGameType == .player2{
167          if location.y > 0 {
168            enemy.run(SKAction.moveTo(x: location.x, duration: 0.2))}
169          if location.y < 0{
170            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
171        }
172          //case for playing against cpu moving the paddle
173          else {
174            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
175        }
176    }
177
```

Με την touchesBegan η επαφή στην οθόνη αντιστοιχίζεται σε κίνηση ρακέτας. Στην περίπτωση των δύο παικτών η επαφή κάτω από το κέντρο κινεί την κάτω ρακέτα, αλλιώς έχουμε κίνηση της πάνω. Σε κάθε άλλη περίπτωση η επαφής κινεί την κάτω αποκλειστικά.

Game Scene

```
143    //activate action
144    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
145      for touch in touches {
146        let location = touch.location(in: self)
147        //case for two players
148        if currentGameType == .player2{
149          if location.y > 0 {
150            enemy.run(SKAction.moveTo(x: location.x, duration: 0.2))}
151          if location.y < 0{
152            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
153        }
154          //case for playing against cpu
155          else {
156            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
157      }
158    }
159
160    //manage touch movement
161    override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
162      for touch in touches {
163
164        let location = touch.location(in: self)
165         //case for two players and moving the paddles
166        if currentGameType == .player2{
167          if location.y > 0 {
168            enemy.run(SKAction.moveTo(x: location.x, duration: 0.2))}
169          if location.y < 0{
170            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
171        }
172          //case for playing against cpu moving the paddle
173          else {
174            friend.run(SKAction.moveTo(x: location.x, duration: 0.2))}
175        }
176    }
177
```

Με την touchesMoved οι συνεχόμενες επαφές κινούν τη ρακέτα προς τη φορά επαφής.
Ισχύουν οι ίδιες συνθήκες για την κίνηση της κάθε ρακέτας.

Game Scene

```
176    override func update(_ currentTime: TimeInterval) {
177      //case for game type
178      switch currentGameType {
179      case .easy:
180          enemy.run(SKAction.moveTo(x: ball.position.x , duration: 0.4))
181        break
182      case .medium:
183          enemy.run(SKAction.moveTo(x: ball.position.x , duration: 0.2))
184        break
185      case .hard:
186          enemy.run(SKAction.moveTo(x: ball.position.x , duration: 0.05))
187        break
188      case .player2:
189        break
190      }
191      //case for adding score to right player
192      if ball.position.y <= friend.position.y - 10 {
193        add_score(player_Won: enemy)}
194      if ball.position.y >= enemy.position.y + 10 {
195        add_score(player_Won: friend)}
196
197      //case for changing direction of the ball
198       if ball.position.y <= friend.position.y + 22 {
199        change(posi:friend)}
200      if ball.position.y >= enemy.position.y - 22 {
201        change(posi:enemy)}
202    }
203  }
```

Η συνάρτηση update είναι αυτή που ελέγχει όλο το παιχνίδι.

Ανάλογα με την τιμή της currentGameType η ρακέτα του υπολογιστή κινείται γρηγορότερα. Στην περίπτωση των δύο παικτών ισχύουν οι συνθήκες κίνησης από την touchesMoved.

Στη συνέχεια, ορίζονται οι συνθήκες απόκτησης πόντου, η ικανοποίηση τους οδηγεί σε κλήση της add_score.

Τέλος έχουμε την συνθήκη ανάκλασης υπο γωνία. Γίνεται κλήση της change.

Στα υπόλοιπα αρχεία που περιέχονται στο παραδοτέο πρόγραμμα δέν έχουν γίνει αλλαγές. Ο κώδικας που υπάρχει δημιουργείται αυτόματα μαζί με το project.

Νίκη Υπολογιστή

Κίνηση Υπολογιστή

Αρχικά είναι σημαντικό να γίνει κατανοητός ο τρόπος κίνησης του υπολογιστή.
Η ρακέτα του υπολογιστή δε κινείται τυχαία, αλλά ακολουθεί την κίνηση κατά τον άξονα χ της μπάλας. Σε όποιο σημείο της οθόνης είναι η μπάλα η ρακέτα του υπολογιστή κινείται παράλληλα προς αυτό.
Όσο μικρότερη είναι η διάρκεια κίνησης τόσο πιο αντιληπτό γίνεται από τον χρήστη.

Νίκη Υπολογιστή

Από όσα έχουμε αναλύσει γίνεται κατανοητό, πως όσο γρηγορότερα κινείται η ρακέτα του υπολογιστή τόσο δυσκολότερο είναι να χάσει την μπάλα. Συνεπώς αν η διάρκεια κίνησης γίνει μικρότερη από αυτή του παίκτη ο υπολογιστής δε θα χάνει ποτέ.

Μελλοντικές Βελτιώσεις

Μελλοντικές Βελτιώσεις

Σε επόμενες εκδόσεις τις εφαρμογής επιδιώκουμε:

- ⋆ Προσθήκη ήχου υποβάθρου και ανάκλασης μπάλας.
- ⋆ Προσθήκη επιπέδου με δύο μπάλες.
- ⋆ Επιλογή σκορ λήξης παρτίδας από τον χρήστη.
- ⋆ Επιλογή επιπέδων δυσκολίας για το παιχνίδι με δύο παίκτες.
- ⋆ Επιλογές σε εμφάνιση μπάλας και ρακετών
- ⋆ Επιπλέον ρυθμίσεις επιπέδων. Ιδανικά θέλουμε ο χρήστης να μπορεί να επιλέξει την ταχύτητα και τον αριθμό των σφαιρών, όπως και την εμφάνιση τους.

Με τις αναβαθμίσεις αυτές θέλουμε να πετύχουμε τόσο την βελτίωση της υπάρχουσας εφαρμογής όσο και τον εκσυγχρονισμό του παιχνιδιού.

Μετρική

Μετρική

Μετρική: Η ποσοτική εκτίμηση του βαθμού κατά τον οποίο ένα σύστημα κατέχει ένα χαρακτηριστικό

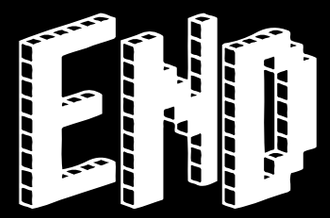
- αριθμός αρχείων : 18
- συνολικός αριθμός γραμμών κώδικα (χωρίς σχόλια) : 185
- αριθμός γραμμών σχολίων : 94
- αριθμός συναρτήσεων : 15
- μέγεθος κώδικα (αν γίνεται compile) : 1.2 MB
- memory footprint (απαιτήσεις σε μνήμη κατά τη διάρκεια εκτέλεσης): 23.8MB
- hot spots (σημεία κώδικα με αυξημένη πολυπλοκότητα και πολλαπλές επαναλήψεις) : 6
- πολυπλοκότητα κώδικα.Υπάρχουν 57 δομές ελέγχου στον κώδικα.

Συμπεράσματα

Συμπεράσματα

Το project φαινομενικά είναι απλό αλλα στην πράξη ήταν αρκετά απαιτητικό. Υπήρξαν αρκετά προβλήματα που έπρεπε να ξεπεραστούν από την κίνηση των ρακετών μέχρι την προσαρμογή μεγέθους οθόνης. Ωστόσο είναι ένα ευχάριστο project που δεν απαιτεί ειδική γνώση. Προσωπικά αποτέλεσε την πρώτη μου επαφή με τη γλώσσα swift και τον προγραμματισμό εφαρμογών. Κατά την πορεία προγραμματισμού και ανάπτυξης διαπίστωσα πως ενώ η swift είναι αρκετά εύκολη στην εκμάθηση υποστηρίζεται από λίγες συσκευές και συγκεκριμένο λογισμικό κάνοντας την εφαρμογή προσιτή σε λίγους. Ξεπερνώντας αυτό το μειονέκτημα θεωρώ πως είναι ένα καλό ξεκίνημα για όποιον θέλει να ασχοληθεί με ανάπτυξη παιχνιδιών ή να πειραματιστεί με εφαρμογές για κινητά.

Η ενασχόληση με το project μου κίνησε το ενδιαφέρον και με ώθησε να διαβασώ αρκετά για τις εφαρμογές σε κινητές συσκεύες. Πλεον επιδιώκω να ασχοληθω περαιτέρω με την ανάπτυξη εφαρμογών τόσο για ios όσο και για android.


END