2007

2019





Programowanie dla iOS

Kartografowie



Kartografowie

Kartografowie

- Gra typu Roll & Write (Flip & Write)

Kartografowie

- Gra typu Roll & Write (Flip & Write)
- Składa się z 41 kart i arkusza do rysowania

Kartografowie

- Gra typu Roll & Write (Flip & Write)
- Składa się z 41 kart i arkusza do rysowania



Zasady

Zasady

- Przygotowując grę:

Zasady

- Przygotowując grę:
  - rozkładamy karty edyktów (A-D),

Zasady

- Przygotowując grę:
  - rozkładamy karty edyktów (A-D),
  - układamy karty pór roku,

Zasady

- Przygotowując grę:
  - rozkładamy karty edyktów (A-D),
  - układamy karty pór roku,
  - dzielimy karty punktacji wg ich rodzaju,
    tasujemy osobno każdy rodzaj
    i wybieramy po 1 z nich, które to następnie
    tasujemy i przydzielamy do kart edyktów

Zasady

Zasady

- tasujemy karty zasadzek,

Zasady

- tasujemy karty zasadzek,
- przygotowując się do pory roku dobieramy I zakrytą kartę zasadzki i tasujemy ją ze wszystkimi (13) kartami odkryć,

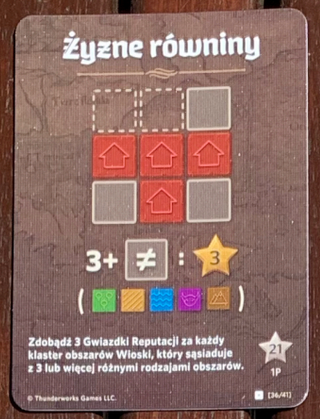
Zasady

- tasujemy karty zasadzek,
- przygotowując się do pory roku dobieramy I zakrytą kartę zasadzki i tasujemy ją ze wszystkimi (13) kartami odkryć,
- Pozostałe karty zasadzek będą sukcesywnie dodawane do talii kart odkryć w następnych porach roku

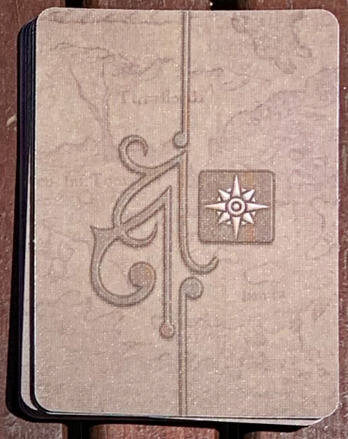
Zasady

- tasujemy karty zasadzek,
- przygotowując się do pory roku dobieramy I zakrytą kartę zasadzki i tasujemy ją ze wszystkimi (13) kartami odkryć,
  - Pozostałe karty zasadzek będą sukcesywnie dodawane do talii kart odkryć w następnych porach roku
- Możemy rozpocząć grę

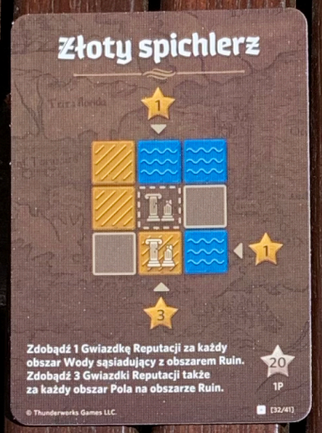

A
© Thunderworks Games LLC.
[22/41]


Żyzne równiny
3+ ≠ : 3
Zdobądź 3 Gwiazdki Reputacji za każdy klaster obszarów Wioski, który sąsiaduje z 3 lub więcej różnymi rodzajami obszarów.
21
1P
© Thunderworks Games LLC.
[36/41]


8
Wiosna
A
B
© Thunderworks Games LLC.
[18/41]




B
© Thunderworks Games LLC.
[23/41]


Złoty spichlerz
1
1
3
Zdobądź 1 Gwiazdkę Reputacji za każdy obszar Wody sąsiadujący z obszarem Ruin. Zdobądź 3 Gwiazdki Reputacji także za każdy obszar Pola na obszarze Ruin.
20
1P
© Thunderworks Games LLC.
[32/41]


2
2
1
2
2
Gospodarstwo
ODKRYCIE
© Thunderworks Games LLC.
[12/41]


C
© Thunderworks Games LLC.
[24/41]


Utracone włości
3
3
Zdobądź 3 Gwiazdki Reputacji za każdy obszar wzdłuż jednej krawędzi największego kwadratu, złożonego z wypełnionych obszarów.
24
1P
© Thunderworks Games LLC.
[39/41]


D
© Thunderworks Games LLC.
[25/41]


Zagajnik
1
1
1
1
1
Zdobądź 1 Gwiazdkę Reputacji za każdy rząd i każdą kolumnę z co najmniej jednym obszarem Lasu. Ten sam obszar Lasu może punktować zarówno w rzędzie, jak i w kolumnie.
22
1P
© Thunderworks Games LLC.
[27/41]

Zasady

Zasady

- W trakcie gry:

Zasady

- W trakcie gry:
  - odkrywamy kartę odkryć,

Zasady

- W trakcie gry:
  - odkrywamy kartę odkryć,
  - rysujemy kształt na arkuszu,

Zasady

- W trakcie gry:
  - odkrywamy kartę odkryć,
  - rysujemy kształt na arkuszu,
  - sprawdzamy warunek końca pory roku (suma czasu dotychczas odsłoniętych kart)

Zasady

- W trakcie gry:
  - odkrywamy kartę odkryć,
  - rysujemy kształt na arkuszu,
  - sprawdzamy warunek końca pory roku (suma czasu dotychczas odsłoniętych kart)
  - jeżeli odkryto kartę zasadzki to teraz odrzucamy ją z gry

Zasady

Zasady

- Jeżeli zakończyła się pora roku:

Zasady

- Jeżeli zakończyła się pora roku:
  - podliczamy punkty za obowiązującą w niej parę kart punktacji,

Zasady

- Jeżeli zakończyła się pora roku:
  - podliczamy punkty za obowiązującą w niej parę kart punktacji,
  - zbieramy karty odkryć, dodajemy kolejną kartę zasadzki i tasujemy,

Zasady

- Jeżeli zakończyła się pora roku:
  - podliczamy punkty za obowiązującą w niej parę kart punktacji,
  - zbieramy karty odkryć, dodajemy kolejną kartę zasadzki i tasujemy,
- Na koniec Zimy podliczamy punkty.

Aplikacja

Aplikacja

- Ma zastąpić w całości karty - potrzebujemy tylko arkusza i czegoś do pisania

Aplikacja

- Ma zastąpić w całości karty - potrzebujemy tylko arkusza i czegoś do pisania
- Koniec z kłopotliwym tasowaniem kart

Aplikacja

- Ma zastąpić w całości karty - potrzebujemy tylko arkusza i czegoś do pisania
- Koniec z kłopotliwym tasowaniem kart
- Napisana w SpriteKit

Wersja 1

Wersja 1

- Jeden ekran z kartami i wszystkim, co potrzebne do gry

Wersja 1

- Jeden ekran z kartami i wszystkim, co potrzebne do gry
- Można dobierać karty,

Wersja 1

- Jeden ekran z kartami i wszystkim, co potrzebne do gry
- Można dobierać karty,
- Kartę można zawsze powiększyć, żeby móc przeczytać tekst,

Wersja 1

- Jeden ekran z kartami i wszystkim, co potrzebne do gry
- Można dobierać karty,
- Kartę można zawsze powiększyć, żeby móc przeczytać tekst,
- Grę można zacząć od nowa.

Zasoby

Zasoby

- Wszystkie karty zeskanowane

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek
  - 5-17 - karty odkryć

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek
  - 5-17 - karty odkryć
  - 18-21 - karty pór roku

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek
  - 5-17 - karty odkryć
  - 18-21 - karty pór roku
  - 22-25 - karty edyktów (niepotrzebne)

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek
  - 5-17 - karty odkryć
  - 18-21 - karty pór roku
  - 22-25 - karty edyktów (niepotrzebne)
  - 26-41 - karty punktacji

Zasoby

- Wszystkie karty zeskanowane
- Każda karta ma numer
- Kart jest 41:
  - 1-4 - karty zasadzek
  - 5-17 - karty odkryć
  - 18-21 - karty pór roku
  - 22-25 - karty edyktów (niepotrzebne)
  - 26-41 - karty punktacji



Zasoby

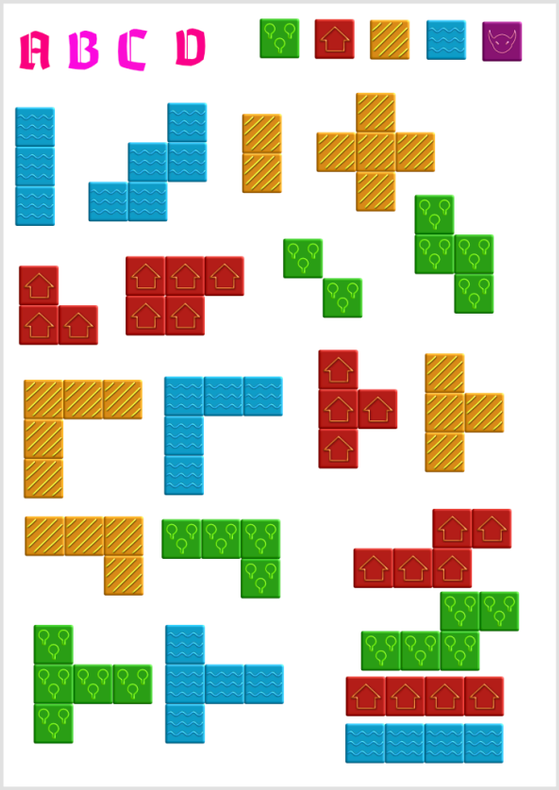
Zasoby

- Dla zwiększenia czytelności wszystkie kształty terenu wykonane jako obrazki

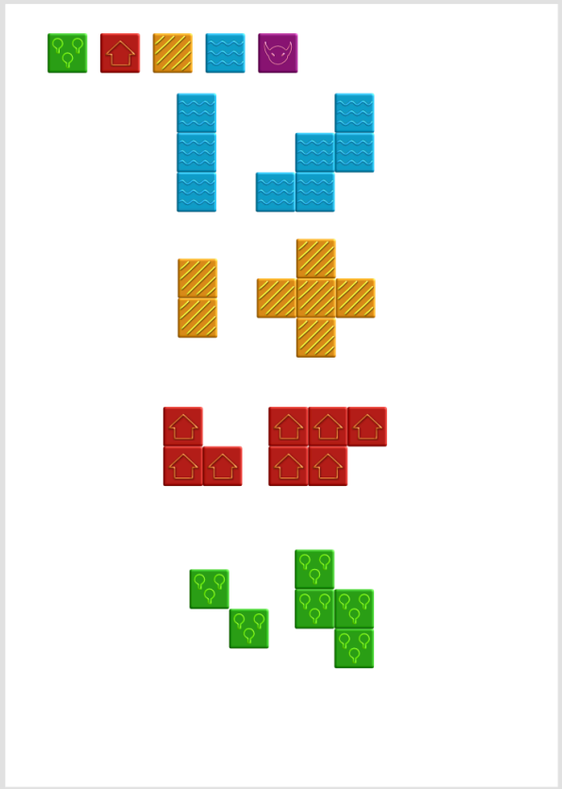
Zasoby

- Dla zwiększenia czytelności wszystkie kształty terenu wykonane jako obrazki

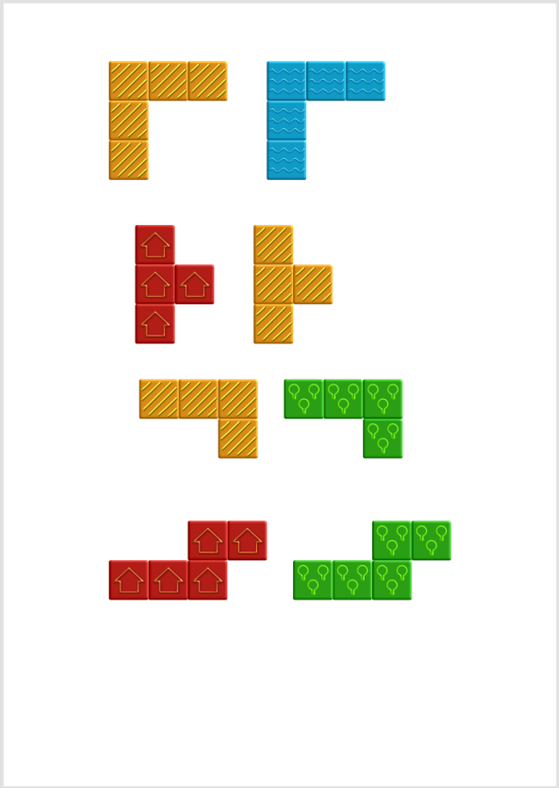
Artboard1



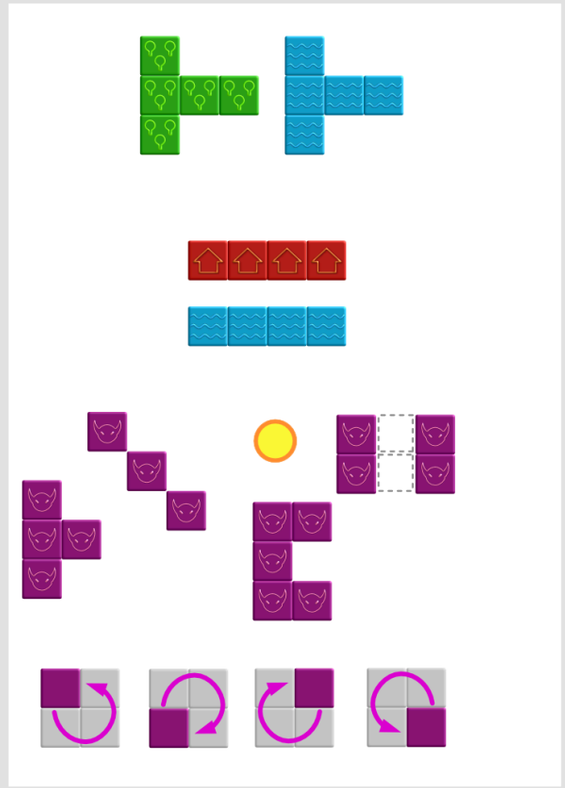
Artboard1

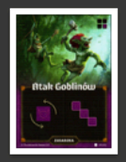


Artboard1



Artboard1

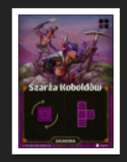




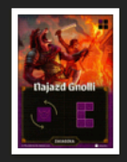
01.png



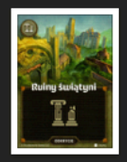
02.png



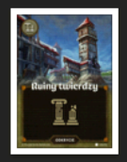
03.png



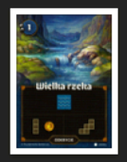
04.png



05.png



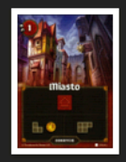
06.png



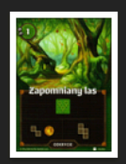
07.png



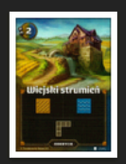
08.png



09.png



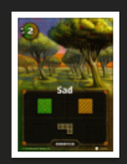
10.png



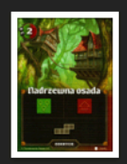
11.png



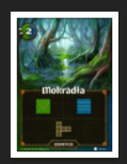
12.png



13.png



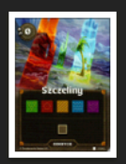
14.png



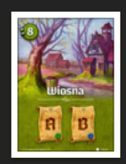
15.png



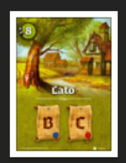
16.png



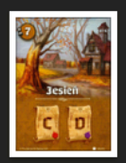
17.png



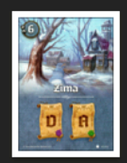
18.png



19.png



20.png



21.png



26.png



27.png



28.png



29.png



30.png



31.png



32.png



33.png



34.png



35.png



36.png



37.png



38.png



39.png



40.png



41.png



A.png



B.png



background.png

C.png

D.png

monster.png

PA01.png

PA02.png

PA03.png

PA04.png

PS1.png

PS2.png

PS3.png

PS4.png

PS5.png

PS6.png

PS7.png

PS8.png

ruins.png

title.png



s01.png



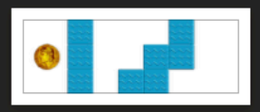
s02.png



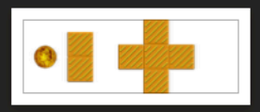
s03.png



s04.png



s07.png



s08.png



s09.png



s10.png



s11.png



s12.png



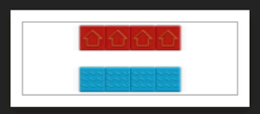
s13.png



s14.png



s15.png



s16.png



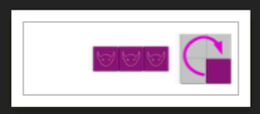
s17.png



sP01.png



sP02.png



sP03.png



sP04.png

Początek

Początek

- Tworzymy nowy projekt z szablonu Game oraz wybieramy SpriteKit i język Swift

Początek

- Tworzymy nowy projekt z szablonu Game oraz wybieramy SpriteKit i język Swift
- Projekt będzie się nazywał Cartographers z identyfikatorem put.Cartographers

Początek

- Tworzymy nowy projekt z szablonu Game oraz wybieramy SpriteKit i język Swift
- Projekt będzie się nazywał Cartographers z identyfikatorem put.Cartographers
- Nie będziemy korzystać z projektu sceny SKS

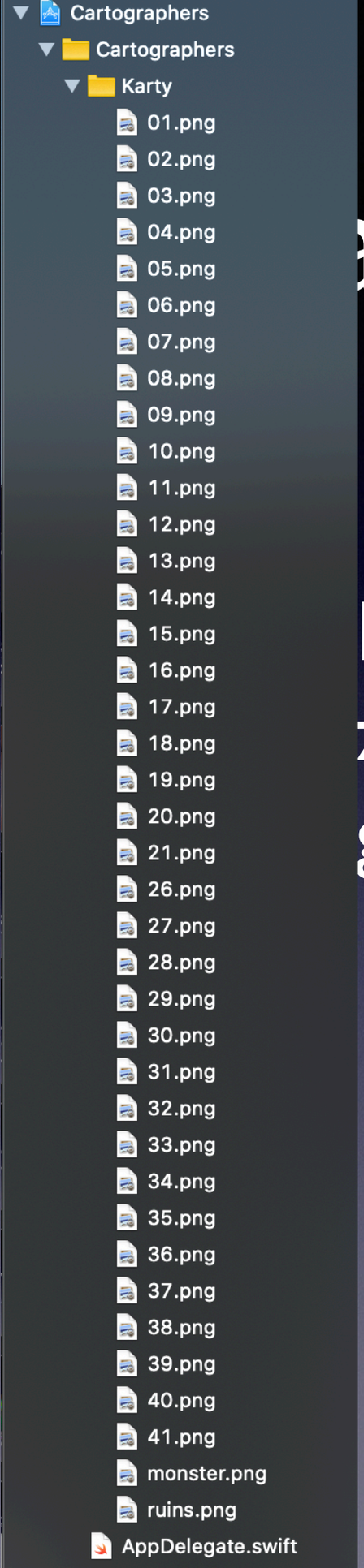
Zasoby i wczytanie sceny

Zasoby i wczytanie sceny

- Do podfolderu Karty wrzucamy wszystkie zeskanowane karty i kilka dodatkowych grafik

asoby i wczytanie sceny

Do podfolderu Karty wrzucamy wszystkie zeskanowane karty i kilka dodatkowych grafik



Zasoby i wczytanie sceny

- Do podfolderu Karty wrzucamy wszystkie zeskanowane karty i kilka dodatkowych grafik
- Zmieniamy metodę viewDidLoad w klasie GameViewController żeby nie wczytywała pliku SKS

Zasoby i wczytanie sceny

- Do
zes
gra
- Zm
Ga
plik

```
override func viewDidLoad() {
    super.viewDidLoad()

    if let view = self.view as! SKView? {
        // Load the SKScene from 'GameScene.sks'
        if let scene = SKScene(fileNamed: "GameScene") {
            // Set the scale mode to scale to fit the window
            scene.scaleMode = .aspectFill

            // Present the scene
            view.presentScene(scene)
        }

        view.ignoresSiblingOrder = true

        //view.showsFPS = true
        //view.showsNodeCount = true
    }
}
```

Klasa GameScene

Klasa GameScene

- Dodajemy pola, które będą obsługiwać nasze sprite'y

Klasa GameScene

- Dodajemy pola, które będą obsługiwać nasze sprite'y

```
private var label : SKLabelNode?
private var spinnyNode : SKShapeNode?
private var goalANode: SKShapeNode?
private var goalBNode: SKShapeNode?
private var goalCNode: SKShapeNode?
private var goalDNode: SKShapeNode?
private var seasonNode: SKShapeNode?
private var seasonLabel: SKLabelNode?
private var newLabel: SKLabelNode?
private var totalLabel: SKLabelNode?
private var modifierNode: SKShapeNode?
private var previewNode: SKShapeNode?
```

Klasa GameScene

- Dodajemy pola, które będą obsługiwać nasze sprite'y
- I dane gry

```
private var label : SKLabelNode?
private var spinnyNode : SKShapeNode?
private var goalANode: SKShapeNode?
private var goalBNode: SKShapeNode?
private var goalCNode: SKShapeNode?
private var goalDNode: SKShapeNode?
private var seasonNode: SKShapeNode?
private var seasonLabel: SKLabelNode?
private var newLabel: SKLabelNode?
private var totalLabel: SKLabelNode?
private var modifierNode: SKShapeNode?
private var previewNode: SKShapeNode?
```

Klasa GameScene

- Dodajemy pola, które będą obsługiwać nasze sprite'y
- I dane gry

```
private var label : SKLabelNode?
private var spinnyNode : SKShapeNode?
private var goalANode: SKShapeNode?
private var goalBNode: SKShapeNode?
private var goalCNode: SKShapeNode?
private var goalDNode: SKShapeNode?
private var seasonNode: SKShapeNode?
private var seasonLabel: SKLabelNode?
private var newLabel: SKLabelNode?
private var totalLabel: SKLabelNode?
private var modifierNode: SKShapeNode?
private var previewNode: SKShapeNode?
```

```
private var currentSeason = 18 //Wiosna
private var exploreCards: [String]?
private var exploreDeck: [String]?
private var ambushCardsLeft: [String]?
private var ambushCards: [String]?
private var h:CGFloat=0
private var w:CGFloat=0
private var s:CGFloat=0
private var originalPosition:CGPoint=CGPoint(x: 0,y: 0)

private var currentSeasonValue=0
private var currentSeasonLimit=8
private var countExplores=0
private var dontHideRuins = false
private var previewShown = false
private var cardValues:NSDictionary?
```

Klasa GameScene

Klasa GameScene

- Cała inicjalizacja widoku odbywa się w metodzie didMove klasy

Klasa GameScene

```
override func didMove(to view: SKView) {
    h = self.size.height/6
    s = h/7
    w = h/88*63
    var totalValue = 0

    if cardValues == nil {
        if let path = Bundle.main.path(forResource: "CardValues", ofType: "plist") {
            cardValues = NSDictionary(contentsOfFile: path)
        }
    }

    //wylosuj karty edyktów
    var goalList = [Int]()
    goalList.append(Int.random(in: 26...29))
    goalList.append(Int.random(in: 30...33))
    goalList.append(Int.random(in: 34...37))
    goalList.append(Int.random(in: 38...41))
```

```
//generowanie UI
let goalMargin = -self.size.width/2+10+w
var currentGoal = 1.5*(s+h)

self.seasonLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let seasonLabel = self.seasonLabel {
    seasonLabel.fontSize = 30
    seasonLabel.fontColor = SKColor.white
    seasonLabel.position=CGPoint(x: -goalMargin, y: h)
    seasonLabel.text = "0/8"
    seasonLabel.name = "SeasonLimit"
    self.addChild(seasonLabel)
}

self.goalANode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w *
    0.05)
if let goalANode = self.goalANode {
    goalANode.position = CGPoint(x: goalMargin, y: currentGoal)
    goalANode.fillColor = SKColor.white
    let num = Int.random(in: 0..<goalList.count)
    let name = String(goalList[num])
    totalValue += cardValues!.object(forKey: name) as! Int
    goalList.remove(at: num)
    goalANode.fillTexture = SKTexture(imageNamed: name)
    goalANode.name = name
    self.addChild(goalANode)
    currentGoal -= (h+s)
}
```

```
self.goalBNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w *
    0.05)
if let goalBNode = self.goalBNode {
    goalBNode.position = CGPoint(x: goalMargin, y: currentGoal)
    goalBNode.fillColor = SKColor.white
    let num = Int.random(in: 0..<goalList.count)
    let name = String(goalList[num])
    totalValue += cardValues!.object(forKey: name) as! Int
    goalList.remove(at: num)
    goalBNode.fillTexture = SKTexture(imageNamed: name)
    goalBNode.name = name
    self.addChild(goalBNode)
    currentGoal -= (h+s)
}
```

```
self.goalCNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w *
    0.05)
if let goalCNode = self.goalCNode {
    goalCNode.position = CGPoint(x: goalMargin, y: currentGoal)
    goalCNode.fillColor = SKColor.gray
    let num = Int.random(in: 0..<goalList.count)
    let name = String(goalList[num])
    totalValue += cardValues!.object(forKey: name) as! Int
    goalList.remove(at: num)
    goalCNode.fillTexture = SKTexture(imageNamed: name)
    goalCNode.name = name
    self.addChild(goalCNode)
    currentGoal -= (h+s)
}
```

```
self.goalDNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w *
    0.05)
if let goalDNode = self.goalDNode {
    goalDNode.position = CGPoint(x: goalMargin, y: currentGoal)
    goalDNode.fillColor = SKColor.gray
    let num = Int.random(in: 0..<goalList.count)
    let name = String(goalList[num])
    totalValue += cardValues!.object(forKey: name) as! Int
    goalList.remove(at: num)
    goalDNode.fillTexture = SKTexture(imageNamed: name)
    goalDNode.name = name
    self.addChild(goalDNode)
    currentGoal -= (h+s)
}

self.seasonNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w *
    0.05)
if let seasonNode = self.seasonNode {
    seasonNode.position = CGPoint(x: -goalMargin, y: 1.5*(s+h))
    seasonNode.fillColor = SKColor.white
    let name = String(currentSeason)
    seasonNode.fillTexture = SKTexture(imageNamed: name)
    seasonNode.name = "Season"
    self.addChild(seasonNode)
}
```

```
self.modifierNode = SKShapeNode.init(rectOf: CGSize.init(width: w, height: w))
if let modifierNode = self.modifierNode {
    modifierNode.position = CGPoint(x: -goalMargin, y: 0)
    modifierNode.fillColor=SKColor.white
    modifierNode.fillTexture = nil
    modifierNode.name = "Modifier"
    modifierNode.strokeColor=SKColor.black
    self.addChild(modifierNode)
    modifierNode.isHidden = true
    let fadeInAction=SKAction.fadeIn(withDuration: 0.5)
    let fadeOutAction=SKAction.fadeOut(withDuration: 0.5)
modifierNode.run(SKAction.repeatForever(SKAction.sequence([fadeInAction,fadeOutAction])))
}

self.newLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let newLabel = self.newLabel {
    newLabel.fontSize = 30
    newLabel.fontColor = SKColor.white
    newLabel.position=CGPoint(x: -goalMargin, y: -2*h)
    newLabel.text = "Wyzeruj"
    newLabel.name = "NewGame"
    self.addChild(newLabel)
}

self.totalLabel = SKLabelNode.init(fontNamed: "Thonburi-Bold")
if let totalLabel = self.totalLabel {
    totalLabel.fontSize = 20
    totalLabel.fontColor = SKColor.white
    totalLabel.position=CGPoint(x: -goalMargin, y: -2*h+50)
    totalLabel.text = "[\(totalValue)]"
    totalLabel.name = "Total"
    self.addChild(totalLabel)
}
```

Klasa GameScene

```
//inicjalizacja logiki
initializeCards()
generateExploreDeck()
}
```

- Cała inicja va się w metodzie didMove klasy

Klasa GameScene

- Cała inicjalizacja widoku odbywa się w metodzie didMove klasy

Obsługa logiki gry

Obsługa logiki gry

```
func initializeCards() {
    self.ambushCards = [String]()
    self.ambushCardsLeft = [String]()
    for c in 1...4 {
        ambushCards?.append(String(format: "%02d", c))
        ambushCardsLeft?.append(String(format: "%02d", c))
    }
    ambushCardsLeft?.shuffle()

    exploreCards = [String]()
    for i in 5...17 {
        exploreCards?.append(String(format: "%02d", i))
    }
    currentSeason = 17 //o 1 mniej niż wiosna
}
```

```
func generateExploreDeck() {
    var toRemove:[SKNode] = [SKNode]()
    for node in self.children {
        if let n=node.name {
            if n.contains("Explore") {
                toRemove.append(node)
            }
        }
    }
    self.removeChildren(in: toRemove)

    currentSeason += 1
    countExplores = 0
    if currentSeason < 22 {
        exploreDeck=[String]()
        //ile dodać zasadzek?
        let val = ambushCardsLeft!.count-21+currentSeason

        for i in 0..<val {
            exploreDeck?.append(ambushCardsLeft![i])
        }

        for s in exploreCards! {
            exploreDeck?.append(s)
        }

        exploreDeck?.shuffle()
```

```
currentSeasonValue = 0
seasonLabel?.fontColor = SKColor.white
let name = String(currentSeason)
seasonNode!.fillTexture = SKTexture(imageNamed: name)

switch currentSeason {
case 18:
    currentSeasonLimit=8
    goalANode?.fillColor=SKColor.white
    goalBNode?.fillColor=SKColor.white
    goalCNode?.fillColor=SKColor.gray
    goalDNode?.fillColor=SKColor.gray
case 19:
    currentSeasonLimit=8
    goalANode?.fillColor=SKColor.gray
    goalBNode?.fillColor=SKColor.white
    goalCNode?.fillColor=SKColor.white
    goalDNode?.fillColor=SKColor.gray
case 20:
    currentSeasonLimit=7
    goalANode?.fillColor=SKColor.gray
    goalBNode?.fillColor=SKColor.gray
    goalCNode?.fillColor=SKColor.white
    goalDNode?.fillColor=SKColor.white
case 21:
    currentSeasonLimit=6
    goalANode?.fillColor=SKColor.white
    goalBNode?.fillColor=SKColor.gray
    goalCNode?.fillColor=SKColor.gray
    goalDNode?.fillColor=SKColor.white
default:
    currentSeasonLimit=0
    goalANode?.fillColor=SKColor.gray
    goalBNode?.fillColor=SKColor.gray
    goalCNode?.fillColor=SKColor.gray
    goalDNode?.fillColor=SKColor.gray
}
nextExplore()
    }
}
```

```
func nextExplore() {
    var finish=false
    if currentSeasonValue < seasonLimit() {
        let currentExplore = exploreDeck?.popLast()!
        countExplores += 1
        let node = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h), cornerRadius: w * 0.05)
        var currentPosition = 1.5*(s+h)+50
        currentPosition -= CGFloat(countExplores)*50

        node.position = CGPoint(x: 0, y: currentPosition)
        node.fillColor = SKColor.white
        let name = currentExplore!
        node.fillTexture = SKTexture(imageNamed: name)
        node.name = "Explore\(name)"
        self.addChild(node)

        let csv=exploreValue(cardNumber: currentExplore!)
        if csv > 0 {
            currentSeasonValue += csv
            if currentSeasonValue >= currentSeasonLimit {
                seasonLabel?.fontColor = SKColor.red
                if currentSeason==21 {
                    seasonLabel?.text = "Koniec"
                    finish=true
                }
            }
        }
```

```
        if csv < 0 { //Ruiny i zasadzki
            if csv == -1 {
                modifierNode?.fillTexture=SKTexture(imageNamed: "ruins")
                dontHideRuins = true
                nextExplore()
            }
            else {
                modifierNode?.fillTexture=SKTexture(imageNamed: "monster")
                var index=0
                for i in 0..<ambushCardsLeft!.count {
                    if ambushCardsLeft![i] == name {
                        index = i
                    }
                }
                ambushCardsLeft!.remove(at: index)
            }
            modifierNode?.isHidden = false
        }
        else {
            if !dontHideRuins {
                modifierNode?.isHidden = true
            }
            else
            {
                modifierNode?.fillTexture=SKTexture(imageNamed: "ruins")
                dontHideRuins = false
            }
        }

        if !finish {
            seasonLabel?.text = "\(currentSeasonValue)/\(currentSeasonLimit)"
        }
    }
    else {
        generateExploreDeck()
    }
}
```

Obsługa logiki gry

Metody pomocnicze

Metody pomocnicze

```
func exploreValue(cardNumber: String) -> Int {
    let cn = Int(cardNumber)
    switch cn! {
    case 17:
        return 0
    case 11...16:
        return 2
    case 7...10:
        return 1
    case 5...6:
        return -1 //Ruiny
    case 1...4:
        return -2 //Zasadzka
    default:
        return 0
    }
}
```

Metody pomocnicze

```
func exploreValue(cardNumber: String) -> Int {
    let cn = Int(cardNumber)
    switch cn! {
    case 17:
        return 0
    case 11...16:
        return 2
    case 7...10:
        return 1
    case 5...6:
        return -1 //Ruiny
    case 1...4:
        return -2 //Zasadzka
    default:
        return 0
    }
}
```

```
func seasonLimit() -> Int {
    switch currentSeason {
    case 18...19:
        return 8
    case 20:
        return 7
    case 21:
        return 6
    default:
        return 0
    }
}
```

Obsługa dotyku

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

    guard let touch = touches.first else {
        return
    }

    let location = touch.location(in: self)
    let frontTouchedNode = atPoint(location)
    if let name=frontTouchedNode.name {
        switch name {
        case "NewGame":
            showAlert(withTitle: "Nowa gra", message: "Czy zacząć od nowa?", function:
                self.okActionFunc)
            return
        case "Preview":
            dismissPreview()
        case "Season":
            if previewShown {
                dismissPreview()
            }
            nextExplore()
        case let str where str.contains("Explore"):
            //let n = name.suffix(2)
            showPreview(toShow: frontTouchedNode as! SKShapeNode)
            return
        default:
            showPreview(toShow: frontTouchedNode as! SKShapeNode)
        }
    }
}
```

Powiększanie
i pomniejszanie

Powiekszanie

```
func showPreview(toShow node: SKShapeNode) {
    if !previewShown {
        self.previewNode = node.copy() as? SKShapeNode
        if let previewNode = self.previewNode {
            originalPosition = node.position
            let actionMove = SKAction.move(to: CGPoint(x: 0, y: 0), duration: 0.5)
            let actionScale = SKAction.scale(by: 3.0, duration: 0.5)
            previewNode.name = "Preview"
            previewNode.fillColor=SKColor.white
            previewNode.zPosition = 1
            self.addChild(previewNode)
            previewNode.run(actionMove)
            previewNode.run(actionScale)
            previewShown = true
        }
    } else
    {
        if let previewNode=self.previewNode {
            previewNode.fillTexture = node.fillTexture
            originalPosition = node.position
        }
    }
}
```

Powiększanie i pomniejszanie

```
func dismissPreview() {
    if let preview = self.previewNode {
        let moveAction=SKAction.move(to: originalPosition, duration: 0.5)
        let scaleAction = SKAction.scale(by: 0.3, duration: 0.5)
        let removeAction = SKAction.removeFromParent()
        preview.run(SKAction.sequence(
            [SKAction.group([scaleAction,moveAction])
            ,removeAction]))
        self.previewNode=nil
        previewShown=false
    }
}
```

Powiększanie
i pomniejszanie

Nowa gra

Nowa gra

- Przed wyzerowaniem gry aplikacja ma się zapytać o potwierdzenie użytkownika

Nowa gra

- Przed wyzerowaniem gry aplikacja ma się zapytać o potwierdzenie użytkownika
- Scena w SpriteKit ma problem z dostępem do elementów UIKit

Nowa gra

- Przed wyzerowaniem gry aplikacja ma się zapytać o potwierdzenie użytkownika
- Scena w SpriteKit ma problem z dostępem do elementów UIKit
- Rozszerzymy klasę SKScene o możliwość wyświetlania alertów

Nowa gra

```
import SpriteKit

protocol Alertable { }
extension Alertable where Self: SKScene {

    func showAlert(withTitle title: String, message: String, function okActionFunc: @escaping
    (UIAlertAction)-> Void) {

        let alertController = UIAlertController(title: title, message: message, preferredStyle: .alert)

        let okAction = UIAlertAction(title: "OK", style: .default, handler: okActionFunc)
        alertController.addAction(okAction)
        let cancelAction = UIAlertAction(title: "Anuluj", style: .cancel, handler: nil)
        alertController.addAction(cancelAction)

        view?.window?.rootViewController?.present(alertController, animated: true)
    }
}
```

8:55 AM Wed Jun 10

100%

Shortcuts

Cartographers

8:55 AM Wed Jun 10

100%

Shortcuts

Cartographers

Wersja 2

Wersja 2

- Ekran początkowy:

Wersja 2

- Ekran początkowy:
  - Możliwość włączenia rozszerzeń

Wersja 2

- Ekran początkowy:
  - Możliwość włączenia rozszerzeń
- Ekran końcowy:

Wersja 2

- Ekran początkowy:
  - Możliwość włączenia rozszerzeń
- Ekran końcowy:
  - Możliwość podliczania punktów

Wersja 2

- Ekran początkowy:
  - Możliwość włączenia rozszerzeń
- Ekran końcowy:
  - Możliwość podliczania punktów
- Odczyt stanu gry (gra miesiąca)

Klasa StartScene

Klasa StartScene

- Dodajemy nową klasę, która będzie dziedziczyć po SKScene

Klasa StartScene

- Dodajemy nową klasę, która będzie dziedziczyć po SKScene

```
import Foundation
import SpriteKit

class StartScene: SKScene {

    var skillsLabel:SKLabelNode?
    var promoALabel:SKLabelNode?


    var background = SKSpriteNode(imageNamed: "title")
```

Klasa StartScene

Klasa StartScene

- Tworzenie widoku w metodzie didMove

```
override func didMove(to view: SKView) {
    background.position = CGPoint(x: 0, y: 0)
    background.size = self.size
    background.zPosition = -1
    addChild(background)
    self.anchorPoint=CGPoint(x: 0.5,y: 0.5)
    print(self.size)

    skillsLabel = SKLabelNode.init(fontNamed: "Chalkduster")
    if let skillsLabel=self.skillsLabel {
        skillsLabel.fontSize = 0.03*self.size.height
        skillsLabel.fontColor = SKColor.white
        skillsLabel.position=CGPoint(x: 0+self.size.width*0.2, y: 0)
        skillsLabel.text = "Umiejętności"
        skillsLabel.name = "Skills"
        self.addChild(skillsLabel)
    }

    promoALabel = SKLabelNode.init(fontNamed: "Chalkduster")
    if let promoALabel=self.promoALabel {
        promoALabel.fontSize = 0.03*self.size.height
        promoALabel.fontColor = SKColor.white
        promoALabel.position=CGPoint(x: 0+self.size.width*0.2, y:
            -self.size.height/2*0.2)
        promoALabel.text = "Zasadzki promo"
        promoALabel.name = "PromoA"
        self.addChild(promoALabel)
    }
    let startLabel = SKLabelNode.init(fontNamed: "Chalkduster")
    startLabel.fontSize = 0.03*self.size.height
    startLabel.fontColor = SKColor.white
    startLabel.position=CGPoint(x: 0+self.size.width*0.2, y: -self.size.height/2*0.8)
    startLabel.text = "Rozpocznij"
    startLabel.name = "Start"
    self.addChild(startLabel)
}
```

Klasa StartScene

Klasa StartScene

- Obsługa dotyku

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

 guard let touch = touches.first else {
     return
 }

 let location = touch.location(in: self)
 let frontTouchedNode = atPoint(location)
 if let name=frontTouchedNode.name {
    print(name)
     switch name {
     case "Start":
        let nextScene = GameScene(size: self.scene!.size)
        var useSkills = false
        var usePromoA = false
        if skillsLabel!.fontColor == .red {
            useSkills = true
        }
        if promoALabel!.fontColor == .red {
            usePromoA = true
        }
        let gameState = GameState(skills: useSkills, promoA: usePromoA)
        nextScene.gameState = gameState
        nextScene.scaleMode = self.scaleMode
        nextScene.anchorPoint = self.anchorPoint
        nextScene.backgroundColor = UIColor.black
        self.view?.presentScene(nextScene, transition: SKTransition.fade(with:
            UIColor.black, duration: 0.5))
```

Klasa StartScene

- Obsługa dotyku

Klasa StartScene

- C

```
    case "Skills":
        if skillsLabel!.fontColor == .red {
                skillsLabel!.fontColor = .white
        }
        else {
            skillsLabel!.fontColor = .red
        }
    case "PromoA":
        if promoALabel!.fontColor == .red {
            promoALabel!.fontColor = .white
        }
        else {
            promoALabel!.fontColor = .red
        }
     default:
        return
    }
    }
}
```

Klasa StartScene

- Obsługa dotyku

Wyliczenie Seasons

Wyliczenie Seasons

- Umożliwia zapamiętanie pory roku

V S

- U

```swift
import Foundation

enum Seasons:Int {
    case Spring=1
    case Summer
    case Autumn
    case Winter

    mutating func next() {
        if self.rawValue < 4 {
            self = Seasons(rawValue: self.rawValue+1)!
        }
    }

    func cardNumber() -> String {
        switch self {
        case .Spring:
            return "18"
        case .Summer:
            return "19"
        case .Autumn:
            return "20"
        case .Winter:
            return "21"
        }
    }
}
```

Rozszerzenie klasy String

Rozszerzenie klasy String

- W Swift klasa String i pochodne potrafi wiele, ale czasami ciężko z niej korzystać,

Rozszerzenie klasy String

- W Swift klasa String i pochodne potrafi wiele, ale czasami ciężko z niej korzystać,
- Np. przy wycinaniu fragmentów.

Rozszerzenie klasy String

```
import Foundation

extension String {
    subscript(_ range: CountableRange<Int>) -> String {
        let start = index(startIndex, offsetBy: max(0, range.lowerBound))
        let end = index(startIndex, offsetBy: min(self.count, range.upperBound))
        return String(self[start..<end])
    }

    subscript(_ range: CountablePartialRangeFrom<Int>) -> String {
        let start = index(startIndex, offsetBy: max(0, range.lowerBound))
         return String(self[start...])
    }
}
```

Klasa GameState

Klasa GameState

- Klasa ma służyć do zapamiętania instancji gry:

Klasa GameState

- Klasa ma służyć do zapamiętania instancji gry:
- Ustawienia opcji,

Klasa GameState

- Klasa ma służyć do zapamiętania instancji gry:
  - Ustawienia opcji,
  - Kolejność kart,

Klasa GameState

- Klasa ma służyć do zapamiętania instancji gry:
  - Ustawienia opcji,
  - Kolejność kart,
  - Aktualny stan gry.

Klasa GameState

```swift
import Foundation
import UIKit

class GameState {
    var GoalA = ""
    var GoalB = ""
    var GoalC = ""
    var GoalD = ""

    var springCards:[String]?
    var summerCards:[String]?
    var autumnCards:[String]?
    var winterCards:[String]?
    var skillCards:[String]?

    var useSkills = false
    var usePromoAmbush = false
    var sheet = "A"

    var currentSeason = Seasons.Spring

    private var currentCard = 0
```

Klasa GameState

```
init(skills useSkills:Bool, promoA usePromoA:Bool) {
    currentSeason = Seasons.Spring
    currentCard = 0

    var ambushCards = [String]()
    var ambushCardsLeft = [String]()
    for c in 1...4 {
        ambushCards.append(String(format: "%02d", c))
        ambushCardsLeft.append(String(format: "%02d", c))
    }
    ambushCardsLeft.shuffle()

    var goalList = [Int]()
    goalList.append(Int.random(in: 26...29))
    goalList.append(Int.random(in: 30...33))
    goalList.append(Int.random(in: 34...37))
    goalList.append(Int.random(in: 38...41))
    goalList.shuffle()

    GoalA = String(format: "%02d", goalList[0])
    GoalB = String(format: "%02d", goalList[1])
    GoalC = String(format: "%02d", goalList[2])
    GoalD = String(format: "%02d", goalList[3])

    springCards = generateSeason(season: .Spring, ambushes: &ambushCardsLeft)
    summerCards = generateSeason(season: .Summer, ambushes: &ambushCardsLeft)
    autumnCards = generateSeason(season: .Autumn, ambushes: &ambushCardsLeft)
    winterCards = generateSeason(season: .Winter, ambushes: &ambushCardsLeft)

    if useSkills {
        self.useSkills = true
        skillCards = [String]()
        for i in 1...8 {
            skillCards!.append("PS\(i)")
        }
        skillCards!.shuffle()
    }
    if usePromoA {
        self.usePromoAmbush = true
    }
}
```

Klasa GameState

Stan gry

Stan gry

- Stan gry może być zapisany w stringu o postaci:

Stan gry

- Stan gry może być zapisany w stringu o postaci:

```
put.Cartographers://gotm/31|26|35|40|9|10|11|
12|13|3|6|5|16|14|15|17|7|10|6|16|9|5|13|14|8|
11|17|12|6|17|10|16|4|7|2|11|15|13|14|9|1|13|
15|12|11|9|7|16|5|10|8|17|A|-|-|-
```

Stan gry

- Stan gry może być zapisany w stringu o postaci:

```
put.Cartographers://gotm/31|26|35|40|9|10|11|
12|13|3|6|5|16|14|15|17|7|10|6|16|9|5|13|14|8|
11|17|12|6|17|10|16|4|7|2|11|15|13|14|9|1|13|
15|12|11|9|7|16|5|10|8|17|A|-|-|-
```

  - Dlaczego taki, okaże się za chwilę

```
init(state data:String) {
    currentSeason = Seasons.Spring
    currentCard = 0

    let separators = CharacterSet(["|"])
    let cards = data.components(separatedBy: separators)
    GoalA = cards[0]
    GoalB = cards[1]
    GoalC = cards[2]
    GoalD = cards[3]

    var index=4
    springCards=loadSeason(season: .Spring, cards: cards, index: &index)
    summerCards=loadSeason(season: .Summer, cards: cards, index: &index)
    autumnCards=loadSeason(season: .Autumn, cards: cards, index: &index)
    winterCards=loadSeason(season: .Winter, cards: cards, index: &index)

    currentSeason = .Spring

    sheet = cards[index]
    index += 1
    if cards[index] != "-" {
        useSkills = true
        skillCards=[String]()
        for i in 0...2 {
            skillCards?.append(cards[index+i])
        }
    }
    else
    {
        useSkills = false
        skillCards = nil
    }
}
```

Metoda zapisująca stan gry do stringa

```
public func toString() -> String {
    var result = "put.Cartographers://gotm/"
    result += GoalA
    result += "|"
    result += GoalB
    result += "|"
    result += GoalC
    result += "|"
    result += GoalD
    result += "|"
    for c in springCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
    for c in summerCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
```

Metoda zapisująca stan gry do stringa

```
public func toString() -> String {
    var result = "put.Cartographers://gotm/"
    result += GoalA
    result += "|"
    result += GoalB
    result += "|"
    result += GoalC
    result += "|"
    result += GoalD
    result += "|"
    for c in springCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
    for c in summerCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
    for c in autumnCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
    for c in winterCards! {
        if c[0..<1] == "0" {
            result += c[1...]
        }
        else {
            result += c
        }
        result += "|"
    }
    result += "A|-|-|-"
    return result
}
```

Metoda zapisująca stan gry do stringa

Metody obsługujące grę

Metody obsługujące grę

```
private func loadSeason(season cSeason:Seasons,cards:[String],index cIndex:inout Int)
    -> [String] {
    var result=[String]()
    currentSeason=cSeason
    var sum=0
    while sum<seasonLimit() {
        var card = cards[cIndex]
        let time = GameState.exploreValue(cardNumber: card)
        if time > 0 {
            sum += time
        }
        if card.count==1 {
            card = "0\(card)"
        }
        result.append(card)
        cIndex += 1
    }
    cIndex -= 1
    return result
}
```

Metody obsługujące grę

Metody obsługujące grę

```
private func generateSeason(season currentSeason:Seasons, ambushes ambushCardsLeft:
    inout [String]) -> [String] {

    var result=[String]()
    //przygotuj listę kart odkryć razem z ruinami
    var exploreCards = [String]()
    for i in 5...17 {
        exploreCards.append(String(format: "%02d", i))
    }

    let val = ambushCardsLeft.count-4+currentSeason.rawValue

    for i in 0..<val {
        exploreCards.append(ambushCardsLeft[i])
    }
    exploreCards.shuffle()

    for i in 0..<12 {
        result.append(exploreCards[i])
        if let index=ambushCardsLeft.firstIndex(of: exploreCards[i]) {
            ambushCardsLeft.remove(at: index)
        }
    }

    return result
}
```

Metody obsługujące grę

Metody obsługujące grę

```
public func getNextCard() -> String {
    if currentCard < 12
    {
        var s = ""
        switch currentSeason {
        case .Spring:
            s = springCards![currentCard]
        case .Summer:
            s = summerCards![currentCard]
        case .Autumn:
            s = autumnCards![currentCard]
        case .Winter:
            s = winterCards![currentCard]
        }
        currentCard += 1
        return s
    }
    else
    {
        return ""
    }
}
```

Metody obsługujące grę

Metody obsługujące grę

```
public func nextSeason() {
    if currentSeason != .Winter {
        currentSeason.next()
        currentCard = 0
    }
}

public func noOfCards() -> Int {
    return currentCard+1
}
```

Metody obsługujące grę

Metody obsługujące grę

```
public static func exploreValue(cardNumber: String) -> Int {
    let first=Character(cardNumber[0..<1])
    if first.isNumber {
        if let cn = Int(cardNumber) {
            switch cn {
            case 11...16:
                return 2
            case 7...10:
                return 1
            case 5...6:
                return -1 //Ruiny
            case 1...4:
                return -2 //Zasadzka
            default:
                return 0
            }
        }
    }
    else
    {
        return -3 //Karta promo PS1, PA01, ...
    }
    return 0
}
```

Metody obsługujące grę

Metody obsługujące grę

```
public func seasonLimit() -> Int {
    switch currentSeason {
    case .Spring,.Summer:
        return 8
    case .Autumn:
        return 7
    case .Winter:
        return 6
    }
}
```

Metody obsługujące grę

Ekran końcowy

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

```swift
import Foundation
import SpriteKit

class ScoreScene: SKScene, UIGestureRecognizerDelegate {

    var h :CGFloat = 0
    var w :CGFloat = 0
    var seasonSpring:SKShapeNode?
    var seasonSummer:SKShapeNode?
    var seasonAutumn:SKShapeNode?
    var seasonWinter:SKShapeNode?
    var labelSpring:SKLabelNode?
    var labelSummer:SKLabelNode?
    var labelAutumn:SKLabelNode?
    var labelWinter:SKLabelNode?
    var selectedSeason:Seasons?
    var newLabel:SKLabelNode?
    var totalLabel:SKLabelNode?
    var titleLabel:SKLabelNode?
    var deductionLabel:SKLabelNode?
    var qrCode:SKShapeNode?

    var goalDeduction = 0
    var gameState:GameState?
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

```
override func didMove(to view: SKView) {
    h = self.size.height/5
    w = h/88*63
    if w*4.5>self.size.width {
        w=self.size.width/4.5
        h=w/63*88
    }
    let s = 0.23*h
    var currentSeason = 1.5*(s+h)
    let seasonMargin = -self.size.width/2+10+w

    self.seasonSpring = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h),
        cornerRadius: w * 0.05)
    if let seasonSpring = self.seasonSpring {
        seasonSpring.position = CGPoint(x: seasonMargin, y: currentSeason)
        seasonSpring.fillColor = SKColor.white
        seasonSpring.fillTexture = SKTexture(imageNamed: "18")
        seasonSpring.name = "Spring"
        self.addChild(seasonSpring)
    }

    self.labelSpring = SKLabelNode.init(fontNamed: "Chalkduster")
    if let labelSpring = self.labelSpring {
        labelSpring.fontSize = 0.05*self.size.height
        labelSpring.fontColor = SKColor.white
        labelSpring.position=CGPoint(x: seasonMargin+w/0.9, y: currentSeason)
        labelSpring.text = "0"
        labelSpring.name = "SpringLabel"
        self.addChild(labelSpring)
        currentSeason -= (h+s)
    }
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

- D
S

```
self.seasonSummer = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h),
    cornerRadius: w * 0.05)
if let seasonSummer = self.seasonSummer {
    seasonSummer.position = CGPoint(x: seasonMargin, y: currentSeason)
    seasonSummer.fillColor = SKColor.white
    seasonSummer.fillTexture = SKTexture(imageNamed: "19")
    seasonSummer.name = "Summer"
    self.addChild(seasonSummer)
}

self.labelSummer = SKLabelNode.init(fontNamed: "Chalkduster")
if let labelSummer = self.labelSummer {
    labelSummer.fontSize = 0.05*self.size.height
    labelSummer.fontColor = SKColor.white
    labelSummer.position=CGPoint(x: seasonMargin+w/0.9, y: currentSeason)
    labelSummer.text = "0"
    labelSummer.name = "SummerLabel"
    self.addChild(labelSummer)
    currentSeason -= (h+s)
}
```

```
self.seasonAutumn = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h),
    cornerRadius: w * 0.05)
if let seasonAutumn = self.seasonAutumn {
    seasonAutumn.position = CGPoint(x: seasonMargin, y: currentSeason)
    seasonAutumn.fillColor = SKColor.white
    seasonAutumn.fillTexture = SKTexture(imageNamed: "20")
    seasonAutumn.name = "Autumn"
    self.addChild(seasonAutumn)
}

self.labelAutumn = SKLabelNode.init(fontNamed: "Chalkduster")
if let labelAutumn = self.labelAutumn {
    labelAutumn.fontSize = 0.05*self.size.height
    labelAutumn.fontColor = SKColor.white
    labelAutumn.position=CGPoint(x: seasonMargin+w/0.9, y: currentSeason)
    labelAutumn.text = "0"
    labelAutumn.name = "AutumnLabel"
    self.addChild(labelAutumn)
    currentSeason -= (h+s)
}
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

```
self.seasonWinter = SKShapeNode.init(rectOf: CGSize.init(width: w, height: h),
    cornerRadius: w * 0.05)
if let seasonWinter = self.seasonWinter {
    seasonWinter.position = CGPoint(x: seasonMargin, y: currentSeason)
    seasonWinter.fillColor = SKColor.white
    seasonWinter.fillTexture = SKTexture(imageNamed: "21")
    seasonWinter.name = "Winter"
    self.addChild(seasonWinter)
}

self.labelWinter = SKLabelNode.init(fontNamed: "Chalkduster")
if let labelWinter = self.labelWinter {
    labelWinter.fontSize = 0.05*self.size.height
    labelWinter.fontColor = SKColor.white
    labelWinter.position=CGPoint(x: seasonMargin+w/0.9, y: currentSeason)
    labelWinter.text = "0"
    labelWinter.name = "WinterLabel"
    self.addChild(labelWinter)
}
```

```
self.newLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let newLabel = self.newLabel {
    newLabel.fontSize = 0.03*self.size.height
    newLabel.fontColor = SKColor.white
    newLabel.position=CGPoint(x: -seasonMargin, y: -2*h)
    newLabel.text = "Nowa gra"
    newLabel.name = "NewGame"
    self.addChild(newLabel)
}

self.deductionLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let deductionLabel = self.deductionLabel {
    deductionLabel.fontSize = 0.02*self.size.height
    deductionLabel.fontColor = SKColor.white
    deductionLabel.position=CGPoint(x: -seasonMargin, y: 1.6*h)
    deductionLabel.text = "Edykty: -\(goalDeduction)"
    deductionLabel.name = "Deduction"
    self.addChild(deductionLabel)
}
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

```
self.totalLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let totalLabel = self.totalLabel {
    totalLabel.fontSize = 0.03*self.size.height
    totalLabel.fontColor = SKColor.white
    totalLabel.position=CGPoint(x: -seasonMargin, y: 2*h)
    totalLabel.text = "Wynik: 0"
    totalLabel.name = "TotalScore"
    self.addChild(totalLabel)
}

self.titleLabel = SKLabelNode.init(fontNamed: "Chalkduster")
if let titleLabel = self.titleLabel {
    titleLabel.fontSize = 0.02*self.size.height
    titleLabel.fontColor = SKColor.white
    titleLabel.position=CGPoint(x: -seasonMargin, y: 1.8*h)
    titleLabel.text = ""
    titleLabel.name = "TitleScore"
    self.addChild(titleLabel)
}

createCalculatorKeyboard()
calculateScore()
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

- Do
SK

po

```
func createCalculatorKeyboard() {

    let startX:CGFloat = 0.4*w
    let startY:CGFloat = 0.8*w
    var currentX:CGFloat = startX
    var currentY:CGFloat = startY
    createButton(label: "1", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "2", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "3", x: currentX, y: currentY)
    currentX = startX
    currentY -= 0.4*w
    createButton(label: "4", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "5", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "6", x: currentX, y: currentY)
    currentX = startX
    currentY -= 0.4*w
    createButton(label: "7", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "8", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "9", x: currentX, y: currentY)
    currentX = startX
    currentY -= 0.4*w
    createButton(label: "←", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "0", x: currentX, y: currentY)
    currentX += 0.4*w
    createButton(label: "C", x: currentX, y: currentY)
}
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

Ekran końcowy

```
func createButton(label text:String,x X:CGFloat,y Y:CGFloat) {
    let node=SKShapeNode.init(rectOf: CGSize.init(width: 0.4*w, height: 0.4*w))
    node.position = CGPoint(x: X, y: Y)
    node.strokeColor=SKColor.white
    node.name = "B\(text)"

    self.addChild(node)
    let textLabel=SKLabelNode.init(fontNamed: "Thonburi-Bold")
    textLabel.fontSize = 0.03*self.size.height
    textLabel.fontColor = SKColor.white
    textLabel.position=CGPoint(x: 0, y: 0)
    textLabel.text = text
    textLabel.name = "B\(text)"
    node.addChild(textLabel)

    if text == "C" {
        let pressed:UILongPressGestureRecognizer = UILongPressGestureRecognizer(target:
            self, action: #selector(longPress(sender:)))
        pressed.delegate = self
        pressed.minimumPressDuration = 2
        self.view!.addGestureRecognizer(pressed)
    }
}
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SK

```
func calculateScore() {
    var score = -goalDeduction
    score += Int(labelSpring!.text!)!
    score += Int(labelSummer!.text!)!
    score += Int(labelAutumn!.text!)!
    score += Int(labelWinter!.text!)!
    totalLabel!.text = "Wynik: \(score)"
    titleLabel!.text = Titles.getTitle(score: score)
}
```

Ekran końcowy

- Dodajemy kolejną klasę dziedziczącą po SKScene

Obsługa dotyku

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

    guard let touch = touches.first else {
        return
    }

    let location = touch.location(in: self)
    let frontTouchedNode = atPoint(location)
    if let name=frontTouchedNode.name {
        switch name {
        case let str where str.contains("B"):
            buttonPress(button: frontTouchedNode)
        case let str where str.contains("Spring"):
            selectedSeason = .Spring
            labelSpring!.fontColor = SKColor.green
            labelSummer!.fontColor = SKColor.white
            labelAutumn!.fontColor = SKColor.white
            labelWinter!.fontColor = SKColor.white
        case let str where str.contains("Summer"):
            selectedSeason = .Summer
            labelSpring!.fontColor = SKColor.white
            labelSummer!.fontColor = SKColor.green
            labelAutumn!.fontColor = SKColor.white
            labelWinter!.fontColor = SKColor.white
        case let str where str.contains("Autumn"):
            selectedSeason = .Autumn
            labelSpring!.fontColor = SKColor.white
            labelSummer!.fontColor = SKColor.white
            labelAutumn!.fontColor = SKColor.green
            labelWinter!.fontColor = SKColor.white
        case let str where str.contains("Winter"):
            selectedSeason = .Winter
            labelSpring!.fontColor = SKColor.white
            labelSummer!.fontColor = SKColor.white
            labelAutumn!.fontColor = SKColor.white
            labelWinter!.fontColor = SKColor.green
        case "NewGame":
            newGame()
        default:
            return
        }
    }
}
```

Obsługa dotyku

```swift
func buttonPress(button node:SKNode) {
    if selectedSeason != nil {
        switch node.name {
        case "B1":
            addToSelectedSeason(value: "1")
        case "B2":
            addToSelectedSeason(value: "2")
        case "B3":
            addToSelectedSeason(value: "3")
        case "B4":
            addToSelectedSeason(value: "4")
        case "B5":
            addToSelectedSeason(value: "5")
        case "B6":
            addToSelectedSeason(value: "6")
        case "B7":
            addToSelectedSeason(value: "7")
        case "B8":
            addToSelectedSeason(value: "8")
        case "B9":
            addToSelectedSeason(value: "9")
        case "B0":
            addToSelectedSeason(value: "0")
        case "BC":
            resetSelectedSeason()
        case "B←":
            backspaceSelectedSeason()
        default:
            return
        }
        let actionScaleDown = SKAction.scale(by: 0.7, duration: 0.3)
        let actionScaleUp = SKAction.scale(by: 1.43, duration: 0.3)
        if node.parent!.name == node.name {
            node.parent!.run(SKAction.sequence(
                [actionScaleDown,actionScaleUp]))
        } else {
            node.run(SKAction.sequence(
                [actionScaleDown,actionScaleUp]))
        }
        calculateScore()
    }
}
```

Obsługa dotyku

```swift
func addToSelectedSeason(value val:String) {
    if let selectedSeason=self.selectedSeason {
        switch selectedSeason {
        case .Spring:
            let text=labelSpring!.text!
            if text == "0" {
                labelSpring!.text = val
            } else {
                if text.count<3 {
                    labelSpring!.text! += val
                }
            }
        case .Summer:
            let text=labelSummer!.text!
            if text == "0" {
                labelSummer!.text = val
            } else {
                if text.count<3 {
                    labelSummer!.text! += val
                }
            }
        case .Autumn:
            let text=labelAutumn!.text!
            if text == "0" {
                labelAutumn!.text = val
            } else {
                if text.count<3 {
                    labelAutumn!.text! += val
                }
            }
        case .Winter:
            let text=labelWinter!.text!
            if text == "0" {
                labelWinter!.text = val
            } else {
                if text.count<3 {
                    labelWinter!.text! += val
                }
            }
        }
    }
}
```

Obsługa dotyku

```swift
func resetSelectedSeason() {
    switch selectedSeason! {
    case .Spring:
        labelSpring!.text="0"
    case .Summer:
        labelSummer!.text="0"
    case .Autumn:
        labelAutumn!.text="0"
    case .Winter:
        labelWinter!.text="0"
    }
}

func backspaceSelectedSeason() {
    var label=labelSpring!
    switch selectedSeason! {

    case .Spring:
        label=labelSpring!
    case .Summer:
        label=labelSummer!
    case .Autumn:
        label=labelAutumn!
    case .Winter:
        label=labelWinter!
    }
    let text = label.text!
    if text.count>1 {
        let newText = text.prefix(text.count-1)
        label.text = String(newText)
    }
    else {
        label.text = "0"
    }

}
```

Obsługa dotyku

Obsługa dotyku

```
@objc func longPress(sender: UILongPressGestureRecognizer) {

    if sender.state == .ended {
        labelSpring!.text = "0"
        labelSummer!.text = "0"
        labelAutumn!.text = "0"
        labelWinter!.text = "0"
        selectedSeason = nil
        labelSpring!.fontColor = SKColor.white
        labelSummer!.fontColor = SKColor.white
        labelAutumn!.fontColor = SKColor.white
        labelWinter!.fontColor = SKColor.white
        calculateScore()
    }

}
```

Obsługa dotyku

Obsługa dotyku

```
func newGame() {
    let nextScene = GameScene(size: self.scene!.size)
    nextScene.scaleMode = self.scaleMode
    nextScene.anchorPoint = self.anchorPoint
    nextScene.backgroundColor = UIColor.black
    self.view?.presentScene(nextScene, transition: SKTransition.fade(with:
        UIColor.black, duration: 1.5))
}
```

Obsługa dotyku

Tytuły

Tytuły

- W grze solo, w zależności od liczby zdobytych punktów, pomniejszonej o wartość kart punktacji zdobywa się tytuł.

Tytuły

- W grze solo, w zależności od liczby zdobytych punktów, pomniejszonej o wartość kart punktacji zdobywa się tytuł.

ZMIANY W PUNKTACJI

Ustal swój wynik, zrób to dokładnie tak samo jak w grze podstawowej. Następnie zsumuj wartości znajdujące się w prawym dolnym rogu 4 kart Punktacji używanych w tej rozgrywce (wartości w gwiazdkach). Odejmij zsumowaną wartość z gwiazdek z kart Punktacji od osiągniętego przez Ciebie wyniku, aby sprawdzić jak ci poszło.

Królowa Gimnax przyzna Tobie tytuł na podstawie najwyższej oceny, którą osiągnąłeś lub przekroczyłeś zgodnie z tabelą obok.

30+	Legendarny Kartograf
20	Mistrz Kartografii
10	Wędrowny Topograf
0	Aplikant Geodeta
-5	Amator Hobbysta
-10	Nieudolny Asystent
-20	Gryzmolący Niezdara
-30	Poplamiony Ochlapus

- V
z
o ytuł.

```swift
import Foundation

class Titles {

    static func getTitle(score myScore:Int) -> String {
        switch myScore {
        case -30 ..< -20:
            return "Poplamiony Ochlapus"
        case -20 ..< -10:
            return "Gryzmolący Niezdara"
        case -10 ..< -5:
            return "Nieudolny Asystent"
        case -5 ..< 0:
            return "Amator Hobbysta"
        case 0 ..< 10:
            return "Aplikant Geodeta"
        case 10 ..< 20:
            return "Wędrowny topograf"
        case 20 ..< 30:
            return "Mistrz Kartografii"
        case 30 ... Int.max:
            return "Legendarny kartograf"
        default:
            return "Masakra"
        }
    }
}
```

Wartości kart punktacji

Wartości kart punktacji

- Aby widzieć, ile karty punktacji są warte, do projektu dołączony jest słownik PLIST

Wartości kart punktacji

- Aby widzieć, ile karty punktacji są warte, do projektu dołączony jest słownik PLIST

Key	Type	Value
▼ Root	Dictionary	(16 items)
26	Number	25
27	Number	22
28	Number	17
29	Number	18
30	Number	24
31	Number	22
32	Number	20
33	Number	27
34	Number	16
35	Number	16
36	Number	20
37	Number	21
38	Number	24
39	Number	24
40	Number	24
41	Number	20

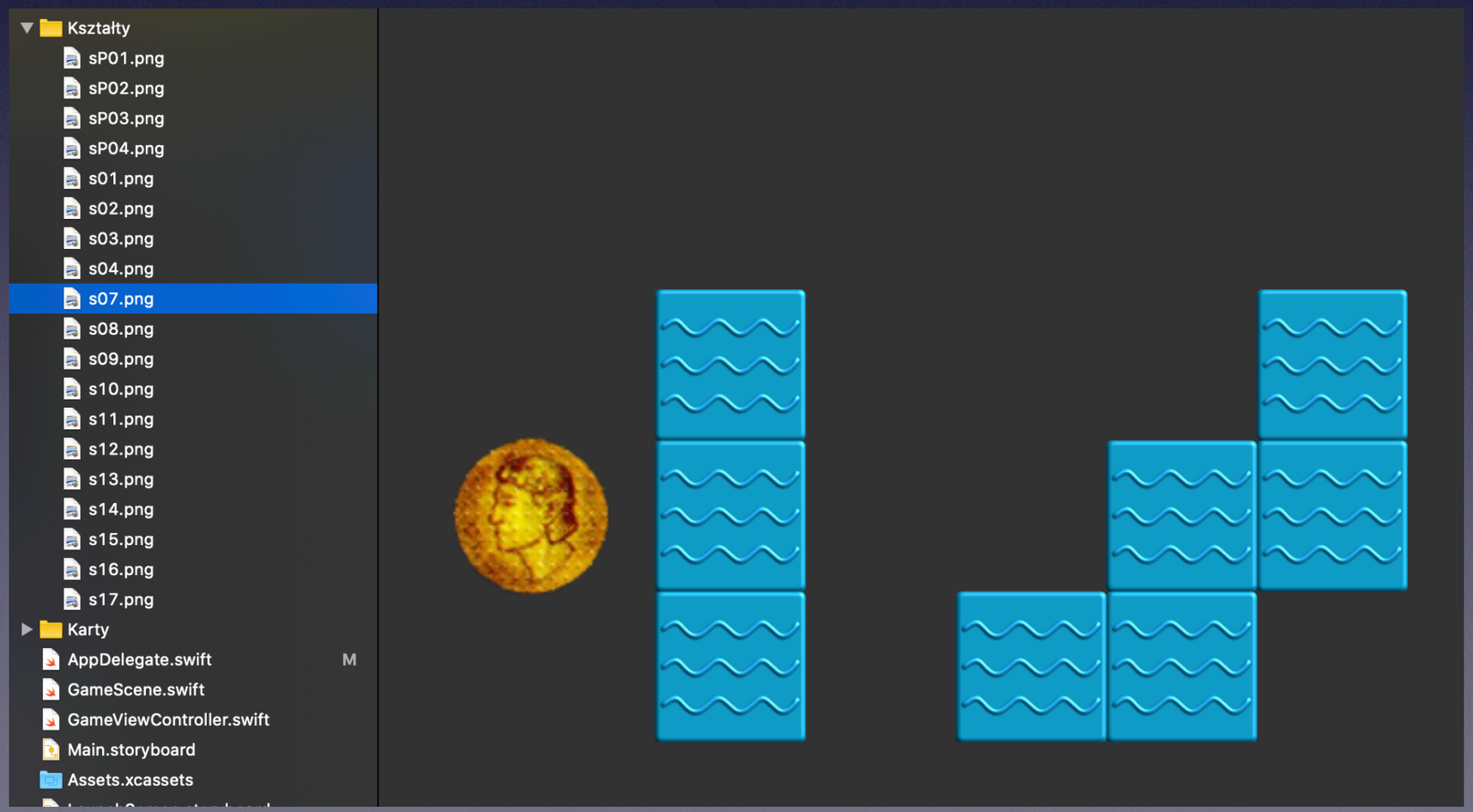
Kształty

Kształty

- Do projektu zostały też dodane kształty z odpowiednim symbolem terenu

Kształty

- Do projektu zostały też dodane kształty z odpowiednim symbolem terenu



Zmiany w klasie GameScene

Zmiany w klasie GameScene

- Klasa nie przechowuje już informacji o stanie gry tylko obiekt GameState

Zmiany w klasie GameScene

- Klasa nie przechowuje już informacji o stanie gry tylko obiekt GameState
- Jeżeli gra się kończy następuje przejście do ekranu końcowego

Zmiany w klasie GameScene

- I
- J do
(

```
if gameFinished {
    let nextScene = ScoreScene(size: self.scene!.size)
    nextScene.scaleMode = self.scaleMode
    nextScene.anchorPoint = self.anchorPoint
    nextScene.goalDeduction = totalValue
    nextScene.backgroundColor = UIColor.black
    self.view?.presentScene(nextScene, transition:
        SKTransition.fade(with: UIColor.black, duration: 1.5))
}
else
{
    nextSeason()
}
```

Zmiany w klasie GameScene

- Klasa nie przechowuje już informacji o stanie gry tylko obiekt GameState
- Jeżeli gra się kończy następuje przejście do ekranu końcowego
- Drobne zmiany usprawniające działanie

Game of the Month

Game of the Month

- Na stronie BoardGameGeek co miesiąc jeden z użytkowników publikuje wyzwanie, kto zdobędzie najwięcej punktów w konkretnej grze



Cartographers Solo Challenge June 2020

Cartographers: A Roll Player Tale on Amazon.de 26,89 €

Mathias N @cirdan · Jun 1 (edited)

Challenge Start

Now

Challenge End

06/30/2020

Description

Here's Cartographers solo challenge #7!

The rules of the solo version can be found e. g. in the download section of Thunderworks.
Please note for German players, that there is a small but subtle difference between the original English and the German version: You're not allowed to rotate or mirror monsters! So we play the **rules of the original version**.

Please also note: Only your first blind play, in which you do not know which cards are coming up, counts for the challenge.

Please post your results in this thread below. It would be awesome if you wanted to include photos of your final board as well as comments on your strategy and/or how your game went, but please post them in spoiler boxes. If you post your score, please use following fields:

Game of the Month

- Na stronie BoardGameGeek co miesiąc jeden z użytkowników publikuje wyzwanie, kto zdobędzie najwięcej punktów w konkretnej grze
- Jak to przenieść do naszej aplikacji?

A) 地大物博 - Shoreside Expanse - Ausgedehnte Strände - Rozległe nabrzeże - Expansão Beira-Mar - Extensión Costera (27)
B) 樹木哨塔 - Treetower - Düsterwald - Leśna wieża - Torre Arbórea - Torre del Árbol (17)
C) 人丁興旺 - Wildholds - Bastionen in der Wildnis - Kolonia - Terras Selvagens - Baluarte (16)
D) 遺忘封地 - Lost Barony - Unzugängliche Baronie - Utracone włości - Baronado Perdido - Baronía Perdida (24)

Total of the numbers in the lower right corner: **84**
At the end of your game you have to substract that from your score to find your rating!

Edit 06/09/2020: Please read carefully the handling of the scorecards, especially the scoring of the Shoreside Expanse (not adjacent to a farm/water space or the edge of the map). There are unfortunately faults in some plays this month, concerning this scoring card.

Spring:

Spring - 1/8: 田野 - Farmland - Ackerland - Pole uprawne - Fazenda - Tierra de Cultivo

Farm



Spring - 3/8: 樹屋村莊 - Treetop Village - Baumwipfeldorf - Nadrzewna osada - Vila Arbórea - Pueblo Arbóreo

Forest | Village



Game of the Month

- Na stronie BoardGameGeek co miesiąc jeden z użytkowników publikuje wyzwanie, kto zdobędzie najwięcej punktów w konkretnej grze
- Jak to przenieść do naszej aplikacji?
- Odpowiedź: kod QR i Deep Linking

C n

- N ąc
  je anie,
  k
  w
- Ja
- C

Jak wygenerować QR?

Jak wygenerować QR?

- Aplikacja pod Windows,

Jak wygenerować QR?

- Aplikacja pod Windows,
- napisana w C#,

Jak wygenerować QR?

- Aplikacja pod Windows,
- napisana w C#,
- Parsuje kod strony i generuje instancję gry w postaci pokazanego wcześniej stringa

Jak wygenerować QR?

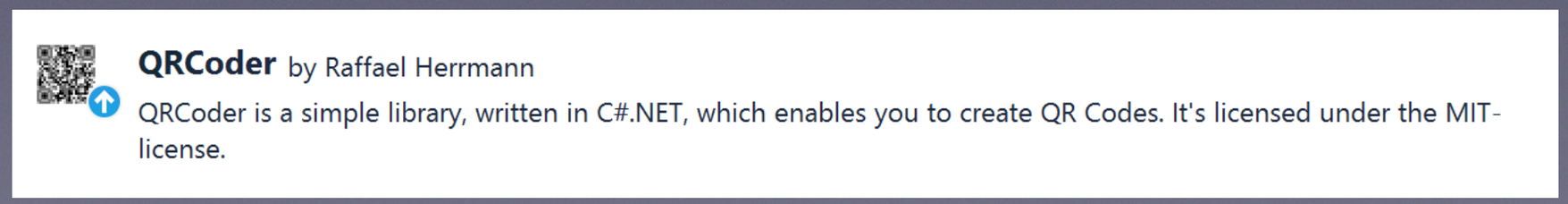
- Aplikacja pod Windows,
- napisana w C#,
- Parsuje kod strony i generuje instancję gry w postaci pokazanego wcześniej stringa
- Koduje go do kodu QR i zapisuje do pliku

Jak wygenerować QR?

- Aplikacja pod Windows,
- napisana w C#,
- Parsuje kod strony i generuje instancję gry w postaci pokazanego wcześniej stringa
- Koduje go do kodu QR i zapisuje do pliku
- Korzysta z biblioteki QRCoder

Jak wygenerować QR?

- Aplikacja pod Windows,
- napisana w C#,
- Parsuje kod strony i generuje instancję gry w postaci pokazanego wcześniej stringa
- Koduje go do kodu QR i zapisuje do pliku
- Korzysta z biblioteki QRCoder

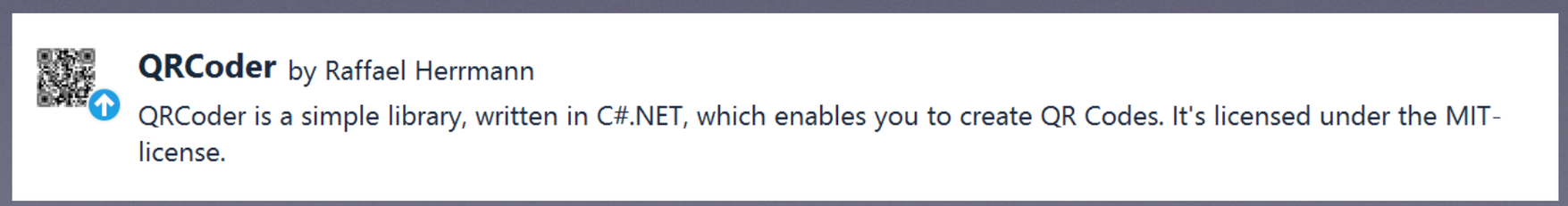


Jak wygenerować QR?

- Aplikacja pod Windows,
- napisana w C#

```
string data = "put.Cartographers://gotm/" + sb.ToString();
Console.WriteLine(data);
QRCodeGenerator qrGenerator = new QRCodeGenerator();
QRCodeData qrCodeData = qrGenerator.CreateQrCode(data, QRCodeGenerator.ECCLevel.Q);
QRCode qrCode = new QRCode(qrCodeData);
Bitmap qrCodeImage = qrCode.GetGraphic(20);
qrCodeImage.Save("c:\\Temp\\qrcode.png", ImageFormat.Png);
```

- Koduje go do kodu QR i zapisuje do pliku
- Korzysta z biblioteki QRCoder



Jak wczytać QR?

Jak wczytać QR?

- Można by dodać w aplikacji opcję wczytywania kodu, ale to wymusiłoby przejście do UIKit - kłopotliwe

Jak wczytać QR?

- Można by dodać w aplikacji opcję wczytywania kodu, ale to wymusiłoby przejście do UIKit - kłopotliwe
- Można skorzystać z DeepLinking - umieszczanie w kodach QR URL'i uruchamianych skojarzoną aplikację

Jak wczytać QR?

- Można by dodać w aplikacji opcję wczytywania kodu, ale to wymusiłoby przejście do UIKit - kłopotliwe
- Można skorzystać z DeepLinking - umieszczanie w kodach QR URL'i uruchamianych skojarzoną aplikację
- Dlatego w stringu jest put.Cartographers://

Jak wczytać QR?

Jak wczytać QR?

- Aplikacja Aparat w iOS automatycznie rozpoznaje kody QR,
- W pliku Info.plist musimy dodać skojarzenie z takim URL, żeby Aparat zasugerował uruchomienie aplikacji

Jak wczytać QR?

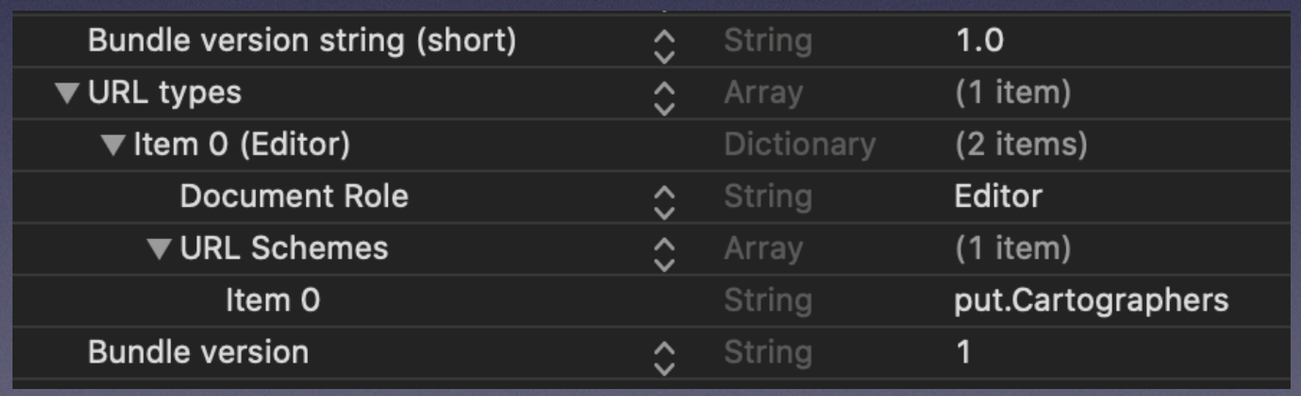
- Aplikacja Aparat w iOS automatycznie rozpoznaje kody QR,
- W pliku Info.plist musimy dodać skojarzenie z takim URL, żeby Aparat zasugerował uruchomienie aplikacji

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>put.Cartographers</string>
        </array>
    </dict>
</array>
```

Jak wczytać QR?

- Aplikacja Aparat w iOS automatycznie rozpoznaje kody QR,
- W pliku Info.plist musimy dodać skojarzenie z takim URL, żeby Aparat zasugerował uruchomienie aplikacji

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>Editor</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>put.Cartographers</string>
        </array>
    </dict>
</array>
```



Jak wczytać kod QR?

Jak wczytać kod QR?

- Dodamy sobie nowe rozszerzenie

Jak wczytać kod QR?

- Dodamy sobie nowe rozszerzenie

```swift
import Foundation

extension Notification.Name {
    static let didReadQR = Notification.Name("didReadQR")

}
```

Jak wczytać kod QR?

- Dodamy sobie nowe rozszerzenie
- W klasie AppDelegate dodajemy metodę:

Jak wczytać kod QR?

- Dodamy sobie nowe rozszerzenie
- W klasie AppDelegate dodajemy metodę:

```
internal func application(_ application: UIApplication,
                 open url: URL,
                 options: [UIApplication.OpenURLOptionsKey : Any] = [:] ) -> Bool {
    //print(url.path)
    let data=url.path[1...]
    AppDelegate.gameData = data
    //print(AppDelegate.gameData!)

    NotificationCenter.default.post(name: .didReadQR, object: data)
    return true
}
```

Jak wczytać kod QR?

- Dodamy sobie nowe rozszerzenie
- W klasie AppDelegate dodajemy metodę:

Reakcja na QR

Reakcja na QR

- W klasie GameScene zmienimy metodę inicjalizującą obiekt GameState

Reakcja na QR

- W klasie GameScene zmienimy metodę inicjalizującą obiekt GameState

```
func initializeCards() {
    if AppDelegate.gameData != nil {
        let data=AppDelegate.gameData!
        gameState = GameState(state: data)
        AppDelegate.gameData = nil
    }
    else
    {
        gameState = GameState(skills: gameState!.useSkills,promoA:
            gameState!.usePromoAmbush)
    }
    gameFinished = false
}
```

Reakcja na QR

- W klasie GameScene zmienimy metodę inicjalizującą obiekt GameState
- Oraz dodamy reakcję na otrzymanie komunikatu o kodzie QR

Reakcja na QR

- W klasie GameScene zmienimy metodę inicjalizującą obiekt GameState
- Oraz dodamy reakcję na otrzymanie komunikatu o kodzie QR

```
@objc func onDidReceiveData(_ notification:Notification) {
    newGame()
}
```

Reakcja na QR

- W klasie GameScene zmienimy metodę inicjalizującą obiekt GameState
- Oraz dodamy reakcję na otrzymanie komunikatu o kodzie QR

Metoda zapisując stan gry do kodu QR

Metoda zapisując stan gry do kodu QR

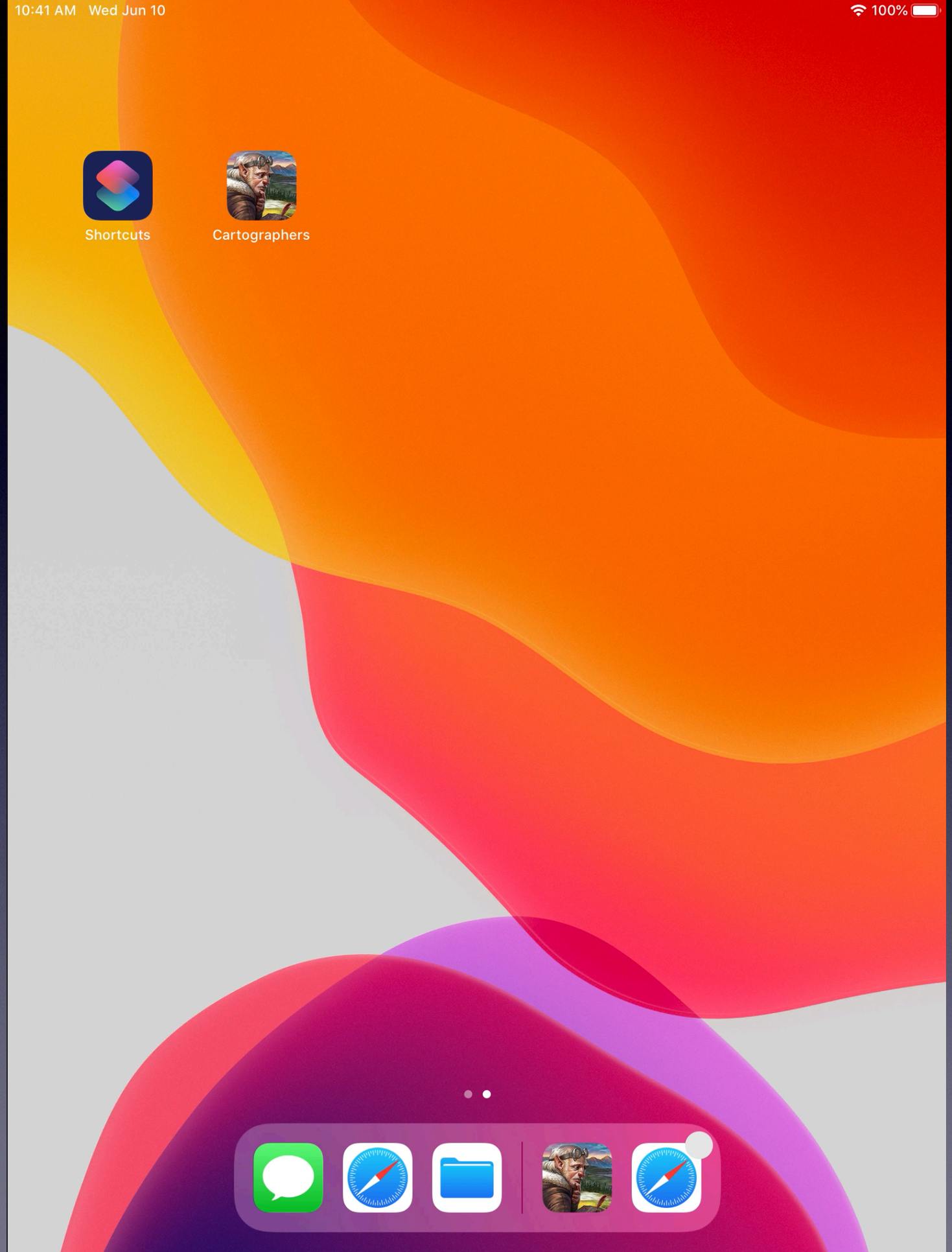
```swift
public func toQRCode() -> UIImage {
    let data=self.toString()
    let image = generateQRCode(from: data)!
    return image
}

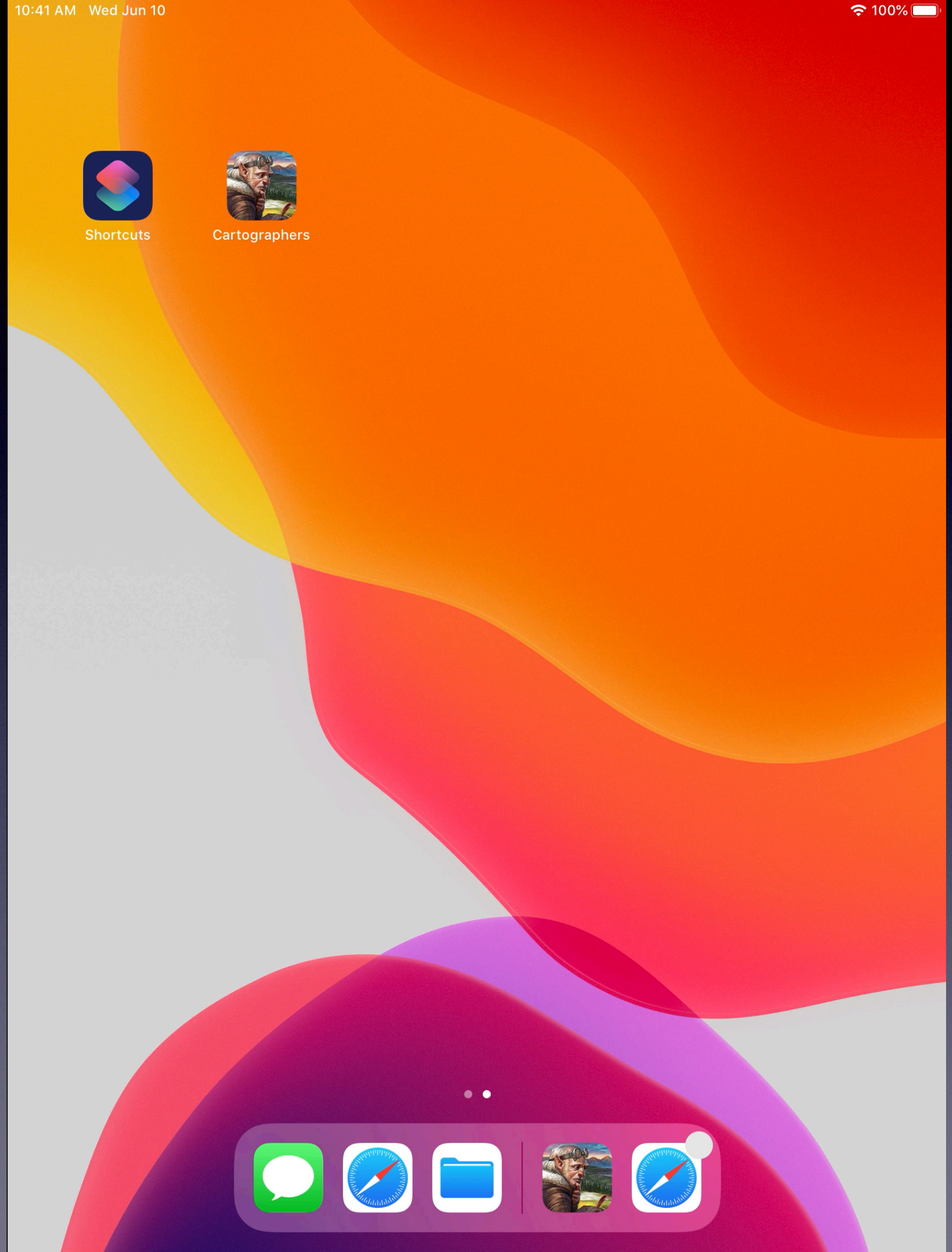
private func generateQRCode(from string: String) -> UIImage? {
    let data = string.data(using: String.Encoding.ascii)

    if let filter = CIFilter(name: "CIQRCodeGenerator") {
        filter.setValue(data, forKey: "inputMessage")
        let transform = CGAffineTransform(scaleX: 4, y: 4)

        if let output = filter.outputImage?.transformed(by: transform) {
            return UIImage(ciImage: output)
        }
    }

    return nil
}
```


10:41 AM Wed Jun 10
100%
Shortcuts
Cartographers


10:41 AM Wed Jun 10
100%
Shortcuts
Cartographers

Wersja 3

Wersja 3

- Zapamiętywanie wyników

Wersja 3

- Zapamiętywanie wyników
- Możliwość powtórnego rozegrania gry

Wersja 3

- Zapamiętywanie wyników
- Możliwość powtórnego rozegrania gry
- Synchronizacja między urządzeniami

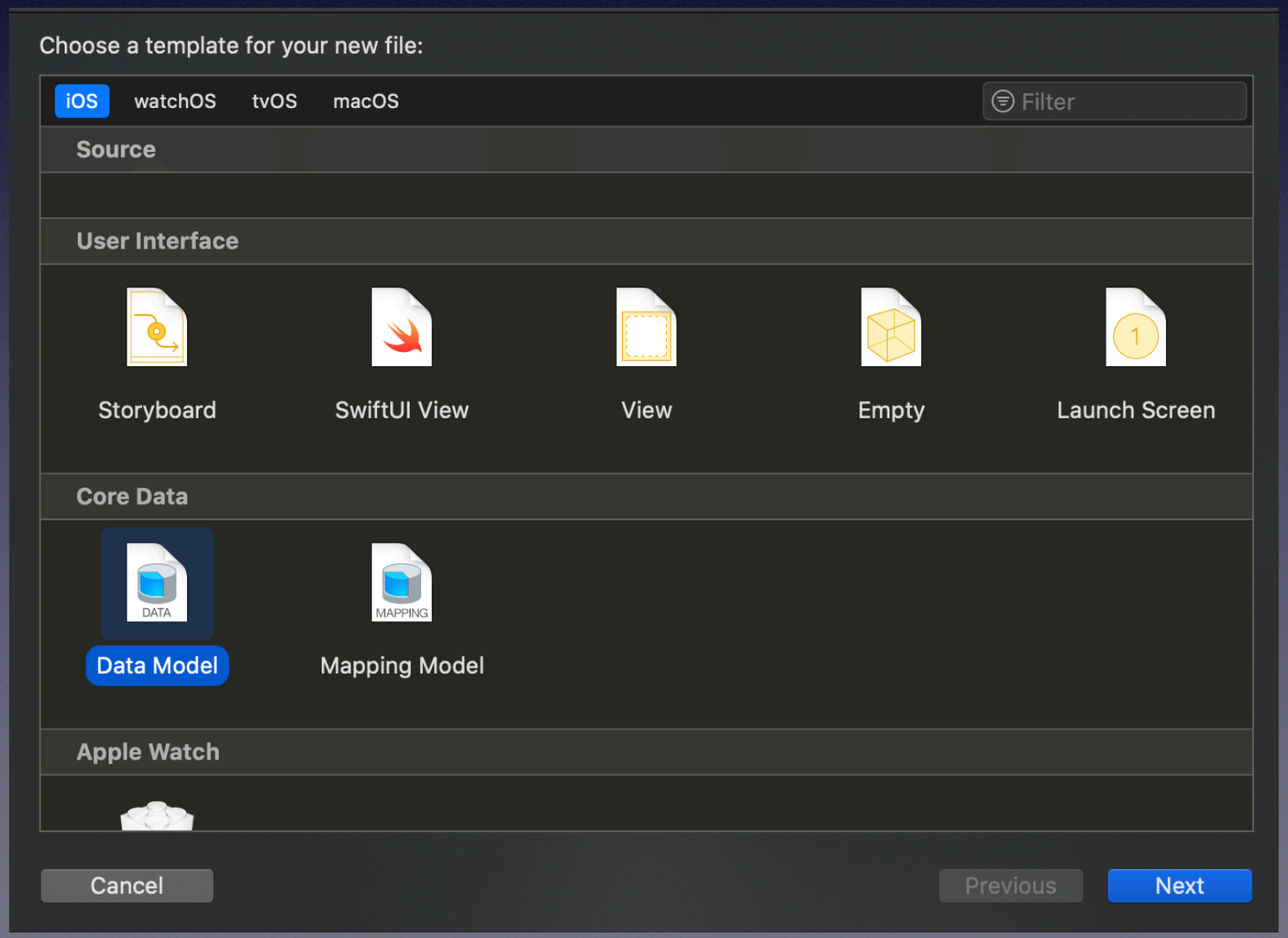
Zapisywanie wyników

Zapisywanie wyników

- Dodamy model danych zapisany w bibliotece CoreData

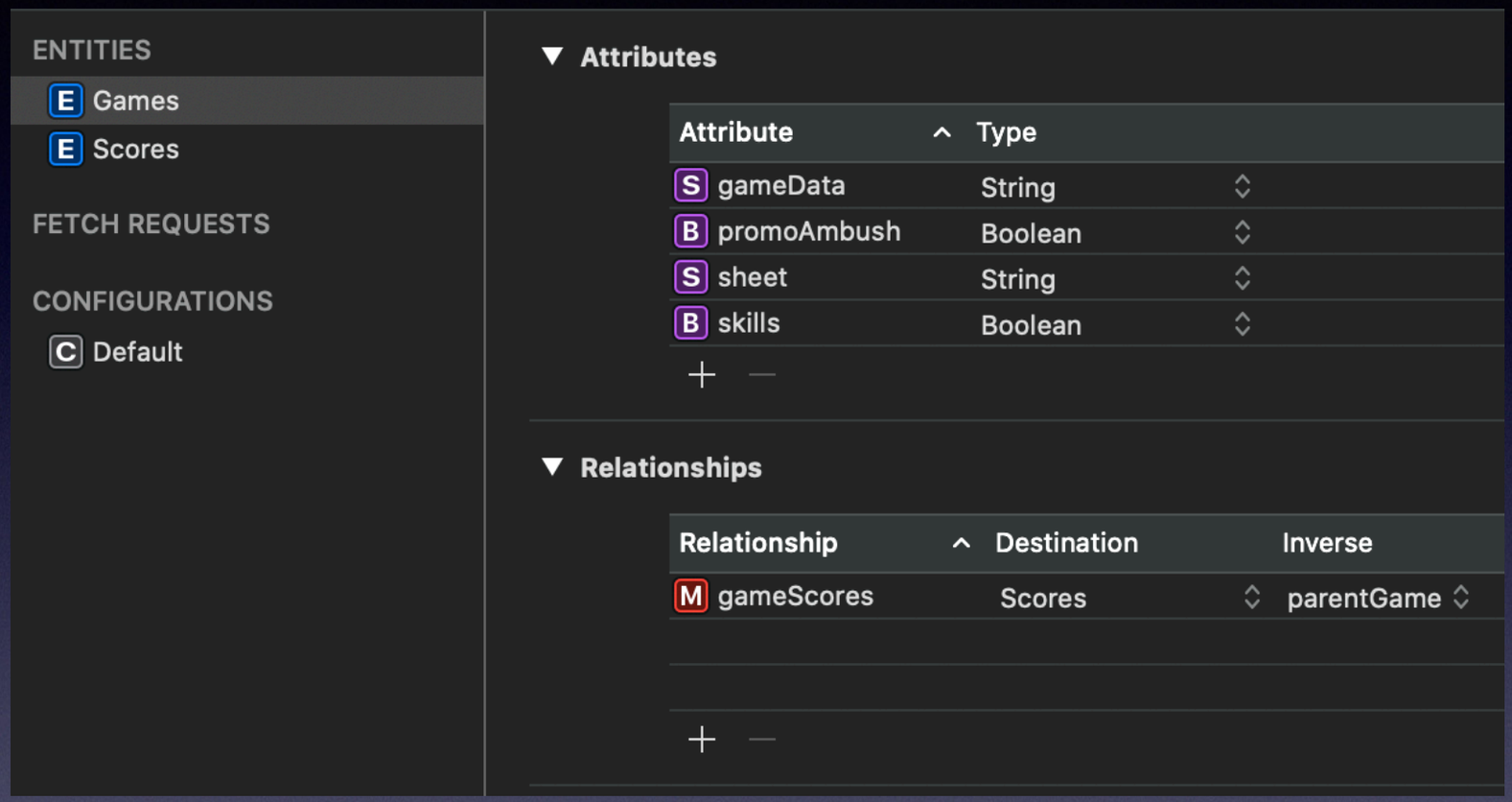
Zapisywanie wyników

- Dodamy model danych zapisany w bibliotece CoreData

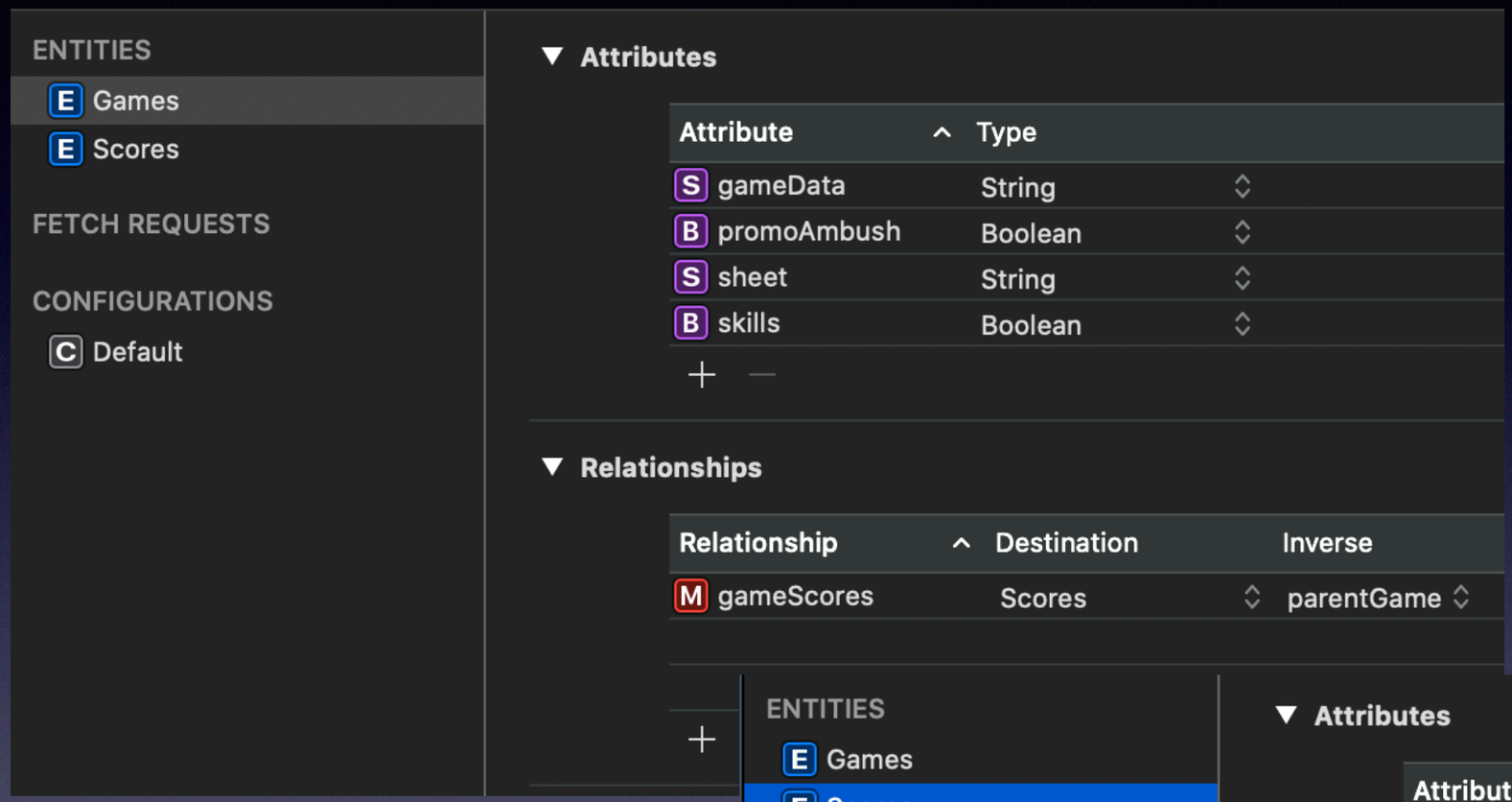


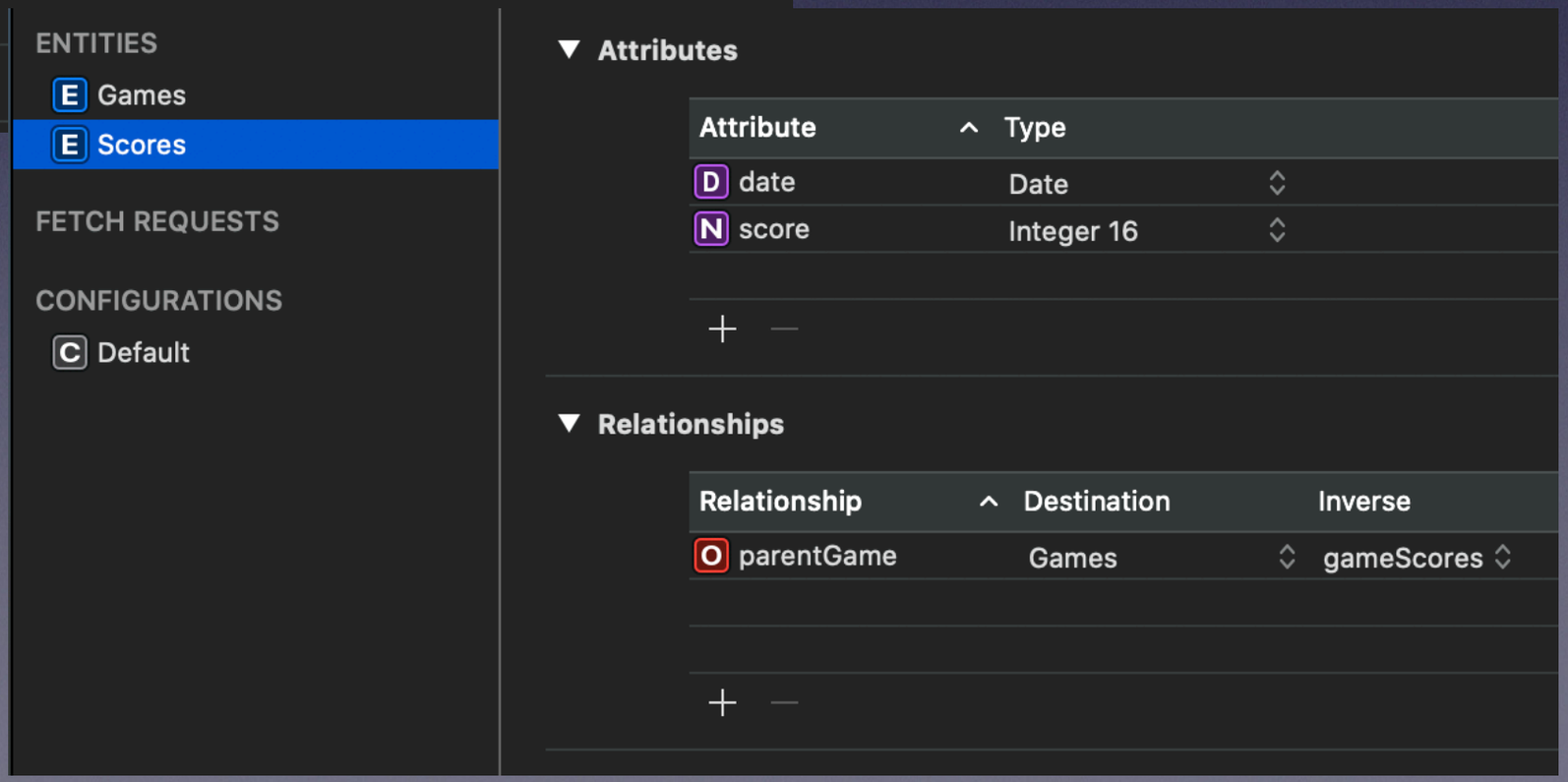
Model danych

Model danych



Model danych



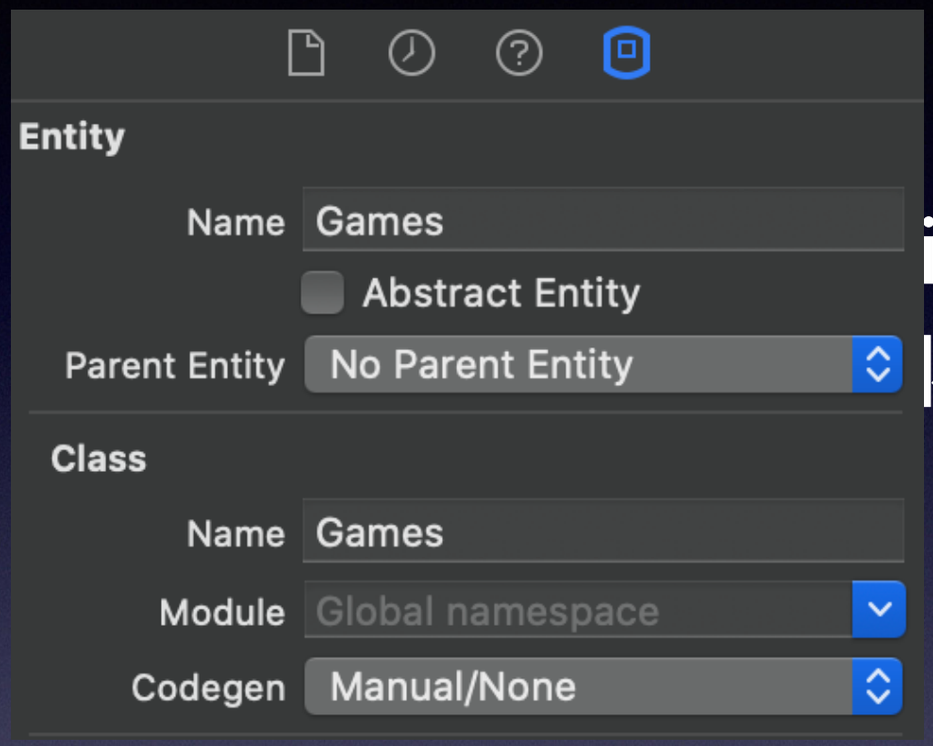


Model danych

Model danych

- Dla obu Entities ustawiamy opcję generowania kodu Codegen na Manual/None

Model danych



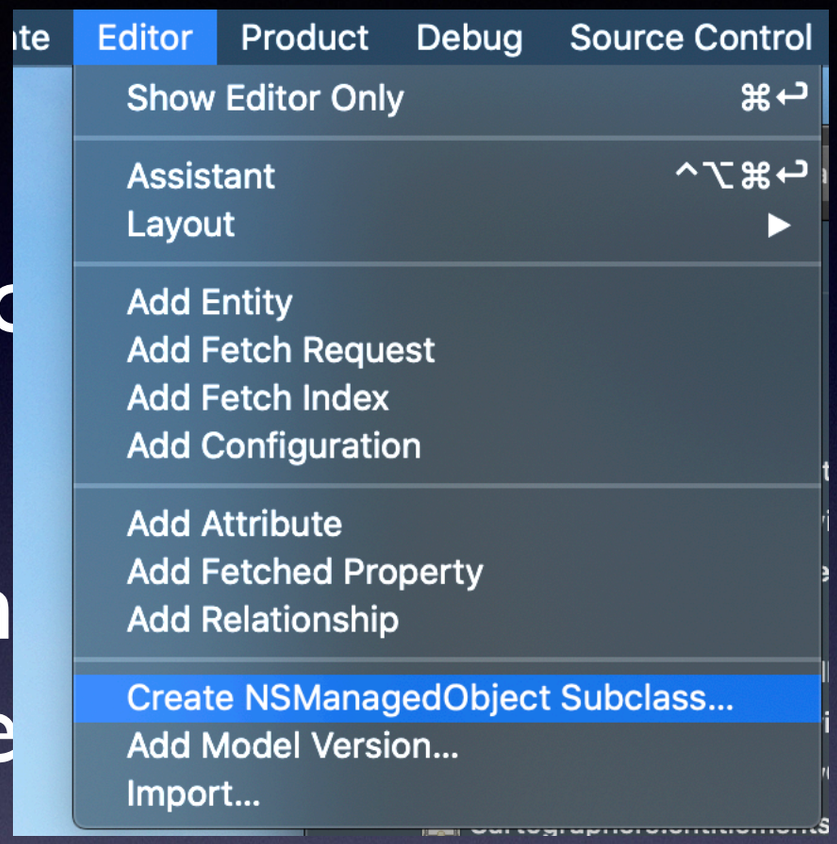
ies ustawiamy opcję
kodu Codegen na Manual/None

Model danych

- Dla obu Entities ustawiamy opcję generowania kodu Codegen na Manual/None
- Następnie wybieramy z menu Editor polecenie Create NSManagedObject Subclass

Model danych

- Dla obu Entities ustawiamy c
generowania kodu Codegen
- Następnie wybieramy z men
polecenie Create NSManage

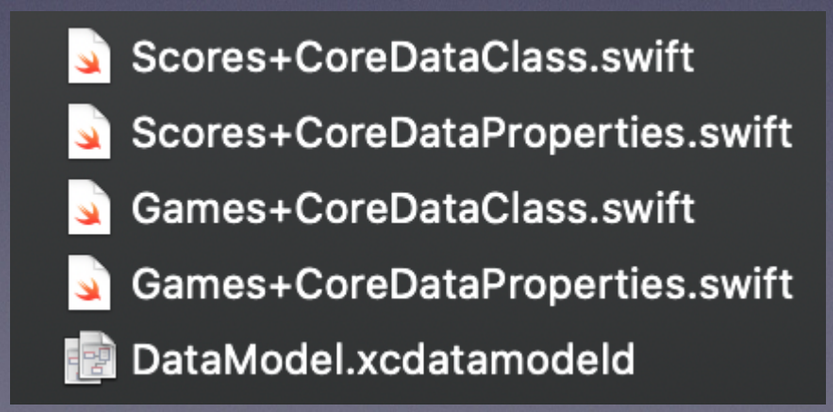


Model danych

- Dla obu Entities ustawiamy opcję generowania kodu Codegen na Manual/None
- Następnie wybieramy z menu Editor polecenie Create NSManagedObject Subclass
- W rezultacie powstają 4 klasy:

Model danych

- Dla obu Entities ustawiamy opcję generowania kodu Codegen na Manual/None
- Następnie wybieramy z menu Editor polecenie Create NSManagedObject Subclass
- W rezultacie powstają 4 klasy:



Model danych

Model danych

- W odpowiednich miejscach klas podmieniamy Set<NSManagedObject> na odpowiednio Set<Scores> i Set<Games>

Model danych

```
import Foundation
import CoreData


extension Scores {

    @nonobjc public class func fetchRequest() -> NSFetchRequest<Scores> {
        return NSFetchRequest<Scores>(entityName: "Scores")
    }

    @NSManaged public var date: Date?
    @NSManaged public var score: Int16
    @NSManaged public var parentGame: Games?

}
```

```swift
import Foundation
import CoreData


extension Games {

    @nonobjc public class func fetchRequest() -> NSFetchRequest<Games> {
        return NSFetchRequest<Games>(entityName: "Games")
    }

    @NSManaged public var gameData: String?
    @NSManaged public var sheet: String?
    @NSManaged public var skills: Bool
    @NSManaged public var promoAmbush: Bool
    @NSManaged public var gameScores: Set<Scores>?

}

// MARK: Generated accessors for gameScores
extension Games {

    @objc(addGameScoresObject:)
    @NSManaged public func addToGameScores(_ value: Scores)

    @objc(removeGameScoresObject:)
    @NSManaged public func removeFromGameScores(_ value: Scores)

    @objc(addGameScores:)
    @NSManaged public func addToGameScores(_ values: Set<Scores>)

    @objc(removeGameScores:)
    @NSManaged public func removeFromGameScores(_ values: Set<Scores>)

}
```

Model danych

- W odpowiednich miejscach klas podmieniamy Set<NSManagedObject> na odpowiednio Set<Scores> i Set<Games>
- Tworzymy nową klasę DataController

Model.datamodel

```
import Foundation
import CoreData
import MessageUI

class DataController {

    var context: NSManagedObjectContext {
        return self.persistentContainer.viewContext
    }

    let persistentContainer = NSPersistentCloudKitContainer(name: "DataModel")

    func initalizeStack() {
        self.persistentContainer.loadPersistentStores { description, error in
            if let error = error {
                print("could not load store \(error.localizedDescription)")
                return
            }
        }
    }
```

Model danych

- W odpowiednich miejscach klas podmieniamy Set<NSManagedObject> na odpowiednio Set<Scores> i Set<Games>
- Tworzymy nową klasę DataController

Model danych

Model danych

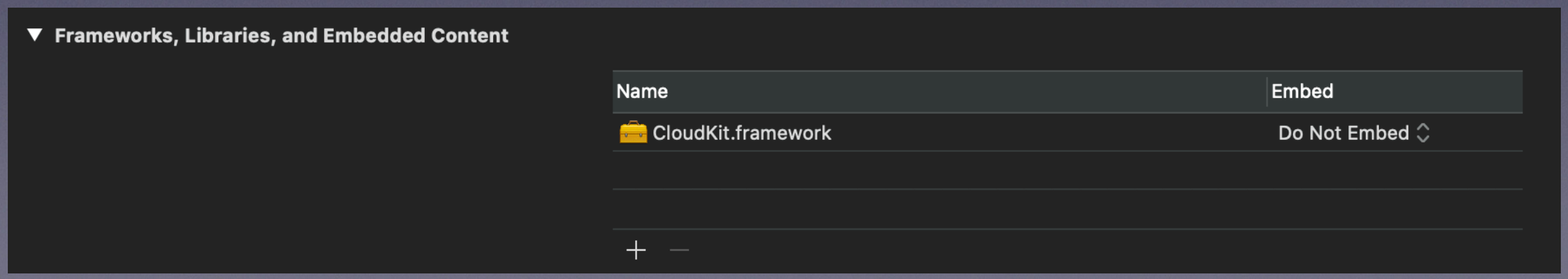
- Dane będziemy synchronizować z chmurą iCloud, dlatego jest NSPersistentCloudKitContainer, a nie NSPersistentContainer

Model danych

- Dane będziemy synchronizować z chmurą iCloud, dlatego jest NSPersistentCloudKitContainer, a nie NSPersistentContainer
- Żeby to działało musimy dodać do projektu bibliotekę CloudKit

Model danych

- Dane będziemy synchronizować z chmurą iCloud, dlatego jest NSPersistentCloudKitContainer, a nie NSPersistentContainer
- Żeby to działało musimy dodać do projektu bibliotekę CloudKit



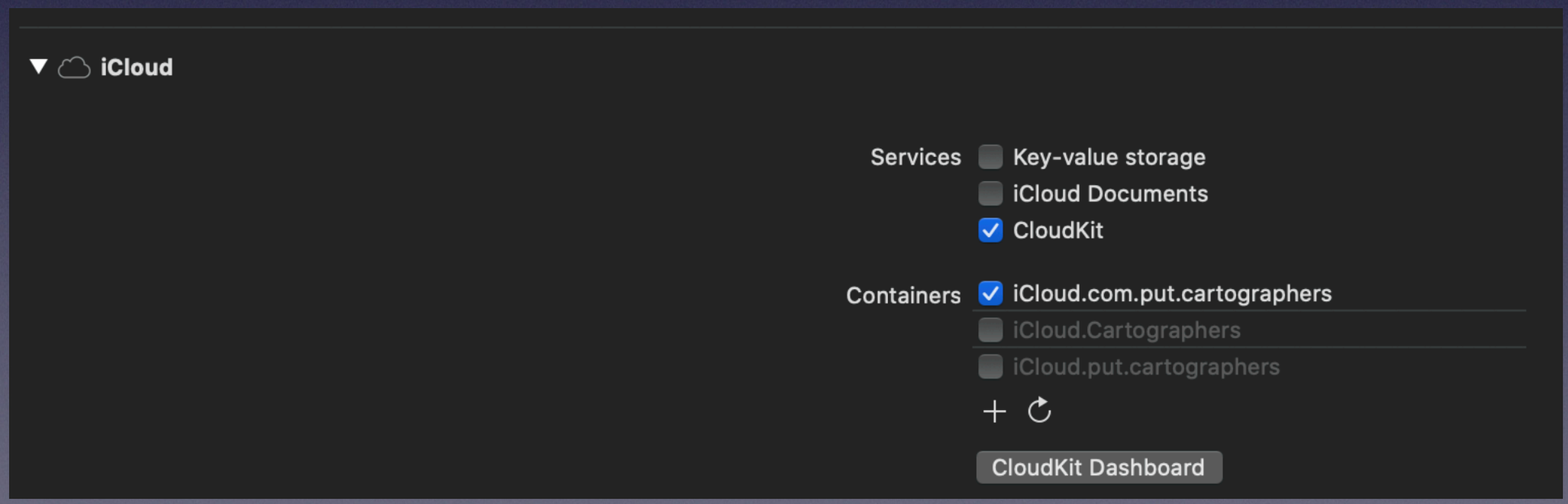
Model danych

Model danych

- A w zakładce Signing & Capabilities ustawić opcje iCloud

Model danych

- A w zakładce Signing & Capabilities ustawić opcje iCloud



Klasa DataController

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

Klasa DataController

```
func insertGame(gameData data:String, sheet:String, useSkills skills:Bool, useAmbushes ambush:Bool, withScore score: Int16) throws {
    let game = Games(context: self.context)
    game.gameData = data
    game.skills = skills
    game.promoAmbush = ambush
    game.sheet=sheet

    let scoreData = Scores(context: self.context)
    scoreData.date = Date()
    scoreData.score = score
    game.addToGameScores(scoreData)

    self.context.insert(game)
    try self.context.save()
}

func insertScore(forGame game:Games, withScore score:Int16) throws {
    let scoreData = Scores(context: self.context)
    scoreData.date = Date()
    scoreData.score = score
    game.addToGameScores(scoreData)

    try self.context.save()
}
```

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

Klasa DataController

```
func fetchAllGames() throws -> [Games] {
    let games = try self.context.fetch(Games.fetchRequest() as NSFetchRequest<Games>)
    return games
}

func fetchGamesWithParams(sheet:String,useSkills skills:Bool,useAmbushes ambush:Bool) throws -> [Games] {

    let predicateSheet = NSPredicate(format: "sheet == %@", sheet)
    let predicateSkills = NSPredicate(format: "skills == %@", NSNumber(value: skills))
    let predicateAmbushes = NSPredicate(format: "promoAmbush == %@", NSNumber(value: ambush))
    let andPredicate = NSCompoundPredicate(type: .and, subpredicates: [predicateSheet, predicateSkills,predicateAmbushes])


    let request = NSFetchRequest<Games>(entityName: "Games")
    request.predicate = andPredicate
    let games = try! self.context.fetch(request)

    return games
}
```

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

k r

```swift
func fetchGame(gameData data:String) -> Games? {
    let request = NSFetchRequest<Games>(entityName: "Games")
    request.predicate = NSPredicate(format: "gameData == %@", data)
    let games = try! self.context.fetch(request)
    if games.count == 0 {
        return nil
    }
    return games[0]
}

func fetchScores(forGame game: Games) throws -> Set<Scores>? {
    let scores = game.gameScores
    return scores
}
```

```swift
func fetchAllScores() throws -> [Scores] {
    let fetchRequest =
        NSFetchRequest<Scores>(entityName: "Scores")
    let sort = NSSortDescriptor(key: "date", ascending: false)
    fetchRequest.sortDescriptors = [sort]

    do {
        let  scores = try self.context.fetch(fetchRequest)
        return scores
    } catch let error as NSError {
        print("Could not fetch. \(error), \(error.userInfo)")
    }
    return []
}
```

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

```swift
func fetchAllScores(order: HistorySortOrder, sheetFilter: HistorySheetFilter, promoFilter: HistoryPromoFilter) throws -> [Scores] {
    let fetchRequest =
        NSFetchRequest<Scores>(entityName: "Scores")
    var sort:NSSortDescriptor?
    switch order {
    case .dateDescending:
        sort = NSSortDescriptor(key: "date", ascending: false)
    case .dateAscending:
        sort = NSSortDescriptor(key: "date", ascending: true)
    case .pointsDescending:
        sort = NSSortDescriptor(key: "score", ascending: false)
    case .pointsAscending:
        sort = NSSortDescriptor(key: "score", ascending: true)
    }
    fetchRequest.sortDescriptors = [sort!]
    switch sheetFilter {
    case .all:
        print("All")
    case .A:
        fetchRequest.predicate = NSPredicate(format: "parentGame.sheet=\"A\"")
    case .B:
        fetchRequest.predicate = NSPredicate(format: "parentGame.sheet=\"B\"")
    }
    switch promoFilter {
    case .none:
        print("No promos")
    case .skills:
        fetchRequest.predicate = NSPredicate(format: "parentGame.skills=true")
    case .ambush:
        fetchRequest.predicate = NSPredicate(format: "parentGame.promoAmbush=true")
    case .skillsambush:
        fetchRequest.predicate = NSPredicate(format: "parentGame.skills=true")
        fetchRequest.predicate = NSPredicate(format: "parentGame.promoAmbush=true")
    }

    do {
        let  scores = try self.context.fetch(fetchRequest)
        return scores
    } catch let error as NSError {
        print("Could not fetch. \(error), \(error.userInfo)")
    }

    return []
}
```

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

Klasa DataController

- Do klasy dodajemy metody które będą

```
func deleteScore(score: Scores) throws {
    self.context.delete(score)
    try self.context.save()
}
```

```
func countScores() throws -> Int {
    let count = try self.context.count(for: Scores.fetchRequest() as NSFetchRequest<Scores>)
    return count
}
```

Klasa DataController

- Do klasy dodajemy metody, które będą operowały na danych

Widok historii

Widok historii

- Zrobimy coś, co jest odradzane - pomieszamy SpriteKit i UIKit

Widok historii

- Zrobimy coś, co jest odradzane - pomieszamy SpriteKit i UIKit
- Projektujemy kolejne widoki w storyboardzie

Game View Controller

History Table View Controller

9:41

Prototype Cells

Punkty | Data ↓ | Arkusz | Promo | Losowy

100 2020-06-01 00:00 A Replay QR

Table View

Prototype Content

Prototype Cells

Arkusz A

Umiejętności

Zasadzki promo

UIImageView

100 02.06.2020 16:56

Table View

Prototype Content

Game View Controller

9:41

Tu jest scena
SpriteKit

History Table View Controller

Prototype Cells

Punkty | Data ↓ | Arkusz | Promo | Losowy

100 2020-06-01 00:00 A Replay QR

Table View
Prototype Content

Prototype Cells

Arkusz A
Umiejętności
Zasadzki promo

UIImageView

100 02.06.2020 16:56

Table View
Prototype Content

Game View Controller

9:41

Tu jest scena SpriteKit

History Table View Controller

Prototype Cells

Punkty | Data ↓ | Arkusz | Promo | Losowy

100 2020-06-01 00:00 A Replay QR

Tu jest widok tabelaryczny z 2 komórkami prototypowymi: 1 nagłówek 2 wyświetlanie wyników

Prototype Cells

Arkusz A

Umiejętności

Zasadzki promo

UIImageView

100 02.06.2020 16:56

Table View

Prototype Content

Game View Controller

9:41

Tu jest scena SpriteKit

History Table View Controller

Prototype Cells

Punkty | Data ↓ | Arkusz | Promo | Losowy

100 2020-06-01 00:00 A Replay QR

Tu jest widok tabelaryczny z 2 komórkami prototypowymi:
1 nagłówek
2 wyświetlanie wyników

Table View
Prototype Content

Prototype Cells

Arkusz A
Umiejętności
Zasadzki promo

UIImageView

100 02.06.2020 16:56

Tu jest widok tabelaryczny z 2 komórkami prototypowymi:
1 informacje o grze
2 wyświetlanie wyników takich samych gier

Table View
Prototype Content

Jak wykonać przejścia?

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół

```
import Foundation

@objc protocol UIViewControllerDelegate
{
    @objc optional  func transitionToQR()
}
```

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku

```
class GameViewController: UIViewController,UIViewControllerDelegate {

    var scene : SKScene?
```

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku
- Implementujemy metodę

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku
- Implementujemy metodę

```
func transitionToQR() {
    performSegue(withIdentifier: "presentHistory", sender: self)
}
```

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku
- Implementujemy metodę
- Do każdej ze scen dodamy pole wskazujące na nadrzędny ViewController

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku
- Implementujemy metodę
- Do każdej ze scen dodamy pole wskazujące na na

```
class GameScene: SKScene, Alertable {

    var delegateRoot : UIViewControllerDelegate?
```

Jak wykonać przejścia?

- Scena w SpriteKit nie jest w stanie wywołać przejścia
- Zdefiniujemy nowy protokół
- Dodajemy go do naszego głównego kontrolera widoku
- Implementujemy metodę
- Do każdej ze scen dodamy pole wskazujące na nadrzędny ViewController

Wywołanie przejścia

Wywołanie przejścia

- W reakcji na dotyk na odpowiednim elemencie

Wywołanie przejścia

- W reakcji na dotyk na odpowiednim elemencie
- Musimy pamiętać o przekazywaniu tego delegateRoot pomiędzy scenami

Wywołanie przejścia

```
let location = touch.location(in: self)
let frontTouchedNode = atPoint(location)
if let name=frontTouchedNode.name {
   //print(name)
    switch name {
   case "Start":
      let nextScene = GameScene(size: self.scene!.size)
      nextScene.gameState = GameState(skills: useSkills, promoA: promoAmbushes)
      nextScene.gameState?.sheet = side
      nextScene.delegateRoot = delegateRoot
      nextScene.scaleMode = self.scaleMode
      nextScene.anchorPoint = self.anchorPoint
      nextScene.backgroundColor = UIColor.black
      self.view?.presentScene(nextScene, transition: SKTransition.fade(with: UIColor.black, duration: 0.5))
   case "History":
      delegateRoot?.transitionToQR?()
   case "Side":
```

Wywołanie przejścia

```
let location = touch.location(in: self)
let frontTouchedNode = atPoint(location)
if let name=frontTouchedNode.name {
   //print(name)
    switch name {
   case "Start":
      let nextScene = GameScene(size: self.scene!.size)
      nextScene.gameState = GameState(skills: useSkills, promoA: promoAmbushes)
      nextScene.gameState?.sheet = side
      nextScene.delegateRoot = delegateRoot
      nextScene.scaleMode = self.scaleMode
      nextScene.anchorPoint = self.anchorPoint
      nextScene.backgroundColor = UIColor.black
      self.view?.presentScene(nextScene, transition: SKTransition.fade(with: UIColor.black, duration: 0.5))
   case "History":
      delegateRoot?.transitionToQR?()
   case "Side":
```

Wywołanie przejścia

```
let location = touch.location(in: self)
let frontTouchedNode = atPoint(location)
if let name=frontTouchedNode.name {
   //print(name)
    switch name {
   case "Start":
      let nextScene = GameScene(size: self.scene!.size)
      nextScene.gameState = GameState(skills: useSkills, promoA: promoAmbushes)
      nextScene.gameState?.sheet = side
      nextScene.delegateRoot = delegateRoot
      nextScene.scaleMode = self.scaleMode
      nextScene.anchorPoint = self.anchorPoint
      nextScene.backgroundColor = UIColor.black
      self.view?.presentScene(nextScene, transition: SKTransition.fade(with: UIColor.black, duration: 0.5))
   case "History":
      delegateRoot?.transitionToQR?()
   case "Side":
```

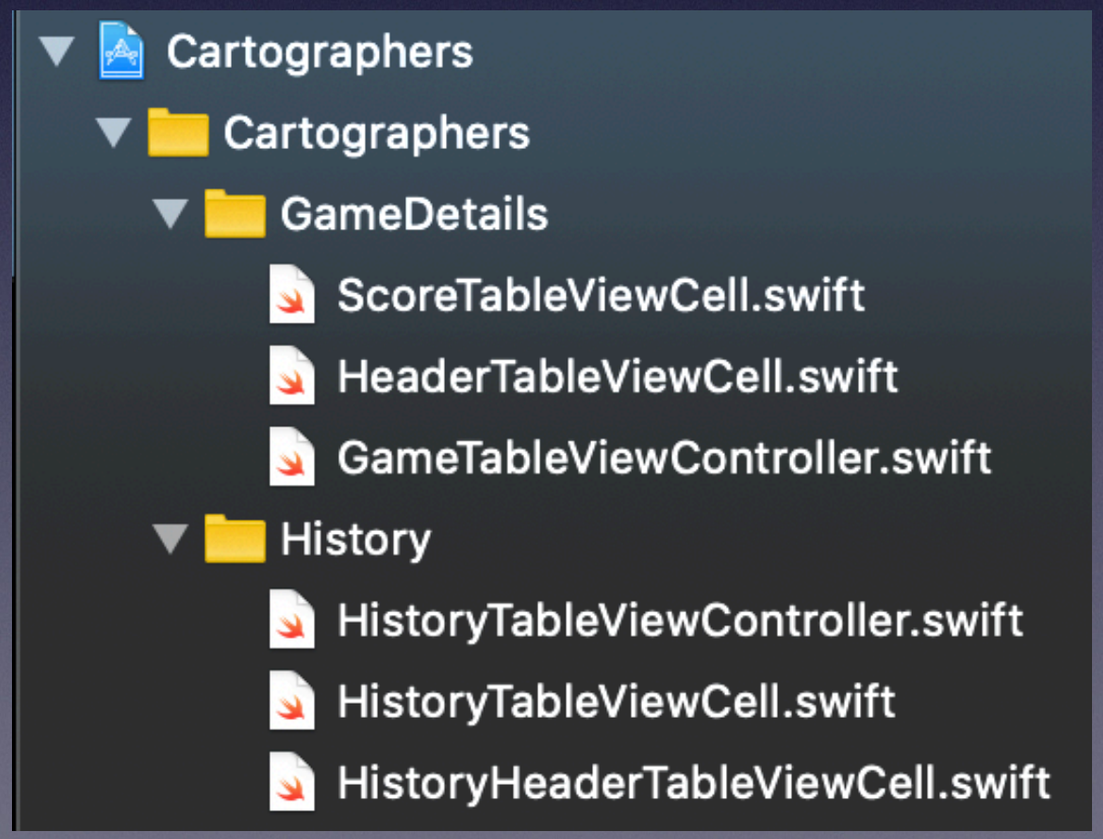
Klasy do obsługi widoków

Klasy do obsługi widoków

- Musimy zaimplementować obsługę komórek i widoków tabelarycznych

Klasy do obsługi widoków

- Musimy zaimplementować obsługę komórek i widoków tabelarycznych



KARTOGRAFOWIE



KARTOGRAFOWIE



I to by było na tyle …

速