GameplayKit

— Entity-Component —

Copenhagen Cocoa – Tobias Due Munk – April 23, 2017

Tobias Due Munk

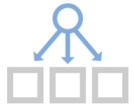
@tobiasdm

github.com/duemunk

developmunk.dk

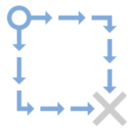
Let's get started

 Randomization.

 Entities and Components.

 State Machines.

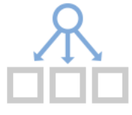
 The Minmax Strategist.

 Pathfinding.

 Agents, Goals, and Behaviors.

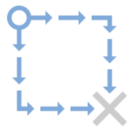
 Rule Systems.

 **Randomization.**

 ~~Entities and Components.~~

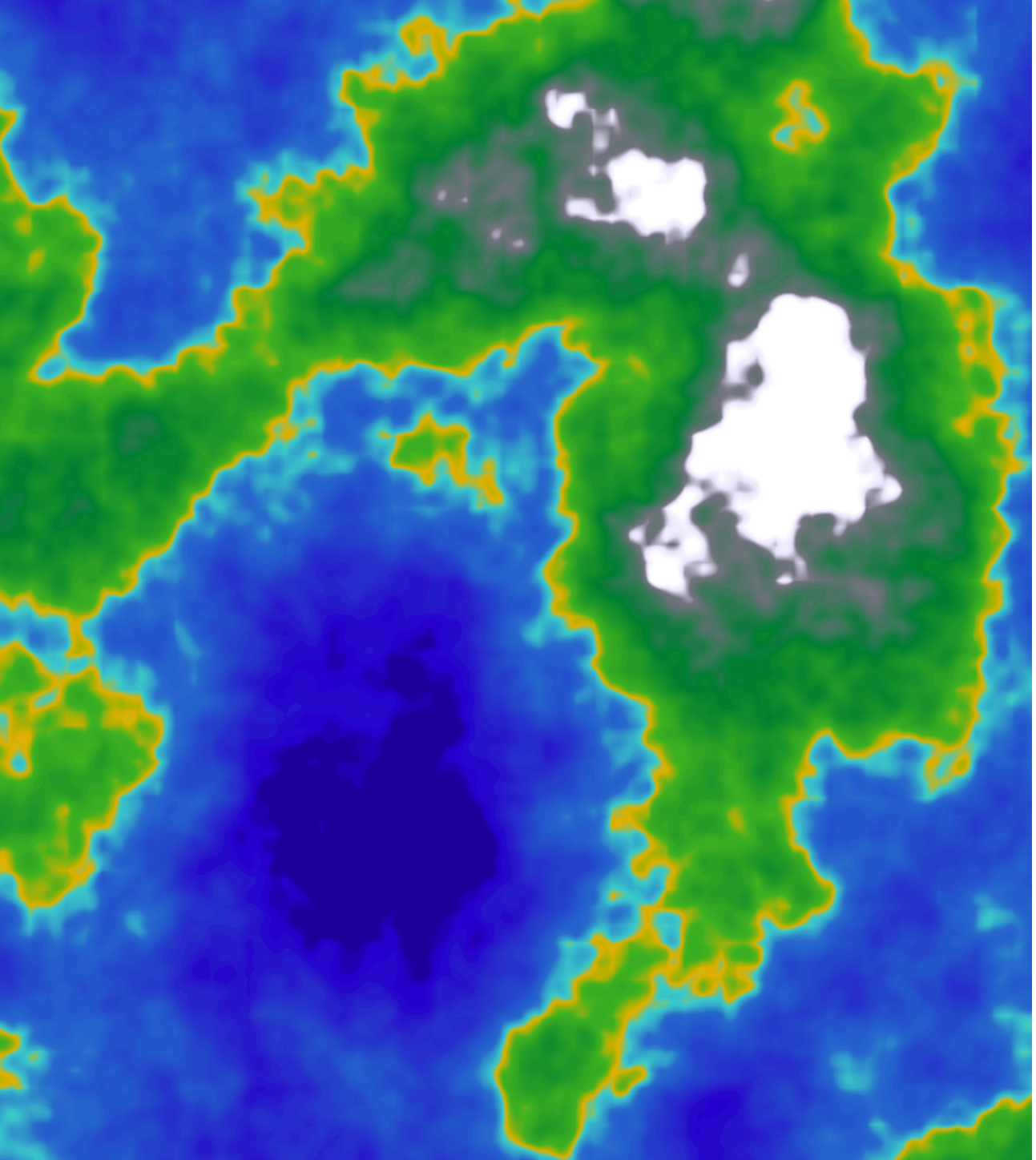
 ~~State Machines.~~

 ~~The Minmax Strategist.~~

 **Pathfinding.**

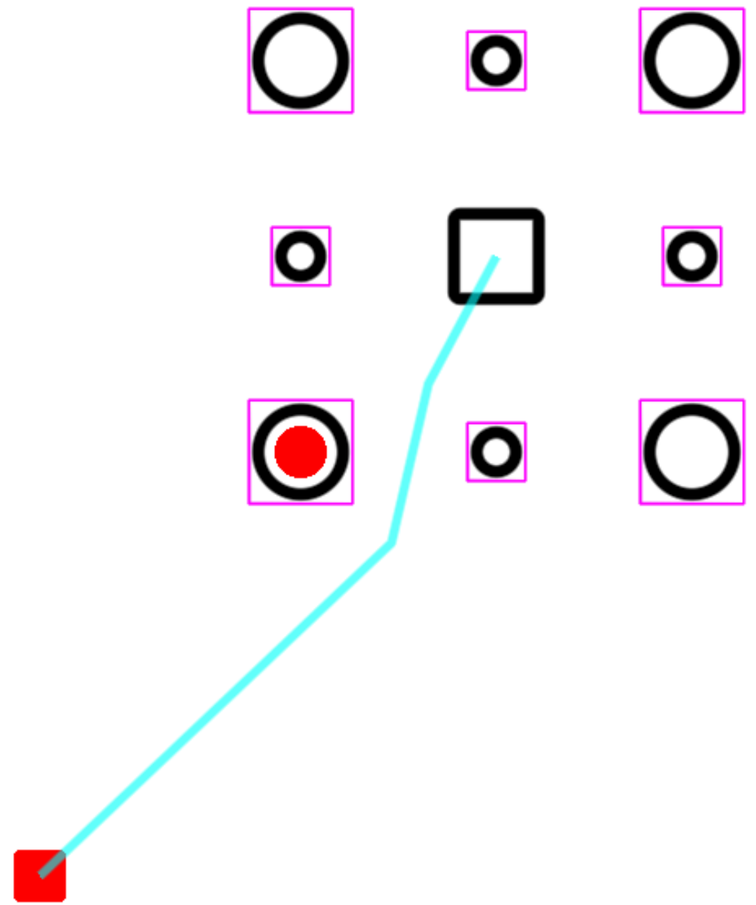
 ~~Agents, Goals, and Behaviors.~~

 ~~Rule Systems.~~

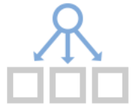


Random
ization

Path finding

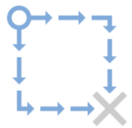


 ~~Randomization.~~

 **Entities and Components.**

 ~~State Machines.~~

 ~~The Minmax Strategist.~~

 ~~Pathfinding.~~

 ~~Agents, Goals, and Behaviors.~~

 ~~Rule Systems.~~

Entities & Components

An entity

```
import GameplayKit

public class Enemy: GKEntity { }
```

A component

```
public final class Sprite: GKComponent {
    public let node: SKNode

    public init(node: SKNode) {
        self.node = node
        super.init()
    }
    @available(*, unavailable)
    public required init?(coder: NSCoder) { fatalError("") }
}
```

Compose

```
let enemy = Enemy()
let enemyNode = SKSpriteNode(imageNamed: "enemy.png")
let sprite = Sprite(node: enemyNode)
enemy.addComponent(sprite)
//
enemy.removeComponent(sprite)
```

Use

```
let sprite = enemy.component(ofType: Sprite.self)

extension Sprite {
    public func breathe() {
        let grow = SKAction.scale(to: 1.2, duration: 0.2)
        let shrink = SKAction.scale(to: 1, duration: 0.2)
        let pulse = SKAction.sequence([grow, shrink])
        node.run(pulse)
    }
}

sprite.breathe()
```

Composition instead of Inheritance

```
public class Player: GKEntity { }

let player = Player()
let playerNode = SKSpriteNode(imageNamed: "player.png")
let sprite = Sprite(node: playerNode)
player.addComponent(sprite)

player.component(ofType: Sprite.self)?.breathe()
```

It works!

Runtime Adaption Attempt A

```
public final class Mortal: GKComponent { }
    public var life: Int
    public init(life: Int) {
        self.life = life
        super.init()
    }
    @available(*, unavailable)
    public required init?(coder: NSCoder) { fatalError("") }
    public func hit(withDamage damage: Int) {
        life = max(life - damage, 0)
    }
}
```

Runtime Adaption Attempt A

```
let enemy = Enemy()
enemy.addComponent(Sprite(node: ...))
enemy.addComponent(Mortal(life: 100))

enemy.component(ofType: Mortal.self)?.hit(damage: 25)
enemy.component(ofType: Mortal.self)?.life // 75

enemy.removeComponent(ofType: Mortal.self)
enemy.component(ofType: Mortal.self)?.hit(damage: 25)
enemy.component(ofType: Mortal.self)?.life // nil
```

Runtime Adaption Attempt B

```
public final class Health: GKComponent {
    public var life: Int
    public var mortal: Bool
    public init(life: Int, mortal: Bool) {
        self.life = life
        self.mortal = mortal
        super.init()
    }
    @available(*, unavailable)
    public required init?(coder: NSCoder) { fatalError("") }
    public func hit(withDamage damage: Int) {
        if mortal {
            life = max(life - damage, 0)
        }
    }
}
```

Runtime Adaption Attempt B

```
let enemy = Enemy()
enemy.addComponent(Sprite(node: ...))
enemy.addComponent(Health(life: 100, mortal: true))

enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75

enemy.component(ofType: Health.self)?.mortal = false
enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75
```

Is that it?

Timing

```
public final class Health: GKComponent {
    public var life: Int
    public var immortalityBoost: TimeInterval
    public init(life: Int, immortalityBoost: TimeInterval) {
        self.life = life
        self.immortalityBoost = immortalityBoost
        super.init()
    }
    ...
    public func hit(withDamage damage: Int) {
        if immortalityBoost <= 0 {
            life = max(life - damage, 0)
        }
    }
}
```

Timing

```
public final class Health: GKComponent {
    public var life: Int
    public var immortalityBoost: TimeInterval
    ...
    public func hit(withDamage damage: Int) {
        if immortalityBoost <= 0 {
            life = max(life - damage, 0)
        }
    }
    public override func update(deltaTime seconds: TimeInterval) {
        immortalityBoost = max(immortalityBoost - seconds, 0)
    }
}
```

Timing

`GKEntity` automatically forwards to `.components`

— or —

```
class Enemy: GKEntity {

    public override func update(deltaTime seconds: TimeInterval) {
        components.forEach { $0.update(deltaTime: seconds) }
    }
}
```

But *Who* Calls

 `GKEntity` 

You,

That's Who!

Update Time Calls Alternative A

```
class Game {
    var enemies: [Enemy] = []
    var scene: SKScene

    init(scene: Scene) {
        self.scene = scene
        super.init()
        scene.delegate = self
    }
}

extension Game: SKSceneDelegate {
    public func update(_ currentTime: TimeInterval, for scene: SKScene) {
        guard let previousTime = previousTime else { return }
        let deltaTime = currentTime - previousTime
        self.previousTime = currentTime
        for enemy in enemies { enemy.update(deltaTime: deltaTime) }
    }
}
```

Update Time Calls Alternative B

```
class Game {
    let mortalSystem = GKComponentSystem(componentClass: Mortal.self)
    var scene: SKScene

    init(scene: Scene) {
        self.scene = scene
        super.init()
        scene.delegate = self
    }
}

extension Game: SKSceneDelegate {
    public func update(_ currentTime: TimeInterval, for scene: SKScene) {
        guard let previousTime = previousTime else { return }
        let deltaTime = currentTime - previousTime
        self.previousTime = currentTime
        mortalSystem.update(deltaTime: deltaTime)
    }
}
```

Tips & Tricks

Tips & tricks Coming back to Entities & Components

```
extension Game: SKPhysicsContactDelegate {
    public func didBegin(_ contact: SKPhysicsContact) {
        let nodeA = contact.bodyA.node
        let nodeB = contact.bodyB.node
        //
    }
}
```

Tips & tricks Coming back to Entities & Components

```
public final class Sprite: GKComponent {
    public let node: SKNode

    public init(node: SKNode) {
        self.node = node
        super.init()
        self.node.owner = self
    }
    @available(*, unavailable)
    public required
    init?(coder: NSCoder) { fatalError("") }
}
```

Tips & tricks Coming back to Entities & Components

```
extension SKNode {
    private static var ownerKey = "dk.developmunk.app_name.SKNode_owner"

    public var owner: GKComponent? {
        get {
            return objc_getAssociatedObject(self, &SKNode.ownerKey) as? GKComponent
        }
        set {
            objc_setAssociatedObject(
                self,
                &SKNode.ownerKey,
                newValue,
                .OBJC_ASSOCIATION_RETAIN_NONATOMIC
            )
        }
    }
}
```

Tips & Tricks Verbose API

```
let enemy = Enemy()
enemy.addComponent(Sprite(node: ...))
enemy.addComponent(Health(life: 100, mortal: true))

enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75

enemy.component(ofType: Health.self)?.immortalityBoost = 10
enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75
```

Tips & Tricks Verbose API

```
extension GKEntity {
    var sprite: Sprite? { return component(ofType: Sprite.self) }
    var mortal: Health? { return component(ofType: Mortal.self) }
    var shooter: Shooter? { return component(ofType: Shooter.self) }
    var movable: Movable? { return component(ofType: Movable.self) }
}
```

Tips & Tricks Verbose API

```
let enemy = Enemy()
enemy.addComponent(Sprite(node: ...))
enemy.addComponent(Health(life: 100, mortal: true))

enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75

enemy.component(ofType: Health.self)?.immortalityBoost = 10
enemy.component(ofType: Health.self)?.hit(damage: 25)
enemy.component(ofType: Health.self)?.life // 75
```

Tips & tricks

```
let enemy = Enemy()
enemy.addComponent(Sprite(node: ...))
enemy.addComponent(Health(life: 100, mortal: true))

enemy.health?.hit(damage: 25)
enemy.health?.life // 75

enemy.health?.immortalityBoost = 10
enemy.health?.hit(damage: 25)
enemy.health?.life // 75
```

Links

— About GameplayKit *by Apple*


The End

GameplayKit

Copenhagen Cocoa – Tobias Due Munk – April 23, 2017