



TRUNG TÂM ĐÀO TẠO TIN HỌC KHOA PHẠM

Website: <http://khoapham.vn>

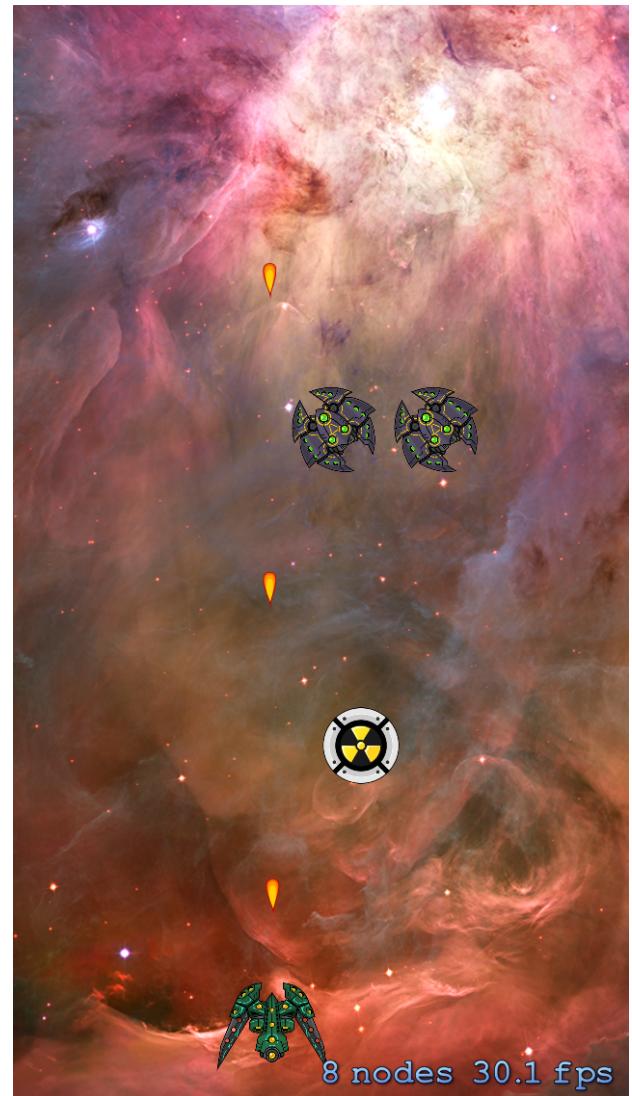
Địa chỉ: 90 Lê Thị Riêng, P.Bến Thành, Q.1, TP.HCM

Điện thoại: 0966 908 907 - 094 276 4080

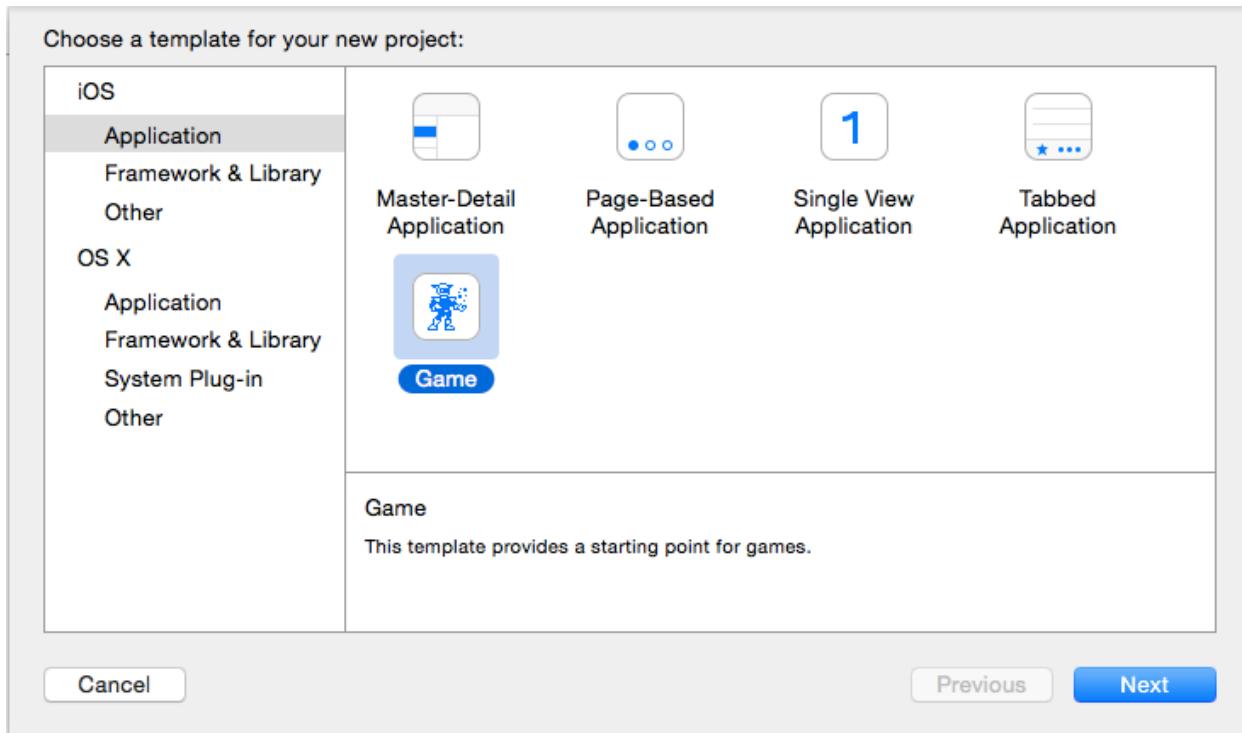
Facebook: <http://facebook.com/khoapham.vn>

LẬP TRÌNH GAME BẮN MÁY BAY VỚI SPRITE KIT

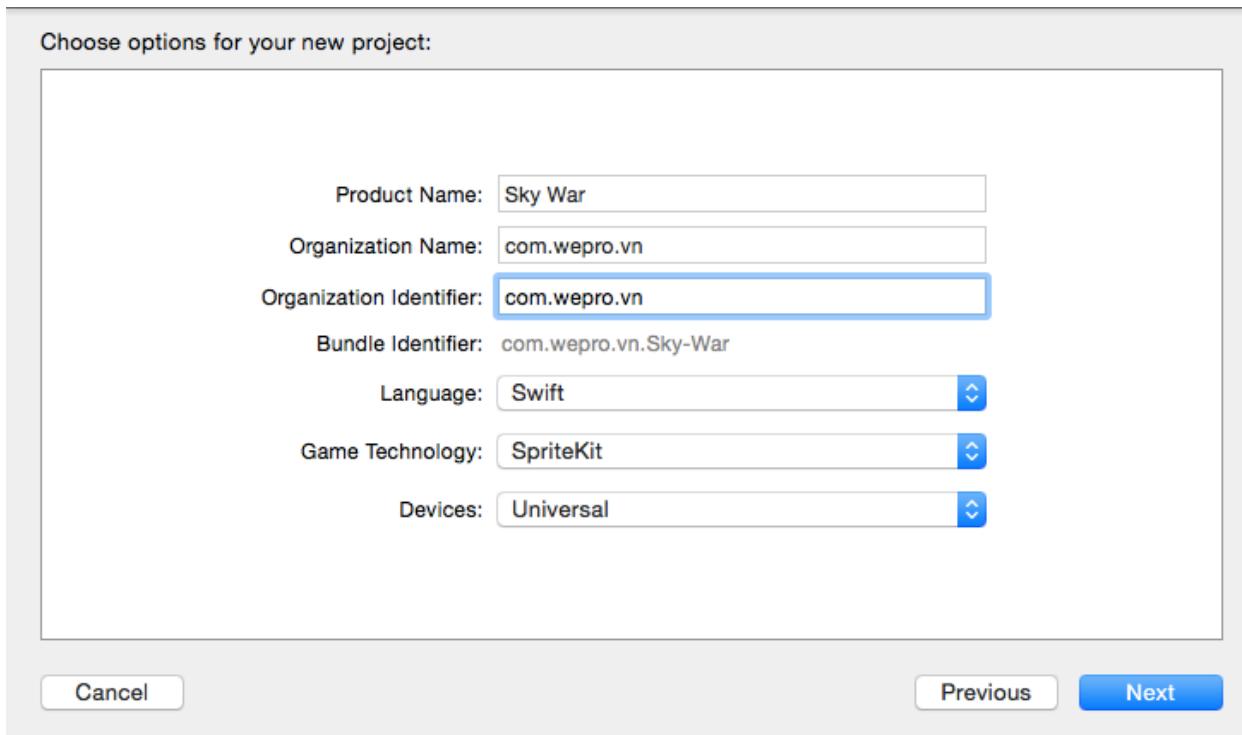
Phần 1: Setting up the project



Bước 1: Tạo project : Chọn “Game” -> Next



- + Đặt tên project là “Sky War”
- + Game Technology là “SpriteKit”
- + Ngôn ngữ là “Swift”



Bước 2: Tạo phi thuyền và cho phi thuyền chuyển di chuyển

```
var ship:SKSpriteNode! // khai báo phi thuyền là một biến toàn cục có kiểu SKSpriteNode
```

```
override func didMoveToView(view: SKView) {
    // hàm tạo phi thuyền
    create_ship()
```

```
func create_ship(){
    // gán image cho ship
    ship = SKSpriteNode(imageNamed: "Ship (2).png")
    // gán size cho ship scale lại cho vừa mắt
    ship.size = CGSize(width: self.size.width/6, height: ship.size.height * (self.size.width / ship.size.width / 6))
    // gán vị trí cho ship lúc mới tạo ship
    ship.position = CGPoint(x: self.size.width/2, y: self.size.height * 0.1)
    addChild(ship)
}
```

```
override func touchesMoved(touches: Set<NSObject>, withEvent event: UIEvent) {
    for touch in (touches as! Set<UITouch>) {
        let location = touch.locationInNode(self)
        ship.position = location
    }
}
```

Bước 3: Tạo background chuyển động + Trong didMoveToView

```
// hàm tạo background chuyển động
background_move()
```

```
func background_move(){
background = SKSpriteNode(imageNamed: "bk01.png")
background.size = CGSize(width: self.size.width, height: background.size.height *
    (self.size.width / background.size.width))
// xác định tâm của node ở góc trái bên dưới màn hình
background.anchorPoint = CGPoint(x: 0, y: 0)
// gán vị trí của background ở tọa độ (0;0)
background.position = CGPoint(x: 0, y: 0)
// gán mặt phẳng z của background = -10 để background nằm dưới
background.zPosition = -10
// tạo action move_bk để tạo cảm giác background di chuyển xuống
// => cho cảm giác phi thuyền bay lên
var move_bk:SKAction = SKAction.moveToY(-background.size.height, duration: 100)
background.runAction(move_bk)
addChild(background)
}
```

Bước 4: Tạo đạn và cho phi thuyền bắn đạn + Trong didMoveToView

```
// gọi timer_shooter để tạo đạn bắn liên tục
timer_shooter = NSTimer.scheduledTimerWithTimeInterval(0.2, target: self, selector:
    "create_bullet", userInfo: nil, repeats: true)
}
```

```
func create_bullet(){
bullet = SKSpriteNode(imageNamed: "Bullet0.png")
bullet.size = CGSizeMake(width: self.size.width/45, height: bullet.size.height * (self.size.width /
    bullet.size.width / 45))
bullet.position = ship.position
bullet.zPosition = -2
addChild(bullet)
shoot_bullet()
}
```

```
func shoot_bullet(){
    // tạo action cho bullet bắn ra khỏi màn hình
    var shoot:SKAction = SKAction.moveToY(1000, duration: 1)
    bullet.runAction(shoot)
}
```

Bước 5: Tạo monster và diễn hoạt cho monster

```
var monster:SKSpriteNode!
```

+ Trong didMoveToView

```
// tạo monster lần đầu
create_monster()
```

```
// gọi timer_create_monster để 3s gọi monster
timer_create_monster = NSTimer.scheduledTimerWithTimeInterval(3, target: self, selector:
    "create_monster", userInfo: nil, repeats: true)
```

```
func create_monster(){
// vòng for random số lượng quái vật từ 1 đến 4
for i in 0...random(min: 0, 3){
monster = SKSpriteNode(imageNamed: "Ship (3).png")
monster.size = CGSizeMake(width: self.size.width/6, height: ship.size.height * (self.size.width /
    ship.size.width / 6))
// random vị trí quái vật
monster.position = CGPointMake(x: random(min: self.size.width * 0.1, self.size.width * 0.9),
    y: self.size.height)
// tạo đường đi cho quái vật
var path = setPath(monster.position.x, toaDoY: monster.position.y, view: self.view!)
// tạo action remove quái vật
var monsterRemove:SKAction = SKAction.removeFromParent()
// tạo action cho quái vật di theo đường đi đã tạo
monster.runAction(SKAction.sequence([SKAction.followPath(path[random(min: 0, path.count - 1)].
    CGPath, duration: 4), monsterRemove]))
addChild(monster)
}
}
```

```

func setPath(toaDoX: CGFloat, toaDoY: CGFloat, view: UIView) -> [UIBezierPath]{
    var path1: UIBezierPath = UIBezierPath()
    path1.moveToPoint(CGPoint(x: 0, y: 0))
    path1.addLineToPoint(CGPoint(x: -100, y: -view.frame.size.height * 0.7))
    path1.addLineToPoint(CGPoint(x: 100, y: -view.frame.size.height * 0.3))
    path1.addLineToPoint(CGPoint(x: 0, y: -view.frame.size.height - 30))

    var path2: UIBezierPath = UIBezierPath()
    path2.moveToPoint(CGPoint(x: 0, y: 0))
    path2.addLineToPoint(CGPoint(x: 100, y: -view.frame.size.height * 0.7))
    path2.addLineToPoint(CGPoint(x: -100, y: -view.frame.size.height * 0.3))
    path2.addLineToPoint(CGPoint(x: 0, y: -view.frame.size.height - 30))

    var path3xTrai: UIBezierPath = UIBezierPath()
    path3xTrai.moveToPoint(CGPoint(x: 0, y: 0))
    path3xTrai.addCurveToPoint(CGPoint(x: 0, y: -view.frame.size.height), controlPoint1:
        CGPoint(x: view.frame.size.width * 0.8, y: -view.frame.size.height * 0.3), controlPoint2:
        CGPoint(x: -view.frame.size.width * 0.8, y: -view.frame.size.height * 0.7))

    var path3xPhai: UIBezierPath = UIBezierPath()
    path3xPhai.moveToPoint(CGPoint(x: 0, y: 0))
    path3xPhai.addCurveToPoint(CGPoint(x: 0, y: -view.frame.size.height), controlPoint1:
        CGPoint(x: -view.frame.size.width * 0.8, y: -view.frame.size.height * 0.3), controlPoint2:
        CGPoint(x: view.frame.size.width * 0.8, y: -view.frame.size.height * 0.7))

    if toaDoX < view.frame.size.width / 2 {
        return [path1, path2, path3xPhai]
    } else {
        return [path1, path2, path3xTrai]
    }
}

```

```

// hàm random tọa độ
func random(#min: CGFloat, max: CGFloat) -> CGFloat {
    return CGFloat(Float(arc4random_uniform(UInt32(max - min + 1)))) + min
}
// hàm |random số nguyên
func random_int(#min: Int, max: Int) -> Int {
    return Int(Int(arc4random_uniform(UInt32(max - min + 1)))) + min
}

```

Bước 6: Hàm tạo hiệu ứng nổ

```

func explosion (location: CGPoint){
    // tạo 1 mảng có kiểu là SKTexture
    var explosion_texture:[SKTexture] = []
    // add 10 hình nổ vào mảng explosion_texture
    for i in 0...9 {
        explosion_texture.append(SKTexture(imageNamed: "Explo_00\(\(i)\).png"))
    }
    // tạo explosion_node là node để run action nổ
    var explosion_node:SKSpriteNode = SKSpriteNode(imageNamed: "Explo_000.png")
    // vị trí nổ là biến location truyền vào
    explosion_node.position = location
    explosion_node.size = CGSize(width: self.size.width / 3, height: explosion_node.size.height *
        (self.size.width / explosion_node.size.width / 3) )
    // tạo action nổ
    var actionExplosion:SKAction = SKAction.animateWithTextures(explosion_texture, timePerFrame:
        0.05)
    // nổ xong remove
    var actionRemove:SKAction = SKAction.removeFromParent()
    // explosion_node sẽ thực hiện lần lượt action nổ và remove
    explosion_node.runAction(SKAction.sequence([actionExplosion, actionRemove]))
    self.addChild(explosion_node)
}

```

Bước 7. Tạo va chạm giữa đạn và monster
+ Khai báo thêm cho class SKPhysicsContactDelegate

```
class GameScene: SKScene, SKPhysicsContactDelegate {
```

+ Khai báo các điều kiện cần thiết để kiểm tra va chạm

```
var check_contact = 0

enum vatThe: UInt32 {
    case ship = 0
    case monster = 1
    case bullet = 4
}
```

+ Trong didMoveToView

```
override func didMoveToView(view: SKView) {
    physicsWorld.contactDelegate = self
```

+ Trong hàm create_ship khi báo physicsBody cho ship

```
ship.name = "name_ship"
ship.physicsBody = SKPhysicsBody(texture: ship.texture, size: ship.size)
ship.physicsBody?.dynamic = true
ship.physicsBody?.affectedByGravity = false
ship.physicsBody?.categoryBitMask = vatThe.ship.rawValue
ship.physicsBody?.contactTestBitMask = vatThe.monster.rawValue
ship.physicsBody?.collisionBitMask = 0
```

+ Trong hàm create_bullet khai báo physicsBody cho bullet

```
bullet.name = "bullet_name"
bullet.physicsBody = SKPhysicsBody(texture: bullet.texture, size: bullet.size)
bullet.physicsBody?.dynamic = true
bullet.physicsBody?.affectedByGravity = false
bullet.physicsBody?.categoryBitMask = vatThe.bullet.rawValue
bullet.physicsBody?.contactTestBitMask = vatThe.monster.rawValue
bullet.physicsBody?.collisionBitMask = 0
```

+ Tương tự khai báo physicsBody cho monster

```
monster.name = "name_monster"
monster.physicsBody = SKPhysicsBody(texture: monster.texture, size: monster.size)
monster.physicsBody?.dynamic = true
monster.physicsBody?.affectedByGravity = false
monster.physicsBody?.categoryBitMask = vatThe.monster.rawValue
monster.physicsBody?.contactTestBitMask = vatThe.ship.rawValue
monster.physicsBody?.contactTestBitMask = vatThe.bullet.rawValue
monster.physicsBody?.collisionBitMask = 0
```

+ Hàm để bắt va chạm

```
func didBeginContact(contact: SKPhysicsContact) {  
    var bodyA = contact.bodyA.categoryBitMask  
    var bodyB = contact.bodyB.categoryBitMask  
    if (bodyA == vatThe.bullet.rawValue && bodyB == vatThe.monster.rawValue) || (bodyA ==  
        vatThe.monster.rawValue && bodyB == vatThe.bullet.rawValue){  
        contact_bullet_monster(contact.bodyA.node as! SKSpriteNode, nodeB: contact.bodyB.node  
            as! SKSpriteNode, location: contact.contactPoint)  
    }  
    if (bodyA == vatThe.monster.rawValue && bodyB == vatThe.ship.rawValue) || (bodyA ==  
        vatThe.ship.rawValue && bodyB == vatThe.monster.rawValue){  
        game_over()  
    }  
}
```

+ Khi va chạm gọi hàm contact_bullet_monster

```
func contact_bullet_monster(nodeA: SKSpriteNode, nodeB: SKSpriteNode, location: CGPoint){  
    if check_contact == 0{  
        check_contact = 1  
        explosion(location)  
  
        nodeA.removeFromParent()  
        nodeB.removeFromParent()  
    }  
}
```

+ Hàm kiểm tra va chạm để xác định va chạm chỉ xảy ra 1 lần

```
override func update(currentTime: CFTimeInterval) {  
    // trong hàm update cho check_contact = 0 để check va chạm , khi va chạm chỉ tính 1 lần  
    check_contact = 0  
}
```