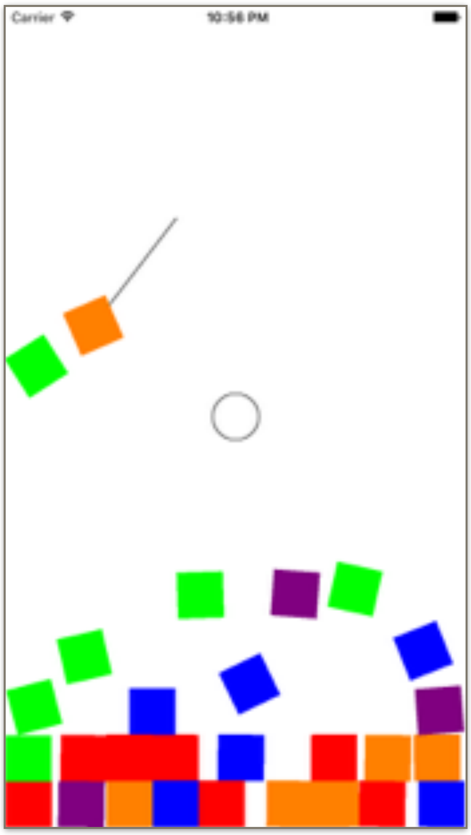
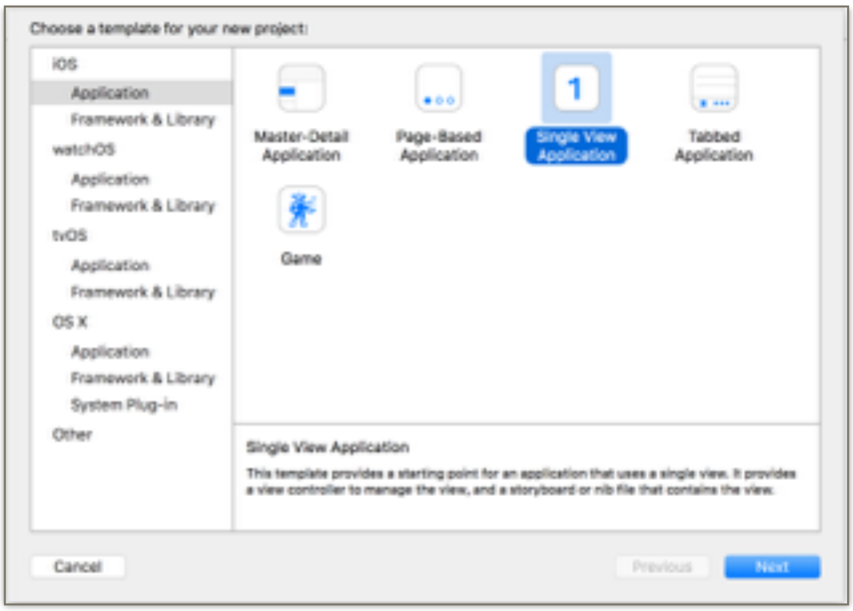
# iOS开发实验手册

## Animation

本章实验将介绍动画，并编写一个Dropit应用程序。应用程序界面如下：



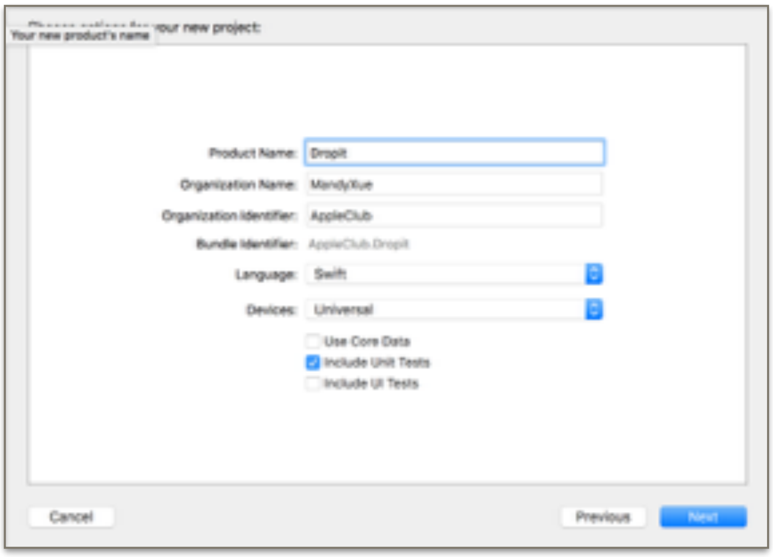
运行Xcode并新建一个SingleView项目：



Single View Application

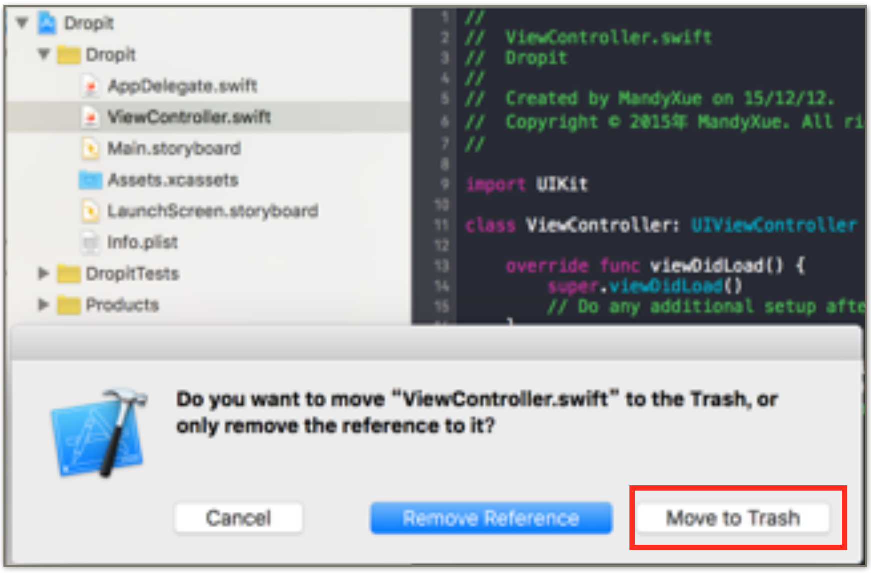
This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

将项目命名为Dropit并设置好相应参数：



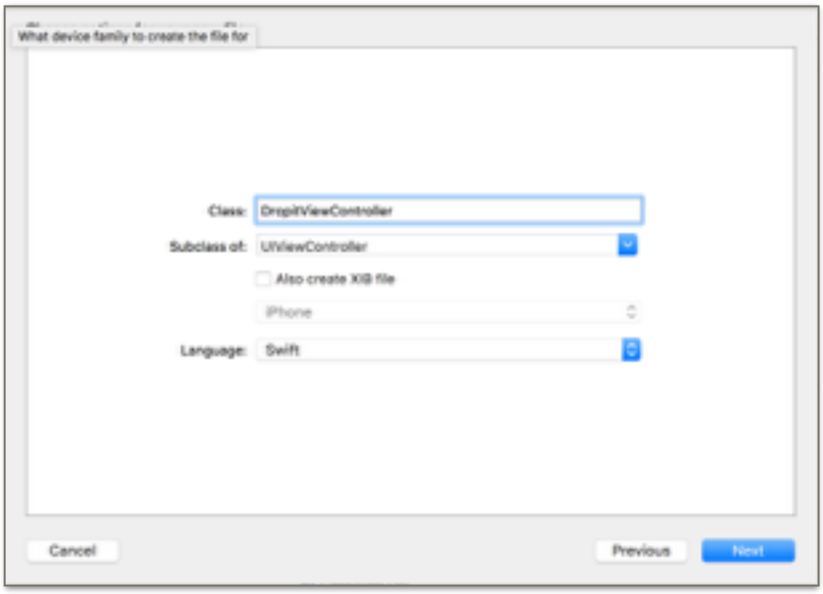
Product Name: Dropit
Organization Name: MandyXue
Organization Identifier: AppleClub
Bundle Identifier: AppleClub.Dropit
Language: Swift
Devices: Universal
Use Core Data
[✓ Include Unit Tests]
Include UI Tests

将项目中的ViewController.swift文件删除并创建用户自定义的控制类，命名为DropitViewController：



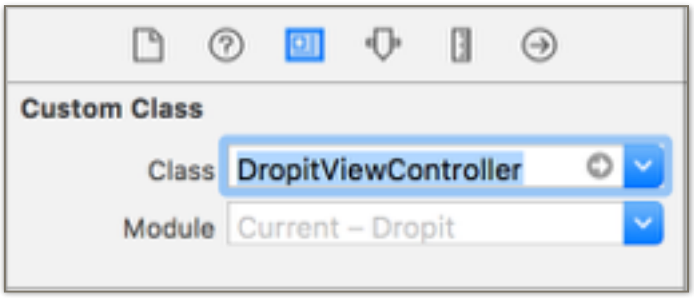
```
//
//  ViewController.swift
//  Dropit
//
//  Created by MandyXue on 15/12/12.
//  Copyright © 2015年 MandyXue. All ri
//

import UIKit

class ViewController: UIViewController

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup afte
```

Do you want to move "ViewController.swift" to the Trash, or only remove the reference to it?



Class: DropitViewController
Subclass of: UIViewController
Also create XIB file
iPhone
Language: Swift

在Main.storyboard中将原先ViewController的类设置为DropitViewController：

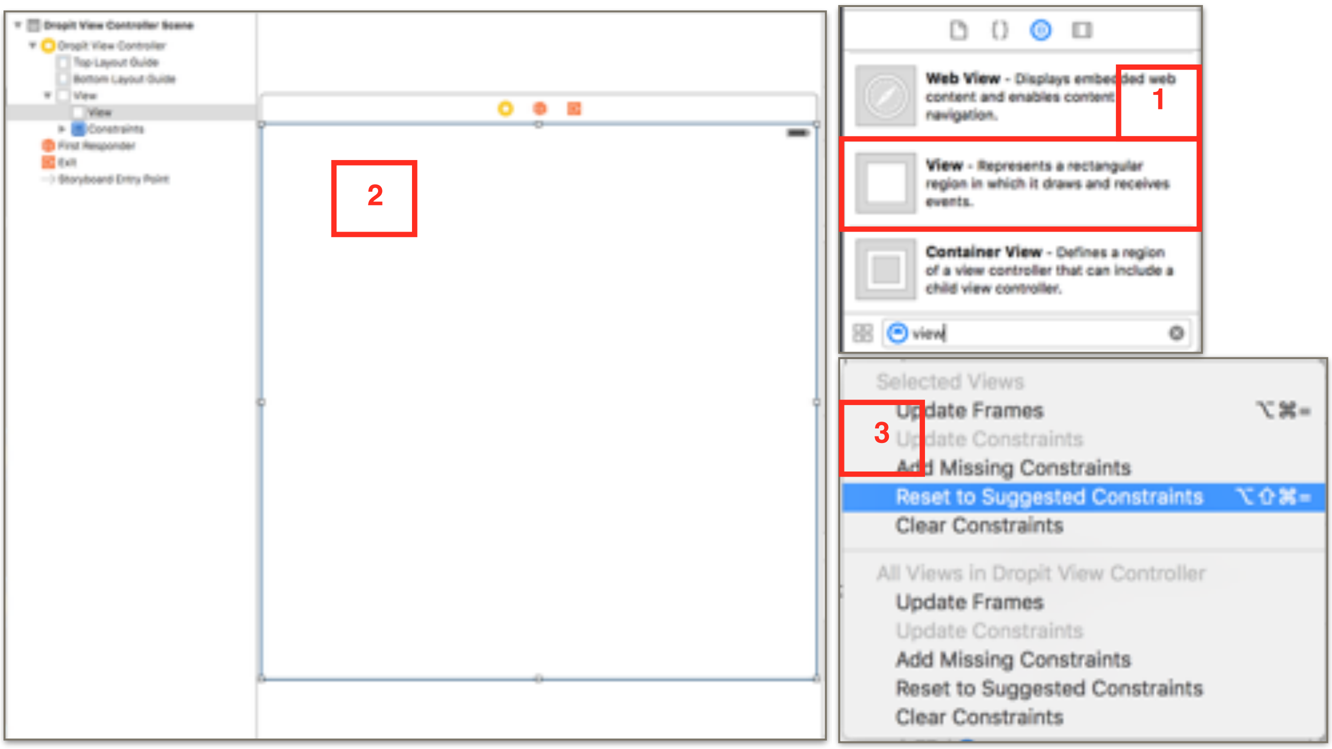


Custom Class

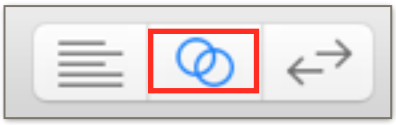
Class: DropitViewController

Module: Current – Dropit

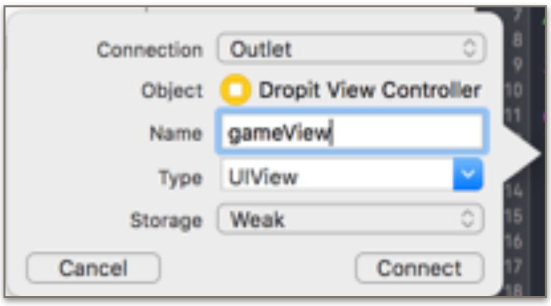
拖动一个范型视图进入storyboard，并设置好autolayout：


Dropit View Controller Scene
Dropit View Controller
Top Layout Guide
Bottom Layout Guide
View
View
Constraints
First Responder
Exit
Storyboard Entry Point
2
Web View - Displays embedded web content and enables content navigation.
1
View - Represents a rectangular region in which it draws and receives events.
Container View - Defines a region of a view controller that can include a child view controller.
view
Selected Views
Update Frames
Update Constraints
3
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints
All Views in Dropit View Controller
Update Frames
Update Constraints
Add Missing Constraints
Reset to Suggested Constraints
Clear Constraints

选择show the assistant editor视图模式以同时展示代码与storyboard：



将刚拖入的view按住control并拖至代码中：



Connection: Outlet

Object: Dropit View Controller

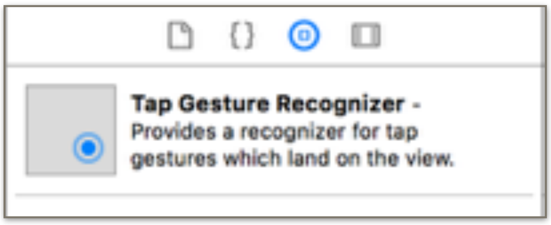
Name: gameView

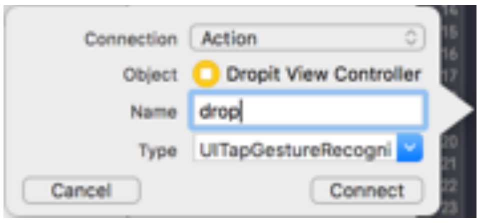
Type: UIView

Storage: Weak

```
@IBOutlet weak var gameView: UIView!
```

再拖入一个手势点击识别器（tap gesture recognizer），并与代码相连：


**Tap Gesture Recognizer** - Provides a recognizer for tap gestures which land on the view.


Connection: Action
Object: Dropit View Controller
Name: drop
Type: UITapGestureRecogni

```
@IBAction func drop(sender: UITapGestureRecognizer) {
    drop()
}
```

打开DropitViewController.swift代码窗口，并在class DropitViewController: UIViewController {} 中添加如下代码来绘制随机小方块：

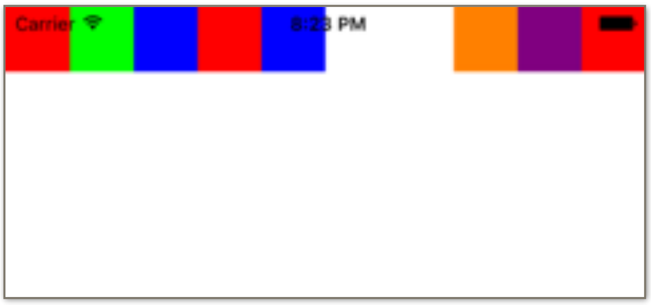
```
var dropsPerRow = 10

var dropSize: CGSize {
    let size = gameView.bounds.size.width / CGFloat(dropsPerRow)
    return CGSize(width: size, height: size)
}

func drop(){
    var frame = CGRect(origin: CGPointZero, size: dropSize)
    frame.origin.x = CGFloat.random(dropsPerRow) * dropSize.width

    let dropView = UIView(frame: frame)
    dropView.backgroundColor = UIColor.random

    gameView.addSubview(dropView)
}

override func viewDidLoad() {
    super.viewDidLoad()
    animator.addBehavior(gravity)
}
```

在class外添加如下代码实现扩展：

```
private extension CGFloat {
    static func random(max: Int) -> CGFloat {
        return CGFloat(arc4random() % UInt32(max))
    }
}

private extension UIColor {
    class var random: UIColor {
        switch arc4random()%5 {
        case 0: return UIColor.greenColor()
        case 1: return UIColor.blueColor()
```

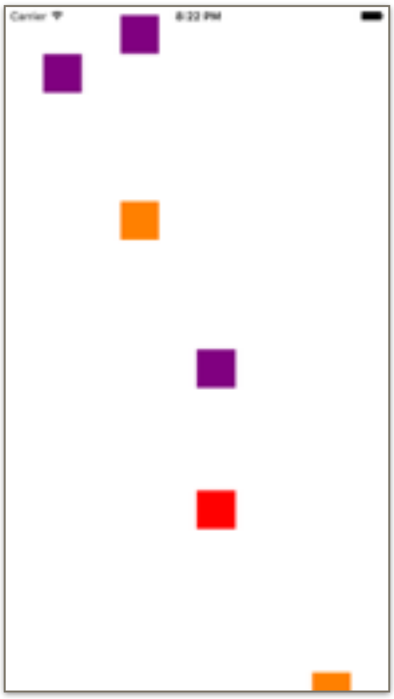
```
                case 2: return UIColor.orangeColor()
                case 3: return UIColor.redColor()
                case 4: return UIColor.purpleColor()
                default: return UIColor.blackColor()
                }
            }
        }
```

点击运行，多次点击iPhone模拟器屏幕，查看效果如下：



接下来，在class DropitViewController: UIViewController {} 中添加如下代码来给小方块增加重力下降效果，其中需要使用lazy initialization的方法来初始化animator：

```
let gravity = UIGravityBehavior()

lazy var animator: UIDynamicAnimator = {
    let lazilyCreatedDynamicAnimator =
UIDynamicAnimator(referenceView:
self.gameView)
    return lazilyCreatedDynamicAnimator
}()

override func viewDidLoad() {
    super.viewDidLoad()
    animator.addBehavior(gravity)
}
```



在drop()方法末尾添加如下代码以增加重力效果：

```
    gravity.addItem(dropView)
```

点击运行，多次点击iPhone模拟器屏幕，效果如右图。

此时，在class DropitViewController: UIViewController {} 中添加如下代码来使小方块不落出屏幕：

```
lazy var collider: UICollisionBehavior = {
        let lazilyCreatedCollision = UICollisionBehavior()
        //configure here
        //edges of reference view are going to be a boundary
        lazilyCreatedCollision.translatesReferenceBoundsIntooundary = true
        return lazilyCreatedCollision
    }()
```

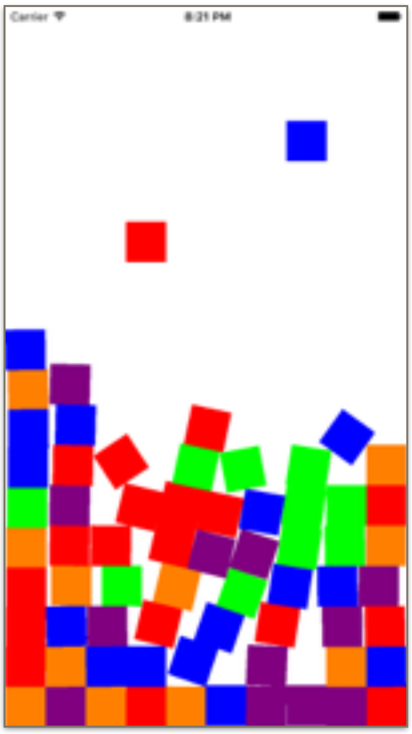
在viewDidLoad()末尾添加如下代码：

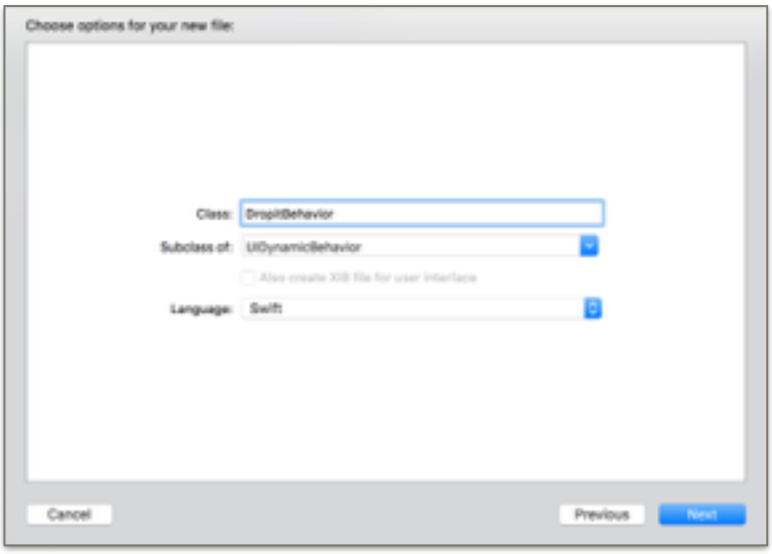
```
animator.addBehavior(collider)
```

在drop()末尾添加如下代码：

```
collider.addItem(dropView)
```

点击运行，多次点击iPhone模拟器屏幕，效果如右图：




Choose options for your new file:

Class: DropitBehavior
Subclass of: UIDynamicBehavior
Also create XIB file for user interface
Language: Swift

此时创建一个DropitBehavior类，用于统一设置小方块的动画，创建方式如上图，代码如下：

```
class DropitBehavior: UIDynamicBehavior {
    let gravity = UIGravityBehavior()

    lazy var collider: UICollisionBehavior = {
        let lazilyCreatedCollision = UICollisionBehavior()
        lazilyCreatedCollision.translatesReferenceBoundsIntoBoundary = true
        return lazilyCreatedCollision
    }()

    override init() {
        super.init()
        addChildBehavior(gravity)
        addChildBehavior(collider)
    }

    func addDrop(drop: UIView){
        dynamicAnimator?.referenceView?.addSubview(drop)
        gravity.addItem(drop)
        collider.addItem(drop)
    }

    func removeDrop(drop: UIView){
        gravity.removeItem(drop)
        collider.removeItem(drop)
        drop.removeFromSuperview()
    }
}
```

在DropitViewController.swift中进行如下修改（红色为修改部分）：

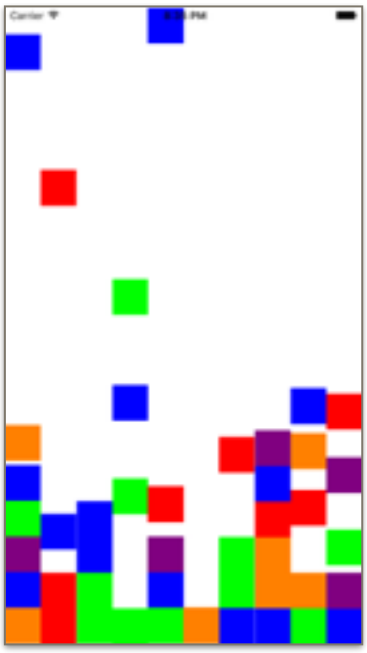
```
    var dropitBehavior = DropitBehavior()
    func drop(){
            var frame = CGRect(origin: CGPointZero, size: dropSize)
            frame.origin.x = CGFloat.random(dropsPerRow) * dropSize.width

            let dropView = UIView(frame: frame)
            dropView.backgroundColor = UIColor.random

            gameView.addSubview(dropView)

            dropitBehavior.addDrop(dropView)
     }
     override func viewDidLoad() {
            super.viewDidLoad()
            animator.addBehavior(dropitBehavior)
     }
```

再次运行，查看结果，与之前相符即可。

此时在DropitViewController.swift中添加如下代码（红色部分为添加），添加小方块的跳跃程度并设置碰撞后不旋转：

```
    lazy var dropBehavior: UIDynamicItemBehavior = {
            let lazilyCreatedDropBehavior = UIDynamicItemBehavior()
            lazilyCreatedDropBehavior.allowsRotation = false
            lazilyCreatedDropBehavior.elasticity = 0.75
            return lazilyCreatedDropBehavior
    }()

    override init() {
            super.init()
            addChildBehavior(gravity)
            addChildBehavior(collider)
            addChildBehavior(dropBehavior)
    }

    func addDrop(drop: UIView){
            dynamicAnimator?.referenceView?.addSubview(drop)
            gravity.addItem(drop)
            collider.addItem(drop)
            dropBehavior.addItem(drop)
    }
```

```
func removeDrop(drop: UIView){
        gravity.removeItem(drop)
        collider.removeItem(drop)
        dropBehavior.removeItem(drop)
        drop.removeFromSuperview()
}
```



运行，小方块碰撞后不再旋转，并且弹跳力增强，结果如右图：

此时，在DropitViewController.swift中添加如下代码，使得小方块在动画结束后从最底层开始检查，若一层叠满则消除一层（类似俄罗斯方块的效果）：

```
fun removeCompletedRow() {
    var dropsToRemove = [UIView]()
    var dropFrame = CGRect(x: 0, y: gameView.frame.maxY, width: dropSize.width, height: 
dropSize.height)

    repeat {
        dropFrame.origin.y -= dropSize.height
        dropFrame.origin.x = 0
        var dropsFound = [UIView]()
        var rowIsComplete = true
        for _ in 0 ..< dropsPerRow {
            if let hitView = gameView.hitTest(CGPoint(x: dropFrame.midX, y: dropFrame.midY), 
withEvent: nil) {
                if hitView.superview == gameView {
                    dropsFound.append(hitView)
                } else {
                    rowIsComplete = false
                }
            }
            dropFrame.origin.x += dropSize.width
        }
        if rowIsComplete {
            dropsToRemove += dropsFound
        }
    } while dropsToRemove.count == 0 && dropFrame.origin.y > 0

    for drop in dropsToRemove {
        dropitBehavior.removeDrop(drop)
    }
}
```

给DropitViewController类添加代理方法，使animator成为自己的代理：

```
class DropitViewController: UIViewController, UIDynamicAnimatorDelegate

lazy var animator: UIDynamicAnimator = {
        let lazilyCreatedDynamicAnimator = UIDynamicAnimator(referenceView:
self.gameView)
        lazilyCreatedDynamicAnimator.delegate = self
        return lazilyCreatedDynamicAnimator
}()
```

在DropitViewController.swift中添加如下代码，使动画全部结束后开始进行消除工作：

```
fun dynamicAnimatorDidPause(animator: UIDynamicAnimator) {
        removeCompletedRow()
}
```

在DropitBehavior.swift中将旋转打开：

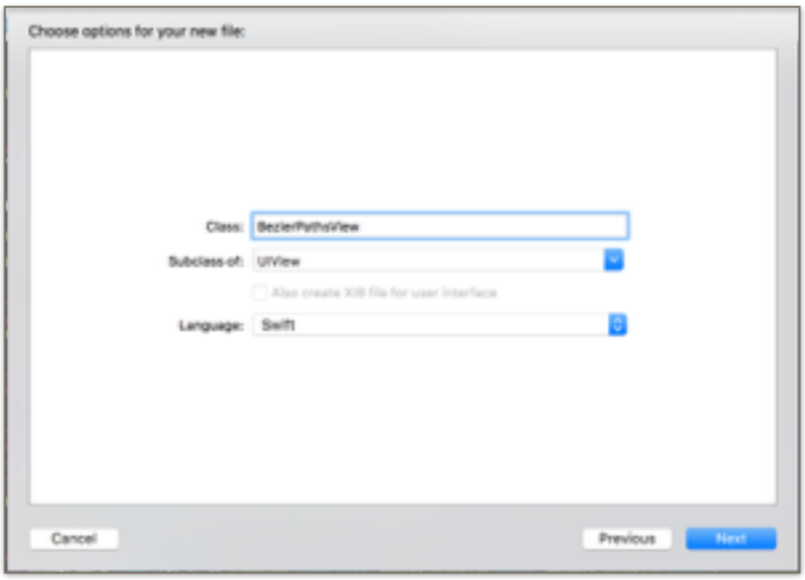
```
lazy var dropBehavior: UIDynamicItemBehavior = {
        let lazilyCreatedDropBehavior = UIDynamicItemBehavior()
        lazilyCreatedDropBehavior.allowsRotation = false
        lazilyCreatedDropBehavior.elasticity = 0.75
        return lazilyCreatedDropBehavior
}()
```

运行，并查看结果。

接下来为界面中心添加一个圆形的障碍物，在DropitBehavior.swift中添加如下代码：

```
func addBarrier(path: UIBezierPath, named name: String) {
    collider.removeBoundaryWithIdentifier(name)
    collider.addBoundaryWithIdentifier(name, forPath: path)
}
```

创建一个BezierPathsView类，用于障碍物的绘制，过程如下：



Choose options for your new file:

Class: BezierPathsView

Subclass of: UIView

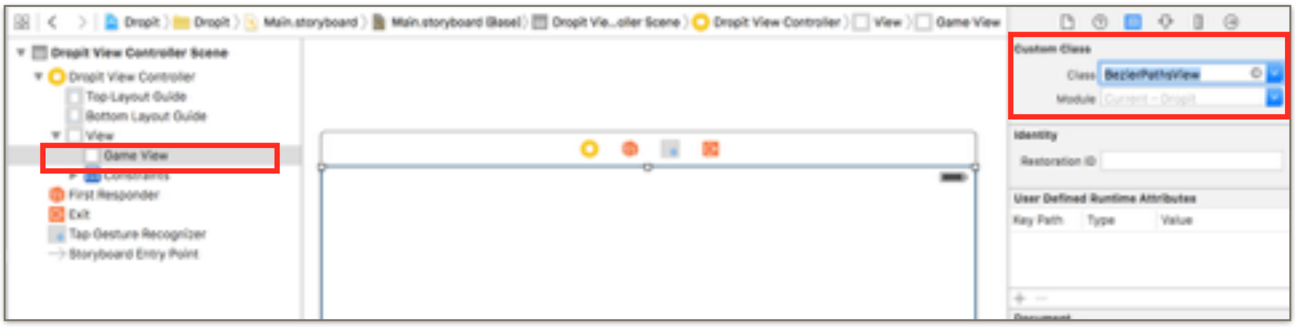
Also create XIB file for user interface

Language: Swift

BezierPathView的实现：

```
class BezierPathsView: UIView {

    private var bezierPaths = [String:UIBezierPath]()

    func setPath(path: UIBezierPath?, named name: String) {
        bezierPaths[name] = path
        setNeedsDisplay()
    }

    override func drawRect(rect: CGRect) {
        for (_, path) in bezierPaths {
            path.stroke()
        }
    }
}
```

切换至storyboard，使gameView继承BezierPathView：



Custom Class
Class BezierPathsView
Module Current – Dropit

在DropitViewController.swift中修改如下代码使gameView继承BezierPathView：

```
@IBOutlet weak var gameView: BezierPathsView!
```

在DropitViewController.swift中添加如下代码，绘制连线：

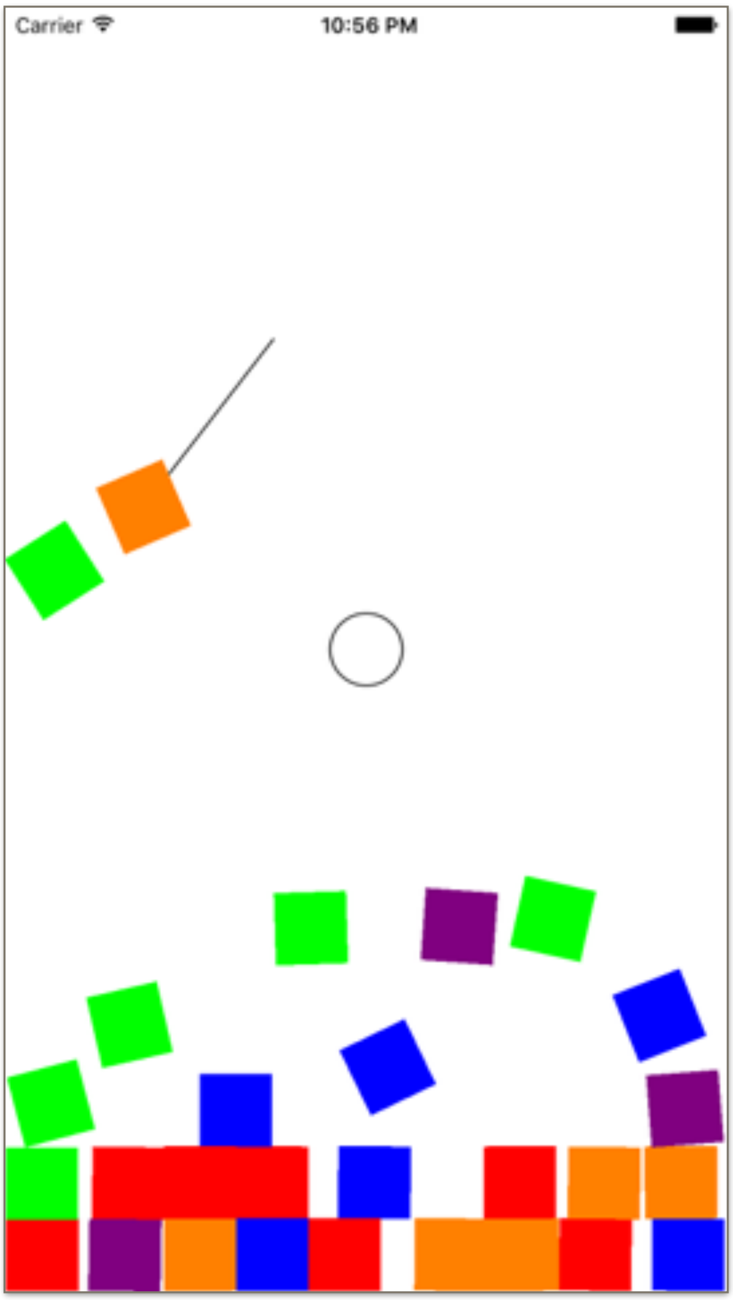
```
override func viewDidLayoutSubviews() {
    super.viewDidLayoutSubviews()
    let barrierSize = dropSize
    let barrierOrigin = CGPoint(x: gameView.bounds.midX-barrierSize.width/2, y: 
gameView.bounds.midY-barrierSize.height/2)
    let path = UIBezierPath(ovalInRect: CGRect(origin: barrierOrigin, size: 
barrierSize))
    dropitBehavior.addBarrier(path, named: PathNames.MiddleBarrier)
    gameView.setPath(path, named: PathNames.MiddleBarrier)
}

var lastDroppedView: UIView?

@IBAction func grabDrop(sender: UIPanGestureRecognizer) {
    let gesturePoint = sender.locationInView(gameView)
```

```
        switch sender.state {
        case .Began:
            if let viewToAttachTo = lastDroppedView {
                attachment = UIAttachmentBehavior(item: viewToAttachTo,
    attachedToAnchor: gesturePoint)
                lastDroppedView = nil
            }
        case .Changed:
            attachment?.anchorPoint = gesturePoint
        case .Ended:
            attachment = nil
        default:
            break
        }
    }

    var attachment: UIAttachmentBehavior? {
        willSet {
            if attachment != nil {
                animator.removeBehavior(attachment!)
                gameView.setPath(nil, named: PathNames.Attachment)
            }
        }
        didSet {
            if attachment != nil {
                animator.addBehavior(attachment!)
                attachment?.action = { [unowned self] in  //fix memory cycle problem
                    if let attachedView = self.attachment?.items.first as? UIView {
                        let path = UIBezierPath()
                        path.moveToPoint(self.attachment!.anchorPoint)
                        path.addLineToPoint(attachedView.center)
                        self.gameView.setPath(path, named: PathNames.Attachment)
                    }
                }
            }
        }
    }
```

在drop()和PathNames中做如下修改：

```
    func drop(){
        var frame = CGRect(origin: CGPointZero, size: dropSize)
        frame.origin.x = CGFloat.random(dropsPerRow) * dropSize.width

        let dropView = UIView(frame: frame)
        dropView.backgroundColor = UIColor.random

        gameView.addSubview(dropView)

        lastDroppedView = dropView
        dropitBehavior.addDrop(dropView)
    }

    struct PathNames {
        static let MiddleBarrier = "Middle Barrier"
        static let Attachment = "Attachment"
    }
```

点击运行，动画小游戏完成：


Carrier
10:56 PM