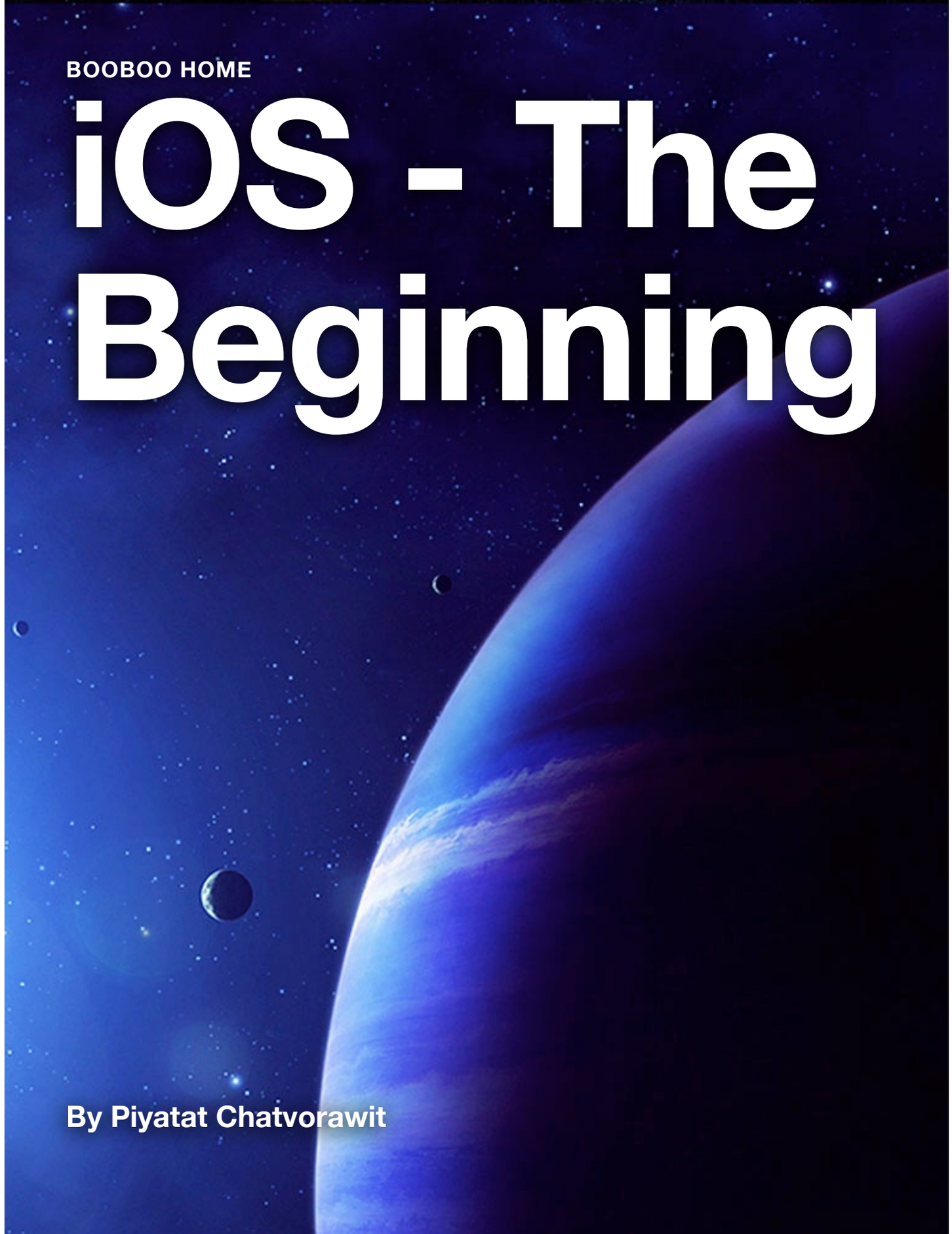

BOOBOO HOME
iOS - The Beginning
By Piyatat Chatvorawit

Chapter 4

iOS - UIKit Part 1: View Component

ในบทนี้เราจะเปลี่ยนมาพูดถึงเกี่ยวกับ iOS กันบ้าง เราจะกล่าวถึงเกี่ยวกับ UIKit ซึ่งจะเป็น framework หลักของ iOS ที่ร่วมคลาสต่างๆที่เกี่ยวข้องกับการแสดงผลและติดต่อกับผู้ใช้งานทางส่วนแสดงผลของโปรแกรม โดยในส่วนนี้เราจะพูดกันถึง UIView, UIViewController และ UIWindow กัน ซึ่ง 3 คลาสนี้จะคลาสหลักๆที่ประกอบขึ้นเป็นหน้าต่างของโปรแกรมทุกโปรแกรม

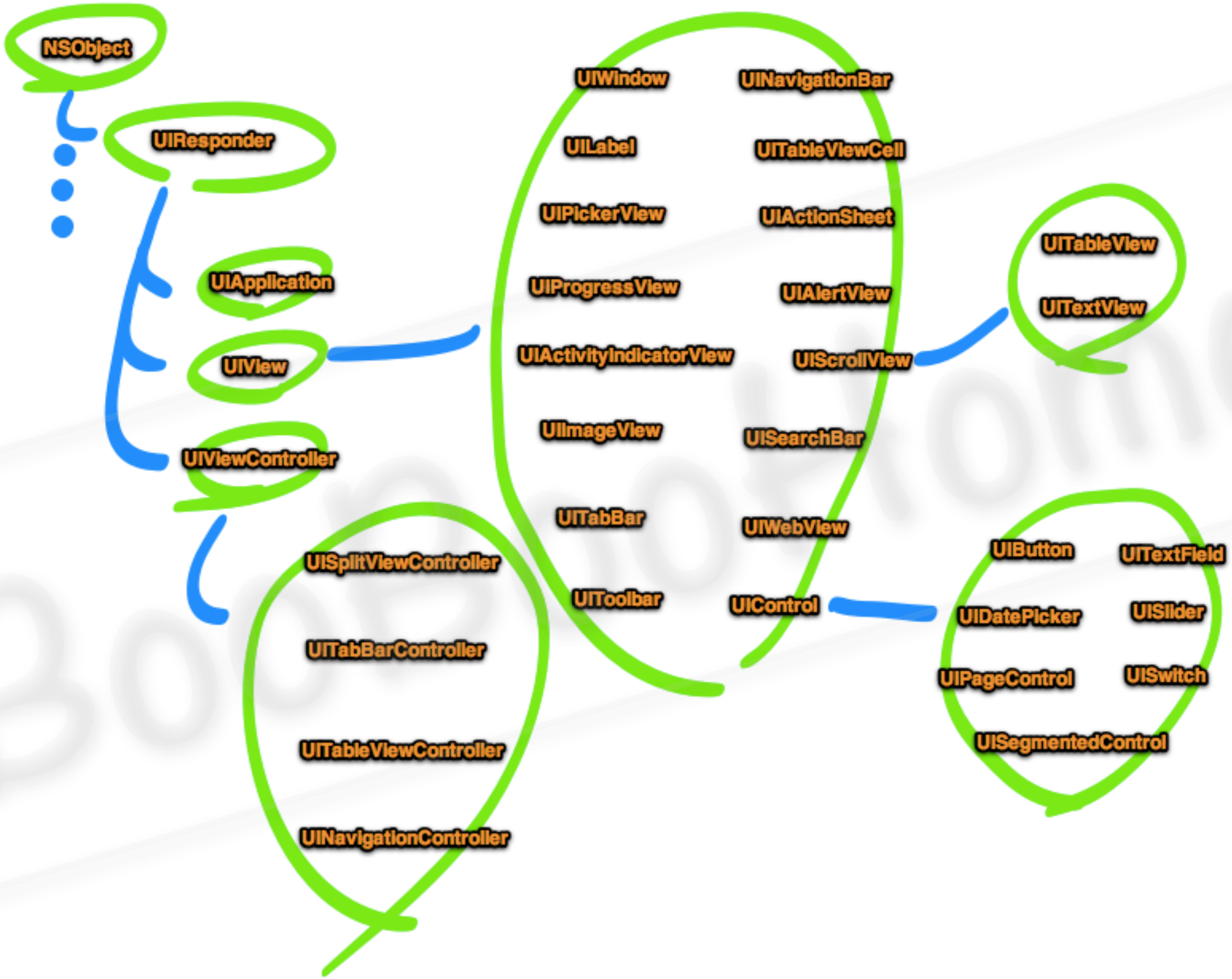
UIKIT FRAMEWORK

1. เป็น framework พื้นฐานของแทบจะทุกโปรแกรมบน iOS ใช้สำหรับติดต่อกับผู้ใช้ผ่านทางหน้าจอ
2. คลาสที่เราจะได้ใช้งานบ่อยๆ จะเป็นคลาสที่ขยายมาจาก UIView และ UIViewController
3. Coordinate system สำหรับ iOS จะแตกต่างกันในแต่ละ framework โดยใน UIKit จะใช้จุดบนซ้ายเป็นจุด (0, 0) และทำการคำนวนพิกัดโดยอ้างอิงกับ superview

อย่างที่เราได้กล่าวไปแล้วข้างต้นว่า UIKit คือ framework พื้นฐานสำหรับใช้ในการติดต่อกับผู้ใช้งาน ซึ่ง framework นี่จะมีให้ใช้งานได้เฉพาะใน iOS เท่านั้น ไม่สามารถนำไปใช้งานบน Mac OS ได้ เนื่องจากพื้นฐานของรูปแบบการติดต่อกับผู้ใช้งานต่างกัน โดยบน Mac OS จะใช้ mouse กับ keyboard สำหรับรับ input จากผู้ใช้งาน ในขณะที่ iOS จะใช้การสัมผัสหน้าจอโดยตรงเพื่อรับ input จากผู้ใช้งาน ทำให้ UIKit ถูกออกแบบมาเพื่อรองรับการติดต่อกับผู้ใช้งานด้วยการสัมผัสบนหน้าจอของโปรแกรมโดยตรง

คลาสที่อยู่ใน framework นี่ที่เราจะต้องใช้งานบ่อยครั้งนั้น จะมีดังรูปด้านล่างนี้

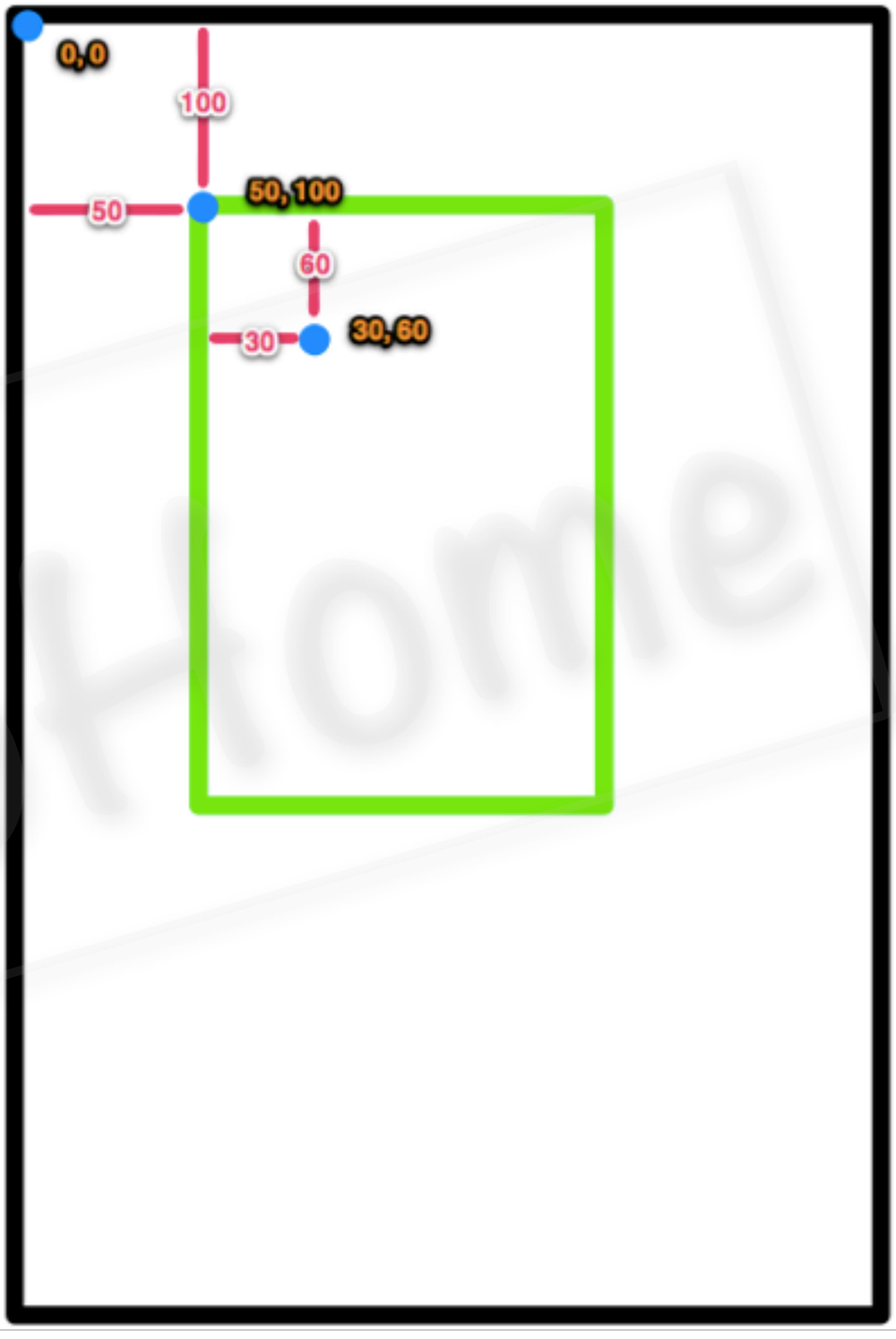
BooBooHome


NSObject
UIResponder
UIApplication
UIView
UIViewController
UISplitViewController
UITabBarController
UITableViewController
UINavigationController
UIWindow
UINavigationBar
UILabel
UITableViewCell
UIPickerView
UIActionSheet
UIProgressView
UIAlertView
UIActivityIndicatorView
UIScrollView
UIImageView
UISearchBar
UITabBar
UIWebView
UIToolbar
UIControl
UITableView
UITextView
UIButton
UITextField
UIDatePicker
UISlider
UIPageControl
UISwitch
UISegmentedControl

สำหรับผู้ที่ต้องการดู class hierarchy เต็มๆสำหรับ UIKit framework นั้น สามารถเข้าไปดูได้ที่เว็บของทาง Apple ตามลิงค์นี้ได้เลยครับ

http://developer.apple.com/library/ios/documentation/uikit/reference/UIKit_Framework/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006955-CH1-SW1

ตามมาจะเป็นเรื่องของ coordinate system โดยใน iOS นั้น จะแตกต่างกันไปในแต่ละ framework ซึ่งสำหรับ UIKit ซึ่งเป็น framework ที่จะถูกใช้โดยทั่วไปนั้นจะมีระบบ coordinate system เป็นดังรูปด้านล่างนี้



BooBook.com

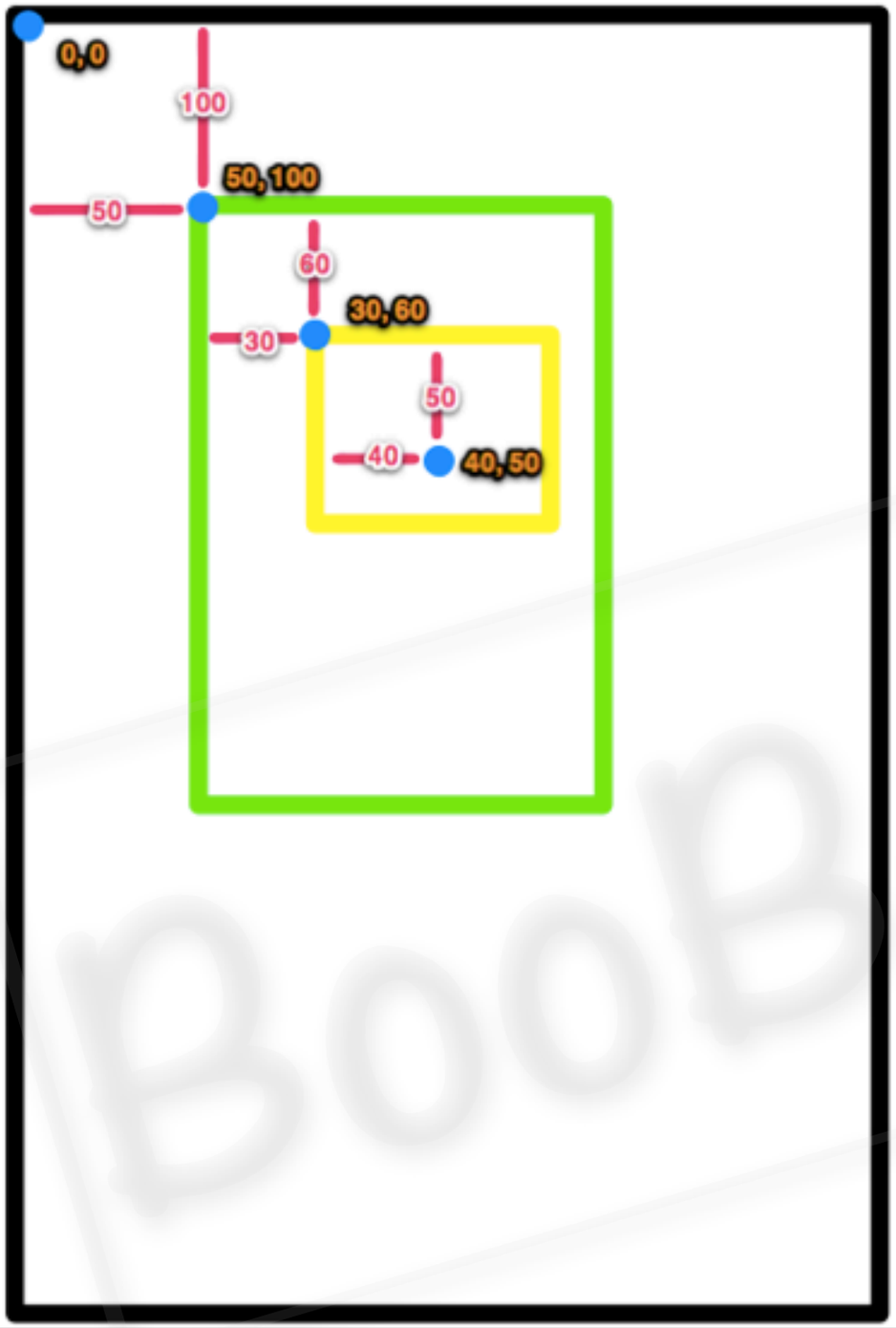
UIVIEW

1. **เป็น super class ของคลาสต่างๆที่ใช้ในการแสดงผลบนหน้าจอ และรับ input จากผู้ใช้งาน**
2. **UIView จะเป็นเพียงพื้นที่สี่เหลี่ยมผืนผ้าเปล่าๆ เท่านั้น ซึ่งโดยส่วนใหญ่เราจะใช้งานคลาสที่ขยายออกมาจาก UIView แทน**
3. **UIView จะมีคลาสเมธอดสำหรับใช้ในการสร้าง animation แบบง่ายๆไว้ให้เรียกใช้งาน**

คลาสแรกที่เราจะมาพูดถึงกันนี้ คือ UIView ซึ่งคลาสนี้เป็นเหมือนกับคลาสพื้นฐานของหลายๆ คลาสใน iOS ที่เกี่ยวข้องกับส่วนของการแสดงผลเพื่อใช้ติดต่อกับผู้ใช้งานผ่านทางหน้าจอของโปรแกรม โดย UIView จะมีพื้นที่แสดงผลเป็นสี่เหลี่ยมผืนผ้า และจะรับผิดชอบต่อการแสดงผลต่างๆในพื้นที่บริเวณนี้ ร่วมถึงรับ input ต่างๆที่เข้ามาในบริเวณนี้อีกด้วย

object UIView นึงสามารถที่จะเก็บ object UIView อันอื่นๆเข้ามาไว้ได้ เพื่อสร้างการแสดงผลที่ซับซ้อนขึ้น โดย UIView ที่เก็บ UIView อื่นเราจะเรียกว่า superview ส่วน UIView ที่ถูกใส่ลงไปจะเรียกว่า subview โดย subview แต่ละอันจะมี superview ได้เพียงอันเดียวเท่านั้น ตำแหน่งของ subview นั้นจะถูกอ้างอิงกับ superview ของมัน ดังรูปด้านล่างนี้

BooBooHome



โดยปกติการแสดงผลของ subview จะไม่ถูกจำกัดโดยขนาดของ superview นั่นคือ subview สามารถที่จะแสดงผลเกินออกมาจากขนาดของ superview ได้ แต่เราก็สามารถที่จะกำหนดให้ subview ไม่แสดงผลส่วนที่เกินออกมาก็ได้เช่นกัน

สำหรับการตอบสนองต่อ input นั้น subview จะสามารถตอบสนองได้เพียงบริเวณที่อยู่ใน superview เท่านั้น นั้นคือ ถึงแม้ว่า subview จะมีขนาดใหญ่กว่า superview แต่ก็จะรับ input ได้เพียงแค่ส่วนที่อยู่ในขอบเขตของ superview เท่านั้น

สำหรับคุณสมบัติต่างๆที่เราสามารถกำหนดหรือเรียกดูให้กับ UIView ได้นั้นจะมีดังนี้

BooBook

1. คุณสมบัติที่เกี่ยวข้องกับการแสดงผล

Property	Type	Description
frame	CGRect	ตำแหน่งและขนาดของ view โดยอ้างอิงกับ superview
bounds	CGRect	ตำแหน่งและขนาดของ view โดยอ้างอิงกับ view ตัวเอง
center	CGPoint	ตำแหน่งจุดศูนย์กลางของ view โดยอ้างอิงกับ superview
transform	CGAffineTransform	การปรับเปลี่ยน view โดยอ้างอิงกับตำแหน่ง center ของ view
backgroundColor	UIColor	สีพื้นหลังของ view
hidden	BOOL	กำหนดให้ view แสดงหรือไม่
alpha	CGFloat	ความโปร่งใสของ view โดยกำหนดเป้นค่า 0.0 ถึง 1.0
opaque	BOOL	กำหนด opaque ของ view
clipsToBounds	BOOL	กำหนดให้ view แสดงผลเฉพาะในขอบเขตของ view
clearsContextBeforeDrawing	BOOL	กำหนดให้ view ลบข้อมูลการแสดงผลเก่าก่อนที่จะทำการ draw ใหม่
layer	CALayer	Core Animation layer ของ view
tag	NSInteger	ตัวเลขสำหรับใช้ในการอ้างอิงถึง view ภายในโปรแกรม

2. คุณสมบัติที่เกี่ยวข้องกับการปรับเปลี่ยนขนาดของ view

Property	Type	Description
autoresizingMask	UIViewAutoresizing	กำหนดรูปแบบการเปลี่ยนแปลงขนาดของ view เมื่อ superview มีการเปลี่ยนแปลงขนาด
autoresizesSubviews	BOOL	กำหนดให้ view ทำการเปลี่ยนแปลงขนาดของ subview เมื่อ view มีการเปลี่ยนแปลงขนาด
contentMode	UIViewContentMode	กำหนดรูปแบบ layout ของ subview
contentStretch	CGRect	กำหนดตำแหน่งและขนาดของ content ของ view โดยจะใช้ค่า normalize 0.0 - 1.0
contentScaleFactor	CGFloat	scale factor ของ view ซึ่งจะมีค่าเป็น 1.0 (normal display) หรือ 2.0 (retina display)

3. คุณสมบัติที่เกี่ยวข้องกับการติดต่อกับผู้ใช้

Property	Type	Description
userInteractionEnabled	BOOL	กำหนดให้ view รับ input จากผู้ใช้ได้หรือไม่
multipleTouchEnabled	BOOL	กำหนดให้ view รับ multi-touch input ได้หรือไม่
exclusiveTouch	BOOL	กำหนดให้ view ทำการ block input ไม่ให้ส่งผ่านให้กับ view อื่นหรือไม่
gestureRecognizers	NSArray	array ของ gesture-recognizer ที่ผูกกับ view

4. คุณสมบัติที่เกี่ยวข้องกับ view hierarchy

Property	Type	Description
superview	UIView	superview ของ view หรือ nil ถ้าไม่มี
subviews	NSArray	array ของ subview ของ view
window	UIWindow	window ที่ view นี้บรรจุอยู่ หรือ nil ถ้าไม่มี

และเมธอดของ UIView ที่เราจะเรียกใช้งานบ่อยๆ จะมีดังนี้

Method	Description
- (id)initWithFrame:(CGRect)aRect	ใช้ในการสร้าง view ใหม่ขึ้นมาตาม frame ที่กำหนด
- (CGSize)sizeThatFits:(CGSize)size	คำนวนขนาดที่เหมาะสมสำหรับ subview ที่มีขนาดที่กำหนด
- (void)sizeToFit	เปลี่ยนขนาดของ view ให้เหมาะกับขนาดของ subviews ที่มีอยู่
- (void)addSubview:(UIView *)view	เพิ่ม subview เข้าไป
- (void)bringSubviewToFront:(UIView *)view	ย้าย subview มาอยู่ด้านหน้า
- (void)sendSubviewToBack:(UIView *)view	ย้าย subview ไปไว้ด้านหลัง
- (void)removeFromSuperview	นำ view ออกมาจาก superview
- (void)insertSubview:(UIView *)view atIndex:(NSInteger)index	เพิ่ม subview เข้าไปที่ลำดับที่ระบุ
- (void)insertSubview:(UIView *)view aboveSubview:(UIView *)siblingSubview	เพิ่ม subview เข้าไปด้านหน้าของ subview ที่ระบุ
- (void)insertSubview:(UIView *)view belowSubview:(UIView *)siblingSubview	เพิ่ม subview เข้าไปที่ข้างหลัง subview ที่ระบุ
- (void)exchangeSubviewAtIndex:(NSInteger)index1 withSubviewAtIndex:(NSInteger)index2	สลับลำดับของ subviews
- (BOOL)isDescendantOfView:(UIView *)view	ตรวจสอบว่า view เป็น subview ของ view ที่ระบุหรือไม่

Method	Description
- (void)addGestureRecognizer:(UIGestureRecognizer *)gestureRecognizer	ผูก gesture-recognizer ที่ระบุให้กับ view
- (void)removeGestureRecognizer:(UIGestureRecognizer *)gestureRecognizer	เอา gesture-recognizer ที่ระบุออกจาก view
- (UIView *)viewWithTag:(NSInteger)tag	หา subview ที่มี tag ตรงกับที่ระบุ
- (CGPoint)convertPoint:(CGPoint)point toView:(UIView *)view	แปลง CGPoint จากระบบพิกัดของ view ไปเป็นระบบพิกัดของ view ที่ระบุ
- (CGPoint)convertPoint:(CGPoint)point fromView:(UIView *)view	แปลง CGPoint จากระบบพิกัดของ view ที่ระบุ ไปเป็นระบบพิกัดของ view
- (CGRect)convertRect:(CGRect)rect toView:(UIView *)view	แปลง CGRect จากระบบพิกัดของ view ไปเป็นระบบพิกัดของ view ที่ระบุ
- (CGRect)convertRect:(CGRect)rect fromView:(UIView *)view	แปลง CGRect จากระบบพิกัดของ view ที่ระบุ ไปเป็นระบบพิกัดของ view

นอกจากนี้เรายังสามารถเพิ่ม animation ให้กับ UIView ได้อีกด้วย โดยเราสามารถสั่งให้ UIView มีการเปลี่ยนแปลงคุณสมบัติบางอย่าง ด้วยแสดง animations ของการเปลี่ยนแปลงที่เกิดขึ้นไปด้วย ซึ่งจะทำให้ผู้ใช้เข้าใจถึงสิ่งที่มีการเปลี่ยนแปลงได้โดยง่าย โดยคุณสมบัติของ UIView ที่เราสามารถกำหนด animation ได้นั้นจะมีดังนี้

1. frame
2. bounds
3. center
4. transform
5. alpha
6. backgroundColor
7. contentStretch

โดยการเรียกใช้งาน animation นั้นจะเรียกผ่านเมธอดต่อไปนี้

+ (void)animationWithDuration:(NSTimeInterval)duration animation:(void (^)(void))animations

+ (void)animationWithDuration:(NSTimeInterval)duration animation:(void (^)(void))animations completion:(void (^)(BOOL finished)completion

+ (void)animationWithDuration:(NSTimeInterval)duration delay:(NSTimeInterval)delay options:(UIViewAnimationOptions)options animation:(void (^)(void))animations completion:(void (^)(BOOL finished))completion

โดยเมธอดเหล่านี้จะเป็นคลาสเมธอด นั้นคือเราสามารถที่จะเรียกใช้งานได้โดยตรงโดยไม่ต้องสร้าง object ขึ้นมาก่อน สำหรับพารามิเตอร์ต่างๆจะมีความหมายดังนี้

1. (NSTimeInterval) duration คือ ระยะเวลาในการเล่น animation
2. (NSTimeInterval) delay คือ ระยะเวลาก่อนที่จะเริ่มเล่น animation
3. (UIViewAnimationOptions) option คือ ตัวเลือกในการทำงานของ animation โดยค่าที่สามารถกำหนดให้ได้จะถูกกำหนดไว้ใน enum UIViewAnimationOptions
4. (void (^)(void)) animations คือ code block สำหรับสร้าง animation
5. (void (^)(BOOL finished)) completion คือ code block ที่จะทำงานเหมือน animation ทำงานจบลง

โดยการใช้งานจะมีรูปแบบดังนี้

```
[UIView animateWithDuration:0.3
                 animations:^{
                     // animation code
                 }
                 completion:^(BOOL finished) {
                     // completion code
                 }];
```

นอกจากนี้ยังมีอีกเรื่องที่สำคัญที่เราจะต้องคำนึงถึงในการสร้าง UIView นั่นก็คือ เรื่องของการเปลี่ยนแปลงขนาดของหน้าจอ เนื่องจากว่าใน iPhone/iPad นั้นเราสามารถที่จะทำการพลิกตัวเครื่องให้เปลี่ยนการแสดงผลจากแนวตั้ง (Portrait mode) ไปเป็นการแสดงผลในแนวนอน (Landscape mode) หรือในทางกลับกันได้อีกด้วย ทำให้เวลาที่เปลี่ยนแนวการแสดงผลนี้จะทำให้พื้นที่การแสดงผลบนหน้าจอเปลี่ยนตามไปด้วย ซึ่งหากเราไม่เตรียมการรองรับการเปลี่ยนแปลงนี้ ก็จะทำให้โปรแกรมของเราดูเหมือนมีการแสดงผลที่ผิดพลาด โดยจะมีคุณสมบัติที่เกี่ยวข้องดังนี้

1. autoResizesSubview จะกำหนดให้ view ทำการปรับขนาดของ subviews เมื่อ view มีการเปลี่ยนแปลงขนาดหรือไม่ โดยจะมีค่า default เป็น YES
2. autoResizingMask ใช้กำหนดรูปแบบการเปลี่ยนแปลงของ view เมื่อมีการปรับขนาด โดยจะมีค่าที่สามารถกำหนดได้ดังนี้

a. UIViewAutoresizingNone คือ ไม่มีการปรับขนาด

b. UIViewAutoresizingFlexibleLeftMargin คือ ให้ย่อหรือขยายไปทางด้านซ้ายได้

c. UIViewAutoresizingFlexibleRightMargin คือ ให้ย่อหรือขยายไปทางด้านขวาได้

d. UIViewAutoresizingFlexibleTopMargin คือ ให้ย่อหรือขยายไปทางด้านบนได้

e. UIViewAutoresizingFlexibleBottomMargin คือ ให้ย่อหรือขยายไปทางด้านล่างได้

f. UIViewAutoresizingFlexibleWidth คือ ให้ปรับความกว้างของ view ได้

g. UIViewAutoresizingFlexibleHeight คือ ให้ปรับความสูงของ view ได้

โดยการกำหนดค่านั้น เราสามารถกำหนดได้มากกว่า 1 รูปแบบ โดยการนำค่าที่ต้องการมาทำ bitwise OR operator กัน และกำหนดให้กับ autoResizingMask ตัวอย่างเช่น

```
view.autoresizingMask = UIViewAutoresizingFlexible-
Width | UIViewAutoresizingFlexibleHeight | UIViewAu-
toresizingFlexibleTopMargin | UIViewAutoresizing-
FlexibleLeftMargin;
```

อันนี้จะเป็นการกำหนดให้ view ทำการเปลี่ยนแปลงความกว้าง ความสูง และให้ยืดหรือหดในทางด้านซ้าย และด้านบนได้

การสร้าง UIView จะทำได้โดยใช้โค๊ดดังนี้

```
UIView *myView = [[UIView alloc] init];
```

เป็นการสร้าง view ขึ้นมาโดยยังไม่กำหนดตำแหน่งและขนาดให้ ซึ่งเราจะต้องมาทำการกำหนดให้ในภายหลังก่อนนำขึ้นมาแสดงผล

```
UIView *myView = [[UIView alloc]
initWithFrame:CGRectMake(0, 0, 100, 100)];
```

อันนี้จะเป็นการสร้าง view ขึ้นมาในตำแหน่งและขนาดที่กำหนด

ซึ่งเราสามารถใช้โค๊ดอันไหนก็ได้ในการสร้าง view ขึ้นมา ซึ่งจะให้ผลเหมือนกัน

หลังจากนั้นก็ทำการกำหนดคุณสมบัติต่างๆที่ต้องการให้กับ view ที่สร้างขึ้นมา โดยจะมีรูปแบบดังนี้

```
myView.frame = CGRectMake(0, 0, 100, 100);
myView.backgroundColor = [UIColor blueColor];
myView.autoresizesSubviews = YES;
myView.autoresizingMask = UIViewAutoresizingFlexible-
Width | UIViewAutoresizingFlexibleHeight | UIViewAu-
toresizingFlexibleLeftMargin | UIViewAutoresizing-
FlexibleRightMargin | UIViewAutoresizingFlexibleTop-
Margin | UIViewAutoresizingFlexibleBottomMargin;
```

เมื่อกำหนดคุณสมบัติที่ต้องการเรียบร้อยแล้วก็ทำการนำ view ที่สร้างขึ้นมาใส่ลงไปใน superview ที่เราต้องการ ซึ่งจะใช้โค๊ดดังนี้

```
[mySuperview addSubview:myView];
```

BooBookHome

UIVIEWCONTROLLER

1. เป็นส่วน **control** ในการแบบโปรแกรมแบบ **MVC**
2. ใช้สำหรับควบคุมการทำงานของ **view** ในการนำข้อมูลขึ้นแสดง และรับ **input** เข้ามาในโปรแกรม
3. โดยปกติเราจะสร้างคลาสลูกของคลาสนี้เพื่อใช้งาน

คลาสถัดมาที่เราจะพูดถึงก็คือคลาส UIViewController ซึ่งดูจากชื่อ ก็คงจะคาดเดากันได้ว่าคลาสนี้น่าจะเกี่ยวข้องกับคลาส UIView ที่เรากล่าวถึงไปแล้ว คลาส UIViewController นี้จะทำหน้าที่ควบคุมคลาส UIView อีกที โดยจะถูกใช้ในการจัดเตรียมข้อมูลต่างๆ ที่จะนำขึ้นมาแสดงบน UIView และรับ input จากผู้ใช้ที่ป้อนเข้ามาผ่านทาง UIView เพื่อนำมาประมวลผลภายในโปรแกรมต่อไป ซึ่งตรงนี้ก็จะเป็นรูปแบบในการออกแบบโปรแกรมในลักษณะของ MVC (Model-View-Control)

สำหรับผู้ที่ยังไม่เคยศึกษาเรื่อง MVC มาก่อน ก็จะขออธิบายคร่าวๆว่า MVC จะแบ่งการทำงานของโปรแกรมออกเป็น 3 ส่วนหลักๆ นั้นคือ

1. Model สำหรับใช้เก็บข้อมูล

2. View สำหรับใช้ในการแสดงข้อมูล และส่วนติดต่อกับผู้ใช้

3. Control สำหรับเชื่อมการทำงานระหว่าง Model และ View เข้าด้วยกัน

โดยปกติแล้วเราจะไม่ทำการสร้าง object จากคลาส UIViewController โดยตรง แต่เราจะทำการ subclass คลาสนี้ขึ้นมาแทน จากนี้ไปเราจะขอเรียกคลาสที่ขยายมาจากคลาส UIViewController ว่าเป็น ViewController เพื่อให้ไม่สับสนกับคลาส UIViewController

โดยทั่วไปแล้วในแต่ละหน้าของโปรแกรม เราจะทำการสร้างคลาส ViewController ขึ้นมาเพื่อควบคุมการทำงานของหน้านั้นๆ และถ้าโปรแกรมมีหลายหน้า ก็จะทำให้มีคลาส ViewController หลายคลาสด้วยเช่นกัน

ก่อนที่เราจะพูดต่อไปถึงรายละเอียดเกี่ยวกับการสร้างคลาส ViewController นั้น เราจะมาพูดถึงคุณสมบัติและเมธอดต่างๆ ที่คลาส UIViewController ได้เตรียมไว้ให้แล้ว

โดยคุณสมบัติที่สำคัญๆจะมีดังนี้

Property	Type	Description
view	UIView	view ที่ viewController นี้ควบคุมอยู่
wantsFullScreenLayout	BOOL	กำหนดให้แสดงผลเต็มหน้าจอ
interfaceOrientation	UIInterfaceOrientation	orientation ของโปรแกรม
modalTransitionStyle	UIModelTransitionStyle	กำหนด transition ของ view สำหรับการนำขึ้นแสดงแบบ modal view
modalPresentationStyle	UIModalPresentationStyle	กำหนดรูปแบบการแสดงผลของ view สำหรับการนำขึ้นแสดงแบบ modal view
parentViewController	UIViewController	viewController ที่เก็บ viewController นี้ไว้ หรือ nil
navigationController	UINavigationController	navigationViewController ที่เก็บ viewController นี้ไว้ (เป็น child ลำดับใดก็ได้)
splitViewController	UISplitViewController	splitViewController ที่เก็บ viewController นี้ไว้
tabBarController	UITabBarController	tabBarController ที่เก็บ viewController นี้ไว้
childViewControllers	NSArray	array ของ viewController ที่ถูกเก็บเอาไว้ใน viewController นี้
hidesBottomBarWhenPushed	BOOL	กำหนดให้ซ่อน toolbar ที่อยู่ด้านล่างของหน้าจอ เมื่อ viewController นี้ถูกนำขึ้นมาแสดง
navigationItem	UINavigationItem	naviationItem ของ viewController นี้ สำหรับแสดงใน navigationBar
toolbarItems	NSArray	array ของ UIBarButtonItem ที่แสดงใน navigationBar
tabBarItem	UITabBarItem	tabBarItem ของ viewController นี้ สำหรับแสดงผลใน tabBar

ในส่วนของเมธอดหลักๆก็จะมีดังนี้

Method	Description
- (void)presentViewController:(UIViewController *)viewController animated:(BOOL)flag completion:(void (^)(void))completion	นำ viewController ขึ้นแสดง โดยกำหนดให้มีการแสดง animation ด้วยหรือไม่ และเมื่อทำงานเสร็จแล้วให้ทำงานใน completion block
- (void)dismissViewControllerAnimated:(BOOL)animated completion:(void (^)(void))completion	นำ viewController ที่แสดงอยู่ออก โดยกำหนดให้มีการแสดง animation ด้วยหรือไม่ และเมื่อทำงานเสร็จแล้วก็จะทำงานใน completion block

สำหรับการสร้างคลาส ViewController นั้น สิ่งสำคัญที่เราจะต้องทำหลักๆ ก็คือ การ implement เมธอดต่างๆที่เกี่ยวข้องกับการแสดงผลของ View ซึ่งได้แก่

1. - (id)init เป็นเมธอดสำหรับใช้ในการสร้าง viewController ขึ้นมา โดยส่วนใหญ่เราจะทำการสร้างและกำหนดค่าให้กับตัวแปรต่างๆ พร้อมทั้งทำการสร้าง view ขึ้นมาเพื่อใช้ในการแสดงผล

2. - (void)viewWillAppear:(BOOL)animated เมธอดนี้จะถูกเรียกเมื่อ view ของ viewController นี้กำลังจะถูกนำมาแสดงบนหน้าจอ

3. - (void)viewDidAppear:(BOOL)animated เมธอดนี้จะถูกเรียกเมื่อ view ของ viewController ถูกนำขึ้นแสดงบนหน้าจอแล้ว

4. - (void)viewWillDisappear:(BOOL)animated เมธอดนี้จะถูกเรียกเมื่อ view ของ viewController นี้กำลังจะถูกนำออกจากการแสดงผลบนหน้าจอ

5. - (void)viewDidDisappear:(BOOL)animated เมธอดนี้จะถูกเรียกเมื่อ view ของ viewController นี้ถูกนำออกจากการแสดงผลบนหน้าจอแล้ว

6. - (void)viewDidLoad เมธอดนี้จะถูกเรียกเมื่อ view ถูกโหลดเข้าสู่ memory แล้ว

7. - (void)viewWillUnload เมธอดนี้จะถูกเรียกเมื่อ view กำลังจะถูก release และคืนหน่วยความจำให้กับระบบ

8. - (void)viewDidUnload เมธอดนี้จะถูกเรียกเมื่อ view ถูก release และคืนหน่วยความจำให้กับระบบแล้ว

9. - (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation เมธอดนี้จะถูกเรียกเพื่อตรวจสอบว่า view ของ viewController นี้จะถูกใช้บน orientation ที่ระบุได้หรือไม่

10. - (void)dealloc เมธอดนี้จะถูกเรียกเมื่อ viewController ถูกทำลายเพื่อให้เราทำการคืนหน่วยความจำและทรัพยากรของระบบต่างๆ

สำหรับตัวอย่างในการสร้างคลาส ViewController นั้น เราจะได้เห็นกันไปแล้วจากตัวอย่างโปรแกรมต่างๆ ซึ่งล้วนแล้วแต่เป็นการสร้างคลาส ViewController ขึ้นมาใหม่เพื่อควบคุมการทำงานของโปรแกรมทั้งนั้น ซึ่งตรงนี้ก็อาจจะทำให้เกิดข้อสงสัยได้ว่า ถ้าหากเรามีคลาส ViewController มากกว่า 1 คลาส แล้วเราจะทำการเปลี่ยน ViewController และเปลี่ยนหน้า View ยังไง ซึ่งตรงจุดนี้จะพูดถึงอีกทีในภายหลัง

BooBook Home

UIWindow

UIWINDOW

1. ทำหน้าที่เป็นหน้าต่างของโปรแกรมสำหรับนำ view ต่างมาประกอบขึ้นแสดงผล
2. โดยมากทุกโปรแกรมจะมี object ของ UIWindow เพียงอันเดียวเท่านั้น

คลาสสุดท้ายที่เราจะนำมาพูดถึงกันในบทนี้ก็คือ คลาส UIWindow ซึ่งคลาสนี้จะทำหน้าที่เป็นหน้าต่างสำหรับแสดงโปรแกรมของเรานั้นเองซึ่งฟังดูแล้วอาจจะดูเหมือนว่าคลาสนี้จะคล้ายกับคลาส UIView ที่เราพูดถึงไปแล้ว ซึ่งก็เป็นเช่นนั้น เพราะว่า UIWindow นั้นเป็นคลาสที่ขยายมาจากคลาส UIView นั้นเอง แต่คลาส UIWindow จะทำหน้าที่แตกต่างจาก UIView นั่นคือ UIWindow จะทำหน้าที่เป็นหน้าต่างแสดงผลของโปรแกรม โดยมี UIView เป็นสิ่งที่แสดงผลอยู่ภายในหน้าต่างนี้ ซึ่งโดยส่วนมากทุกโปรแกรมจะมี object จากคลาส UIWindow เพียงตัวเดียวเท่านั้น นั้นคือหน้าต่างหลักของโปรแกรม

คลาสนี้จะเป็นสำคัญที่จะต้องใช้งานในแทบจะทุกโปรแกรม แต่ในขณะเดียวกัน ก็เป็นคลาสที่เราแทบไม่ต้องเข้าไปจัดการอะไรกับมันเลยอีกเช่นกัน โดยทั่วไปแล้วสิ่งที่เราต้องทำก็คือการสร้าง view ของโปรแกรมขึ้นมา และนำขึ้นมาแสดงผลบน UIWindow ซึ่งเราได้พูดถึงไปแล้วในช่วงต้นของบทที่ 2

```
UIWindow *myWindow = [[[UIWindow alloc] initWithFrame:[[UIScreen
mainScreen] bounds]] autorelease];
myWindow.backgroundColor = [UIColor whiteColor];
[myWindow addSubview:myView];
[myWindow makeKeyAndVisible];
```

สรุปสิ่งที่กล่าวถึงในบทนี้

1. แนะนำ UIKit Framework
2. คลาส UIView
3. คลาส UIViewController
4. คลาส UIWindow

ในบทนี้เราได้พูดถึงคลาสพื้นฐาน 3 คลาสด้วยกัน คือ UIView, UIViewController และ UIWindow ซึ่งคลาสเหล่านี้จะเป็นองค์ประกอบพื้นฐานของทุกโปรแกรมที่ต้องใช้ส่วนแสดงผลสำหรับติดต่อกับผู้ใช้

โดยส่วนใหญ่แล้วเราอาจจะไม่ได้ใช้งานคลาส UIView โดยตรง เพราะ UIView นั้นเป็นแค่พื้นที่สี่เหลี่ยมว่างๆเท่านั้น ซึ่งทำให้การนำมาใช้งานไม่สะดวกนัก เราจะทำงานกับคลาสลูกที่ขยายมาจาก UIView มากกว่า อย่างเช่น UIButton, UITextField, UILabel เป็นต้น ซึ่งคลาสลูกเหล่านี้จะถูกออกแบบมาเพื่อแสดงผลและรับข้อมูลจากผู้ใช้ในรูปแบบต่างๆที่เหมาะสมไว้อยู่แล้ว เราจะมากล่าวถึงคลาสต่างๆเหล่านี้กันในภายหลัง

ในส่วนของคลาส UIViewController นั้น สิ่งที่เราจะนำมาใช้นั้นจะเป็นคลาสที่เราขยายออกมาเพื่อใช้สำหรับควบคุมการทำงานของโปรแกรมเรา

สำหรับคลาส UIWindow เราแทบจะไม่ต้องเข้าไปจัดการใดๆเลย เพราะคลาสนี้จะใช้สำหรับเป็นหน้าต่างสำหรับให้โปรแกรมแสดงผลเท่านั้น

ซึ่งที่พูดมานี้เป็นเพียงองค์ประกอบเล็กๆของ iOS เท่านั้น ในบทต่อๆไป เราจะมาพูดถึงองค์ประกอบอื่นๆ เริ่มจากองค์ประกอบง่ายๆที่จะถูกเรียกใช้งานบ่อย