# Custom Window in IOS 8

The app's window is the top of the view hierarchy. It is an instance of UIWindow (or a subclass of UIWindow, which is a subclass of UIView. Our app should have only one main window. It is created at launch time and is never destroyed or replaced. It occupies the entire screen and forms the background, and is the superview of all visible views on the interface. The window should fill the device's screen. Therefore, its size and position must be identical to the size and position of the screen.

It is extremely unlikely that we would ever need to subclass UIWindow. In this tutorial, we will subclass UIWindow, just for the fun of it and to learn a few things about how a window is created and customized. We will consider two ways of achieving our objective: using a storyboard and doing everything entirely in code.

## 1. Custom Window in Storyboard

We create an Xcode project using the single view application template and choosing Swift as the language. The application will start with a main storyboard and two files: AppDelegate.swift and ViewController.swift. We need a UIWindow subclass. So we add a Cocoa Touch class file, name it MyWindow and choose it to be subclass of UIWindow. We will not add any code to the class:

```
import UIKit

class MyWindow: UIWindow {

   }
```

In an app with a storyboard, there is a property: `var window: UIWindow?` In the AppDelegate class. When the application is started, an instance of UIWindow is automatically created and assigned to this property. We don't have to do anything. Since we want to use our custom window MyWindow, we need to add some code to the AppDelegate class:

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window:UIWindow = {
```

```
//CGRect created using bounds of mainscreen
   let frame: CGRect = UIScreen.mainScreen().bounds
//MyWindow is given a frame with a margin of 50 all around:
   let win = MyWindow(frame: CGRect(x: 50,y: 50,width:
             frame.width - 100,height: frame.height - 100))
   return win
 }()

    func application(application: UIApplication,
              didFinishLaunchingWithOptions launchOptions:
                          NSDictionary?) -> Bool {
   //This is necessary to create the instance window
     self.window.backgroundColor = UIColor.whiteColor()
     return true
  }
}
```

A closure is used in the getter for the window variable. This customizes the MyWindow instance. Instead of the window frame covering the entire screen, we leave a margin of 50 all around.

To give our app some functionality, we add some code to ViewController.swift file:

```
override func viewDidAppear(animated: Bool)  {
    super.viewDidAppear(animated)
    //The main window is inset in the custom window
    let deltaW = self.view.window!.frame.width - 40
    let deltaH = self.view.window!.frame.height - 40
    let frame: CGRect = CGRectMake(20, 20, deltaW, deltaH)
    self.view.frame = frame
    self.view.backgroundColor = UIColor.blueColor()
    self.view.window!.backgroundColor = UIColor.redColor()

    println(self.view.window)
    println((UIApplication.sharedApplication().delegate as
                                        AppDelegate).window)
    println(UIApplication.sharedApplication().delegate!
                                                .window)
    println(UIApplication.sharedApplication().keyWindow)
  }
```

We have added code to `override func viewDidAppear(animated: Bool)` to make the main view a little smaller than the window so that we get a glimpse of the

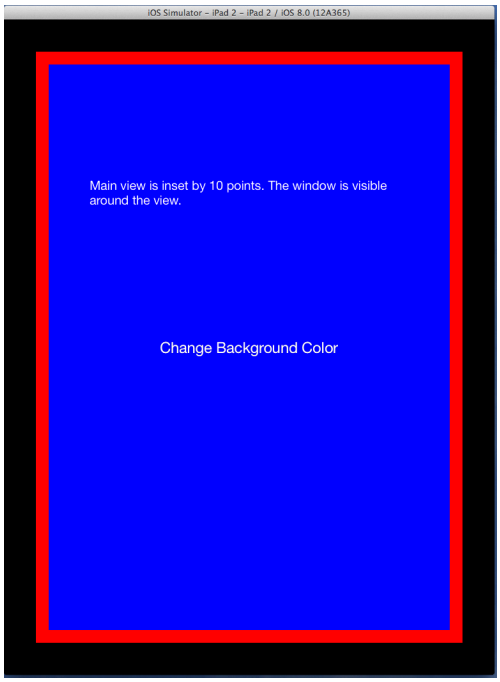
window behind the view. We have also set the background color of the window red and the background color of the main view blue. The window can be referenced a number of ways in the code. We have added code to print description of the window obtained from different ways of refering to the window. All the desription show instance of MyWindow is the window in the app.

We have also added a button on the interface. Pressing the button will change the background colors:

```
@IBAction func buttonPressed(sender : AnyObject) {

  //When button is pressed the balckground colors of the
  // view  and the window are changed
  if self.view.backgroundColor == UIColor.blueColor()
    {
      self.view.backgroundColor=UIColor.redColor()
      self.view.window!.backgroundColor=UIColor.blueColor()
    }
    else
    {
      self.view.backgroundColor=UIColor.blueColor()
      self.view.window!.backgroundColor=UIColor.redColor()
    }
 }
```

Running the application, we will see the following in the simulator:



Observe that the window does not cover the entire screen. So we see the black screen behind the window. The main view is smaller than the window and we can see window with red background behind the view. Clicking the button will exchange the window and view background colors.

## 2. Custom Window in Code

Now we will not use Xcode interface so we have to create everything in code. The template Empty Application has been removed from Xcode 6. So we have to start with a single view application and remove the view controller file, the storyboard and all reference to the storyboard. This would leave us with an application with only AppDelegate file.

We start by adding a Cocoa Touch class file. We make it a subclass of UIView and name it MyView. Later on we will assign an instance of MyView as the main view.

```
import UIkit

class MyView: UIView {

    //Color constants to be used as background colors
    let color1 =  UIColor(red: 0.8, green: 0.8, blue: 0.6,
                                                alpha: 1.0)
    let color2 =  UIColor(red: 0.5, green: 0.4, blue: 0.3,
                                                alpha: 1.0)

    override init(frame: CGRect) {
        super.init(frame: CGRectZero)
        self.frame = frame
        self.backgroundColor = color1

 //An instance of UIButton of width 100 and height 50 is
 //created and place at the center of the view

     var button = UIButton(frame: CGRect(x: (frame.width –
                  100)/2,y: (frame.height - 50)/2,
                   width: 100,height: 50))
     button.setTitle("Click Me", forState:
                                    UIControlState.Normal)
     //Button color set red for normal state
     button.setTitleColor(UIColor.redColor(), forState:
                                    UIControlState.Normal)
```

```
//Button color is set blue for highlighted state
    button.setTitleColor(UIColor.blueColor(), forState:
                            UIControlState.Highlighted)
//func pressMe is set as the target for TouchUpInside event
    button.addTarget(self, action: Selector("pressMe:"),
        forControlEvents: UIControlEvents.TouchUpInside)
//The button is added as the subview of MyView
    self.addSubview(button)
 }

 required init(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
   }

   //Method run when butto is pressed
 func pressMe(sender: AnyObject){

   //Background color changed on button click
    if self.backgroundColor == color1
       {
         self.window!.backgroundColor = color1
         self.backgroundColor = color2
       }
    else
       {
         self.window!.backgroundColor = color2
         self.backgroundColor = color1
       }
  }

}
```

We have place a button and hooked up a method to run when the button is clicked. Each button click will exchange the window and view background colors.

Then we create a UIWindow subclass named MyWindow:

```
import UIKit

class MyWindow: UIWindow {

        override init(frame: CGRect) {
        super.init(frame : CGRectZero)
        self.frame = frame

//Adds a margin around the view
```

```
    let deltaW = frame.width - 40
    let deltaH = frame.height - 40
    var v = MyView(frame: CGRect(x: 20,y:  20,width:
                                    deltaW,height: deltaH))
//Create an instance of UIViewController
    var viewController = UIViewController( )
//Adds v as a subview of viewController view
    viewController.view.addSubview(v)
//sets viewController as the rootViewController
    self.rootViewController = viewController
   }
 required init(coder aDecoder: NSCoder) {
            fatalError("init(coder:) has not been
                    implemented")
    }
 }
```

We need to create an instance of UIViewController, add MyView as a subview of the main view and make the viewController the rootViewController of the window.

Finally we have to set an instance of MyWindow to the window variable in AppDelegate. We make the window frame smaller than the screen to add a margin all around.

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window:MyWindow!

    func application(application: UIApplication,
              didFinishLaunchingWithOptions launchOptions:
                                NSDictionary?) -> Bool {
        let frame: CGRect = UIScreen.mainScreen().bounds
        window = MyWindow(frame: CGRect(x: 50,y: 50,width:
          frame.width - 100,height: frame.height - 100))
        self.window.backgroundColor = UIColor.whiteColor()
        self.window.makeKeyAndVisible()
        return true
     }
}
```