Aufgabe 6b

Anforderungen:

- Klasse ShadowView erbt von UIView
- Draw − Methode überschreiben:
  - Zufällige Anzahl an Rechtecke zeichnen
  - Zufällige Größe, Position, Farbe
  - Rechtecke passen komplett in Zeichenebene
  - Jedes zweite Rechteck mit Schatten
- Hilfsfunktionen für wiederholende Schritte
- Implementierung in Swift

ViewController.swift

```
import UIKit

class ViewController: UIViewController {

   override func viewDidLoad() {
      super.viewDidLoad()

      let shadowView = ShadowView(frame: CGRect(x:0, y:0,width: self.view.bounds.width,height:
               self.view.bounds.height), numberOfRectangles:Int(arc4random_uniform(50)+1))
      self.view = shadowView

   }
}
```

ShadowView.swift (1)

```
import UIKit

class ShadowView : UIView {

   var numberOfRectangles: Int

   init(frame: CGRect, numberOfRectangles:Int) {

      self.numberOfRectangles = numberOfRectangles;
      super.init(frame: frame)

   }

   required init?(coder aDecoder: NSCoder) {

      fatalError("init(coder:) has not been implemented“)
   }
...
```

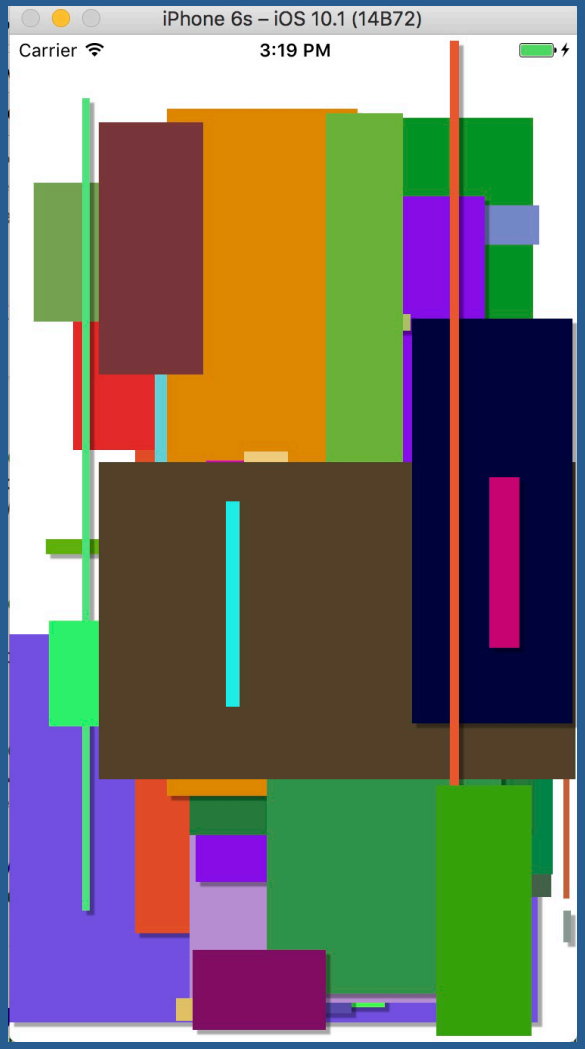
ShadowView.swift (2)

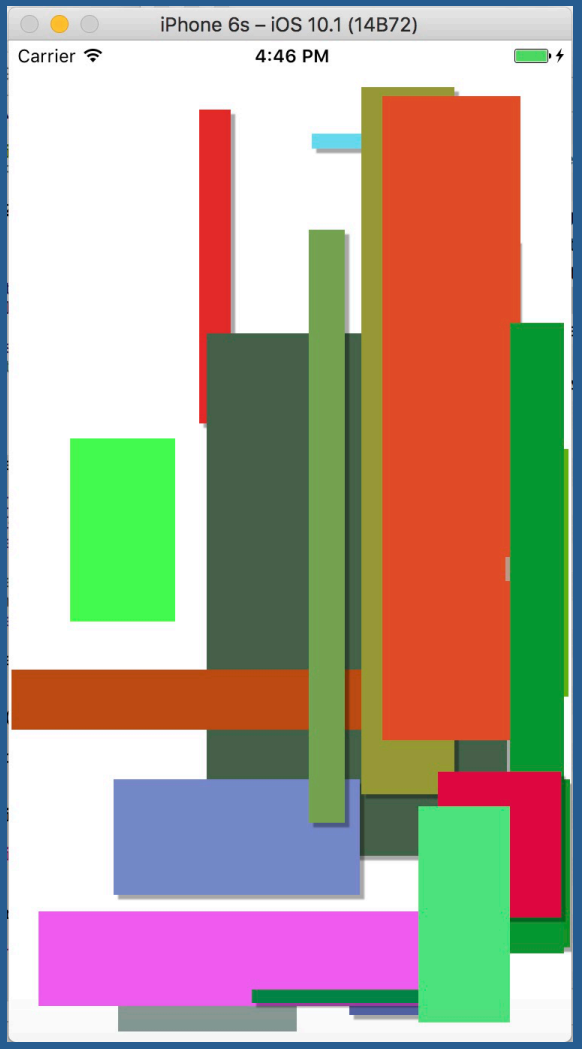
```
override func draw(_ rect: CGRect) {

    let context = UIGraphicsGetCurrentContext()

    //Hintergrund weiß setzen
    context?.setFillColor(CGColor.init(colorSpace: CGColorSpaceCreateDeviceRGB(), components:
        [1.0, 1.0, 1.0, 1.0])!)
    context?.fill(rect)

    while(numberOfRectangles > 0){

        //Speichert den momentanen Context
        context?.saveGState()

        //Jedes zweite Rechteck mit Schatten
        if(numberOfRectangles%2==0){

            context?.setShadow(offset: CGSize(width: 3, height: 3), blur: 1.0)

        }

        context!.setFillColor(getRandomColor())
        context!.fill(createRectangle(rect: rect))

        context?.restoreGState()
        numberOfRectangles -= 1
    }

}
```

ShadowView.swift (3)

```
func createRectangle(rect:CGRect) -> CGRect{

   var rectangle: CGRect
   var position: CGPoint
   var size: CGSize

   position = getRandomPosition(rect: rect)
   size = getRandomSize(rect: rect, position: position)
   rectangle = CGRect(x: position.x, y: position.y, width: size.width, height: size.height)

   return rectangle
}

func getRandomColor() -> CGColor{

   return CGColor(colorSpace: CGColorSpaceCreateDeviceRGB(), components: [CGFloat(drand48()),
              CGFloat(drand48()), CGFloat(drand48()), 1.0])!
}

func getRandomPosition(rect: CGRect) -> CGPoint{

   return CGPoint.init(x: CGFloat(arc4random_uniform(UInt32(rect.width))), y:
              CGFloat(arc4random_uniform(UInt32(rect.height))))
}

func getRandomSize(rect: CGRect, position: CGPoint) -> CGSize{

   return CGSize.init(width: CGFloat(arc4random_uniform(UInt32(rect.width - position.x))), height:
              CGFloat(arc4random_uniform(UInt32(rect.height - position.y))))
}
```

Ergebnis


iPhone 6s – iOS 10.1 (14B72)
Carrier
3:19 PM


iPhone 6s – iOS 10.1 (14B72)
Carrier
4:46 PM