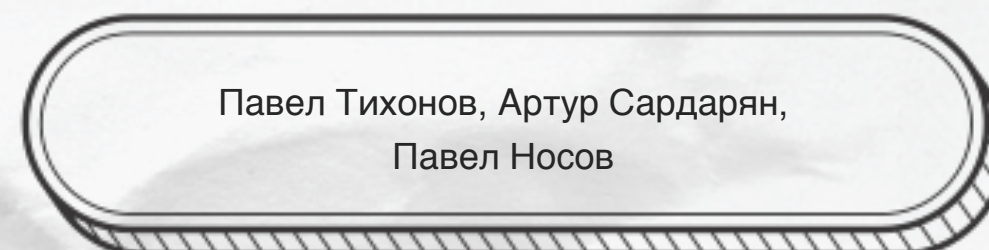


Лекция №7

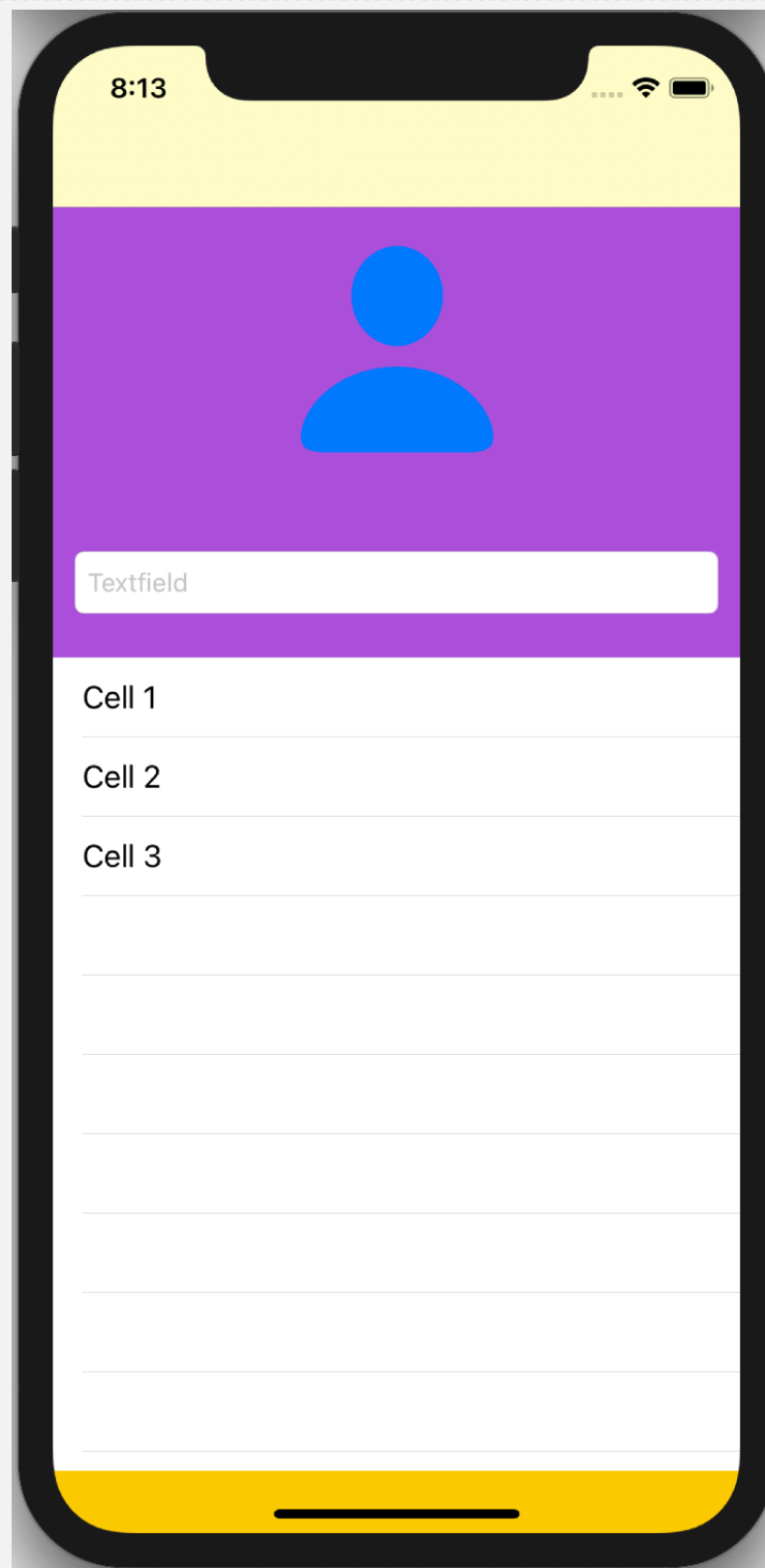


Организационная часть

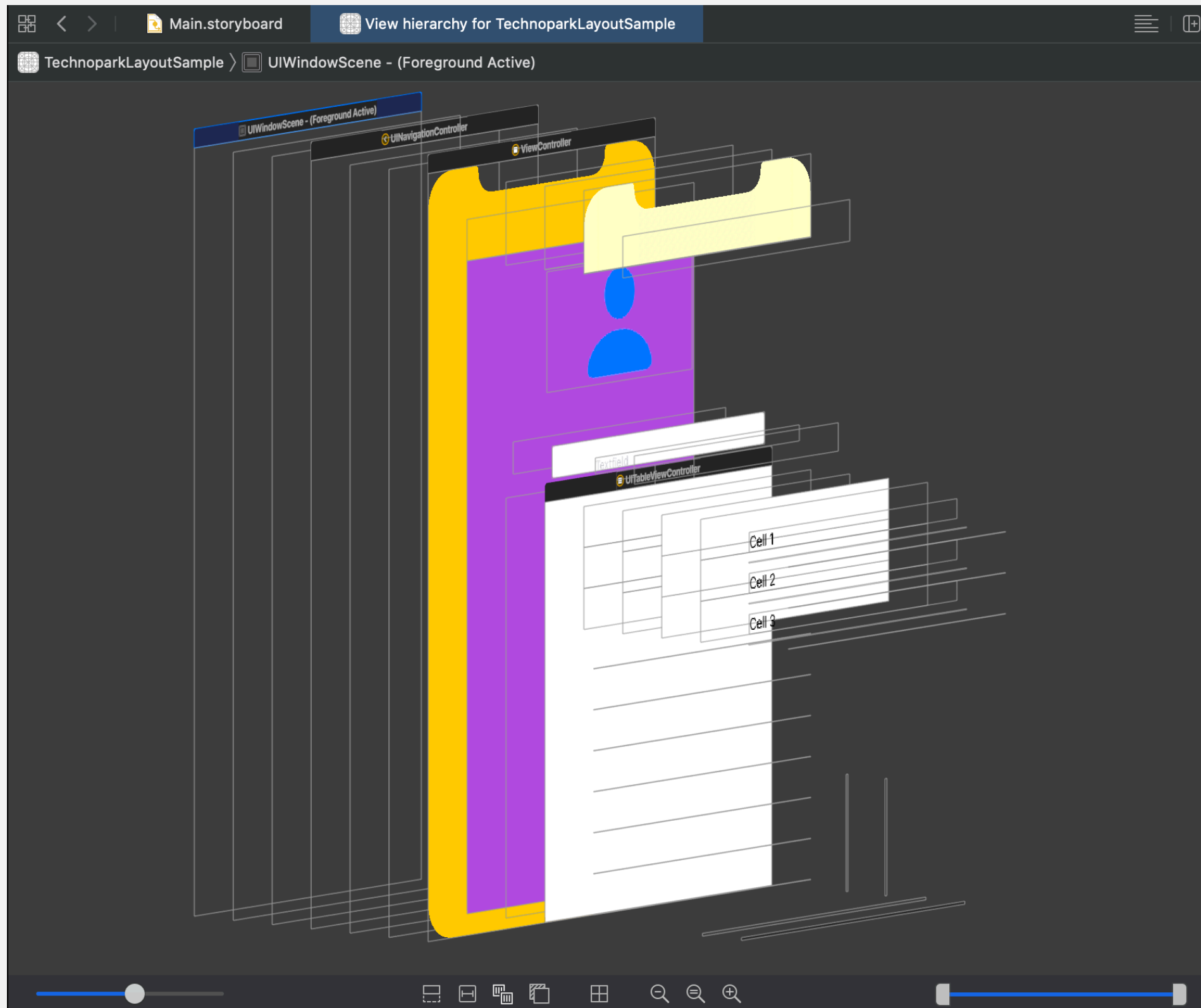


- Не нарушая традиций — отметить
- Чем займемся:
 - Рассмотрим базовые элементы UIKit
 - Поговорим о подходах для создания интерфейса
 - Узнаем про responder chain
 - Распознавание жестов
- Оставить отзыв (после занятия)

Интерфейс приложения

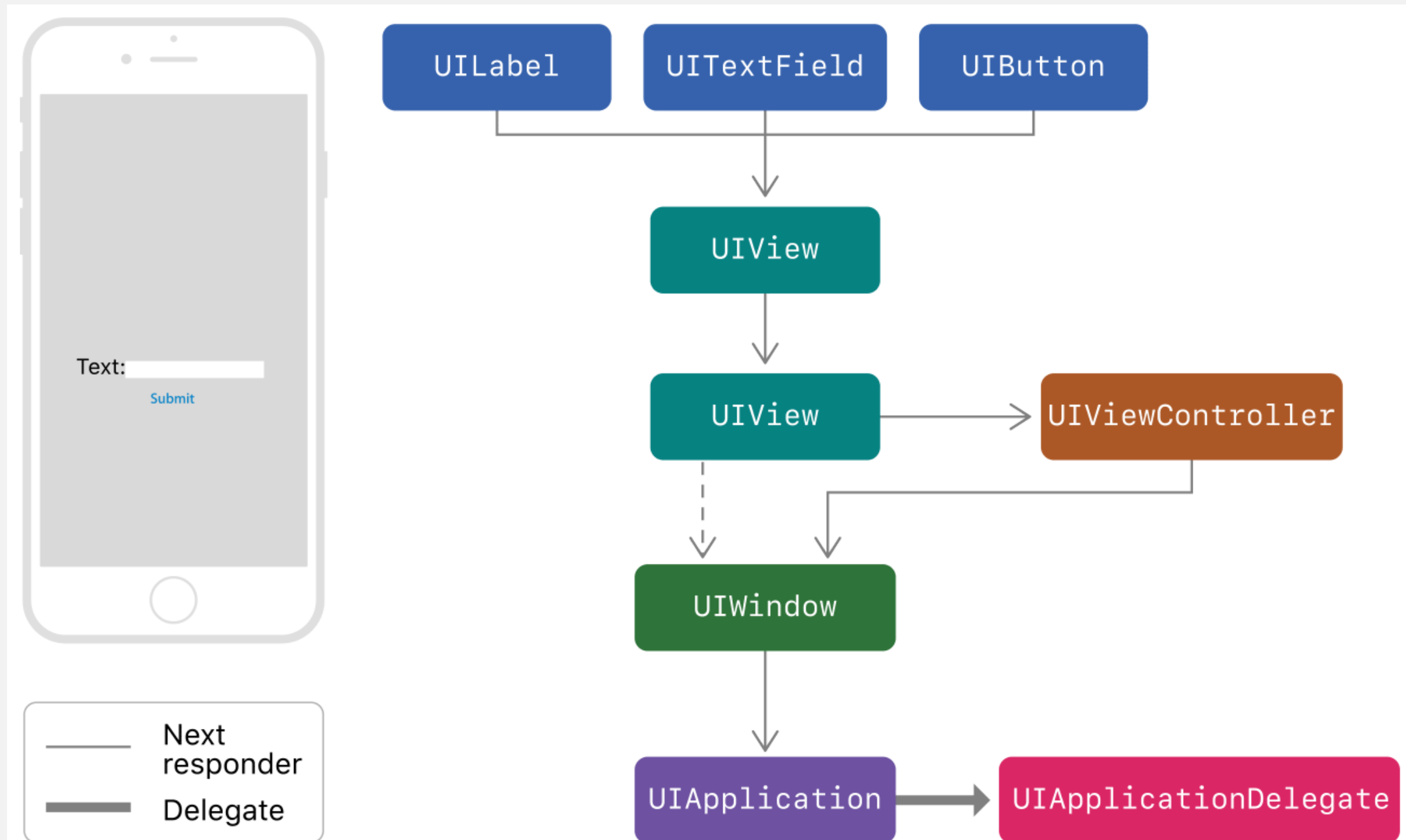


Интерфейс приложения в разрезе



- Базовый класс UIKit («кирпичи» из которых строится интерфейс)
- Потомок UIResponder'a
 - обрабатывает события:
 - касания (touches)
 - движения (motions), например, встряхивание
- Отображает элементы интерфейса
- Может настраивать размеры и позиции своих subview
- Могут добавлять распознавателей жестов (gesture recognizers) для обработки типичных жестов

Responder Chain



https://developer.apple.com/documentation/uikit/understanding_event_handling_responders_and_the_responder_chain

UIView (потомки)

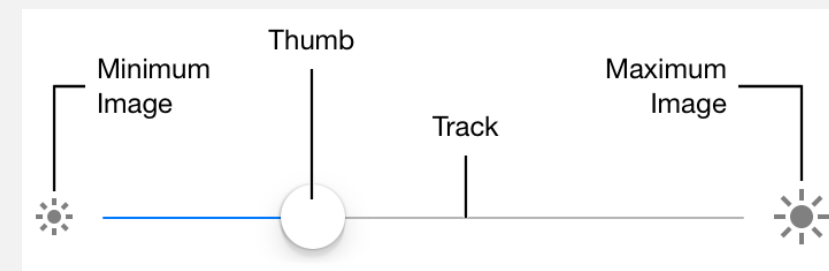
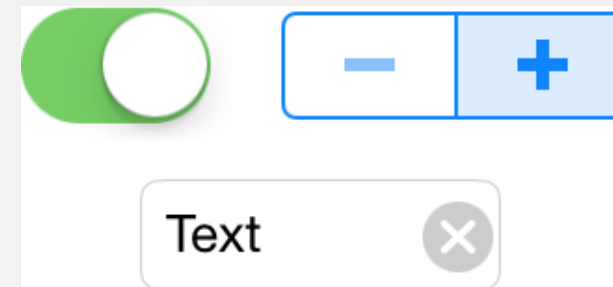


- UIControl
 - UIButton, UITextField и др.
- UIWindow
 - редко нужен в iOS, но может пригодиться, если нужно что-то показать поверх status bar'a
- UILabel
- списки
 - UIScrollView,
 - UITableView,
 - UICollectionView,
 - UITextView
- UIImageView
- MKMapView
- WKWebView

UIControl



- UIControl
 - UIButton
 - UITextField
 - UISwitch
 - UISegmentedControl
 - UISlider
 - UIProgress
- Добавляет к UIView
 - механизм target/action
 - протокол (UITextFieldDelegate и пр.)



<https://developer.apple.com/reference/uikit/uicontrol>
<https://developer.apple.com/reference/uikit/uislider>

UIView (основные понятия)



- один superview
- много subview
- addSubview: (у родительской view)
 - добавляет к нашему view subview
- removeFromSuperview
 - удаляет наш view из его superview

- анимации

```
• @available(iOS 4.0, *)  
  open class func animate(withDuration duration: TimeInterval, delay: TimeInterval,  
    options: UIView.AnimationOptions = [], animations: @escaping () -> Void,  
    completion: ((Bool) -> Void)? = nil)
```

UIView (полезные свойства)



- CGFloat alpha
 - прозрачность (от 0 до 1)
- Bool isOpaque (непрозрачный)
 - true/false
- Bool isHidden (невидимый)
- Bool userInteractionEnabled (отключенный)
- Bool clipsToBounds (обрезать по границам)
- Bool translatesAutoresizingMaskIntoConstraints (при использовании auto layout кодом)

UIView (bounds / frame)



- bounds — положение и размер в собственной системе координат
- frame — положение и размер в системе координат родительской вью
- Если нужно спозиционировать view, лучше менять center, а frame пусть считается сам

- Вручную
 - touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)touchesEnded:withEvent:
 - touchesMoved...
 - touchesCancelled...
- Использование UIGestureRecognizer (точнее, его ПОТОМКОВ)
 - КОДОМ
 - с помощью Interface Builder



Пример распознавания жеста:



```
.....  
  
let gestureRecognizer =  
    UITapGestureRecognizer(target: self, action: #selector(viewTapped(_:)))  
view.addGestureRecognizer(gestureRecognizer)  
  
.....  
  
@objc func viewTapped(_ sender: UITapGestureRecognizer) {  
    print("viewTapped: \(gestureRecognizer.view)")  
}
```

UIGestureRecognizer — основные виды



UIGestureRecognizer — основные виды

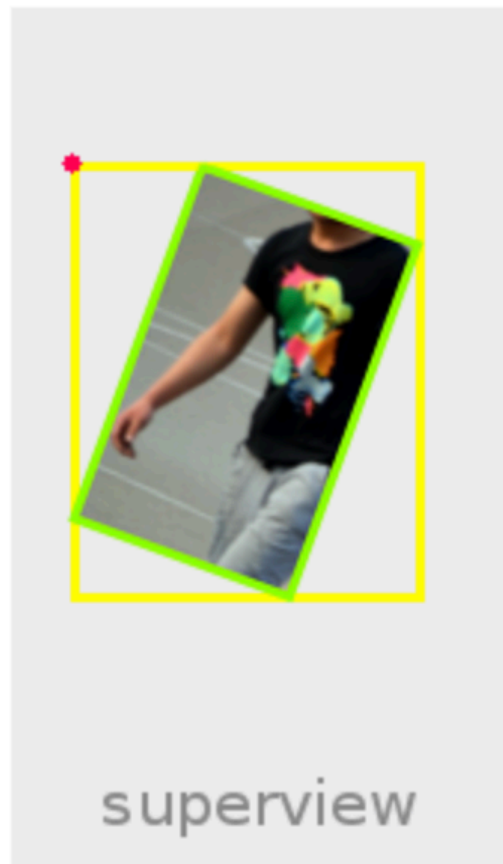


- tap
- swipe
- pan
- long press
- pinch
- rotation

UIView (bounds / frame)



Frame



Bounds



- Кастомизация
 - `init(frame:)` (могут отличаться у наследников UIView, например, у `UITableViewCell`)
 - конструктор используется только для создания view кодом
 - `init(coder:)`
 - для view, созданного через Interface Builder
 - `draw(_:)`
 - если нужно во view что-то нарисовать

Кастомные UIView: -drawRect:



- В UIView можно рисовать:

- **UIBezierPath**

```
let path = UIBezierPath()
```

- moveToPoint:,
- addLineToPoint

- **Core Graphics**

- контекст CGContextGetCurrentContext()
- функции для рисования

- Используется CPU, а не GPU



Пример drawRect:



```
override fun draw(_ rect: CGRect) {
    // получаем ссылку на контекст
    guard let context = UIGraphicsGetCurrentContext() else {
        return
    }

    // очищаем контекст
    context.clear(rect)

    // выставляем цвет
    context.setFill-color(UIColor.yellow.cgColor)

    // заполняем rect
    context.fill(rect)
}
```

```
override fun draw(_ rect: CGRect) {
    let origin: CGPoint = .init(x: 50, y: 50)
    let size: CGSize = .init(width: 200, height: 400)
    let rect = CGRect(origin: origin, size: size)

    UIColor.green.setFill()
    let path = UIBezierPath(roundedRect: rect, cornerRadius: 10)
    path.fill()
}
```

```
override fun draw(_ rect: CGRect) {
    let bezierPath = UIBezierPath()

    UIColor.green.setFill()

    bezierPath.move(to: CGPoint(x: 10, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 10))
    bezierPath.addLine(to: CGPoint(x: 160, y: 160))
    bezierPath.addLine(to: CGPoint(x: 10, y: 160))
    bezierPath.close()

    bezierPath.fill()
}
```

Frame-based верстка



- До iOS 6 (2012) верстка делалась только через изменение frame у view
- С помощью autoresizingMask задавались правила изменения frame, если родительские размеры изменятся



- Система линейных неравенств
- Constraints (ограничения)
 - основной класс NSLayoutConstraint
- Можно задать разными способами:
 - Interface Builder
 - Код (NSLayoutConstraint)
 - Код (visual format language)
 - Библиотеки (PureLayout, SnapKit)



Пример Auto Layout (VFL)



```
let subview = UIView(frame: .zero)
subview.backgroundColor = .purple
subview.translatesAutoresizingMaskIntoConstraints = false

view.addSubview(subview)

let views = ["subview": subview]
let metrics = ["bottomMargin": 50]
let horizontalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "H:|[subview]|",
                                                                              options: [],
                                                                              metrics: metrics,
                                                                              views: views)

view.addConstraints(horizontalConstraints)

let verticalConstraints: [NSLayoutConstraint] = NSLayoutConstraint.constraints(withVisualFormat: "V:|[subview]-bottomMargin-|",
                                                                              options: [],
                                                                              metrics: metrics,
                                                                              views: views)

view.addConstraints(verticalConstraints)
```

- addConstraints — deprecated :)

Layout Anchors



```
// создаем constraints через NSLayoutConstraint-конструктор

NSLayoutConstraint(item: self.view!,
                  attribute: .top,
                  relatedBy: .equal,
                  toItem: subview,
                  attribute: .top,
                  multiplier: 1,
                  constant: 0).isActive = true

NSLayoutConstraint(item: self.view!,
                  attribute: .leading,
                  relatedBy: .equal,
                  toItem: subview,
                  attribute: .leading,
                  multiplier: 1,
                  constant: 0).isActive = true

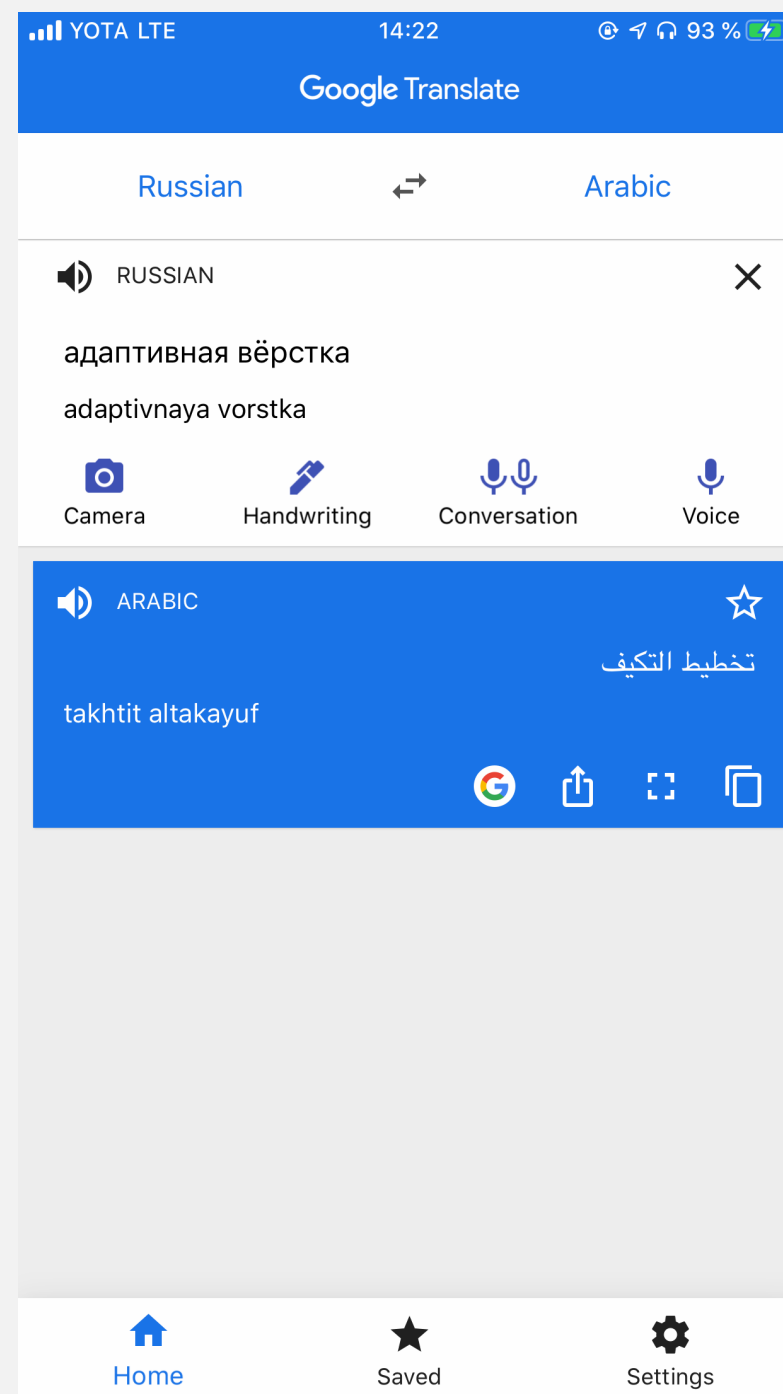
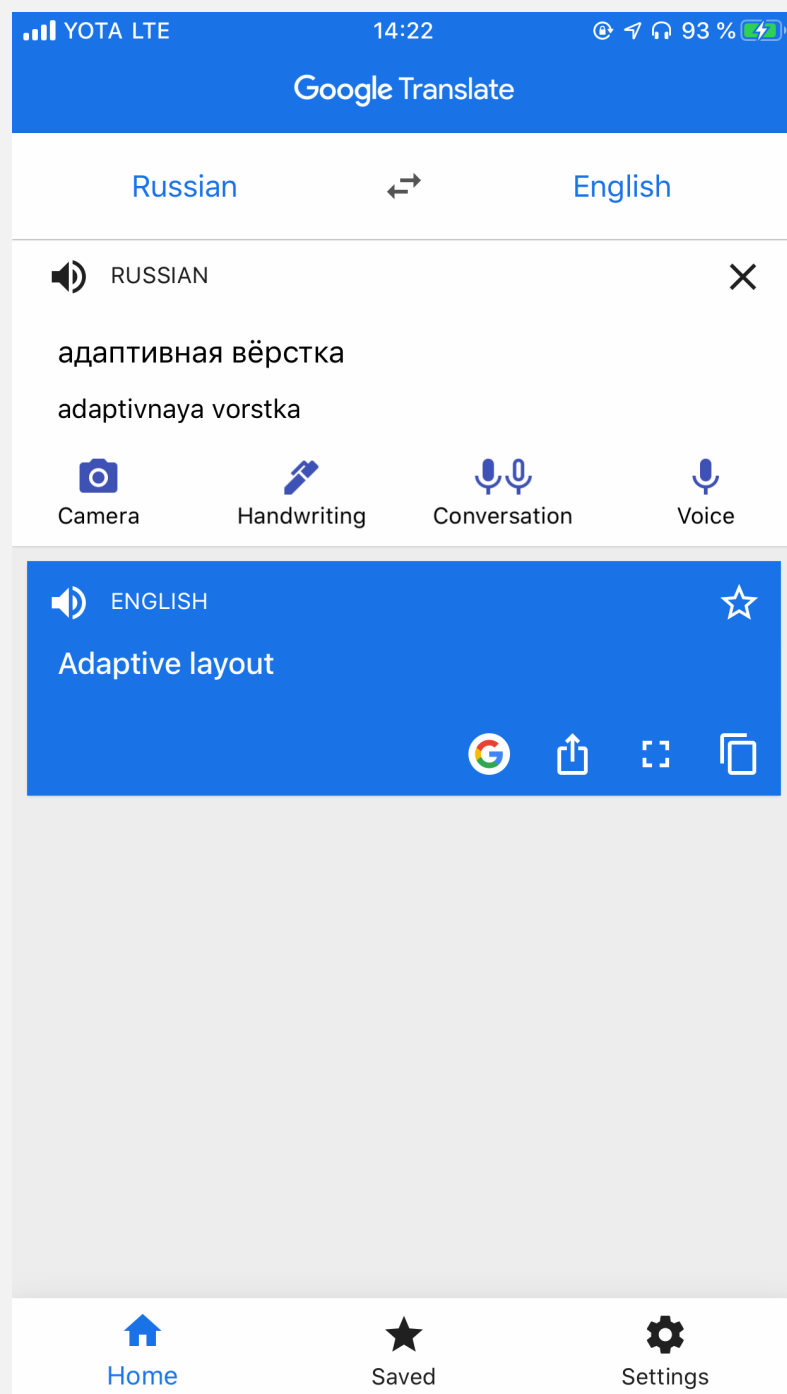
// создаем те же constraints через Anchor-API

self.view.topAnchor.constraint(equalTo: subview.topAnchor).isActive = true
self.view.leadingAnchor.constraint(equalTo: subview.leadingAnchor).isActive = true
```

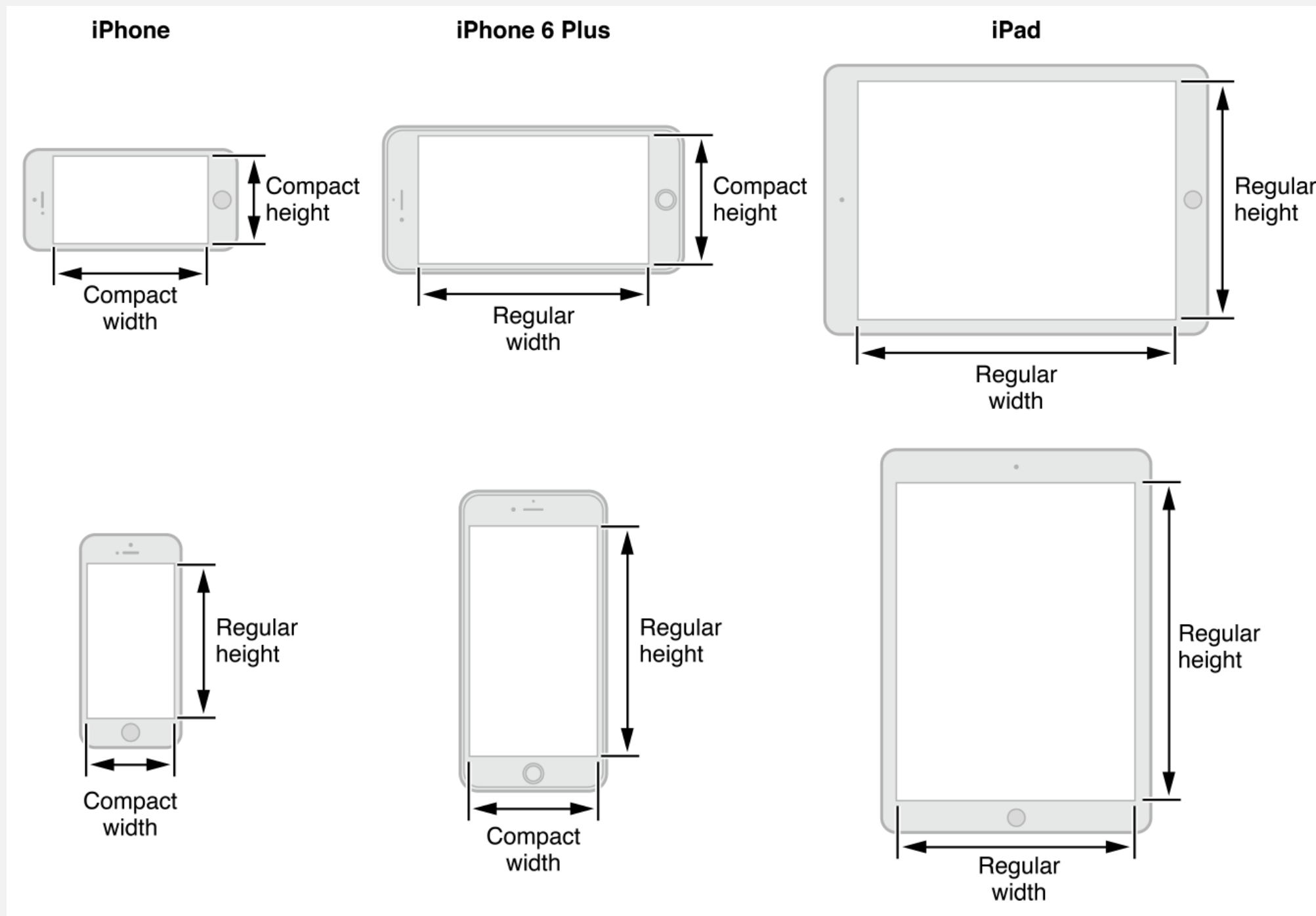
- Входит в состав CoreAnimation framework
- Создается UIView и UIView ставится его делегатом по умолчанию
- Не умеет обрабатывать события (не наследуется от UIResponder)
- Нужен для большего контроля над отображением интерфейса и анимациями
- Пригодится:
 - `cornerRadius`
 - `borderColor`, `borderWidth`
 - `shadowPath`

- Разные размеры экранов
- Разные ориентации (вертикальная, горизонтальная)
- Split View (iPad)
- Меняющийся текст
- Layout direction в зависимости от страны

Adaptivity



Size Classes



Size Classes



- Доступно в Storyboard
- В зависимости от size class можно:
 - добавлять/удалять constraint
 - добавлять/удалять view
 - менять атрибуты (например, шрифт)
- В коде

```
if view.traitCollection.horizontalSizeClass == .compact {  
    // какая-то логика  
}
```

-
- gesture recognizers
 - autolayout
 - draw rect

- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html#//apple_ref/doc/uid/TP40010853 — Autolayout
- <https://www.raywenderlich.com/443-auto-layout-tutorial-in-ios-11-getting-started> - Autolayout
- <https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/adaptivity-and-layout/> — Adaptivity
- <https://developer.apple.com/documentation/uikit/nslayoutanchor> — Anchors
- <https://developer.apple.com/ios/human-interface-guidelines/> — Apple Human Interface Guidelines
- <https://www.objc.io/issues/3-views/custom-controls/> — о кастомных control'ах
- <http://nshipster.com/ibinspectable-ibdesignable/> — IBDesignable / IBInspectable (редко используется, для фанатов Interface Builder'a)
- <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/AutolayoutPG/VisualFormatLanguage.html> — visual format language (для auto layout)
- <https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/Size-ClassSpecificLayout.html> — Size Classes
- <https://developer.apple.com/library/archive/documentation/2DDrawing/Conceptual/DrawingPrintingiOS/BezierPaths/BezierPaths.html> — рисование с помощью Bezier Paths