КРАТКИЙ ОБЗОР ВОЗМОЖНОСТЕЙ VISION KIT В IOS 13

Вчерашний В.Е.

Вчерашний Владислав Евгеньевич - руководитель отдела,
отдел разработки мобильного приложения,
Компания Parler LLC,
преподаватель,
курсы разработки приложений для Apple iOS,
г. Киев, Украина

Аннотация: *в статье анализируется работа библиотеки VisionKit в операционной системе Apple iOS 13.*
Ключевые слова: *apple, iOS, visionkit, камера, сканирование, Swift.*

Введение

Возвращаясь в, уже казалось бы, далёкий 2017-й год на ежегодную конференцию Apple для разработчиков — WWDC мало кто вспомнит, что тогда нам впервые показали фреймворк под названием **Vision**. На тот момент он умел не так уж и много:

1. распознавать лица и штрих-коды;
2. определять текст;
3. идентифицировать плоскости.

Так же фреймворк позволял классифицировать или распознавать объекты, используя Core ML модели.

Решив испытать новые возможности iOS, я быстро побежал создавать новый проект, читать документацию и тестировать новые фичи iOS. Каким же было моё удивление, когда я увидел рамку вокруг текста — но вот текст, к сожалению, так и не был распознан. Как так получилось? — В Apple решили, что мне достаточно указать место, где есть текст, а вот распознавать я должен самостоятельно. На этом мое желание тестировать Vision закончилось. :)

Вскоре, спустя два года, на WWDC 2019 мы получили долгожданное обновление фреймворка, который теперь умеет *распознавать* текст. Важно отметить, что распознавание текста с помощью VisionKit доступно только в iOS 13, а вот для более ранних версий рекомендую использовать WeScan.

Вступление

VisionKit — это небольшой фреймворк, который позволяет распознавать текст в вашем приложении используя системный сканер документов (как в Notes.app).

Давайте на примере кратко ознакомимся с возможностями VisionKit, а также с его помощью научимся распознавать текст с документов.

Приступая к работе

Создайте проект на основе *Single View App*.

! *Не забудьте добавить* ***NSCameraUsageDescription*** *в Info.plist во избежания креша приложения.*

Откройте *ViewController* и импортируйте **Vision** и **VisionKit**:

```
//*
import Vision
import VisionKit
*/
```

Далее откройте **Main.storyboard** и добавьте в него 3 элемента

1. UIButton — для вызова VNDocumentCameraViewController;
2. UIImageView — для отображения сделанной фотографии;
3. UITextView — для отображения распознанного текста.

Сканирование документов

В VisionKit Apple добавила новый ViewController — *VNDocumentCameraViewController*. Как говорит документация — это контроллер, который показывает, что видит “камера документов”.

Добавьте его в вашу иерархию контроллеров для сканирования документов, как показано ниже:

```
@IBAction private func scan(_ sender: Any?) {
    let scannerVC = VNDocumentCameraViewController()
    scannerVC.delegate = self
```

```
        self.present(scannerVC, animated: true, completion: nil)
    }
```

также, не забудьте подписаться и реализовать делегат *VNDocumentCameraViewControllerDelegate*. В нем находится всего 3 метода, каждый отвечающий за состояние контроллера:

optional func documentCameraViewController(_ controller: VNDocumentCameraViewController, didFinishWith scan: VNDocumentCameraScan)

optional func documentCameraViewControllerDidCancel(_ controller: VNDocumentCameraViewController)

optional func documentCameraViewController(_ controller: VNDocumentCameraViewController, didFailWithError error: Error)

! *Важно отметить, что за закрытие контроллера полностью отвечаете только вы. Поэтому вызов функции* **controller.dismiss(animated: true, completion: nil)** *должен происходить во всех трех методах.*

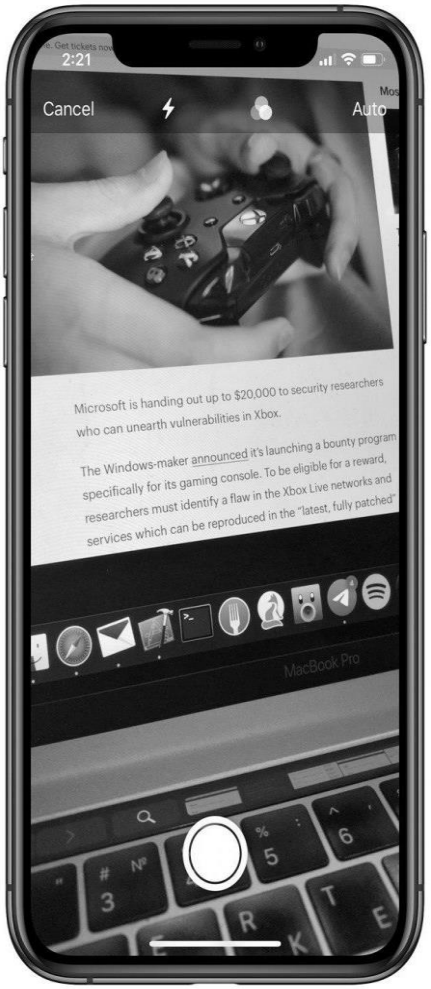

Рис. 1. Работа сканера

Распознавание текста в документах

Как вы могли догадаться, показ сканера документов — это только начало. Теперь текст нужно распознать! Для этого нам понадобится **VNRecognizeTextRequest** — запрос, который будет определять место текста, а также распознавать его на картинке.

```
//*
private var request: VNRecognizeTextRequest!
```

```
    private func setupVisionKit() {
        self.request = VNRecognizeTextRequest(completionHandler: { [weak self] request, error in
            guard let `self` = self else { return }

            if let error = error {
                print("Scanned with error: \(error.localizedDescription)")
                return
            }

            guard let result = request.results as? [VNRecognizedTextObservation], result.count > 0 else {
                print("Nothing found")
                return
            }

            var scannedTextValue = ""
            for observation in result {
                guard let topValue = observation.topCandidates(1).first else { return }
                scannedTextValue += "\(topValue.string)\n"
            }

            DispatchQueue.main.async {
                self.recognizedTextView.text = scannedTextValue
            }

        })
        self.request.recognitionLanguages = ["en-US"]
        self.request.recognitionLevel = .accurate
    }
*/
```

Несколько слов о параметрах VNRecognizeTextRequest

В примере выше мы установили два свойства — recognitionLanguages (массив языков) и recognitionLevel (.fast — быстрый и .accurate — точный). Также существует еще несколько дополнительных опций:

1. customWords — массив слов, дополняющие словари языков;
2. minimumTextHeight — число от 0 до 1. Размер от высоты изображения, при котором текст не будет распознаваться;
3. usesLanguageCorrection — булевое значение. Указывает на использование коррекции языка во время распознавания текста.

Запрос, который мы создали имеет комплишен, который будет запускаться каждый раз, когда нам нужно будет распознать текст с картинки.

Для каждого сканирования нам нужно запускать отдельный VNImageRequestHandler. Во избежание “подтормаживания” интерфейса обработку нужно запсукать в отдельном *.userInitiated* потоке.

```
    private func recognizeText(in image: UIImage) {
        guard let cgImage = image.cgImage else { return }
        self.imageView.image = image

        DispatchQueue.global(qos: .userInitiated).async { [weak self] in
            guard let `self` = self else { return }

            let imageRequestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:])
            do {
                try imageRequestHandler.perform([self.request])
            } catch {
                print(error.localizedDescription)
            }
        }
    }
```

Для лучших результатов рекомендуется использовать **cgImage**.

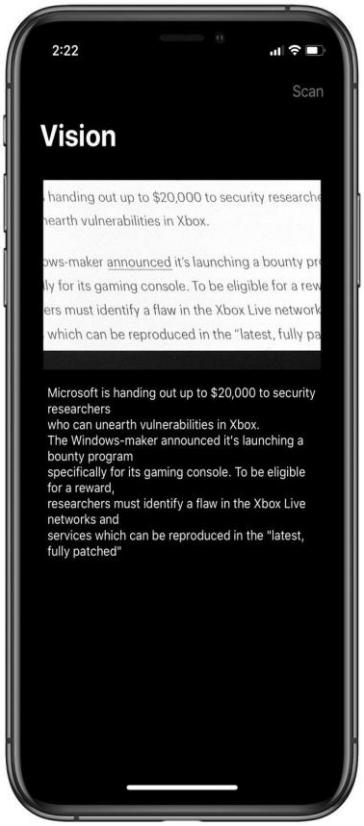

Рис. 2. Результат работы

Выводы

В данной статье мы научились сканировать документы и распознавать на них текст с помощью VisionKit в iOS 13.

Список литературы

1. Документация Apple / VisionKit. [Электронный ресурс], 2019. Режим доступа: https://developer.apple.com/documentation/vision/ (дата обращения: 08.01.2021).
2. Репозиторий GitHub / WeScan. [Электронный ресурс], 2019. Режим доступа: https://github.com/WeTransfer/WeScan/ (дата обращения: 18.01.2021).