SharePlay и другие способы коммуникации

Связываем пользователей друг с другом

Данис Тазетдинов, 2021

Данис Тазетдинов

Chief Software Engineer, EPAM Systems

- Евангелист платформ Apple
- Отвечаю за компетенцию Apple в EPAM
- 21 год в IT
- 10 лет создаю мобильные приложения
- Веду подкасты Mobile People Talks и Apple Treats

@edeniska @edeniska


15

SharePlay

SharePlay

Зачем нужен и как работает

- Позволяет сблизить людей во время звонка FaceTime
- Представлен на WWDC21
- Работает на iOS 15.1, iPadOS 15.1, tvOS 15.1, macOS 12 Monterey (?)
- Часть функций доступны в Safari на Mac
- Еще недоступен для публики

Совместный просмотр кино

Или прослушивание музыки

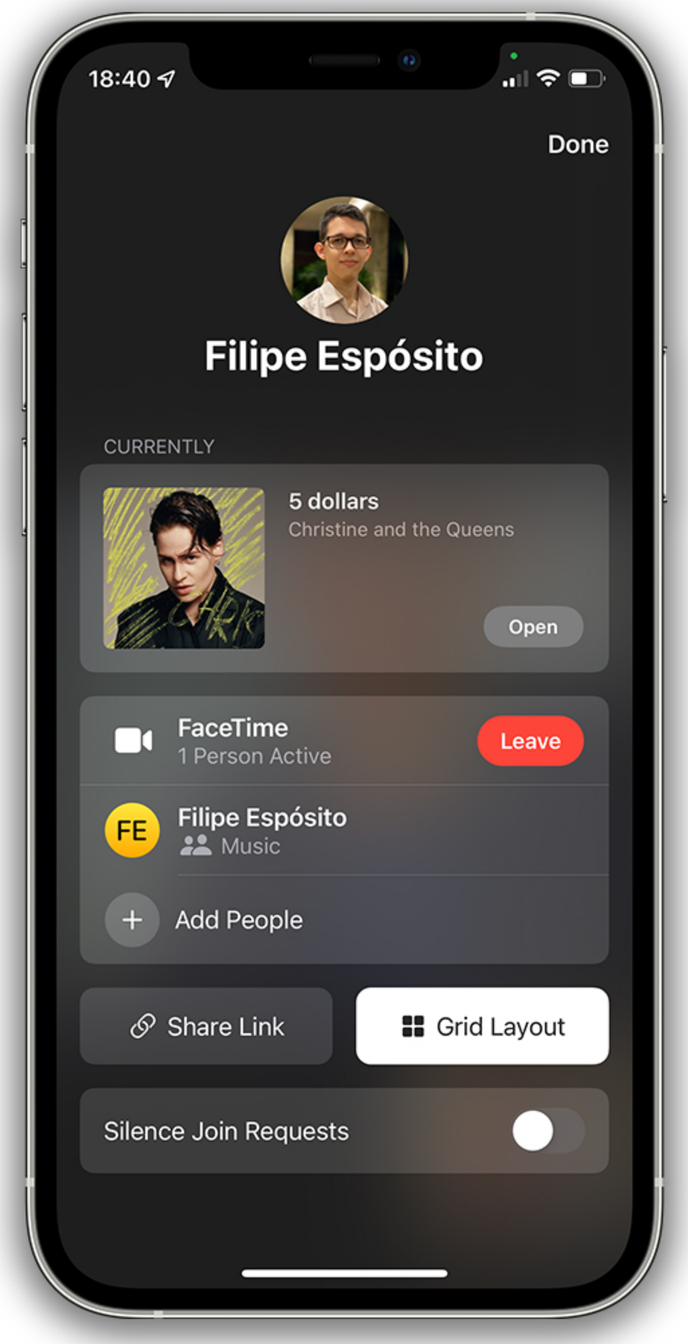
- Тесная интеграция с `AVPlayer`
- Точная синхронизация контента у участников

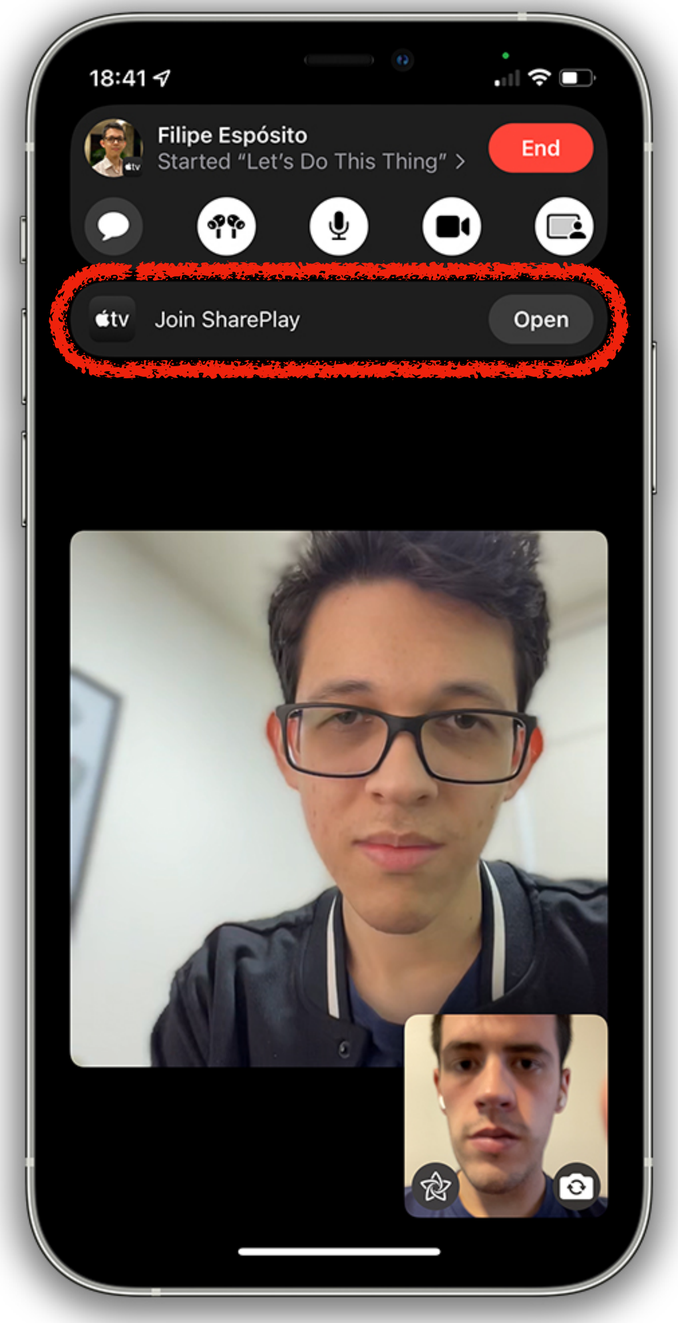
Совместная работа в приложениях

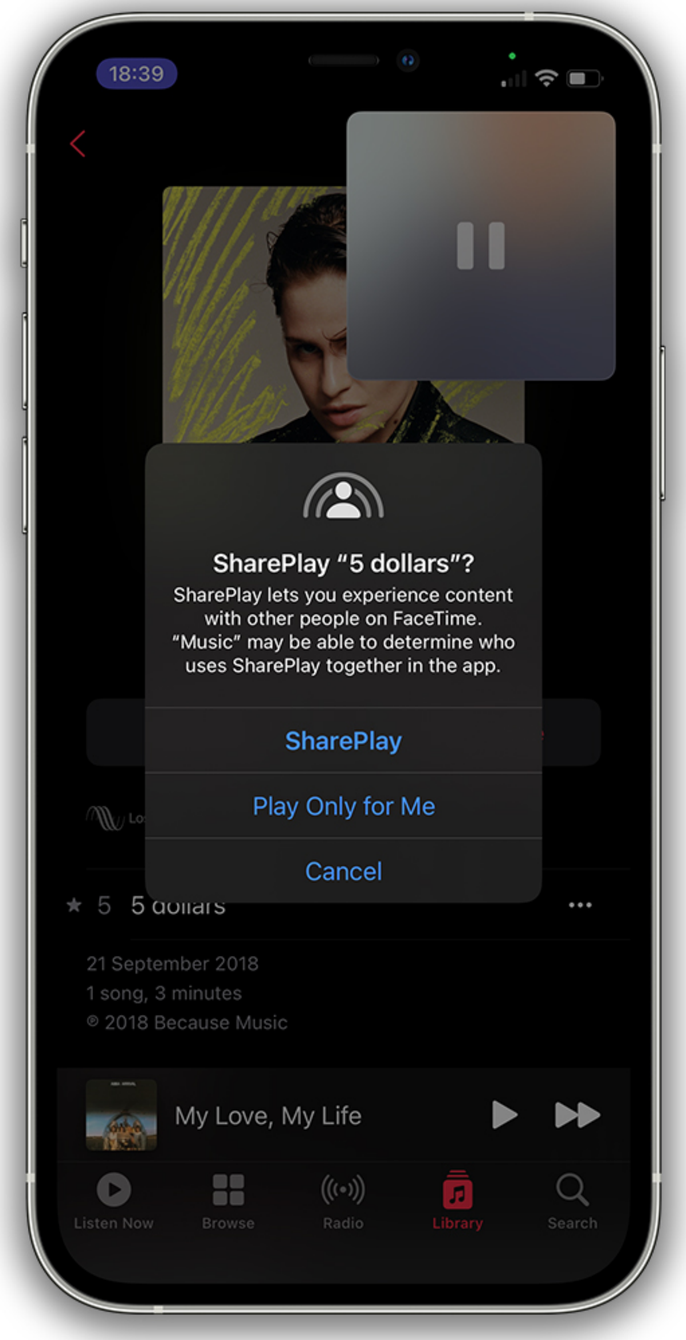
Не только для медиа-контента

- Канал данных для приложений
- Простое управление сессией
- Трафик зашифрован end-to-end

Хорошая реклама приложений


18:40
Done
Filipe Espósito
CURRENTLY
5 dollars
Christine and the Queens
Open
FaceTime
1 Person Active
Leave
FE
Filipe Espósito
Music
Add People
Share Link
Grid Layout
Silence Join Requests


18:41
Filipe Espósito
Started "Let's Do This Thing" >
End
Join SharePlay
Open


18:39
SharePlay "5 dollars"?
SharePlay lets you experience content with other people on FaceTime. "Music" may be able to determine who uses SharePlay together in the app.
SharePlay
Play Only for Me
Cancel
21 September 2018
1 song, 3 minutes
2018 Because Music
My Love, My Life
Listen Now
Browse
Radio
Library
Search

Что такое Group Activities?

Как это могут использовать разработчики

- Есть API для передачи данных другим участникам звонка
- Приложения могут иметь разные «активности»
- Для активности создаются «сессии»
- В каждой сессии есть «участники»
- Участникам можно слать «сообщения»
- Обильно используется Structured Concurrency и AsyncSequence

Использование Group Activity

Подготовительная работа

```
struct DrawingActivity: GroupActivity {
    static let activityIdentifier = "com.epam.poc.some-app"

    var metadata: GroupActivityMetadata {
        var metadata = GroupActivityMetadata()
        metadata.type = .generic
        metadata.title = "Some App"
        metadata.supportsContinuationOnTV = true
        return metadata
    }
}
```

Использование Group Activity

Подготовка сессии

```
// check for availability
if GroupStateObserver().isEligibleForGroupSession {
    // group activities are allowed
}

// activate session
Task {
    if try await DrawingActivity().activate() {
        // ...
    } else {
        // if session is handed over to AppleTV
        // ...
    }
}
```

Использование Group Activity

Использование сессии

```
var messenger = GroupSessionMessenger(session: session)

// join session
Task {
    for await session in DrawingActivity.sessions() {
        // configure session
        messenger = GroupSessionMessenger(session: session)
        // join the session
        session.join()
    }
}
```

Использование Group Activity

Обмен сообщениями

```
// receive messages
Task {
    for await (msg, cxt) in messenger.messages(of: Message.self) {
        // process message, context has sender
    }
}
```

Использование Group Activity

Обмен сообщениями

```
// send messages
let message: Message
let participant: Participant

Task {
    // send to all
    try await messenger.send(message)

    // send to specific participant
    try await messenger.send(message, to: .only(participant))
}
```

Использование Group Activity

Следим за сессией

```
let cancellable = session.$state.sink { state in
    switch state {
    case .waiting:
        break // before joining
    case .joined:
        break // active
    case .invalidated(reason: let reason):
        break // should reset

    @unknown default:
        break // for future use
    }
}
```

Что-то это напоминает…

Есть и другие фреймворки для совместной работы

- Существует Multipeer Connectivity
- Кроме того, был и GameKit
- Но вообще – никто не отменял Bonjour Services
- Или любые другие p2p протоколы

Multipeer Connectivity

Уже существующий фреймворк для взаимодействия

- Позволяет устройствам обмениваться информацией
- Позволяет шифровать трафик
- Устройства должны быть рядом друг с другом
- Общение начиняет «сервер», к которому подключаются «клиенты»
- Дальше обмен данными идёт между равнозначными узлами

Multipeer Connectivity

Создание сессии

```
let peerID = MCPeerID(displayName: "me")

let serviceType = "epam-some-app"

let session = MCSession(peer: peerID,
                        securityIdentity: nil,
                        encryptionPreference: .required)

session.delegate = sessionDelegate
```

Multipeer Connectivity

Ищем и подключаем узлы к сессии

```
let serviceBrowser = MCNearbyServiceBrowser(peer: peerID, serviceType:
serviceType)

serviceBrowser.delegate = serviceBrowserDelegate
serviceBrowser.startBrowsingForPeers()
```

Multipeer Connectivity

Ищем и подключаем узлы к сессии

```
class ServiceBrowserDelegate: NSObject, MCNearbyServiceBrowserDelegate {
    func browser(_ browser: MCNearbyServiceBrowser,
                 foundPeer peerID: MCPeerID,
                 withDiscoveryInfo info: [String : String]?) {
        // found new peer, ask to join session
        browser.invitePeer(peerID,
                           to: session,
                           withContext: nil,
                           timeout: 60)
    }

    func browser(_ browser: MCNearbyServiceBrowser,
                 lostPeer peerID: MCPeerID) {
        // peer is lost
    }
}
```

Multipeer Connectivity

Показываем себя окружающим

```
let advertiser = MCNearbyServiceAdvertiser(peer: peerID,
                                           discoveryInfo: nil,
                                           serviceType: serviceType)
advertiser.delegate = advertiserDelegate
advertiser.startAdvertisingPeer()

class AdvertiserDelegate: NSObject, MCNearbyServiceAdvertiserDelegate {
    func advertiser(_ advertiser: MCNearbyServiceAdvertiser,
                    didReceiveInvitationFromPeer peerID: MCPeerID,
                    withContext context: Data?,
                    invitationHandler: @escaping (Bool, MCSession?) -> Void) {
        invitationHandler(true, session)
    }
}
```

Multipeer Connectivity

Ведем счет участникам

```
// keep list of peers
var connectedPeers = Set<MCPeerID>()

class SessionDelegate: NSObject, MCSessionDelegate {
    // ...

    func session(_ session: MCSession, peer peerID: MCPeerID, didChange state: MCSessionState) {
       switch state {
       case .notConnected:
            connectedPeers.remove(peerID)
       case .connecting:
            break // peer is almost connected
       case .connected:
            connectedPeers.insert(peerID)
       @unknown default:
            break // for future (?) use
        }
    }
}
```

Multipeer Connectivity

Обмен сообщениями

```
// send data
let data: Data
let otherPeer: MCPeerID

try session.send(data, toPeers: [otherPeer], with: .reliable)
```

Multipeer Connectivity

Обмен сообщениями

```
// receive data
class SessionDelegate: NSObject, MCSessionDelegate {
    // ...

    func session(_ session: MCSession,
                 didReceive data: Data,
                 fromPeer peerID: MCPeerID) {
        // process data
    }
}
```

Вроде бы похоже…

… но не так уж просто.

Применим SharePlay


Shared
whiteboard

Shared whiteboard

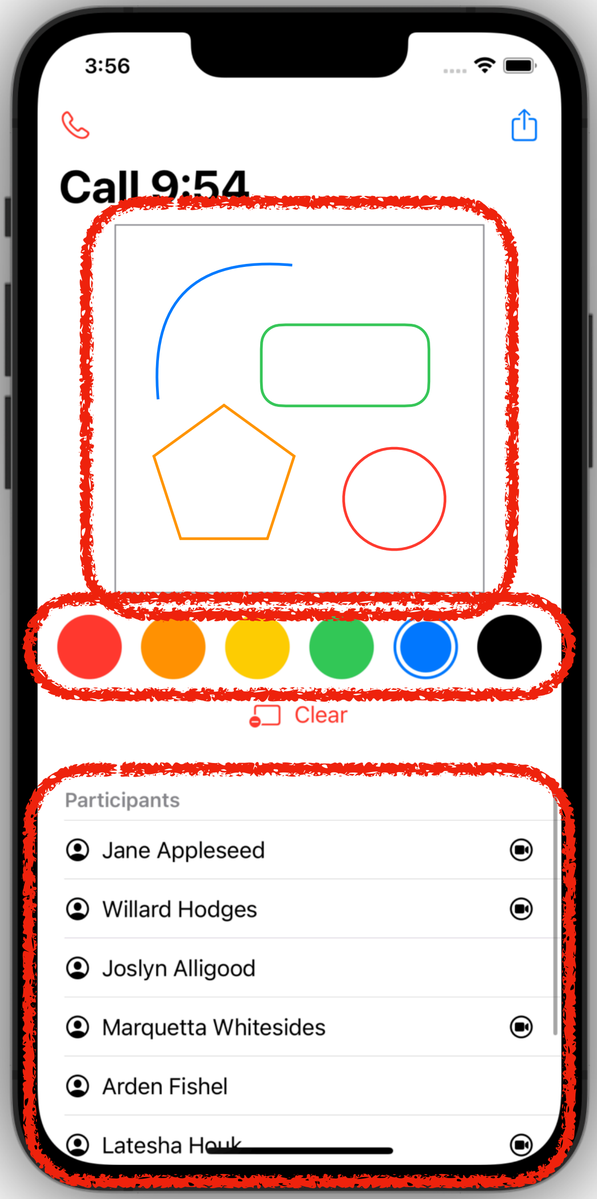
Полноценное приложение

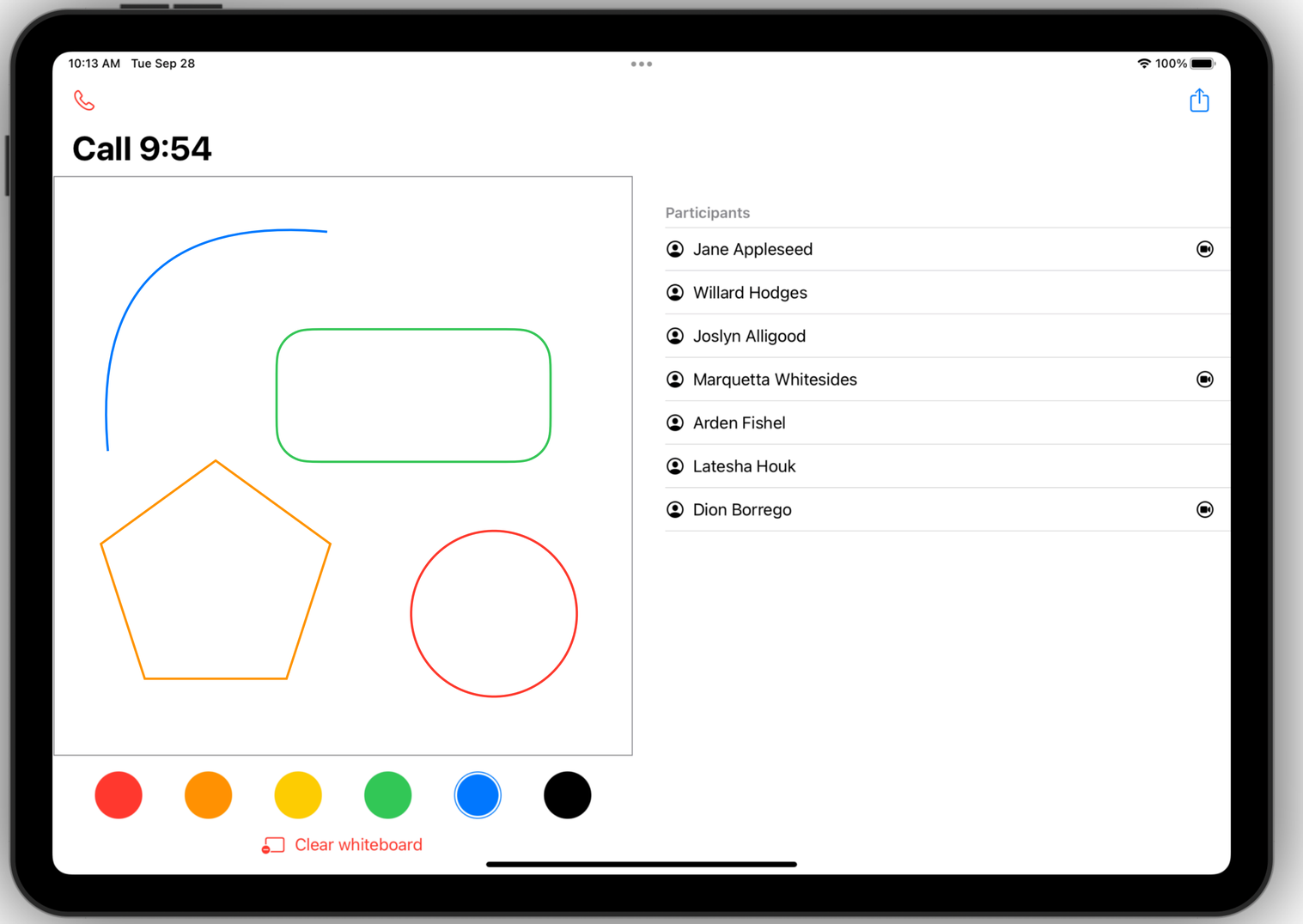
- Для проведения совещаний с общей доской для рисования
- Можно созвониться по FaceTime
- Участники рядом – подключаются к рисованию
- Рисовать могут все одновременно
- Картинка выводится на AppleTV

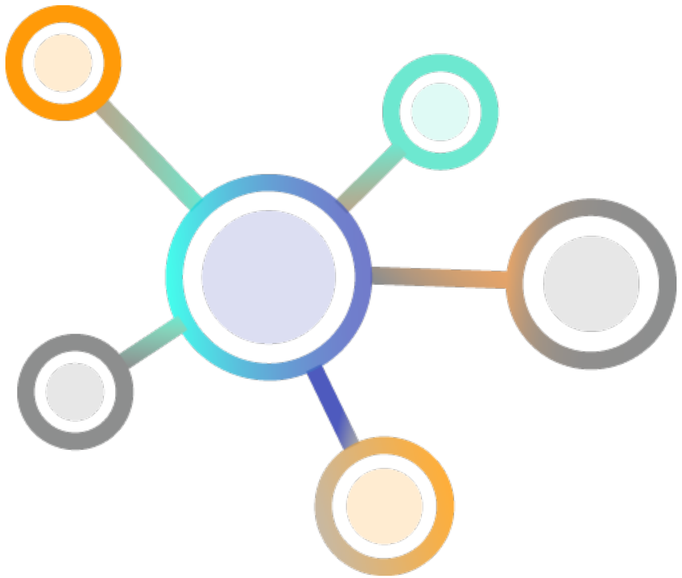
Что должно уметь приложение

Практический пример

- Никакого сервера, никаких сторонних ресурсов
- Рисовать пальцем на экране
- Общий экран для участников звонка FaceTime
- Общий экран для соседей участников звонка


3:56
Call 9:54
Clear
Participants
Jane Appleseed
Willard Hodges
Joslyn Alligood
Marquetta Whitesides
Arden Fishel
Latesha Houk


10:13 AM Tue Sep 28
100%
Call 9:54
Participants
Jane Appleseed
Willard Hodges
Joslyn Alligood
Marquetta Whitesides
Arden Fishel
Latesha Houk
Dion Borrego
Clear whiteboard



Модель данных

Модель данных

Картинка состоит из отдельных штрихов

- Массив штрихов
- Собственно, все

Модель данных

Картинка

```
struct WhiteboardImage: Codable {
    let strokes: [Stroke]
}
```

Модель данных

Штрих

- Уникальный идентификатор
- Автор
- Цвет
- Массив контрольных точек
- Точки будем нормировать ([0..1], [0..1]), поскольку экраны у всех разные

Модель данных

Штрих

```
struct StrokeColor: Codable {
    let red: Double
    let green: Double
    let blue: Double
    let alpha: Double
}

struct Stroke: Codable {
    let uuid: UUID
    let author: String
    let color: StrokeColor
    let points: [CGPoint] // normalized to ([0..1], [0..1])
}
```

Формирование изображения

Формирование изображения

Небольшая заготовка

```
extension CGPoint {
    static func *(lhs: CGPoint, rhs: CGSize) -> CGPoint {
        CGPoint(x: lhs.x * rhs.width, y: lhs.y * rhs.height)
    }
}
```

Формирование изображения

Рисование штриха

```
extension Stroke {
    func draw(with context: inout GraphicsContext, size: CGSize) {
        guard !points.isEmpty else {
            return
        }
        var path = Path()

        path.addLines(points.map { $0 * size })
        context.stroke(path, with: .color(color.color))
    }
}
```

Формирование изображения

Рисуем всю доску

```
extension WhiteboardImage {
    func draw(with context: inout GraphicsContext, size: CGSize) {
        strokes.forEach { $0.draw(with: &context, size: size) }
    }
}
```

Формирование изображения

Собираем все вместе

```
struct Whiteboard: View {
    let image: WhiteboardImage

    var body: some View {
        Canvas(renderer: image.draw)
    }
}
```

SwiftUI – это круто :)

Сетевые сообщения

Сетевые сообщения

Набор сообщений, которыми обмениваются узлы

- Новый штрих
- Очистить доску
- Запрос полной картинки
- Полная картинка

Сетевые сообщения

Набор сообщений, которыми обмениваются узлы

```
enum Message: Codable {
    case add(stroke: Stroke)
    case clear
    case requestImage
    case full(image: WhiteboardImage)
}
```

Swift 5.5 – очень крут :)

Важные задачи

Важные задачи

Не все так просто

- Нужно избегать «закольцованности»
- Не нужно перегружать сеть
- При подключении нового участника – ему нужная полная картина
- Кто-то должен знать, как выглядит полная картинка

Как избежать циклов

Какие проблемы вводят циклы

- Не нужно перегружать сеть
- Нужно упростить роутинг сообщений между сегментами
- Локальные «клиенты» не могут подключаться к другому FaceTime звонку
- Локальный участник может быть «сервером» для других и при этом начать или подключиться к FaceTime звонку
- Звонящие по FaceTime могут выступать только «сервером» для локальных участников и не могут подключаться к другим локальным «серверам»

Кто будет отвечать новичкам?

Люди подключаются к звонку в разное время

- При подключении новый участник спрашивает полную картинку
- Кто-то должен ее прислать
- Прислать ее должен кто-то один
- Все должны знать, что картинку этот «кто-то» отправит

Выбор «главного» среди равных

Алгоритм Raft

- Существует давно
- Создавался для несколько других целей
- Отлично решает наши задачи

Как работает алгоритм Raft

В контексте выбора лидера

Выбор «главного» среди равных

Алгоритм Raft

- Если узел не давно не получал сигналы от «лидера», он переходит в статус «кандидат»
- «Кандидат» начинает голосование за роль «лидера»
- Все узлы голосуют за того «кандидата», от которого получили запрос на голосование раньше других
- Если «кандидат» получил сообщение от «лидера» – он снимает статус «кандидата»

Выбор «главного» среди равных

Алгоритм Raft

- Голоса подсчитывают все участники
- Если голоса разделились поровну – «кандидаты» начнут новое голосование
- Если «кандидат» не получил большинства голосов, и не получил сообщения от «лидера», он ждет случайное время и инициирует голосование вновь
- «Лидер» должен периодически посылать сообщение, идентифицирующее статус «лидера»

Выбор «главного» среди равных

Алгоритм Raft

- Нам нужно добавить еще один тип сообщения – «Я – лидер»
- А также все типы сообщений для проведения голосования
- Участник не может стать кандидатом без получения полной картинки
  - Но как же быть с первыми подключившимися?..
  - Исключим тех, у кого `image.strokes.count` меньше других
- Уже есть готовые реализации Raft
  - Например, https://github.com/makadaw/swift-raft

Так что там про Apple TV?

Так что там про Apple TV?

Как будем выводить картинку?

- Можно «передать» активность SharePlay на AppleTV
- Но как быть с «локальным» рисованием?
- Можно использовать AirPlay
  - Но это немного неудобно…
- В нашем приложении мы сделаем tvOS приложение
  - Которое будет «локальным» клиентом к сессии
  - Подключается с пульта

Чего не хватает приложению?

Чего не хватает приложению?

Другие платформы

- Как быть с Android?
- Там есть FaceTime
  - … но нет SharePlay
- Как насчет других технологий обмена данными?

Чего не хватает приложению?

Перфекционизм

- Неоднозначность рисования
- Порядок штрихов может отличаться для разных участников
- Разная скорость каналов может вызывать проблемы

Подведем итоги

Подведем итоги

Трезвый взгляд на SharePlay

- Технология только появилась
- Очень ограничена экосистемой
- Но проста в использовании
- Увеличивает вовлеченность
- Подходит даже для бизнес-задач
- И относительно легка в разработке

Спасибо за внимание!

@eDeniska @AppleDevTalks @MobilePeopleTalks @AppleTreatsNews