SharePlay

Using FaceTime for Real-Time Networking

Redmond Xcoders — September 23rd, 2021

What is SharePlay?

Several things, actually…

- SharePlay allows you to use the FaceTime infrastructure to do real-time networking between participants.
- There is specific handling of video synchronization.
- But you can use it to send arbitrary messages, too!

How Do I Use SharePlay?

You can’t. Yet.

- SharePlay announced at WWDC ’21.
- SharePlay… un-announced on August 17th.
- But they added a profile to continue using it!
- …which didn’t work.
- But then it did!
- …but they broke it again.
- And now you need iOS 15.1.

```
struct GameActivity: GroupActivity {
    var metadata: GroupActivityMetadata {
        var metadata = GroupActivityMetadata()
        metadata.title = "Rock, Paper, Scissors"
        metadata.type = .generic
        return metadata
    }
}
```

```
func start() async {
    do {
        switch await activity.prepareForActivation() {
        case .activationPreferred:
            try _ = await activity.activate()
        default: break
        }
    } catch {
        // handle error
    }
}
```

```
for await session in GameActivity.sessions() {
    join(session)
}
```

```
private func join(_ session: GroupSession<GameActivity>) {
    session.$state.sink { [weak self] sessionState in
        switch sessionState {
        case .invalidated: self?.currentState = .notJoined
        case .waiting: self?.currentState = .notJoined
        case .joined: self?.handleSessionJoined()
        @unknown default: break
        }
    }.store(in: &cancellables)

    session.$activeParticipants.sink { [weak self] _ in
        self?.updateParticipants()
    }.store(in: &cancellables)

    // …

    session.join()
}
```

```
struct ChoiceMessage: Codable {
    let choice: Choice
}

struct RestartMessage: Codable {}
```

```
let messenger = GroupSessionMessenger(session: session)
self.messenger = messenger

session.join()

Task.detached {
    for await message in messenger.messages(of: ChoiceMessage.self) {
        self.handleTheirChoice(message.0.choice)
    }
}

Task.detached {
    for await _ in messenger.messages(of: RestartMessage.self) {
        self.restart()
    }
}
```

```
func sendChoice(_ choice: Choice) {
    messenger?.send(ChoiceMessage(choice: choice)) { [weak self] error in
        if let error = error {
            // handle error
        } else {
            self?.handleOurChoice(choice)
        }
    }
}
```