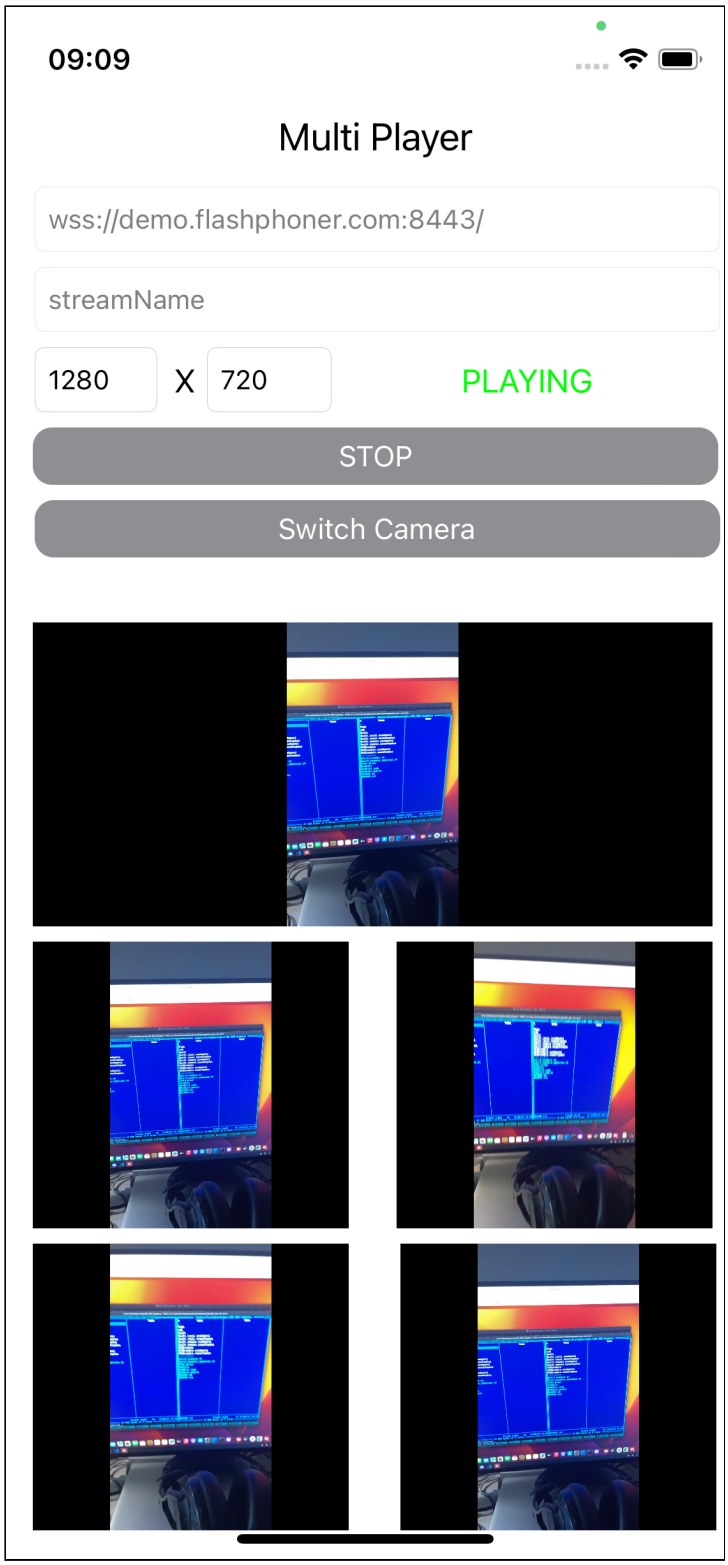
iOS MultiPlayer Swift

Описание

Данный пример демонстрирует публикацию одного WebRTC потока на сервер и проигрывание четырех WebRTC потоков с сервера (видеочат один-ко-многим). Может также использоваться как тест на потребление оперативной памяти при декодировании нескольких видеопотоков в высоком разрешении.

На скриншоте ниже представлен пример публикации и проигрывания потока 1280x720. Сверху отображается видео с камеры устройства, ниже - четыре проигрываемых потока



Работа с кодом примера

Для разбора кода возьмем версию примера MultiPlayerSwift, которая доступна для скачивания на GitHub.

Класс для основного вида приложения: MultiPlayerController (файл имплементации MultiPlayerController.swift).

1. Импорт API

code

```
import FPWCSApi2Swift
```

2. Создание сессии и подключение к серверу

`WCSSession`, `WCSSession.connect` code

В параметрах сессии указываются:

- URL WCS-сервера
- имя серверного REST hook приложения defaultApp

```
@IBAction func startPressed(_ sender: Any) {
    ...

    if (startButton.title(for: .normal) == "PUBLISH AND PLAY") {
        if (session == nil) {
            let options = FPWCSApi2SessionOptions()
            options.urlServer = urlField.text
            options.appKey = "defaultApp"
            do {
                try session = WCSSession(options)
            } catch {
                print(error)
            }
        }
        ...
        session?.connect()
    } else {
        ...
    }

}
```

3. Публикация видеопотока

`WCSSession.createStream`, `WCSStream.publish` code

Методу `createStream` передаются параметры:

- имя публикуемого потока
- вид для локального отображения

- ограничения высоты и ширины публикуемого потока

```
func publish() {
    let options = FPWCSApi2StreamOptions()
    options.name = streamName.text
    options.display = localDisplay.videoView

    let ret = FPWCSApi2MediaConstraints()

    let video = FPWCSApi2VideoConstraints()
    video.minWidth = Int(widthField.text ?? "0") ?? 0
    video.maxWidth = video.minWidth
    video.minHeight = Int(heightField.text ?? "0") ?? 0
    video.maxHeight = video.minHeight
    ret.video = video

    options.constraints = ret
    do {
        publishStream = try session!.createStream(options)
    } catch {
        print(error);
    }

    ...
    do {
        try publishStream?.publish()
    } catch {
        print(error);
    }
}
```

4. Запуск воспроизведения четырех экземпляров потока при успешной публикации

code

```
fileprivate func onPublishing(_ stream:FPWCSApi2Stream) {
    playStreamLT = self.play(display: remoteDisplayLT);
    playStreamRT = self.play(display: remoteDisplayRT);
    playStreamLB = self.play(display: remoteDisplayLB);
    playStreamRB = self.play(display: remoteDisplayRB);

    startButton.setTitle("STOP", for:.normal)
    changeViewState(startButton, true);
    changeViewState(switchCameraButton, true);
}
```

5. Воспроизведение потока

`WCSSession.createStream`, `WCSStream.play` code

Методу `createStream` передаются параметры:

- имя публикуемого потока
- вид для локального отображения

```
func play(display: WebRTCView) -> WCSStream? {
    let options = FPWCSApi2StreamOptions()
    options.name = streamName.text;
    options.display = display.videoView;
    do {
        let playStream = try session!.createStream(options)

        ...
        try playStream.play()
        return playStream;
    } catch {
        print(error);
    }
    return nil;
}
```

6. Переключение камеры

`WCSStream.switchCamera` code

```
@IBAction func switchCameraPressed(_ sender: Any) {
    publishStream?.switchCamera()
}
```

7. Остановка воспроизведения при остановке публикации потока

`WCSStream.stop` code

```
fileprivate func onUnpublished() {
    do {
        try playStreamLT?.stop();
    } catch {
        print(error);
    }
    do {
        try playStreamRT?.stop();
    } catch {
        print(error);
    }
    do {
        try playStreamLB?.stop();
    } catch {
        print(error);
    }
    do {
        try playStreamRB?.stop();
    } catch {
        print(error);
```

```
    }
}
```

8. Закрытие соединения

`WCSSession.disconnect` code

```
@IBAction func startPressed(_ sender: Any) {
    changeViewState(startButton, false)
    changeViewState(urlField, false)
    changeViewState(streamName, false)

    if (startButton.title(for: .normal) == "PUBLISH AND PLAY") {
        ...
        session?.connect()
    } else {
        session?.disconnect()
    }

}
```