Engineering

Recording to disk using AVCaptureScreenInput

On older macOS versions, ScreenCaptureKit isn't available. We created an example project demonstrating screen recording using the older AVCaptureScreenInput.

Tom Lokhorst, Mathijs Kadijk

January 29, 2023 · 2 min read

> *"Note: On macOS 12.3 it is recommended to use ScreenRecordingKit instead, this is more efficient and provides more features. Check out our post "Recording to disk using ScreenCaptureKit""*

tldr; Create an AVCaptureSession, connect the AVCaptureScreenInput and a AVCaptureMovieFileOutput.

Recording to a file

In our recent project to build a new recording backend for Screen Studio, we needed to create a screen recording on macOS and write the output to a file on disk. For newer macOS versions, we use ScreenCaptureKit, but that has only been available since macOS 12.3, and we also wanted to support 10.15 Catalina.

Fortunately for us, while it's quite a bit of work to write a file from a ScreenCaptureKit recording, using the older APIs, it's not a lot of code.

Using AVCaptureSession

We start by creating an AVCaptureSession, and adding an AVCaptureScreenInput as input and adding a AVCaptureMovieFileOuput as output:

```
let captureSession = AVCaptureSession()
let videoInput = AVCaptureScreenInput(displayID: displayID)!
```

```
captureSession.addInput(videoInput)

let output = AVCaptureMovieFileOutput()
captureSession.addOutput(output)

captureSession.startRunning()
```

We can then start the recording by calling **`startRecording(to:recordingDelegate:)`** on AVCaptureMovieFileOutput. Note that this can throw an NSException if there's an issue. You can't directly catch this exception from Swift, you'll need an Objective-C wrapper. Using a library like ExceptionCatcher from Sindre Sorhus makes this convenient to do from Swift.

```
output.startRecording(to: url, recordingDelegate: delegate)
```

To stop the recording, call the **`stopRecording()`** on AVCaptureMovieFileOutput, make sure to wait for the callback in **`RecordingDelegate.fileOutput(_:didFinishRecordingTo:from:error:)`** to really know the recording is finished:

```
func stop() async throws {
    try await withCheckedThrowingContinuation { continuation in
        delegate.finishedContinuation = continuation
        output.stopRecording()
    }
    delegate.finishedContinuation = nil // Don't leak continuation
```

```
    captureSession.stopRunning()
}

private class RecordingDelegate: NSObject, AVCaptureFileOutputRecordi
    var finishedContinuation: CheckedContinuation<Void, Error>?

    func fileOutput(_ output: AVCaptureFileOutput, didFinishRecording
        if let error {
            finishedContinuation?.resume(throwing: error)
        } else {
            finishedContinuation?.resume()
        }
    }
}
```

Detailed Example

On GitHub we have a repository with the full example in a SPM package.

To run this, clone the repo and run:

```
swift run avcrecording
```

References

- Nonstrict. (2023, January 24). AVCaptureScreenInput Recording Example. GitHub. https://github.com/nonstrict-hq/AVCaptureScreenInput-

Recording-example ▸

- Sindre Sorhus. (2022, September 22). ExceptionCatcher. GitHub. https://github.com/sindresorhus/ExceptionCatcher