AVFoundation - Video

Step 1: On whatever view controller you want to allow the user to view and record video in, draw out a `UIView`.

Step 2: Add the custom `CameraPreviewView` class as a separate file in your project:

```
import UIKit
import AVFoundation

class CameraPreviewView: UIView {
    override class var layerClass: AnyClass {
        return AVCaptureVideoPreviewLayer.self
    }
    var videoPreviewLayer: AVCaptureVideoPreviewLayer {
        return layer as! AVCaptureVideoPreviewLayer
    }
}
```

Step 3: Class the `UIView` you dragged out as a `CameraPreviewView`, and drag out an outlet to your accompanying view controller class file `@IBOutlet weak var videoPreviewView: CameraPreviewView!`. You'll also want to drag out a `UIButton` into the camera view and make actions/outlets from it. The outlet will allow you to switch from a circle to a square button when tapped to indicate recording or not recording. The action should allow you to perform work inside the IBAction when the button is tapped.

Step 4: As always, `import AVFoundation` up at the top of your class. This time, have your class conform to `AVCaptureFileOutputRecordingDelegate` if you want to be able to record.

Step 5: Add a few properties to the class that you'll need to record and save video:
`var captureSession: AVCaptureSession!`
`var recordOutput: AVCaptureMovieFileOutput!`
`var id: String?`

```
var recordedVideo: URL {
   let fileManager = FileManager.default
```

```
  let documentsDirectory = try! fileManager.url(for: .documentDirectory,
in: .userDomainMask, appropriateFor: nil, create: true)
  return
documentsDirectory.appendingPathComponent(id).appendingPathExtension("mov")
}
```

Step 6: Some boilerplate code is necessary to get the right camera and have it show up when the view loads. This is reusable code, so saving a snippet of this is a good idea.

```

private func showCamera() {
        let captureSession = AVCaptureSession()
        let videoDevice = bestCamera()
        guard let videoDeviceInput = try? AVCaptureDeviceInput(device:
videoDevice),
            captureSession.canAddInput(videoDeviceInput) else {
                fatalError()
        }
        captureSession.addInput(videoDeviceInput)

        let fileOutput = AVCaptureMovieFileOutput()
        guard captureSession.canAddOutput(fileOutput) else { fatalError() }
        captureSession.addOutput(fileOutput)
        recordOutput = fileOutput

        captureSession.sessionPreset = .hd1920x1080
        captureSession.commitConfiguration()

        self.captureSession = captureSession
        videoPreviewView.videoPreviewLayer.session = captureSession

    }



    private func bestCamera() -> AVCaptureDevice {
        if let device = AVCaptureDevice.default(.builtInDualCamera,
for: .video, position: .back) {
            return device
        } else if let device = AVCaptureDevice.default(.builtInWideAngleCamera,
for: .video, position: .back) {
            return device
        } else {
            fatalError(“Missing expected back camera device.”)
        }
    }
```

Step 7: In `viewDidLoad`, you can add the following code block to get the camera preview rolling once the user has authorized camera use (don’t forget to update the privacy settings in the project’s Info!

```
switch AVCaptureDevice.authorizationStatus(for: .video) {
        case .authorized:
            self.showCamera()
            captureSession.startRunning()

        case .notDetermined:
            AVCaptureDevice.requestAccess(for: .video) { granted in
                if granted {
                    self.showCamera()
                    self.captureSession.startRunning()
                }
            }
        default:
            return
        }
```

Step 8: In your IBAction record method, you can use the following to record and save the video to the URL property you set earlier.

```
if recordOutput.isRecording {
            recordOutput.stopRecording()
        } else {
            recordOutput.startRecording(to: recordedVideoURL,
recordingDelegate: self)
        }
```

Step 9: You can use the `didStartRecordingTo` and `didFinishRecordingTo` delegate methods to perform some work like updating UI if you want to. If everything went well, you should have no problem recording video on a real device and saving it.

` for the sprint challenge, you won't need to show a preview. Just be able to record a video.