俺のカード

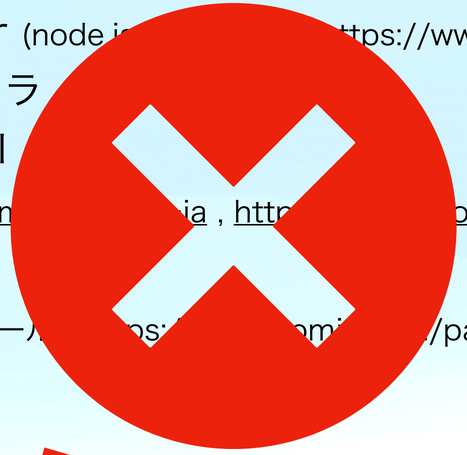
俺のカード

俺のカード

サービス名：俺のカード

概要：ポケモンカードゲームにおいてユーザーの持つカードの一覧作成を補助するツール。
　　手持ちカードの入力をカメラで文字認識して、キーボードの手打ち作業を軽減する。

使用予定技術：カードの撮影→webrtc-adapter (node.j　　ttps://www.npmjs.com/package/webrtc-adapter)
　　macPC搭載カメラ
　　OCR→Google Cloud Vision AI
　　(https://cloud.google.com　　ja , http　　oogle.com/vision/docs/ocr?hl=ja)
　　or
　　Tesseract.js (node.jsモジュール　　s:　　m　　/package/node-tesseract-ocr)



Swift独自の何かがあるんじゃ？

MVP1 ・キーボード入力でカード識別文字と枚数をDBに登録
　　・CRUD(クラッド)作る
MVP2 カメラで画像取得
MVP3 画像をOCRでテキスト化
MVP4 テキストをDBへ入れる

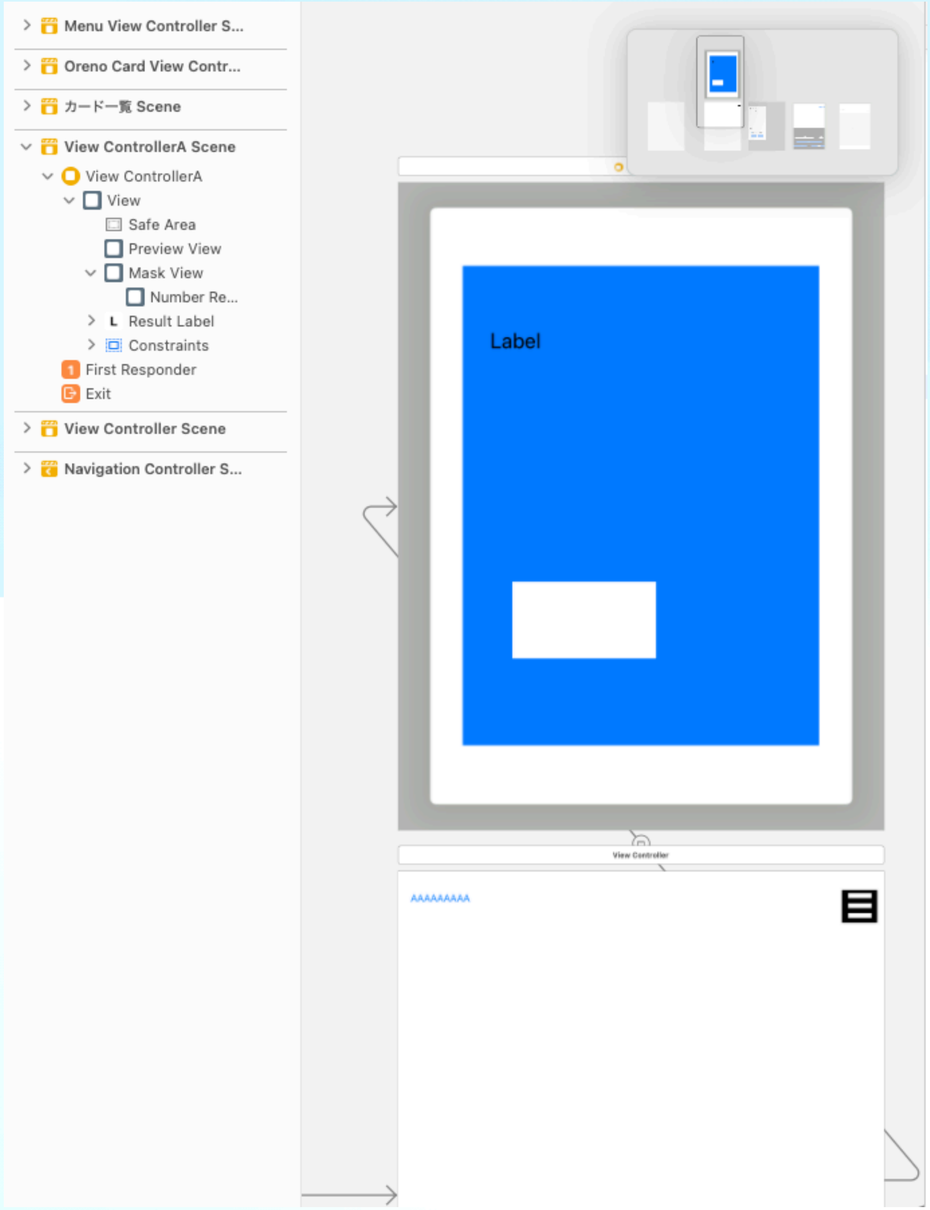
○1週間で学んだこと　～　テキスト検出に必要な映像を取り扱うフレームワーク3選　～

	VisionKit	Vision	AVFoundation
Overview(概要)	ピクセル情報分析→言語データ分離。分析結果をアプリへ。ドキュメントスキャンやOCR用。 DataScannerViewController:映像認識ビューを提示。 キャプチャされた情報を処理のためにアプリに提供。 画像分析IFが画像の上に表示され、ユーザーは認識するコンテンツを操作できるようになります。	Vision フレームワークは、顔と顔のランドマーク検出、テキスト検出、バーコード認識、画像登録、および一般的な特徴追跡を実行します。 Vision では、分類やオブ ジェクト検出などのタスクにカスタム Core ML モデルを使用することもできます。	AVFoundation は、iOS、macOS、watchOS、tvOS 上でタイムベースのオーディオビジュアル メディアを操作するためのフル機能のフレームワークです。 AVFoundation を使用すると、QuickTime ムービーや MPEG-4 ファイルの再生、作成、編集、HLS ストリームの再生、アプリへの強力なメディア機能の構築が簡単に行えます。
What I used (何に使ったか)	画像のスキャン	テキスト検出	カメラキャプチャ、 ビデオフレームワーク処理
Impressions(感想)	それなりに簡単に実装できた。	送られるものによって 難度変わる。	いろいろできるけど、 むずいいぃ。
Developer HP	https://developer.apple.com/documentation/visionkit	https://developer.apple.com/documentation/vision	https://developer.apple.com/av-foundation/

OCR　＜　テキスト検出

○アプリのデモ　→　最後

○アプリのソースコード



映像表示、マスク部、文字読取部の制約と誓約

```
class ViewControllerA: UIViewC
    override func viewDidLoad(
        NSLayoutConstraint.activate([
            PreviewView.topAnchor.constraint(equalTo: superview.topAnchor),
            PreviewView.bottomAnchor.constraint(equalTo: superview.bottomAnchor),
            PreviewView.leadingAnchor.constraint(equalTo: superview.leadingAnchor),
            PreviewView.trailingAnchor.constraint(equalTo: superview.trailingAnchor),

            MaskView.widthAnchor.constraint(equalTo: PreviewView.widthAnchor, multiplier: 0.8),
            MaskView.heightAnchor.constraint(equalTo: PreviewView.widthAnchor, multiplier: 0.8 * (7.0/5.0)),
            MaskView.centerXAnchor.constraint(equalTo: PreviewView.centerXAnchor),
            MaskView.centerYAnchor.constraint(equalTo: PreviewView.centerYAnchor),

            maskViewBorder!.topAnchor.constraint(equalTo: MaskView.topAnchor),
            maskViewBorder!.bottomAnchor.constraint(equalTo: MaskView.bottomAnchor),
            maskViewBorder!.leadingAnchor.constraint(equalTo: MaskView.leadingAnchor),
            maskViewBorder!.trailingAnchor.constraint(equalTo: MaskView.trailingAnchor),

            NumberRegionView.heightAnchor.constraint(equalTo: MaskView.heightAnchor, multiplier: 0.15),
            NumberRegionView.bottomAnchor.constraint(equalTo: MaskView.bottomAnchor),
            NumberRegionView.leadingAnchor.constraint(equalTo: MaskView.leadingAnchor),
            NumberRegionView.widthAnchor.constraint(equalTo: MaskView.widthAnchor, multiplier: 0.4),
        ])
```

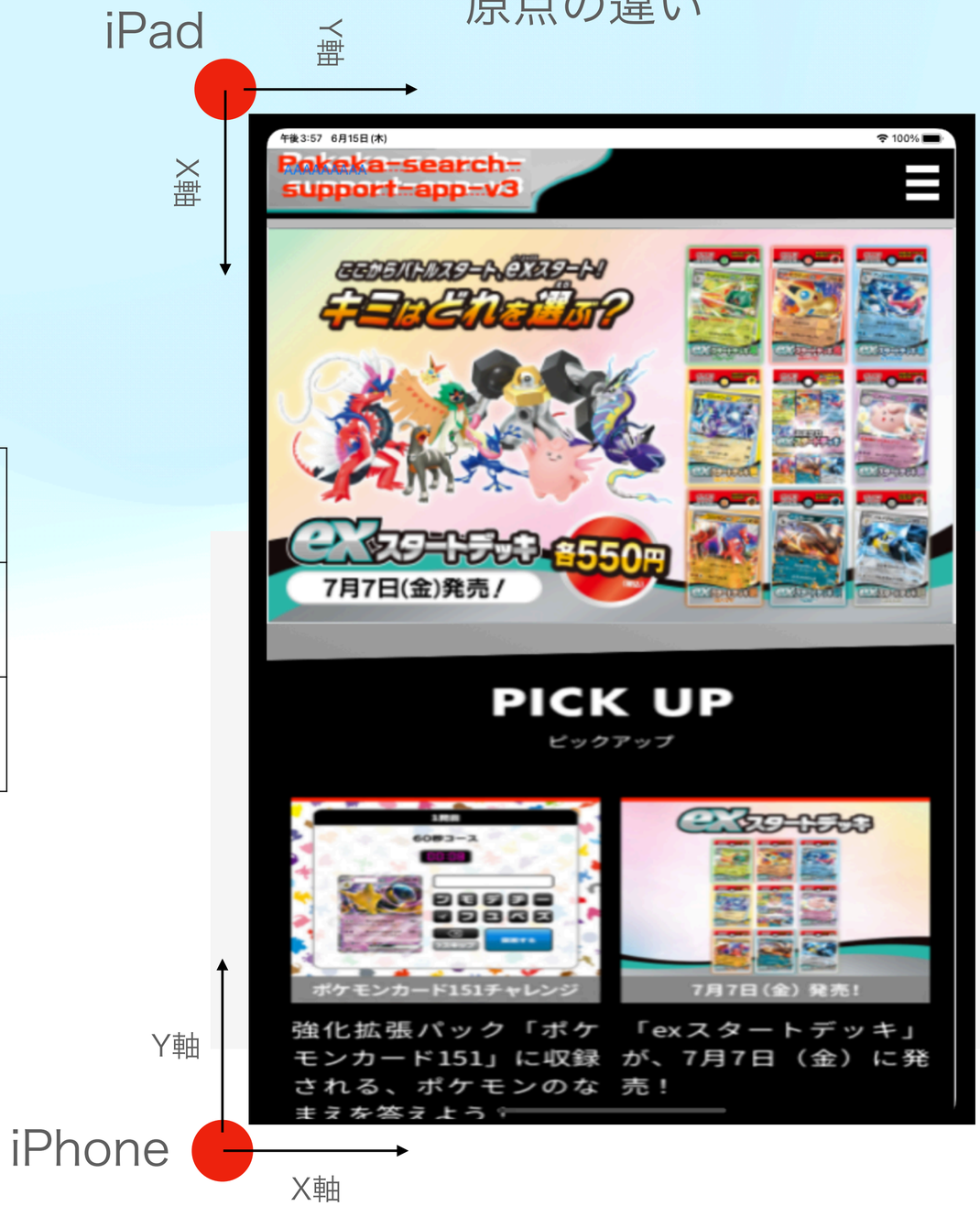
制約からviewDidLayoutSubviewsメソッドのregionOfInterestを設定。

```
    overri  unc viewDidLayout
        super.viewDidLayoutSubv
        let maskViewBounds = Ma
        let numberRegionViewBou

        let roiX = numberRegionViewBounds.origin.x / maskViewBounds.size.width
        let roiY = numberRegionViewBounds.origin.y / maskViewBounds.size.height
        let roiWidth = numberRegionViewBounds.size.width / maskViewBounds.size.width
        let roiHeight = numberRegionViewBounds.size.height / maskViewBounds.size.height

        let convertedROIX = (maskViewBounds.size.height - roiY * maskViewBounds.size.height - roiHeight *
            maskViewBounds.size.height) / PreviewView.bounds.size.height
        let convertedROIY = roiX * maskViewBounds.size.width / PreviewView.bounds.size.width
        let convertedROIWidth = roiHeight * maskViewBounds.size.height / PreviewView.bounds.size.height
        let convertedROIHeight = roiWidth * maskViewBounds.size.width / PreviewView.bounds.size.width

        regionOfInterest = CGRect(x: convertedROIX, y: convertedROIY, width: convertedROIWidth, height:
            convertedROIHeight)

        let convertedFrame = NumberRegionView.convert(NumberRegionView.bounds, to: PreviewView)
/         print(regionOfInterest)

        // NumberRegionViewの座標系変換後の枠線を追加して表示
        numberRegionBorderView?.removeFromSuperview()
        numberRegionBorderView = UIView(frame: convertedFrame)
        numberRegionBorderView!.layer.borderWidth = 2.0
        numberRegionBorderView!.layer.borderColor = UIColor.red.cgColor
        numberRegionBorderView!.layer.masksToBounds = true
        view.addSubview(numberRegionBorderView!)
        view.bringSubviewToFront(numberRegionBorderView!)

        maskViewBorder?.frame = MaskView.frame
        view.bringSubviewToFront(results)
        view.bringSubviewToFront(maskViewBorder!)
```

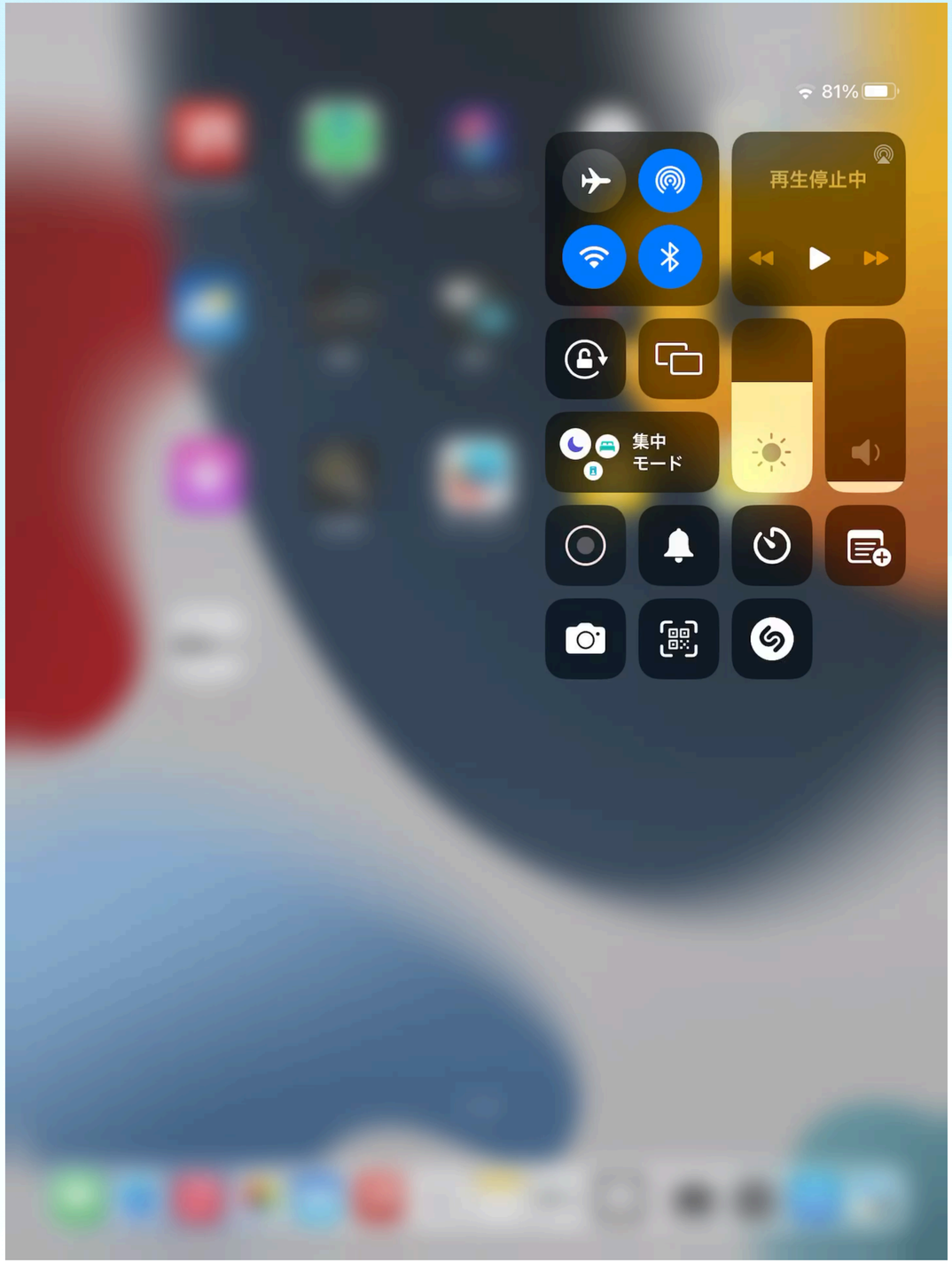
値の扱い

	表示の扱い	例
制約	ピクセル	0～1024
表示メソッド	正規化	0～1

原点の違い



○アプリのデモ


81%
再生停止中
集中
モード


たね
カイオーガ
HP 130
25th
アクアストーム
自分の山札を上から5枚トラッシュし、その中にあるエネルギーの枚数×50ダメージを、相手のベンチポケモン2匹にそれぞれ与える。［ベンチは弱点・抵抗力を計算しない。］
なみのり 120
弱点 ×2
抵抗力
にげる
s8a E 007/028


s8a
E
007/028