コンテンツ工学 最終課題

201920827 野秋裕己

選択課題2

表情だけを伝えるコミュニケーションアプリ

今回のアプリはGitHub上で公開しているため、プログラムを確認する必要がある場合は、以下のURLよりお願いいたします。

https://github.com/krgpi/facecast_ios/tree/prototype/record
(アクセス不可能な場合は、https://github.com/krgpi/facecast_ios/)

コンテンツの概念設計

開発の背景
近年、コミュニケーションデバイスやコミュニケションツールの発達により、遠隔地間の通信手段は多様化している。音声通話、ビデオ通話、文字による通信、さらにはVR空間内でアバターを通してコミュニケーションを行うVRChatなど、実に多種多様なコミュニケーション手段が挙げられる。

特にここでは、ビデオ通話に着目した。ビデオ通話は、遠隔地の相手の様子を映像で確認しながら会話することができるため、相手の表情やうなづきと言った非言語情報が、音声通話に比べると付加情報として得られる。

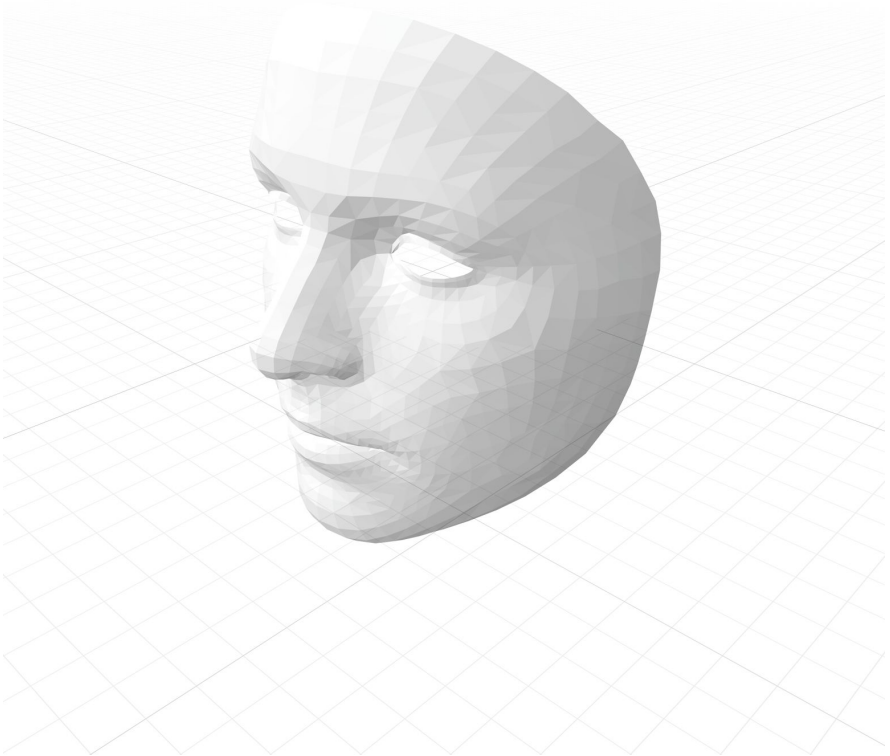
これは、Clark&Brennan 1991 Grounding in Communication で定義されるコミュニケーション手段別の性質による、visibilityが確保されていることによる。（下図）

しかし、しばしばビデオ通話においては、コミュニケーションをとる上で不要な情報もノイズとして付加されてしまう。例えば、マイクが拾ってしまう、呼吸音や環境音、居住空間であれば洗濯機やエアコンなどの家電機器の動作音が入り、コミュニケーションの妨げになる場面はよくある。また、ビデオによってうつされることにより、居住空間の様子や自宅にいるため身支度を指定なく、例えば寝癖などがひどい様子がうつされてしまうことになる。特に、後者はビデオ通話が遠隔コミュニケーション手段の中でも唯一visibilityが確保されていることに起因するものである。これが、ビデオ通話のデメリットである。

そこで、コミュニケーションにあった方が良い非言語情報（表情）を伝えつつ、前述したビデオ通話のデメリットを排除するために、3Dの顔のアバターを媒介とする、遠隔コミュニケーションツールを作成することとした。

詳細設計

本ツールはiOSアプリで作成した。iPhone X以降のiPhoneにはデプスカメラが搭載されており、Apple が配布しているSDKであるAR Kit を用いることで、表情のBlendshapes配列を取得することができる。BlendShapes配列は、顔の22点の動きを数値で表したものであり、この配列の数値で顔の表情が3D顔面モデルに再現することができる。
下図は、全てのblendShapes値が初期値である状態の3D顔面モデルである。



そして、iPhoneのセンサーで取得したblendShapesの配列を、リアルタイムで相手側の端末と送受信するために、TCP通信を行うsocket.ioというライブラリを採用した。

送信側と受信側を媒介するサーバーを立て、送信側のiPhoneからblendshapes配列を、サーバーにTCP通信で送信する。受信側のiPhoneはサーバーからTCP通信でblendshapes配列を受信して、アプリ内で画面上の3D顔面データに数値を当てはめる。

そうすることで、表情と音声だけを使ったコミュニケーションが可能になる。

送信側のコード

FaceTracking中にfaceGeometryがアップデートされた時に呼ばれる関数に、Socketに値を送信するコードを挿入した。

```
   func renderer(_ renderer: SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor) {

        guard let faceGeometry = node.geometry as? ARSCNFaceGeometry,

            let faceAnchor = anchor as? ARFaceAnchor

            else { return }



        faceGeometry.update(from: faceAnchor.geometry)



        var blendShapes = faceAnchor.blendShapes

     //キー名がStringではなく、ARFaceAnchor.BlendShapeLocation（構造体）の要素になるので注意

        guard let jaw: NSNumber = blendShapes[ARFaceAnchor.BlendShapeLocation.jawOpen]

            else { return }
```

```
      //socketで送信するためのStringに変換

        let str: String = jaw.stringValue

      //Socketで送信  このプロジェクトではsocketのクラスを別途用意しインスタンス化して使ってます

        socket.socketIOClient.emit("chat message", str)

    }
```

受信側のコード

socket io の部分は他のViewでも使うためClassを作って使いまわしてます。適宜読み替えてください。

```
//初期状態の顔面のジオメトリを宣言。（「一般的で中立的」な形と表情）

let faceGeometry = ARSCNFaceGeometry.init(device: sceneView.device!)!



let material = faceGeometry.firstMaterial!



//アセットにあるワイヤーフレームテクスチャーを適用させます

material.diffuse.contents = #imageLiteral(resourceName: "wireframeTexture")
```

```
//ライティングを設定 ライティングの種類は https://developer.apple.com/documentation/scenekit/scnmaterial/lightingmodel を参照

material.lightingModel = .physicallyBased




//nodeを初期化

let contentNode = SCNNode.init(geometry: faceGeometry)




let position = SCNVector3(x: 0.0, y: 0.0, z: -0.5)

if let camera = sceneView.pointOfView{

    //視点を設定

     contentNode.position = camera.convertPosition(position, to: nil)

}




//sceneView: ARSCNView に追加

self.sceneView.scene.rootNode.addChildNode(contentNode)




var jawValue: Float!

socket.socketIOClient.on("chat message"){data,ack in //socketから値を受信したら毎回以下を実行

    //受信データをパース
```

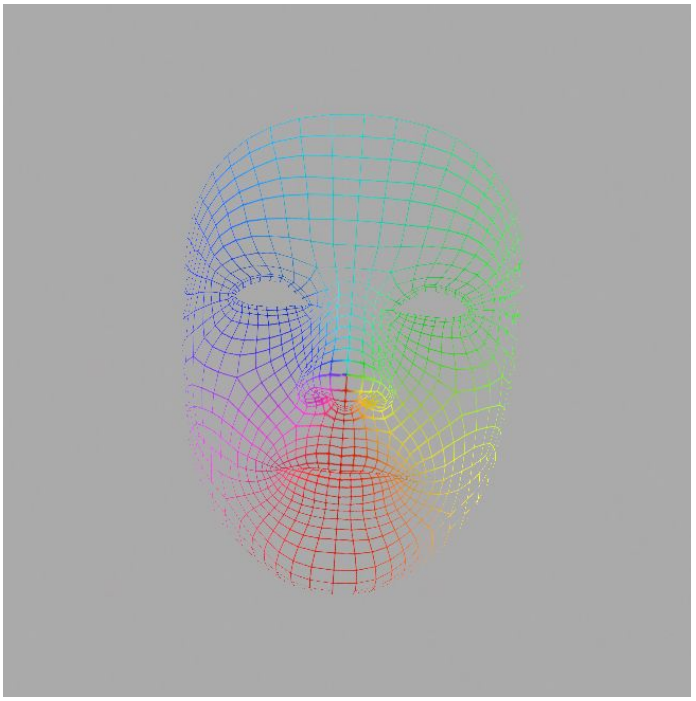
```
        guard let cur = data[2] as? String else { return }

        //Float型に変換
        jawValue = (cur as NSString).floatValue

        //さらにNSNumber型に変換
        let jaw: NSNumber = NSNumber(value: jawValue)

        //jawOpenの値を変えた新たな顔面のジオメトリをセット(他のプロパティは全て初期値=0.0)
        let geo: ARFaceGeometry = ARFaceGeometry.init(blendShapes:
[ARFaceAnchor.BlendShapeLocation.jawOpen:jaw ])!

        //ジオメトリを更新
        faceGeometry.update(from: geo)

    }
```

評価結果

このアプリを友人間のコミュニケーションに使用した。今回は、遠隔でプログラミングを教えるという場面を想定し評価を行った。何点かメリットとデメリットが挙げられた。

メリットとしては、表情以外の余計な情報がないため、相手の顔が常に画面に表示されている違和感や、常につながっている不快感が軽減された。

デメリットとしては、3Dアバターにテクスチャーを貼らなかったため、不気味の谷現象が生じてしまい、不快感を与えてしまったことが挙げられる。(下図)



総括

今回、既存のコミュニケーションツールが抱える問題点に着目し、新たなツールを提案した。コミュニケーションツールを使った遠隔コミュニケーションと、対面のコミュニケーションを比較し、音声通話・ビデオ通話・VRChatと言った多種多様なコミュニケーションツールがもつ性質について分析を行い、対面と比較して感じる違和感がなるべく少ないツールを理想とし、設計を行った。

今回、限られた時間での実装ではあったが、遠隔コミュニケーションについて深く考える時間を作ることができ、また自らの手でiPhoneのSDKを触ったことで、iOSの理解にもつながった。