Escola Politécnica da Universidade de São Paulo

PCS 3559 - Tecnologia para Aplicações Interativas

Documentação do Projeto

Laura Lassance (8588199), Willian Costa (8987847), Amauri Shimabukuro (4156015)

Alfabetização de crianças surdas: Realidade aumentada auxiliando a alfabetização em libras

Sumário

1. Contextualização e requisitos

O presente projeto tem por objetivo criar uma ferramenta para auxiliar a alfabetização de crianças surdas na Língua Brasileira de Sinais (LIBRAS). Assim como qualquer língua, a LIBRAS também é formada por diferentes níveis linguísticos, como fonologia, morfologia, sintaxe e semântica. Não basta apenas saber os sinais que representam cada letra, mas também a gramática de tal língua para combinar frases, permitindo o estabelecimento de uma comunicação correta e completa. Entretanto, o foco desse projeto é no primeiro passo para o aprendizado de uma língua: o alfabeto. *Ênfase é aqui dada não só ao aprendizado dos sinais representando alfabeto em LIBRAS, mas também à associação entre este e o alfabeto de símbolos gráficos da língua portuguesa.* A proposta aqui apresentada atende à surdez, deixando a cargo de outras soluções a inclusão de alunos com outros tipos de deficiência, como a cegueira.

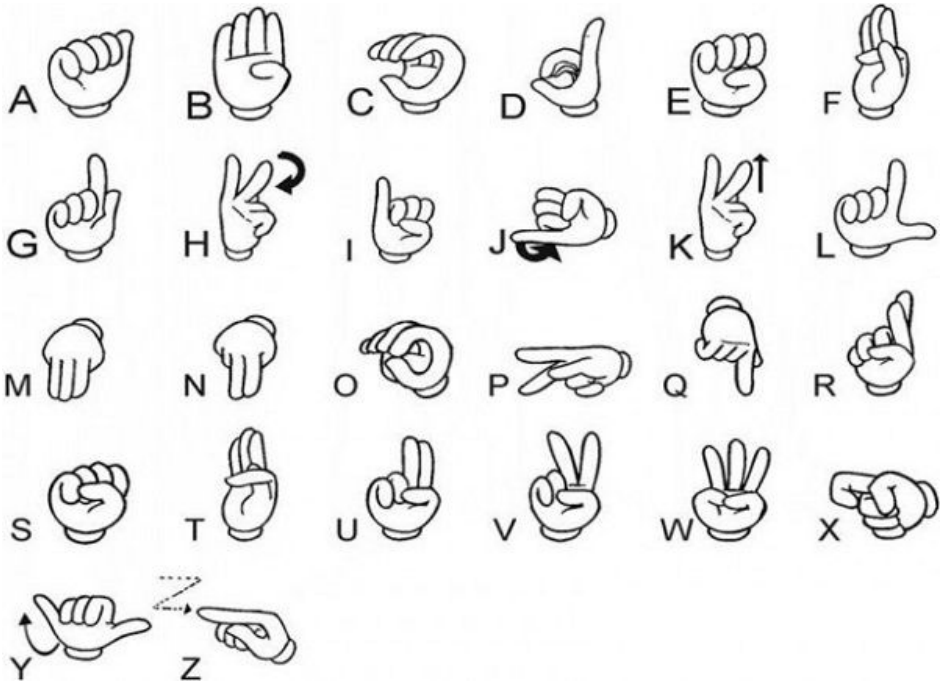


Figura 1 - Representação do alfabeto de LIBRAS

O contexto estudado é o de uma sala de aula inclusiva, onde tanto alunos surdos quanto alunos com audição normal participam juntos. O cenário principal considerado é o de alunos, em um primeiro momento, aprendendo o alfabeto com livros mostrando as letras e sua grafia (Figura 2) e, em um segundo momento, treinando o aprendido por meio de exercícios. A partir desse contexto os requisitos principais para a solução são: *ser facilmente portado por uma criança (peso e tamanho), ser resistente ao uso intensivo por crianças, ser adaptado ao uso em grupo, ser uma interação simples e eficiente para uso interativo por alunos e professores, se adaptar ao contextos de aulas existentes, e motivar por si só os alunos à a utilizarem*. O último quesito sugere a adição de um aspecto lúdico, levando em consideração que o público alvo é formado por crianças em torno de sete anos de idade. Outro aspecto desejável é o da união e cooperação entre alunos de audição normal e alunos deficientes auditivos.

2. Solução proposta

Isto posto, é proposta uma solução envolvendo três etapas: aprendizado livre, treino e jogo. O aprendizado livre trata a primeira situação do cenário, e o treino e o jogo endereçam a outra parte, de exercícios.

Etapa 1: aprendizado livre

Uso de realidade aumentada para aprendizado livre dos sinais manuais. Enquanto lendo o alfabeto em seu livro ou cartilha de alfabetização (Figura 2), a criança poderá direcionar a câmera do dispositivo móvel para letra que deseja aprender (ou marcador associado a essa letra). Nesse momento, com o auxílio de realidade aumentada, a criança visualizará o sinal manual correspondente àquela letra, ou seja, ela visualizará um modelo de mão fazendo o sinal como se este estivesse presente no ambiente visto através do dispositivo móvel. Tal sinal pode ser estático ou apresentar movimento, como o H e o J. O modelo de mão fazendo o movimento seria do tamanho médio da mão de uma criança. Além disso, é proposta a adaptação da cor de tal modelo para a cor de pele da criança. Dessa forma, a criança terá a impressão de que é sua própria mão que está aparecendo, aumentando a sensação de presença corporal e física e, com isso, melhorando a experiência do usuário.

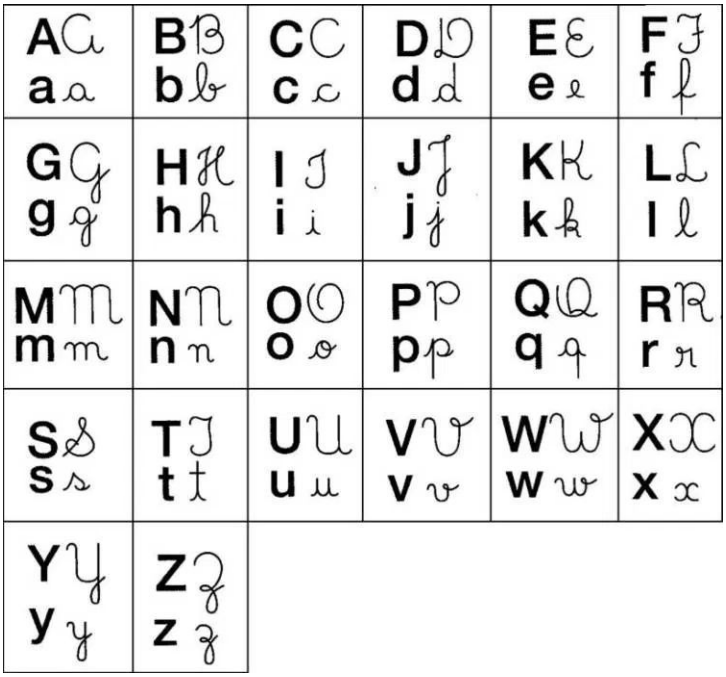


Figura 2 - Exemplo de livro ou cartilha de alfabetização da língua portuguesa

Crianças, atualmente, começam desde de cedo a usar intensivamente dispositivos móveis e estão totalmente adaptadas e confortáveis com sua manipulação e formas de interação. Portanto, o uso do dispositivo móvel aqui é escolhido atendendo aos requisitos de *ser facilmente portado por uma criança (peso e tamanho), ser adaptado ao uso em grupo, ser uma interação simples e eficiente para uso interativo por alunos e professores, e se adaptar ao contextos de aulas existentes*. Em relação ao requisito de resistência à manipulação intensa, os dispositivos e/ou acessórios preparados para esse tipo de uso, já existentes no mercado, serão selecionados.

Etapa 2: treino

Uma estação de treino (ou diversas) estará preparada com um sensor de movimentos das mãos e um programa de treino de símbolos do alfabeto. As letras a serem treinadas serão escolhidas previamente ao início da sessão de treinamento. A criança terá que fazer o sinal manual correspondente à cada uma das letras mostradas por meio de seus símbolos gráficos. Para cada símbolo haverá 3 tentativas. Caso, ao final das 3 tentativas, a criança ainda não tenha acertado o sinal manual, a letra é recolocada na lista de não concluídos e a criança passa para a próxima letra dessa lista.

Devido à necessidade do sensor de movimentos das mãos, foram adotadas as estações fixas formadas por um desktop ou notebook conectado a tal sensor. Dessa maneira, o custo da solução proposta é otimizado sem perda de funcionalidade ou usabilidade, tornando mais palpável sua adoção e implementação. Essas estações poderiam ser compartilhadas por grupos de alunos revezando as sessões de treinamento. O uso conjunto também incentivará a colaboração entre alunos, o que é mais divertido e motivador para as crianças.

Etapa 3: jogo

O objetivo dessa etapa é estimular as crianças ao aprendizado correto dos sinais manuais e da correta associação entre os sinais e os símbolos gráficos das letras. A abordagem lúdica objetiva dar motivação para os alunos passarem pelo processo de aprendizado e aperfeiçoá-lo até atingir um bom resultado. O jogo seria como um *LIBRAS Pump It!* (Figura 3). Por um período determinado, letras "subiriam" na tela e a criança teria que fazer corretamente o sinal correspondente. O mesmo sensor do treino seria utilizado aqui para verificar a corretude do sinal feito pelo aluno. O jogo teria níveis de dificuldade codificados como a velocidade com a qual as letras "sobem" a tela (como no Pump It!). Quando uma letra é feita corretamente, ela entra na caixinha com um brilho - exemplo do Pump It! na imagem abaixo. O jogo também engloba um aspecto colaborativo, pois pode ser jogado em duplas. Fica a cargo do professor como o campeonato será organizado e quais os critérios definem ganhar.

Para essa etapa foi pensado uma única estação fixa, com uma tela grande, como a de um televisor, e dois sensores de movimentos das mãos. O conceito proposto é de que a turma inteira participaria do jogo ao mesmo tempo, seja jogando ou assistindo, e que todos estariam envolvidos pois participarão de maneira ativa em algum momento.

Figura 3 - Tela do jogo Pump It! usado como inspiração para a proposta do LIBRAS Pump It!

3. Prova de Conceito

Nossa Prova de Conceito (POC) é modelar parcialmente a etapa 1 - aprendizado livre usando realidade aumentada e cartões com marcadores. Como POC foram modelados de 9 sinais em LIBRAS representando letras do alfabeto. Três deles têm movimento, os sinais do X, do Y e do Z. A plataforma Unity foi utilizada para construir o aplicativo com o modelo das mãos e lhes dar movimento. Cada sinal é associado a um marcador fixo e pré-determinado. Tais marcadores foram colocados em cartões, para facilitar a manipulação, inclusive nos testes. O código dessa POC pode ser encontrado no repositório do Github[1].

A supermeta era detectar a cor predominante da imagem da câmera quando uma mão estiver presente e aplicá-la ao modelo de mão sendo mostrado. Esse caso só considerará o cenário de uma mão aparecendo por vez. A super meta não pode ser atingida, mas para testar sua viabilidade um botão que controla a cor de pele do modelo da mão foi acrescentado à POC.

4. Especificação do Projeto

O projeto aborda a solução aqui apresentada de maneira simplificada, já que a intenção é provar que o conceito aqui proposto é possível.

4.1 - Programa

O software é um aplicativo para iOS que utiliza realidade aumentada para mostrar um modelo de mão fazendo o sinal de libras equivalente ao marcador detectado. Por isso, o aplicativo funciona apenas para os dispositivos que possuem câmera. Os marcadores possuem os símbolos gráficos das letras do alfabeto latino.

4.2 - Interface

A interface do projeto consiste da visão da câmera sobreposta por um painel de botões para mudança da cor de pele no canto superior esquerdo. Quando uma imagem-alvo é detectada, a ela sobrepõe-se o respectivo modelo 3D de uma mão fazendo o sinal de libra correspondente com animações quando condizente. Esse modelo acompanha qualquer movimentação do cartão, desde que a imagem da letra permaneça visível. É um requisito que cada cartão associe um e somente um modelo 3D.

No caso de modelos com animação, aparece também na interface dois botões: um chamado "RESTART letra" (letra é substituído pela letra representada, por exemplo "RESTART X" para a animação de X) que reinicia a animação do seu ponto inicial, e outro botão "PAUSE/PLAY letra" que pausa e reinicia a animação do mesmo ponto.

4.3 - Modelo de mão

É importante que as mãos tenham gestos claros, de maneira que as crianças sejam capazes de identificar de distinguir os gestos correspondentes às letras. Os movimentos devem ter velocidade adequada, de forma que uma criança seja capaz de acompanhar os movimentos e gestos, já que o intuito é que elas aprendam a os reproduzir, ou seja, aprendam o alfabeto de LIBRAS. O alfabeto manual pode ser feito com as duas mãos, e, no caso do projeto, o modelo usado é de uma mão esquerda.

---

[1] https://github.com/LauraLassance/alfabetizaLIBRAS.git

5. Implementação do projeto

5.1 - Implementação do alfabeto manual de LIBRAS

Devido ao tempo de projeto e à experiência limitada dos componentes do grupo com softwares de desenho 3D, decidiu-se por utilizar um modelo de mão pronto, disponível gratuitamente na Internet[2]. Logo foi pensado de qual forma seriam construídos os gestos e, quando aplicável, os movimentos representando as letras em LIBRAS. As alternativas encontradas pelo grupo foram a manipulação direta do modelo de mão criação dos gestos e animações ou o uso do Leap Motion para captura dos movimentos e de outra ferramenta para sua transformação em uma animação aplicável ao modelo no Unity. A segunda alternativa era preferível, pois permitiria a captura de movimentos mais naturais e fluidos. Mas caso não fosse possível, a primeira seria adotada. O grupo avaliou a viabilidade dessas soluções.

a) Leap Motion

   O dispositivo Leap Motion é um sensor desenvolvido para a captura de movimento de mão e dedos. As engines Unreal e Unity possuem suporte para o Leap Motion e conseguem integrar o movimento capturado com o ambiente de desenvolvimento.

   O Leap Motion possui alguns modelos próprios de mão utilizados nas aplicações de captura e reprodução de movimentos manuais do *plugin* do Leap Motion para Unity. O grupo tentou realizar a acoplação da captura realizada pelo Leap Motion com o modelo de mão usado no projeto. Entretanto, as tentativas realizadas não foram bem sucedidas. O esqueleto do modelo utilizado conseguia acompanhar corretamente o modelo do Leap Motion. Entretanto, ao adicionar o *mesh* o modelo ficava totalmente distorcido, sendo impossível distinguir os gestos sendo feitos.

b) MotionBuilder

   Após a tentativa de usar o Unity para captura do Leap Motion, foi feito mais uma tentativa utilizando um segundo software chamado MotionBuilder. O MotionBuilder permite criar animações em 3D e também possui um *plugin* para Leap Motion. Similar ao Unity, o MotionBuilder permite vincular a captura dos movimentos da mão com um modelo inserido no ambiente de desenvolvimento. Entretanto, ocorreu o mesmo problema do Leap Motion.

c) Animações Unity

   Por fim, voltou-se a alternativa inicial. Os gestos e movimentos das mãos foram manualmente modelados usando o próprio Unity. Para os gestos em que há animação, basta definir as posições dos dedos em instantes de tempo marcados, que o próprio Unity se encarrega de fazer a interpolação, gerando o efeito de animação.

5.2 - Integração AR

O desenvolvimento do projeto foi iniciado com a escolha da engine a ser utilizada para a implementação. As engines Unreal e Unity foram avaliadas, sendo que o critério para a escolha é a facilidade de integração com o sistema de AR e com o dispositivo móvel. Um requisito inicial é que a

2 Modelo de mão abaixado de https://www.turbosquid.com/3d-models/hand-rigged-3d-model/735487

Engine seja capaz de se comunicar com um dispositivo iOS, pois o dispositivo o único dispositivo disponível para utilização pelo grupo que era suportado pelas bibliotecas AR é um iPhone 6s.

Começamos utilizando o Unreal, que dispõe de duas bibliotecas[3]: o ARKit (iOS) e o ARCore (Android). Como o ARCode requer alguns modelos específicos de celular para funcionar, os quais não possuímos, tivemos de usar o ARKit. Infelizmente, não foi possível utilizá-lo em nosso projeto, pois ele requer uma licença para desenvolvimento iOS somente obtível por meio de um registro pago de desenvolvedor. Dessa forma, o Unreal deixou de ser uma alternativa viável para o nosso projeto.

Migramos para o Unity, que há integração com uma engine de AR chamada Vuforia[4], e fácil integração com *build* de aplicativos para iOS. Felizmente o Unity permite exportar o projeto diretamente para o XCode. Dessa forma foi possível elaborar o programa com AR e executá-lo no celular utilizando um registro pessoal de desenvolvedor, que não é pago. Tal registro permite que o aplicativo seja colocado em apenas um dispositivo e o aplicativo expira a cada 7 dias, sendo necessário fazer novo *build* e *deploy* no dispositivo móvel. Como para uma POC essas restrições são aceitáveis, optamos pelo Unity como engine para desenvolvermos a aplicação.

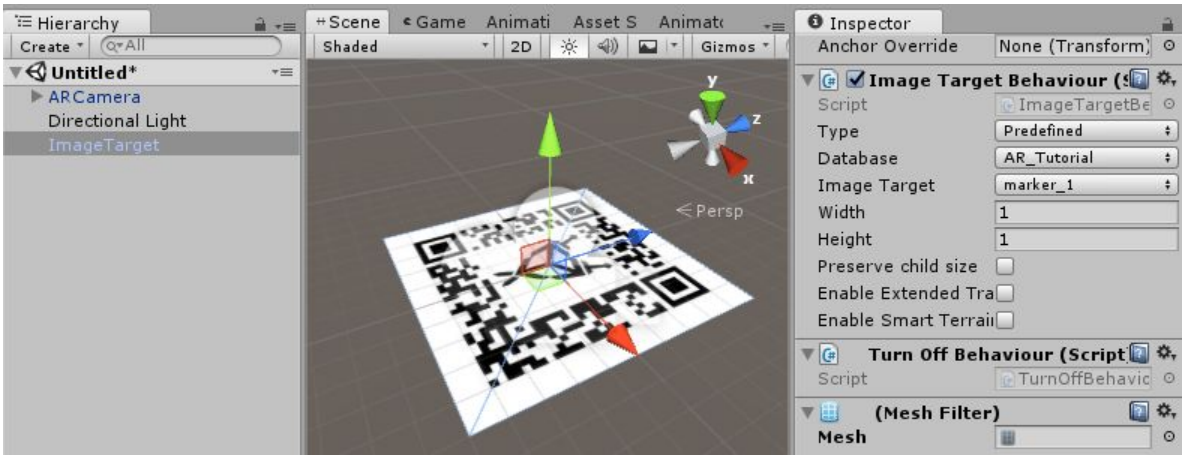


Figura 4 - Exemplo de associação entre modelo 3D e Image Target

Utilizou-se um projeto padrão do Vuforia como base do projeto[5]. Segui-se passo a passo o tutorial para construção de um projeto inicial que funcionasse, para que depois fossem adicionados nossos modelos. A estrutura do aplicativo em si (tela de splash, menus, etc) é complexa e seu entendimento não é completamente dominado pelo grupo. Para adaptá-la ao aplicativo da POC, conseguiu-se modificar o menu, deixando apenas um campo com uma pequena descrição do projeto. Entretanto, não foi possível retirar, por exemplo, a tela de splash do Vuforia. Portanto, devido à limitações técnicas do grupo, não foi possível adaptar a usabilidade do aplicativo para o que acredita-se que seria melhor do contexto desse projeto.

A forma de utilização do Vuforia é bastante intuitiva. Adicionam-se Image Targets (GameObject > Vuforia > Image), ou seja, componentes que conectam um alvo / cartão (Figura 4) a um modelo a ser

[3] https://docs.unrealengine.com/en-us/Platforms/AR/HandheldAR/ARQuickStart
[4] https://www.vuforia.com/
[5] https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html

mostrado quando o alvo for detectado. Após inserir os Image Targets para todas os sinais manuais que modelamos, o resultado é o mostrado na Figura 5. Nela podemos ver alguns modelos de mãos associados a figuras, que correspondem às imagens dos cartões das letras. Dessa forma, sempre que a câmera do celular detectar uma das figuras dos cartões aparecerá o modelo correspondente, com as devidas animações, sobre a imagem do cartão. A disposição dos modelos sobre as figuras (distância, proporção de tamanhos e orientação) corresponde à forma como aparecem na própria visualização do Unity.

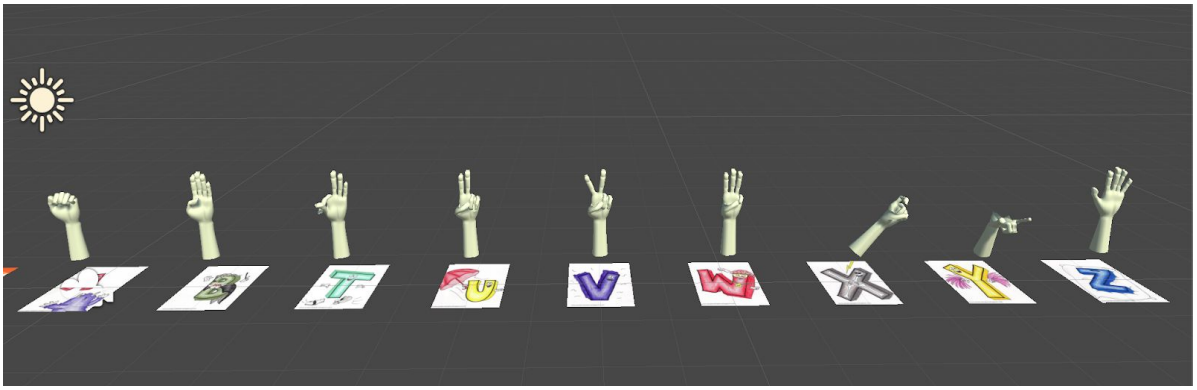


Figura 5 - Modelos 3D das mãos e seus respectivos cartões

5.3 - Modelo de Mãos

Para o modelo de mãos, utilizamos um asset disponível gratuitamente no site TurboSquid[6], chamado "Hand Rigged", conforme a figura abaixo. Com ele, geramos as animações e gestos referentes aos sinais de LIBRAS. Os novos modelos criados foram incluídos no projeto Vuforia para a projeção do modelo correspondente na letra do marcador. Com esse asset, geramos as animações das letras utilizando o próprio Unity.

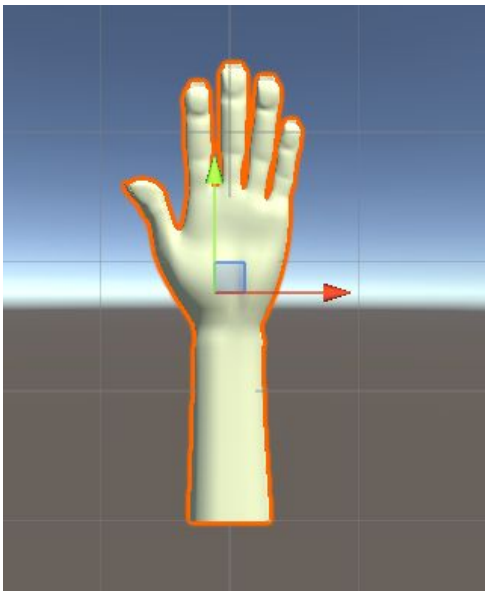


Figura 6 - Modelo "*Hand Rigged*"

[6] https://www.turbosquid.com/3d-models/hand-rigged-3d-model/735487

Um problema estético desse modelo são as unhas. Elas são muito grandes, e achamos que isso pode ser aversivo para as crianças. Utilizando o Blender, encolhemos as unhas para que elas não ficassem tão exageradas. O resultado dessa mão editada foi satisfatório. Infelizmente, o arquivo editado no Blender e exportado não foi aceito pelo Unity. Embora pudéssemos ver o modelo dentro do cenário, as animações não aconteciam. Dessa forma, tivemos de desconsiderar a edição nas unhas pelo Blender.

5.4 - Cartões (Targets)

A primeira ideia de cartão foi usar QR Codes que continham as letras nas bordas, conforme a Figura 7.

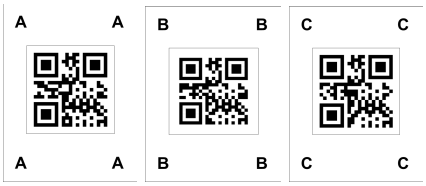


Figura 7 - Cartões com QR Code

A interface do Vuforia para inserção de imagens fornece uma espécie de avaliação para cada imagem que colocamos como *target*. Os QR Codes tiveram nota máxima. Entretanto, quando dispusemos vários cartões e usamos a aplicação VR, vimos que para um mesmo cartão varios modelos de mãos diferentes apareciam. Isso se deve ao fato de que essa avaliação diz o quão boa uma imagem é para ser identificada, e não quão boa é uma imagem para identificar um único modelo. A avaliação do Vuforia não leva em conta a distinção entre imagens. Acontece que, como os QR Codes são muito parecidos macroscopicamente (pois as diferenças são ao nível dos pixels), o Vuforia associa um cartão visualizado com vários targets possíveis. Dessa forma, os QR Codes foram abandonados.

Utilizamos uma segunda opção de imagens de cartões[7], de forma a garantir que as imagens sejam macroscopicamente diferentes também. As cores também ajudam bastante a distinguir um cartão dos outros.

Figura 8 - Cartões figuras coloridas

Fazendo o teste, conseguimos distinguir nitidamente as letras umas das outras. Optamos finalmente então por usar essas figuras coloridas para os cartões. A página onde estão os cartões não menciona explicitamente os direitos de uso. Entretanto, o autor diz “Estar compartilhando os cartões que fez” s.i.c. “I'm sharing the ones I made”. Logo entendemos que o uso desses cartões é livre.

[7] http://projectsbyjess.blogspot.com/2010/09/leap-frog-letter-factory-flash-cards.html

5.5- Botões de interface com usuário

A interface com usuário desenvolvida apresenta dois tipos de botões que realizam ações durante a execução da aplicação. O primeiro possibilita a interrupção e retomada das animações dos sinais de libras e o segundo permite ao usuário trocar a coloração do modelo da mão utilizado (Figura 9, *esq*). O segundo permite a mudança da cor do modelo (Figura 9, *dir*). Ele simula a possível mudança da "cor de pele" do modelo, desejada para melhorar a experiência do usuário através da possibilidade de visualizar uma mão similar à dele.

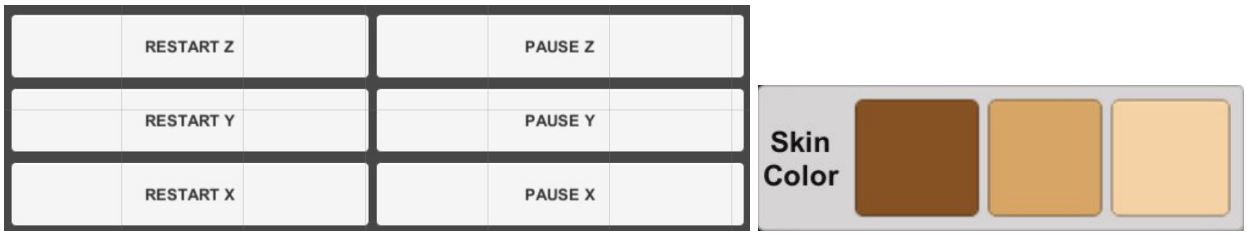


Figura 9 - Botões para reiniciar e parar a animação dos modelos com movimento (*esq*). Botões para mudança da "cor de pele" do modelo de mão (*dir*).

Os botões de animação foram vinculados às letras X, Y e Z, pois essas são as únicas letras implementadas que possuem algum tipo de animação a ser realizada. Cada letra também possui um botão que permite pausar/retomar e um que permite reiniciar a animação.

Os botões são configurados de forma a operar em cima de um modelo de mão animada realizando uma função específica. Um script de C# que implementa as funções de pausar/retomar e reiniciar animação é acoplado ao modelo de cada mão para que os botões possam realizar essas funções.

6. Instalando e executando o projeto

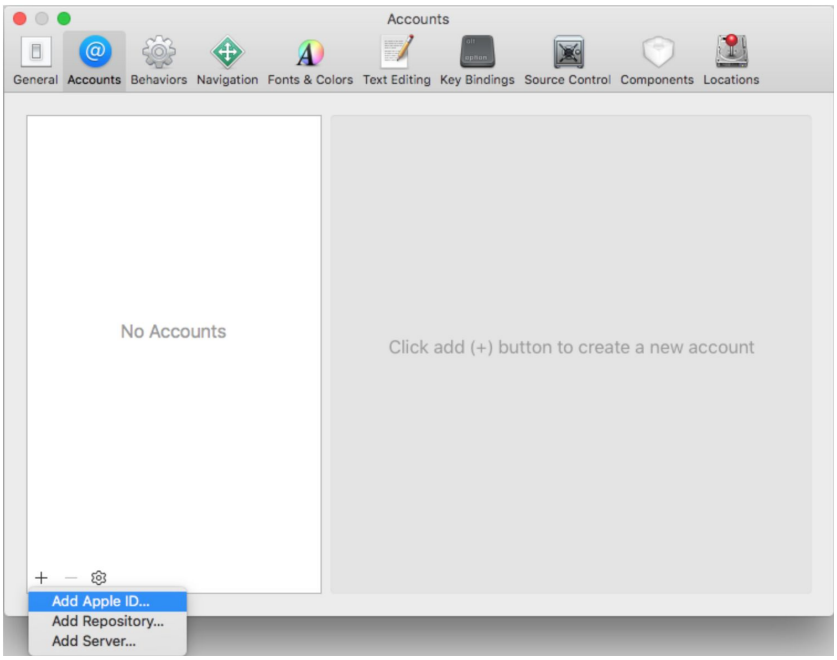
Este capítulo tem como objetivo descrever os passos para utilização do projeto desenvolvido. O repositório se encontra no *link* do Github disponibilizado no final deste relatório e contém os programas desenvolvidos através do software Unity junto com o pacote de desenvolvimento Vuforia.

A seguir iremos descrever as etapas de configuração do projeto.

1. O projeto o Unity na versão mínima 2018.2.13f1 com os componentes: Microsoft Visual Studio, suporte para build em iOS e da engine Vuforia Augmented Reality.
2. Após clonar o repositório, abra o Unity e selecione a pasta "*VuforiaUnityConnectTest*" que contém o projeto de AR implementado.
3. A cena do projeto contém os diversos modelos de mão desenvolvidos pelo grupo junto com os botões implementados para a tela.
4. A simulação do projeto pode ser realizada no próprio computador através do botão de play localizado no menu superior. A *webcam* do computador será utilizada para realizar o reconhecimento dos cartões de realidade aumentada. Caso se queira testar o aplicativo no celular, deve-se seguir os passos descritos no próximo parágrafo.

Após configurado o projeto, iremos descrever o processo de *build* do projeto em uma aplicação iOS.

1. Caso possua-se uma conta de developer no Apple Developer Provisioning Portal, deve-se cadastrá-la no aplicativo. Caso contrário, é necessário possuir o Xcode também instalado e gerar um certificado pessoal (license).
    a. Para gerar a license pessoal, entre em Xcode > *Preferences*. Na aba *Accounts* clique no botão inferior esquerdo de "+" e selecione "Add Apple ID".



    b. Após colocar seu login e senha, seu Apple ID aparecerá na lista da esquerda. Clique nele para ver mais informações.
    c. Do lado direito, abaixo do título Team, você deve ser capaz de ver o seu nome e entre parênteses "Personal Team". Está tudo pronto. Para mais detalhes ou em casos de problemas consultar o guia utilizado pelo grupo[8].
2. Em File > Building Settings selecione a plataforma iOS e clique em *Build.*
3. Será gerada uma pasta de *build* com um projeto *Xcode*. Abrir esse projeto.
4. O projeto gerado será então importado para o Xcode para prosseguir com o *deploy* no dispositivo.
5. O projeto necessita ser configurado com uma conta de desenvolvedor da Apple para que o dispositivo possa confiar na aplicação instalada. Então, clicando sobre o nome do projeto, deve-se alterar a configuração *Signing*, selecionando a opção "*Automatically manage signing*" e escolher seu "*Personal Team*" para o campo *Team*.
6. O próximo passo é conectar o dispositivo móvel iOS ao computador por meio de um cabo (USB, por exemplo). Seleciona-se o dispositivo como alvo (*target*) do build e seleciona-se *run*. O aplicativo será instalado no aparelho. Ele poderá ser usado por 7 dias, necessitando, após esse período, um novo build (devido a *license* pessoal).

8 https://unity3d.com/learn/tutorials/topics/mobile-touch/building-your-unity-game-ios-device-testing?playlist=17138

7. Links Úteis e Referências

1. Projeto AlfabetizaLibras no Github. Disponível em: <https://github.com/LauraLassance/alfabetizaLIBRAS.git>
2. Vídeo demonstrativo disponível na raiz do GitHub.
3. Tutorial de AR no Unreal usando ARKit e ARCore. Disponível em: <https://docs.unrealengine.com/en-us/Platforms/AR/HandheldAR/ARQuickStart>
4. Getting Started with Vuforia Engine in Unity - Vuforia Developer Library. Disponível em: <https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html>
5. Unity AR Tutorial: Augmented Reality Game Development with Vuforia - The Knights of Unity. Disponível em: <https://blog.theknightsofunity.com/unity-vuforia-guide/>
6. Modelo de Mãos “Hand Rigged”. Disponível em <https://www.turbosquid.com/3d-models/hand-rigged-3d-model/735487>
7. Leap Frog Letter Factory Flash Cards - Running With Scissors. Disponível em <http://projectsbyjess.blogspot.com/2010/09/leap-frog-letter-factory-flash-cards.html>