UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

LIBRAR - CONCEITOS BÁSICOS DE LIBRAS USANDO REALIDADE AUMENTADA E REALIDADE VIRTUAL

LUAN RIBEIRO DA SILVA

BLUMENAU
2018

LUAN RIBEIRO DA SILVA

LIBRAR - CONCEITOS BÁSICOS DE LIBRAS USANDO REALIDADE AUMENTADA E REALIDADE VIRTUAL

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof(a). Dalton Solano dos Reis, M.Sc. – Orientador

Prof(a). Fernanda Pelence, Especialista - Coorientadora

BLUMENAU
2018

LIBRAR - CONCEITOS BÁSICOS DE LIBRAS USANDO REALIDADE AUMENTADA E REALIDADE VIRTUAL

Por

LUAN RIBEIRO DA SILVA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Dalton Solano dos Reis, M.Sc. – Orientador, FURB
Membro:	Prof(a). Nome do(a) Professor(a), Titulação – FURB
Membro:	Prof(a). Nome do(a) Professor(a), Titulação – FURB

Blumenau, dia de mês de ano [data da apresentação]

Dedico este trabalho a minha família.

AGRADECIMENTOS

Aos meus pais pelo incentivo durante o curso.

Ao meu orientador pela ajuda oferecida em todo o desenvolvimento deste trabalho.

A co-orientadora pela toda ajuda fornecida com o seu conhecimento em Libras.

O sábio nunca diz tudo o que pensa, mas pensa sempre tudo o que diz.

Aristóteles

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema que utiliza Realidade Virtual e ~~Realidade~~ Aumentada para auxiliar no ensino das letras do alfabeto e algarismos numéricos em Libras. Para isso, o sistema foi dividido em três módulos, sendo o primeiro com o objetivo mostrar ao usuário as letras e algarismos numéricos em Libras através de uma mão 3D. E os outros dois módulos são jogos para que o usuário treine seus conhecimentos em Libras através da Realidade Virtual e ~~Realidade~~ Aumentada. O sistema foi desenvolvido utilizando a plataforma de criação de jogos e aplicativos Unity, em conjunto com a biblioteca de Realidade Aumentada Vuforia e de Realidade Virtual Google VR. O tipo de Realidade Aumentada usada para o sistema foi visão por vídeo baseada em monitor. E o tipo de Realidade Virtual usada para o sistema é imersiva. Durante a implementação foram feitos teste e modificações no sistema para que a biblioteca Vuforia e Google VR trabalhassem juntas em um mesmo sistema. Também são apresentados conceitos, códigos e ideias usadas para o desenvolvimento do sistema e modelagem da mão 3D. Como resultado o sistema foi exportado para a plataforma Android e IOS, possibilitando a utilização para testes com um grupo de alunos. Durante os testes, foi observado uma certa dificuldade no uso das tecnologias de Realidade Aumentada e Virtual, porém todos os alunos conseguiram usar o sistema e se divertiam com as tarefas que lhes eram passadas. Com os resultados dos testes foi possível notar que os objetivos foram alcançados e os usuários tiveram um grande interesse no sistema, por possibilitar uma forma mais tecnológica de se aprender Libras.

Palavras-chave: Libras. Realidade aumentada. Realidade virtual. Unity.

ABSTRACT

This work presents the development of a system that uses Virtual Reality and Augmented Reality to aid in the teaching of alphabet letters and numerical figures in Libras. For this, the system was divided into three modules, the first one with the purpose of showing the user the letters and numerals in Libras using a 3D hand. And the other two modules are games for the user to train their knowledge in Libras through Virtual Reality and Augmented Reality. The system was developed using the Unity game and application creation platform, in conjunction with the Vuforia Augmented Reality library and Google VR Virtual Reality library. The type of Augmented Reality for the system was monitor-based video vision. And the type of Virtual Reality used for the system is immersive. During the implementation, tests and modifications were made to the system so that the Vuforia and Google VR libraries worked together on the same system. Also presented are concepts, codes and ideas used for the development of the 3D hand modeling system. As a result the system was exported to the Android and IOS platform, enabling the use for testing with a group of students. During the tests, a certain difficulty was observed in the use of Augmented and Virtual Reality technologies, but all the students were able to use the system and enjoy the tasks that were done to them. With the results of the tests it was possible to notice that the objectives were reached and the users had a great interest in the system, for enabling a more technological way of learning Libras.

Key-words: Libras. Increased reality. Virtual reality. Unity.

LISTA DE FIGURAS

Figura 1 - Alfabeto e algarismos numéricos em LIBRAS 16
Figura 2 - Inserindo objetos virtuais no mundo real 17
Figura 3 - Posicionando objetos virtuais sobre marcadores 18
Figura 4 - Ciclo de processamento do sistema de RV 19
Figura 5 - Exemplo de RV imersiva com HMD 20
Figura 6 - Exemplo de RV não-imersiva com monitor 20
Figura 7 - Marcador fiducial vazado 21
Figura 8 - Jogo em andamento com suas diversas situações 22
Figura 9 - Marcadores fiduciais 23
Figura 10 - Iniciando o jogo 23
Figura 11 - Associação correta entre os marcadores 24
Figura 12 - Tela inicial do jogo para realizar as configurações 25
Figura 13 - Inserindo novas imagens no sistema 25
Figura 14 - Primeiras etapas do jogo com alguns itens sobrepostos 26
Figura 15 - Diagrama de casos de uso 29
Figura 16 - Parte do diagrama de classes 30
Figura 17 - Parte do diagrama de classes 32
Figura 18 - Página de cadastro da *License Key* 34
Figura 19 - Cadastro dos marcadores 34
Figura 20 - Associando o *database* e o marcador ao *target* 35
Figura 21- Configurações do Vuforia 36
Figura 22 - Configuração da RV no projeto 38
Figura 23 - Exemplo de marcadores do sistema 39
Figura 24 - Molde da mão 3D 39
Figura 25 - Molde da mão 3D com esqueleto 40
Figura 26 - `Animation` referente a letra Z 41
Figura 27 - `Animator Controller` do sistema 41
Figura 28 - Estrutura da mão 3D 42
Figura 29 - Componente `Animator` do `prefab` da mão 3D 42
Figura 30 - Diagrama de atividades da tela inicial 43
Figura 31 - Tela inicial do sistema 43

Figura 32 - Estrutura da tela inicial 44
Figura 33 - Configurações do *script* `Canvas Scaler` 44
Figura 34 - Diagrama de atividades do módulo Aprender os Sinais 45
Figura 35 - Tela para o usuário escolher o conteúdo do módulo 45
Figura 36 - Tela do módulo Aprender os Sinais 46
Figura 37 - Estrutura do módulo Aprender os Sinais 47
Figura 38 - Diagrama de atividades do módulo Jogo Associativo 50
Figura 39 - Exemplo de tela do módulo Jogo Associativo 51
Figura 40 - Estrutura do módulo Jogo Associativo 52
Figura 41 - Atributos do `ImageTarget MarkerSignal1` 53
Figura 42 - Diagrama de atividades do módulo Jogo Raciocínio Rápido 56
Figura 43 - Exemplo de tela do módulo Jogo Raciocínio Rápido 57
Figura 44 - Estrutura do módulo Jogo Raciocínio Rápido 58
Figura 45 - Componentes do `prefab Hand1` 59
Figura 46 - *Script* `ProgressRadialBehaviour` do `prefab ProgressRadialPlain` 60
Figura 47 - Vídeos dos sinais executados pela mão 3D 69
Figura 48 - Mostrando o sistema 70
Figura 49 - Professora e sua aluna usando o segundo módulo 70
Figura 50 - Aluna usando o terceiro módulo 71
Figura 51 - Mostrando o sistema para os alunos 72
Figura 52 - Alunos usando o sistema 72
Figura 53 - Vídeos mostrando a forma correta de alguns sinais 73

LISTA DE QUADROS

Quadro 1 - Significado dos Parâmetros das línguas de sinais 16
Quadro 2 - Tipos de sistemas de RA 18
Quadro 3 - Relação entre RF e casos de uso 29
Quadro 4 - Métodos para inicializar o Vuforia 37
Quadro 5 - Métodos para habilitar e desabilitar a RV no projeto 38
Quadro 6 - Métodos para escolher o conteúdo do módulo 48
Quadro 7 - Métodos para avançar, retornar e executar a animação do sinal 49
Quadro 8 - Método para escolher os sinais aleatoriamente 54
Quadro 9 - Métodos para detectar colisão dos marcadores 55
Quadro 10 - Método para executar a animação ao detectar o marcador 56
Quadro 11 - Métodos referentes as ações do `GvrReticlePointer` 60
Quadro 12 - Comparativo entre os trabalhos correlatos 63

LISTA DE ABREVIATURAS E SIGLAS

HMD – Head Mounted Display

LIBRAS – Língua Brasileira de Sinais

RA – Realidade Aumentada

RF – Requisito Funcional

RNF – Requisito Não Funcional

RV – Realidade Virtual

SDK – Software Development Kit

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVOS 14
1.2 ESTRUTURA 14
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 LIBRAS 15
2.2 REALIDADE AUMENTADA 17
2.3 REALIDADE VIRTUAL 18
2.4 TRABALHOS CORRELATOS 20
2.4.1 Realidade aumentada como ferramenta de apoio na alfabetização de crianças com surdez usuárias da língua brasileira de sinais 21
2.4.2 Aprendendo números em libras com a tecnológica da realidade aumentada 22
2.4.3 Jogando com a realidade aumentada e aprendendo libras 24
3 DESENVOLVIMENTO 27
3.1 CENÁRIO 27
3.2 REQUISITOS 27
3.3 ESPECIFICAÇÃO 28
3.3.1 Diagrama de Casos de Uso 28
3.3.2 Diagrama de Classes 30
3.4 IMPLEMENTAÇÃO 33
3.4.1 Técnicas e ferramentas utilizadas 33
3.5 ANÁLISE DOS RESULTADOS 60
3.5.1 Ideia inicial do sistema 60
3.5.2 Experimento e resultado do sistema 61
3.5.3 Comparação com os trabalhos correlatos 63
4 CONCLUSÕES 65
4.1 EXTENSÕES 65
REFERÊNCIAS 67
APÊNDICE A – VÍDEOS DAS ANIMAÇÕES FEITAS NA MÃO 3D 69
APÊNDICE B – REUNIÃO SOBRE O SISTEMA DESENVOLVIDO 70
APÊNDICE C – TESTE DO SISTEMA NA ABADA 72
ANEXO A – VÍDEOS DOS SINAIS NA LIBRAS 73

1 INTRODUÇÃO

Na década de 1850, um professor surdo francês chamado Ernest Huet chega ao Brasil, trazendo com ele o alfabeto manual francês e alguns sinais. Nessa época os surdos/mudos brasileiros ainda não possuíam um sistema de sinais próprio para se comunicar, e com a chegada da Língua de Sinais Francesa (LSF), foi criada a LIBRAS (MONTEIRO, 2006, p. 296).

De acordo com Lopes (2013, p. 23), "[...] Libras é a língua utilizada como meio de comunicação pelas pessoas Surdas no Brasil. Trata-se de uma língua que não é universal, portanto, cada país possui a sua [...]". A LIBRAS é uma modalidade gestual-visual, porque para comunicar-se através dela, é usado gestos e expressões faciais que são percebidos pela visão. Ao realizarmos a comparação com a Língua Portuguesa, que por sua vez é uma língua oral-auditiva, é usado como meio de comunicação sons articulados que são percebidos pelos nossos ouvidos (REVISTA DA FENEIS, 1999, p. 16).

Com isso, temos o conceito de escola bilíngue, que segundo Marques, Barroco e Silva (2013, p. 514), "[...] escola que se propõe bilíngue e que oportuniza a experiência de inclusão de alunos surdos deve apresentar seus conteúdos, simultaneamente, em língua portuguesa (oral e escrita) e em Libras.". Marques, Barroco e Silva (2013, p. 515) fazem a seguinte teorização:

> Em uma intervenção prática de ensino de Libras que realizamos em 2012, para crianças ouvintes e uma criança surda, em um Centro de Educação Infantil, notamos fatos relevantes. Um deles refere-se à interação da aluna surda com os demais colegas de classe. Após algumas aulas de Libras observamos que houve um aumento significativo na frequência do seu uso na comunicação entre as crianças. Nessa experiência, o ensino dessa língua se deu empregando o próprio conteúdo programático da educação infantil previsto para a turma. Outro fato diz respeito ao emprego de alguns sinais em Libras, por crianças ouvintes ao se comunicarem com outras também ouvintes. Juntamente com a comunicação oral elas se comunicavam também pela Libras.

Segundo Forte e Kirner (2009, p. 1), "Pensar na adoção de recursos tecnológicos como ferramentas facilitadoras no processo educacional pode ser encarado hoje como uma tarefa comum.". Nesse ponto entramos com a RA, que nos possibilita inserir objetos virtuais no nosso ambiente físico em tempo real através de algum dispositivo tecnológico, como por exemplo um smartphone com uma câmera (KIRNER, C.; KIRNER, T., 2007 apud FORTE; KIRNER, 2009, p. 2). Junto com a RA, temos a RV, que possui um conjunto de técnicas e ferramentas gráficas 3D que leva o usuário a um ambiente totalmente virtual gerado por um computador, podendo ainda realizar interações com esse ambiente em tempo real, tudo isso com quase nenhuma percepção que o ambiente onde está não é real (LESTON, 1996, p. 12-

13). Com isso, a RA e RV proporcionam um poder muito grande de ilustração comparado com outras mídias, disponibilizando a oportunidade de realizar experiências e permitir o desenvolvimento do educando no seu próprio ritmo (PANTELIDES, 1995 apud FORTE; KIRNER, 2009, p. 3).

Diante do exposto, desenvolveu-se um sistema para dispositivos móveis que demonstra os conceitos básicos de LIBRAS de uma forma divertida e de fácil entendimento para as crianças, juntando a RA e RV com jogos para que as crianças conheçam a Libras e se divirtam ao mesmo tempo.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar um sistema na plataforma móvel para o aprendizado dos conceitos básicos da LIBRAS.

Os objetivos específicos são:

a) disponibilizar ao usuário a possibilidade de visualizar todas as letras do alfabeto e algarismos numéricos em uma mão virtual 3D;
b) disponibilizar ao usuário um jogo usando RA para poder entender os conceitos básicos de LIBRAS;
c) disponibilizar ao usuário um jogo usando RV para o mesmo ter um ponto de vista diferente do sinal em LIBRAS enquanto entende os conceitos básicos.

1.2 ESTRUTURA

Este trabalho encontra-se divido em quatro capítulos principais. O primeiro apresenta a introdução e o objetivo geral e específicos. O segundo capítulo apresenta a fundamentação teórica que serve de base para o trabalho. No terceiro capítulo é demonstrado questões referentes a implementação do sistema. São apresentados os requisitos e diagramas do sistema. São apresentadas imagens do sistema, demonstrando suas funcionalidades e alguns trechos de código quando necessário para complementar a explicação da lógica usada. Por fim, neste capítulo são apresentadas as análises dos resultados, realizando comentários sobre os testes executados, e uma comparação com os trabalhos correlatos. No quarto capítulo, é apresentada a conclusão do trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção destina-se a apresentar fundamentos e ideias dos assuntos que são necessários para sustentar o projeto desenvolvido. A seção 2.1 descreve sobre o conceito de LIBRAS. Na seção 2.2 é destinada a falar sobre a RA, mostrando conceitos e alguns exemplos. A seção 2.3 comenta-se sobre a RV, mostrando suas características e exemplos. Por fim, a seção 2.4 mostra os trabalhos correlatos.

2.1 LIBRAS

A LIBRAS hoje em dia é a língua oficial das pessoas surdas, conforme descrito Brasil (2002):

> Parágrafo único. Entende-se como Língua Brasileira de Sinais – Libras a forma de comunicação e expressão, em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil.

De acordo com Schlünzen, Benedetto e Santos (2012, p. 46), “As línguas de sinais são chamadas de gestual-visual porque o responsável para emitir a comunicação são as mãos por meio dos sinais, e o receptor são os olhos.”. A LIBRAS pode ser usada por pessoas surdas que entendem os sinais através da visão, por surdo-cegos que captam os sinais segurando a mão do emissor para poder entender, e até por surdos que não possuem os braços fazendo os sinais com seus pés de forma adaptada (SCHLÜNZEN; BENEDETTO; SANTOS, 2012, p. 46).

Segundo Cechinel (2005, p. 32), “A Libras possui estrutura e gramática própria e status linguístico completo, possibilitando expressar não apenas conceitos concretos, mas também abstratos, assim como qualquer outro idioma.”. De acordo com Silva (2011, p. 2), cada sinal na língua de sinais pode ser composto de até cinco componentes chamados de Parâmetros, que são: configuração de mão, ponto de articulação, movimento, orientação e expressão não-manuais. O Quadro 1 mostra o significado de cada um desses Parâmetros mencionados por Silva (2011, p. 2-9).

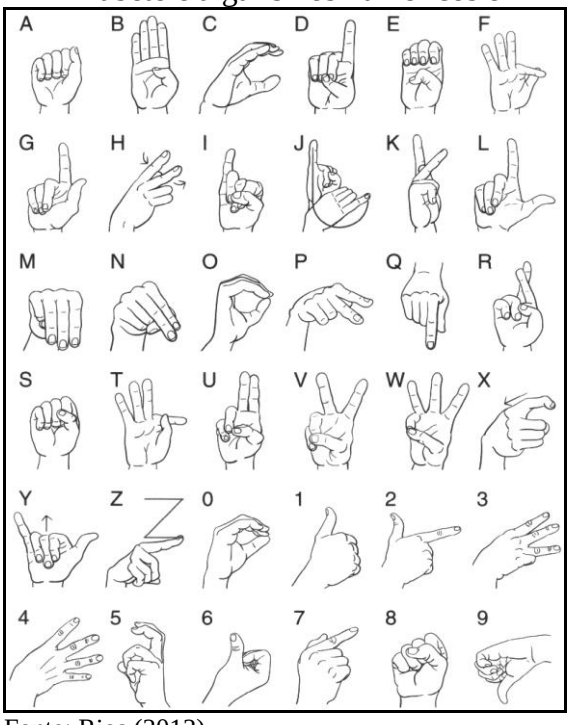
Quadro 1 - Significado dos Parâmetros das línguas de sinais

Parâmetro	Significado
Configuração de mão	É a forma da estrutura da mão que deriva o sinal. A LIBRAS tem aproximadamente 70 configurações de mão.
Ponto de articulação	É a área do corpo na qual ou próxima da qual é realizado o sinal. Nós temos quatro áreas principais: cabeça, mão, tronco e braço (QUADROS; KARNOPP, 2004, p. 57).
Movimento	É o movimento no espaço realizado pelas mãos durante o sinal. Podemos ter o movimento interno da mão, do pulso e direcional no espaço (BRITO, 1995).
Orientação	É a direção na qual a palma da mão aponta durante o sinal, podendo ser para cima, para baixo, para dentro, para fora ou para os lados (QUADROS; KARNOPP, 2004, p. 59-60).
Expressão não-manuais	É as expressões faciais e corporais realizados durante o sinal.

Fonte: elaborado pelo autor.

As Línguas de Sinais possuem uma estrutura frasal muito diferente das outras línguas. Na LIBRAS, para formar uma frase usa-se (objeto → verbo → sujeito) ou (objeto → sujeito → verbo), enquanto que na Língua Portuguesa a frase é formada por (sujeito → verbo → objeto). Com isso, a frase "Eu vou para casa" falada no português ficaria "Casa vou eu" em LIBRAS (SCHLÜNZEN; BENEDETTO; SANTOS, 2012, p. 46). Conforme Schlünzen, Benedetto e Santos (2012, p. 46), "Em todas as línguas de sinais, inclusive na Libras, cada palavra é representada por um sinal, por isso é incorreto caracterizar os sinais da Libras como simples gestos ou mímicas, uma vez que se diferem por regras gramaticais específicas.". Porém, cada letra e algarismo numérico também possui seu respectivo sinal em LIBRAS (Figura 1).

Figura 1 - Alfabeto e algarismos numéricos em LIBRAS



Fonte: Rios (2012).

2.2 REALIDADE AUMENTADA

Segundo Cardoso et al. (2007, p. 8):

> Pode-se definir Realidade Aumentada – RA – como a amplificação da percepção sensorial por meio de recursos computacionais. Assim, associando dados computacionais ao mundo real, a RA permite uma interface mais natural com dados e imagens geradas por computador. Um sistema de RA deve prover ao usuário condições de interagir com estes dados de forma natural.

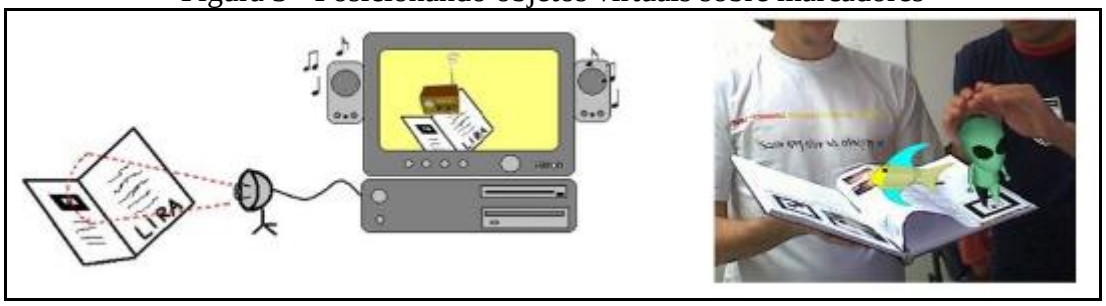
Com isso, RA é basicamente uma interação entre o mundo real e o mundo virtual, realizando isso através da geração de elementos virtuais no mundo real, fazendo o usuário acreditar que aquele elemento virtual faz parte realmente do mundo real. Como existe essa interação com o mundo real, a associação dos objetos virtuais gerados computacionalmente acaba ficando mais natural para os seres humanos, e esse é o grande objetivo da realidade aumentada (CARDOSO et al., 2007, p. 8).

A RA representa, conforme Kirner, C.; Kirner, T. (2011, p. 11), “técnicas de interface computacional que levam em conta o espaço tridimensional. Nesse espaço, o usuário atua de forma multisensorial, explorando aspectos deste espaço por meio da visão, audição e tato.”. Na Figura 2 temos um exemplo de RA, onde foi inserido objetos virtuais no mundo real que podem ser percebidos através de dispositivos tecnológicos, e na Figura 3 temos o uso de um dispositivo tecnológico (webcam) que captura imagens do mundo real, passando para o computador que identifica na imagem os marcadores e posiciona os objetos virtuais sobre eles (KIRNER, C.; KIRNER, T., 2011, p. 15).

Figura 2 - Inserindo objetos virtuais no mundo real

Fonte: Kirner, C. e Kirner, T. (2011, p. 15).

Figura 3 - Posicionando objetos virtuais sobre marcadores



Fonte: Kirner, C. e Kirner, T. (2011, p. 15).

Os sistemas de RA podem ser classificados de quatro formas diferentes, que são sistema de visão ótica direta, sistema de visão direta por vídeo, sistema de visão por vídeo baseado em monitor e sistema de visão ótica por projeção (CARDOSO et al., 2007, p. 9-10). O Quadro 2 descreve cada um desses sistemas.

Quadro 2 - Tipos de sistemas de RA

Sistema	Significado
Visão ótica direta	Utiliza óculos ou capacetes com lentes que recebem imagens do ambiente real ao mesmo tempo que projetam imagens virtuais ajustadas a cena real.
Visão direta por vídeo	Utiliza capacetes com micro câmeras que captam a cena real, passam para o computador que mistura com elementos virtuais e projeta em pequenos monitores acoplados no capacete.
Visão por vídeo baseado em monitor	Utiliza uma webcam que captura o ambiente real, passa para o computador misturar com os objetos virtuais e apresenta a nova imagem no monitor.
Visão ótica por projeção	Utiliza superfícies do ambiente real onde será projetada os objetos virtuais através de um projetor, sem a necessidade de um equipamento junto ao usuário.

Fonte: Cardoso et al. (2007, p. 9-10).

2.3 REALIDADE VIRTUAL

Segundo Kirner e Siscoutto (2007, p. 7):

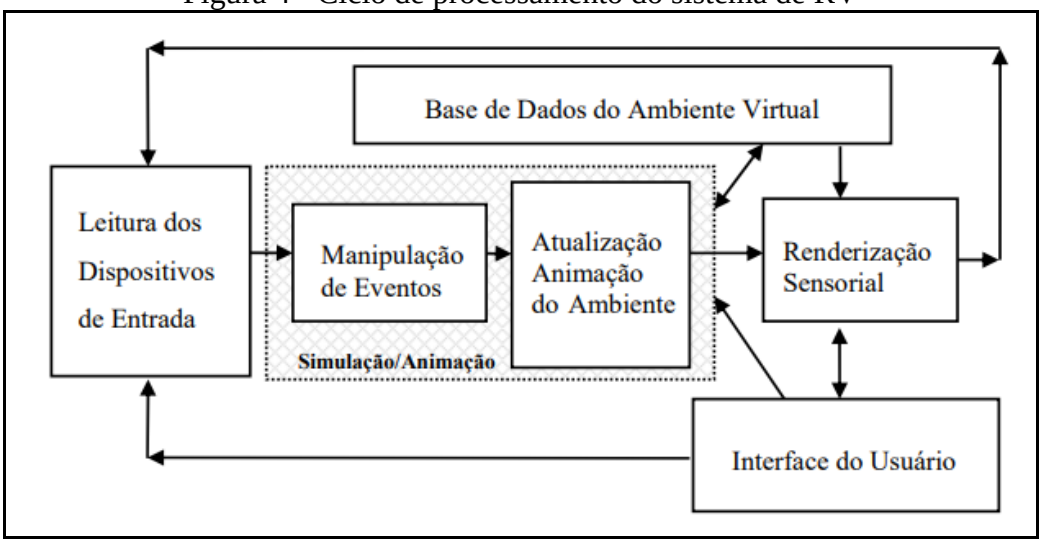
> A Realidade Virtual (RV) é uma "interface avançada do usuário" para acessar aplicações executadas no computador, propiciando a visualização, movimentação e interação do usuário, em tempo real, em ambientes tridimensionais gerados por um computador. O sentido da visão costuma ser preponderante em aplicações de realidade virtual, mas os outros sentidos, como tato, audição, etc. também podem ser usados para enriquecer a experiência do usuário.

Como mencionado acima, o usuário possui várias formas de interação com a RV, e a mais simples que podemos considerar é a movimentação no espaço tridimensional, que pode acontecer através de um mouse 3D, comandos de voz ou a própria movimentação do usuário detectada por um dispositivo de captura (KIRNER; SISCOUTTO, 2007, p. 8). Agora quando falamos de interações mais complexas, como movimentação dos objetos virtuais em tempo real, necessitamos de dispositivos mais complexos, como um capacete de visualização, luvas

eletrônicas, etc. Desta forma o usuário terá impressão de estar no mundo real, porque estará manipulando, pegando e executando ações sobre os objetos virtuais em tempo real (KIRNER; SISCOUTTO, 2007, p. 8).

Além dos dispositivos que o usuário irá usar quando tivermos essas interações mais complexas, o dispositivo que irá realizar todos os cálculos para renderizar esse ambiente tridimensional precisa obedecer algumas taxas mínimas de renderização. Os atrasos admissíveis para visão e reações como tato, força e audição estão em torno de 100 milissegundos, isso significa que o dispositivo terá que rodar com taxas mínimas de 10 quadros por segundo (KIRNER; SISCOUTTO, 2007, p. 8-9). Um sistema de RV possui alguns módulos de processamento como Manipulação de Eventos, Renderização Sensorial, etc., como pode ser observado na Figura 4. Conforme Tori, Kirner e Siscoutto (2006, p. 15), "ciclo de processamento pode ser resumido em: leitura dos dados dos dispositivos de entrada, execução da simulação/animação e renderização sensorial. A renderização sensorial é considerada de forma ampla e engloba: renderização visual, auditiva e háptica.".

Figura 4 - Ciclo de processamento do sistema de RV



Fonte: Tori, Kirner e Siscoutto (2006, p. 15).

De acordo com Tori, Kirner e Siscoutto (2006, p. 8), "Realidade Virtual pode ser classificada, em função do senso de presença do usuário, em imersiva ou não-imersiva". Podemos chamar de RV imersiva quando o usuário é levado de forma inteira para o mundo virtual, através dos dispositivos multissensoriais que iram captar seus movimentos e reações para interagir com o mundo virtual (Figura 5), deixando-o completamente no mundo virtual, porém com impressão que está no mundo real. Agora, a RV não-imersiva é quando o usuário é levado parcialmente para o mundo virtual através de uma janela (monitor por exemplo),

porém ele consegue perceber que está no mundo real (Figura 6) (TORI; KIRNER; SISCOUTTO, 2006, p. 8).

Figura 5 - Exemplo de RV imersiva com HMD

Fonte: Tori, Kirner e Siscoutto (2006, p. 8).

Figura 6 - Exemplo de RV não-imersiva com monitor

Fonte: Tori, Kirner e Siscoutto (2006, p. 8).

2.4 TRABALHOS CORRELATOS

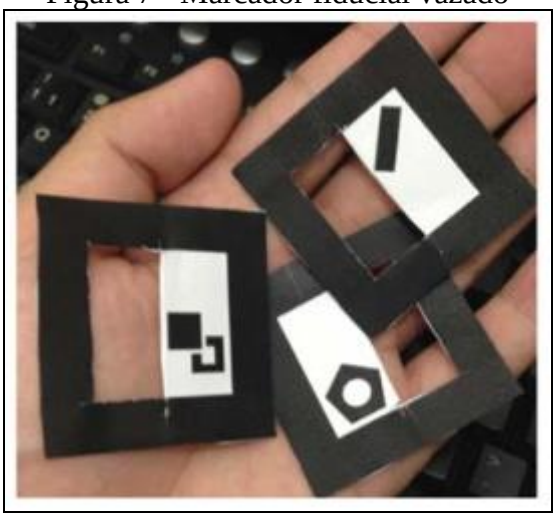
Neste capítulo são apresentados três trabalhos correlatos com características semelhantes aos principais objetivos do estudo proposto. Na seção 2.4.1 é apresentado o artigo de Freire et al. (2015) que desenvolveram um jogo para ajudar a alfabetização de crianças surdas. A seção 2.4.2 detalha o artigo de Santos e Souza et al. (2013), que desenvolveram um jogo para ensinar algarismos numéricos em LIBRAS. Por fim, a seção 2.4.3 descreve o artigo de Santos e Lobo et al. (2013), que desenvolveram um software para os usuários poderem cadastrar seus próprios temas para o aprendizado de LIBRAS.

2.4.1 Realidade aumentada como ferramenta de apoio na alfabetização de crianças com surdez usuárias da língua brasileira de sinais

Segundo Freire et al. (2015, p. 2), “foi desenvolvido um jogo que se vale da realidade aumentada somada a animações 3D, como ferramenta para auxiliar e apoiar o processo de aprendizagem de crianças com surdez usuárias da Língua Brasileira de Sinais.” Para executar o jogo é necessário um computador, uma câmera e alguns marcadores impressos para que as crianças possam interagir com o jogo.

A lógica do jogo segue da distribuição de marcadores fiduciais vazados, como os mostrados na Figura 7, que após serem detectados pela câmera produziram imagens de letras do alfabeto, números e seus correspondentes na LIBRAS. Com isso as crianças podem associar as letras ou números com seus correspondentes na LIBRAS, fazendo isso com a sobreposição dos marcadores fiduciais vazados. Foram usados marcadores fiduciais vazados para que quando o usuário sobrepor os marcadores, o mesmo gere um novo marcador indicando se a sobreposição foi correta ou não (FREIRE et al., 2015, p. 4-5).

Figura 7 - Marcador fiducial vazado



Fonte: Freire et al. (2015, p. 5).

Ao iniciar o jogo os objetos virtuais são carregados em cima dos marcadores, que podem ser as vogais do alfabeto ou algarismos numéricos exibidos em cores diferente, conforme a Figura 8A e 8B. Ou sinais em LIBRAS que são exibidos em todos os lados de um cubo 3D com fundo preto, conforme a Figura 8A. Quando o usuário sobrepoem os marcadores de forma correta, é exibido um cubo 3D com fundo branco que possui em todos os seus lados o respectivo sinal em LIBRAS junto com a letra ou algarismo número indicando que a sobresição foi realizada corretamente (conforme a Figura 8B e 8D). Agora se o marcador foi colocado em um local incorreto é exibido uma cruz vermelha informando o erro (Figura 8A). Na Figura 8C é exibido a associação correta entre os marcadores, porém

nessa é exibido um boneco em 3D que consegue realizar além do sinal a movimentação com os braços que algumas representações de palavras em LIBRAS necessitam.

Figura 8 - Jogo em andamento com suas diversas situações

Fonte: Freire et al. (2015, p. 8).

Para o desenvolvimento deste trabalho, foi utilizado a biblioteca ARToolKit para possiblitar que os objetos 3D fossem renderizados a parti dos marcadores fiduciais. E para desenhar os objetos em 3D foi usuado o software 3ds Max-Autodesk por possuir muitas funcionalidades e também a possibilidade de exportar os objetos para a extensão .wrl, aceitada pelo ARToolKit.

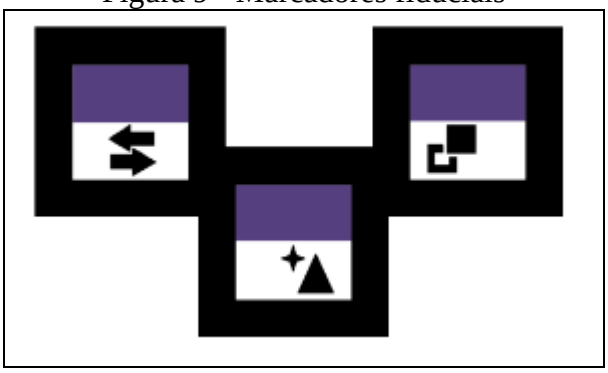
2.4.2 Aprendendo números em libras com a tecnológica da realidade aumentada

Neste trabalho foi desenvolvido um jogo da memória para alunos deficientes auditivos e/ou surdos para auxiliar no ensino dos algarismos numéricos na LIBRAS. O objetivo do jogo consiste em realizar a associação entre os marcadores para o ensino dos algarismos numéricos (SANTOS; SOUZA et al., 2013, p. 22). Para executar o jogo, é necessário somente uma câmera e imprimir os marcadores para a interação dos alunos.

Antes de executar o jogo é necessário imprimir os marcadores fiduciais que serão disponibilizados. Esses terão uma região com a cor lilás (Figura 9) representando a área que precisa ser recortada para se tornar um marcador fiducial. Os marcadores fiduciais são necessários porque quando o usuário realizar a sobreposição deles, irá ser gerado um novo

marcador indicando se foi feita corretamente a associação. Cada marcador pode gerar 5 combinações diferentes, sendo apenas uma correta (SANTOS; SOUZA et al., 2013, p. 22).

Figura 9 - Marcadores fiduciais



Fonte: Santos e Souza et al. (2013, p. 22).

Ao iniciar o jogo e apontar a câmera para os marcadores é exibido os algarismos numéricos e seus respectivos sinais em LIBRAS, todos no formato 2D (Figura 10). Após isso podemos realizar as associações dos objetos virtuais, que ao ser realizado de forma incorreta não é realizada nenhuma alteração nos objetos virtuais. Porém quando a associação está correta, é exibido um cubo 3D com fundo branco que possui em todos os seus lados o algarismo numérico e seu respectivo sinal em LIBRAS (Figura 11).

Figura 10 - Iniciando o jogo

Fonte: Santos e Souza et al. (2013, p. 23).

Figura 11 - Associação correta entre os marcadores

Fonte: Santos e Souza et al. (2013, p. 23).

Segundo Santos e Souza et al. (2013, p. 23), “Essa interação que a RA oferece é bastante motivadora para uma criança e até mesmo um adulto, uma vez que este percebe o seu aprendizado de forma diferenciada com a mistura entre o mundo real e virtual [...]”.

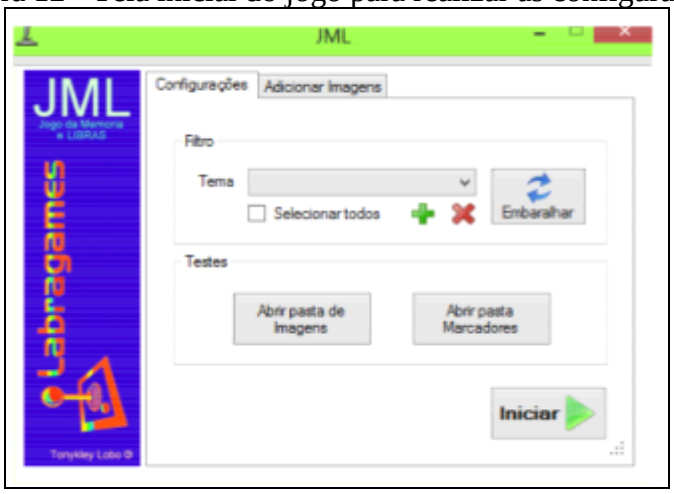
Para o desenvolvimento deste trabalho, foi utilizada a biblioteca ARToolKit que possibilita que objetos virtuais sejam mostrados junto com os marcadores fiduciais. Para modelar os objetos virtuais, foi usado o software Vivaty3D Studio por possuir uma interface intuitiva e exportar objetos no formato .wrl, compreendida pelo ARToolKit.

2.4.3 Jogando com a realidade aumentada e aprendendo libras

Neste trabalho foi desenvolvido um software gratuito que pode ser baixado pela Internet por professores, pais, estudantes ou qualquer pessoa interessada em aprender LIBRAS, sendo necessário ter uma câmera no computador, papel A4 para a impressão dos marcadores e montar pelo software os temas que o jogo irá abordar. Nesse trabalho foi utilizado o alfabeto português associados a objetos do mundo real (SANTOS; LOBO et al, 2013, p. 455-456).

Ao iniciar o software é aberto a tela de configurações, nela é possível escolher um tema já existente, inserir ou remover novos temas clicando nos botões que possui o sinal de “+” e “X” respectivamente. Além disso, podemos também embaralhar as imagens para quando iniciar o jogo não mostrar as mesmas que foram exibidas na última vez, visualizar a pasta que contém as imagens e os marcadores dos temas, e por fim temos o botão para iniciar o jogo. Essa tela pode ser vista na Figura 12.

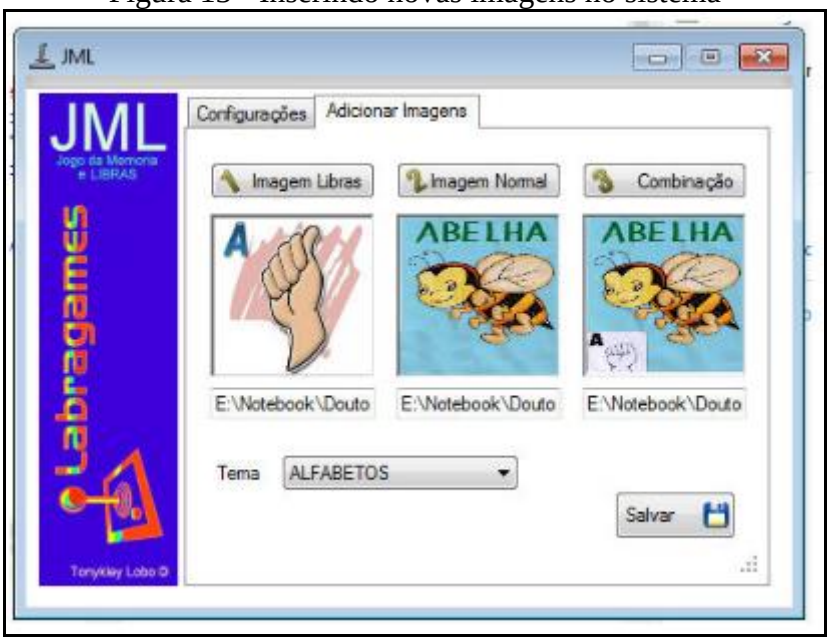
Figura 12 - Tela inicial do jogo para realizar as configurações



Fonte: Santos e Lobo et al. (2013, p. 457).

Como falado anteriormente, esse software disponibiliza para o usuário uma interface para criar os seus próprios marcadores com seus devidos temas. Para isso é necessário informar ao sistema a imagem do sinal em LIBRAS, a imagem do objeto referente ao sinal, uma imagem que será exibida após o usuário associar a imagem do sinal com a do objeto e em qual tema esse conjunto será inserido. Esse processo pode ser observado na Figura 13.

Figura 13 - Inserindo novas imagens no sistema



Fonte: Santos e Lobo et al. (2013, p. 457).

Após ser inserido todas as imagens com seus devidos temas é possível imprimir os marcadores e iniciar o jogo. Ao iniciar o jogo, o usuário terá os marcadores fixos que serão os que contém o sinal em LIBRAS, e os outros contendo a imagem do objeto que precisa ser associado com o seu respectivo sinal, todos exibidos em 2D. Caso a associação estiver incorreta nenhuma mudança será notada. Caso esteja correta, e mostrado um cubo 3D que irá

conter em todos os seus lados a imagem que foi inserida para a respectiva combinação lá nas configurações inicias do jogo. Na Figura 14 é mostrado o jogo em execução.

Figura 14 - Primeiras etapas do jogo com alguns itens sobrepostos

Fonte: Santos e Lobo et al. (2013, p. 457-458).

Para o desenvolvimento deste trabalho, foi utilizada a biblioteca ARToolKit para renderizar os objetos VRML (extensão .wrl) em cima dos marcadores.

3 DESENVOLVIMENTO

Neste capítulo serão apresentados os passos do desenvolvimento do sistema. A seção 3.1 apresenta a ideia geral da ferramenta. A seção 3.2 apresenta os requisitos funcionais e não funcionais do sistema. Na seção 3.3 é apresentado a especificação do sistema. A seção 3.4 apresenta a implementação que foi usada neste trabalho. E na seção 3.5, é apresentado a análise dos resultados obtidos no trabalho.

3.1 CENÁRIO

O cenário proposto para esse sistema está dividido em três módulos. O primeiro módulo é para `Aprender os Sinais`, que tem o objetivo de mostrar para o usuário os sinais na LIBRAS das letras do alfabeto e os algarismos numéricos a partir de uma mão 3D, realizando a animação necessária para o sinal. O segundo módulo é um `Jogo Associativo`, onde o usuário tem o objetivo de associar o sinal na LIBRAS com o seu respectivo significa usando a RA. E o terceiro módulo é o `Jogo de Raciocínio Rápido`, onde o usuário precisa escolher o sinal na LIBRAS correto para a letra ou algarismo numérico gerado aleatoriamente, usando a RV.

3.2 REQUISITOS

O sistema descrito nesse trabalho deverá:

a) utilizar uma mão 3D para exibir o sinal em LIBRAS juntamente com a animação (RF01);
b) possuir um módulo para o usuário poder ver cada letra ou algarismo numérico e seu respectivo sinal em LIBRAS separadamente em ordem alfabética (RF02);
c) disponibilizar ao usuário a possibilidade de rotacionar a mão 3D para poder ver o sinal em diferentes ângulos (RF03);
d) possuir um módulo para o usuário treinar seus conhecimentos em LIBRAS com um jogo associativo de letras ou algarismos numéricos com seu sinal em LIBRAS usando RA (RF04);
e) utilizar oito marcadores de RA para o jogo associativo, sendo quatro deles para exibir o sinal em LIBRAS, e os outros quatro para exibir a letra ou algarismo numérico (RF05);
f) exibir no canto esquerdo da tela os sinais que já foram associados corretamente durante o jogo associativo (RF06);

g) exibir no canto superior esquerdo da tela a quantidade de erros que o usuário cometeu durante o jogo associativo, podendo ser no máximo cinco vezes (RF07);
h) possuir um módulo para o usuário treinar seus conhecimentos em LIBRAS com um jogo de raciocínio rápido com letras ou algarismos numéricos usando RV e HMD (RF08);
i) exibir para o usuário um contador regressivo de vinte e cinco segundos durando o jogo de raciocínio rápido (RF09);
j) exibir para o usuário a quantidade de erros cometidas durante o jogo de raciocínio rápido, podendo ser no máximo duas vezes (RF10);
k) disponibilizar ao usuário, uma tela inicial para escolher se o conteúdo será em cima de letras ou algarismos numéricos (RF11);
l) possuir em cada um dos três módulos botões para o usuário voltar a tela inicial, voltar a tela para escolher o conteúdo do módulo e executar novamente as animações dos sinais (RF12);
m) executar o sistema em dispositivos móveis com sistema operacional Android e iOS (RNF01);
n) utilizar o ambiente de desenvolvimento Unity para o desenvolvimento do sistema (RNF02);
o) utilizar o Vuforia junto com o Unity para o desenvolvimento da RA no sistema (RNF03);
p) utilizar o Google VR junto com o Unity para o desenvolvimento da RV no sistema (RNF04);
q) executar o sistema no modo *offline* (RNF05).

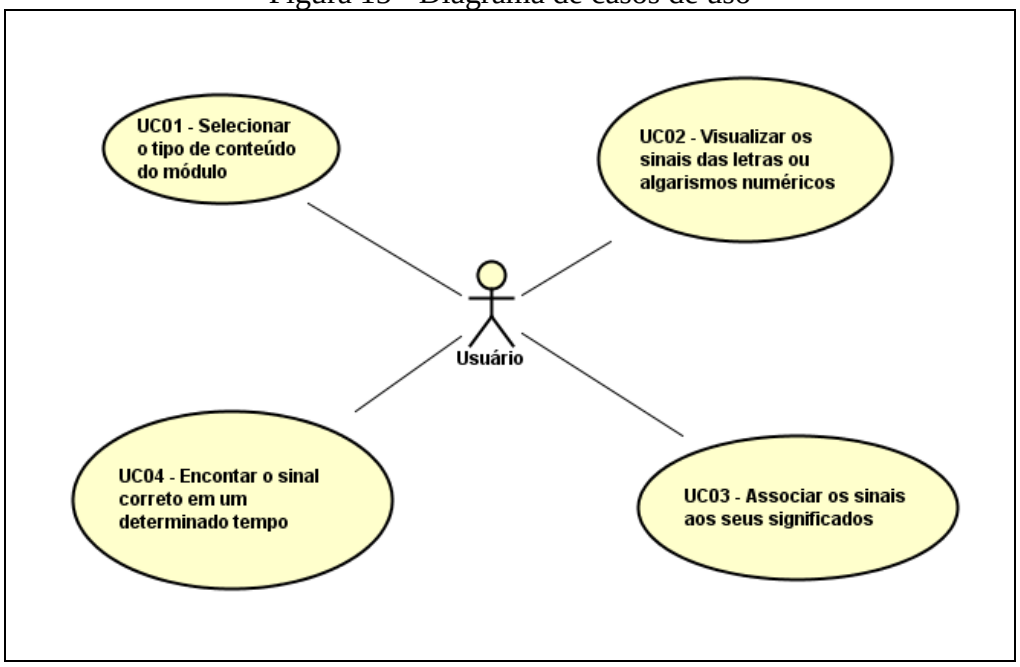
3.3 ESPECIFICAÇÃO

A especificação da ferramenta foi feita utilizando a linguagem UML junto com a ferramenta Astah, com a qual foram feitos os diagramas de casos de uso, diagrama de classes e o diagrama de atividades. O diagrama de atividades será mostrado na seção 3.4 junto com as imagens do sistema para um melhor entendimento do sistema.

3.3.1 Diagrama de Casos de Uso

Nesta seção são apresentados os casos de uso do sistema, conforme Figura 15. O sistema possui apenas o `Usuário` como ator e quatro casos de uso.

Figura 15 - Diagrama de casos de uso



Fonte: elaborado pelo autor.

No diagrama de caso de uso apresentado, o caso `UC01 – Selecionar o tipo de conteúdo do módulo`, o usuário deve escolher entre letras ou algarismos numéricos para o conteúdo do módulo escolhido. No caso de uso `UC02 – Visualizar os sinais das letras ou algarismos numéricos`, o usuário poderá ver a representação do sinal na LIBRAS de acordo com o conteúdo escolhido. O caso de uso `UC03 – Associar os sinais aos seus significados`, o usuário terá o objetivo associar o sinal na LIBRAS ao seu respectivo significado usando os marcadores. No caso de uso `UC04 – Encontrar o sinal correto em determinado tempo`, o usuário terá a tarefa de encontrar o sinal na LIBRAS para uma determinada letra ou algarismo numérico aleatório usando o HMD.

Para facilitar a associação entre os casos de uso e os requisitos do sistema, o Quadro 3 faz uma relação entre eles.

Quadro 3 - Relação entre RF e casos de uso

RFs\Casos de Uso	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12
UC01											X	
UC02	X	X	X								X	X
UC03	X			X	X	X	X				X	X
UC04	X							X	X	X	X	X

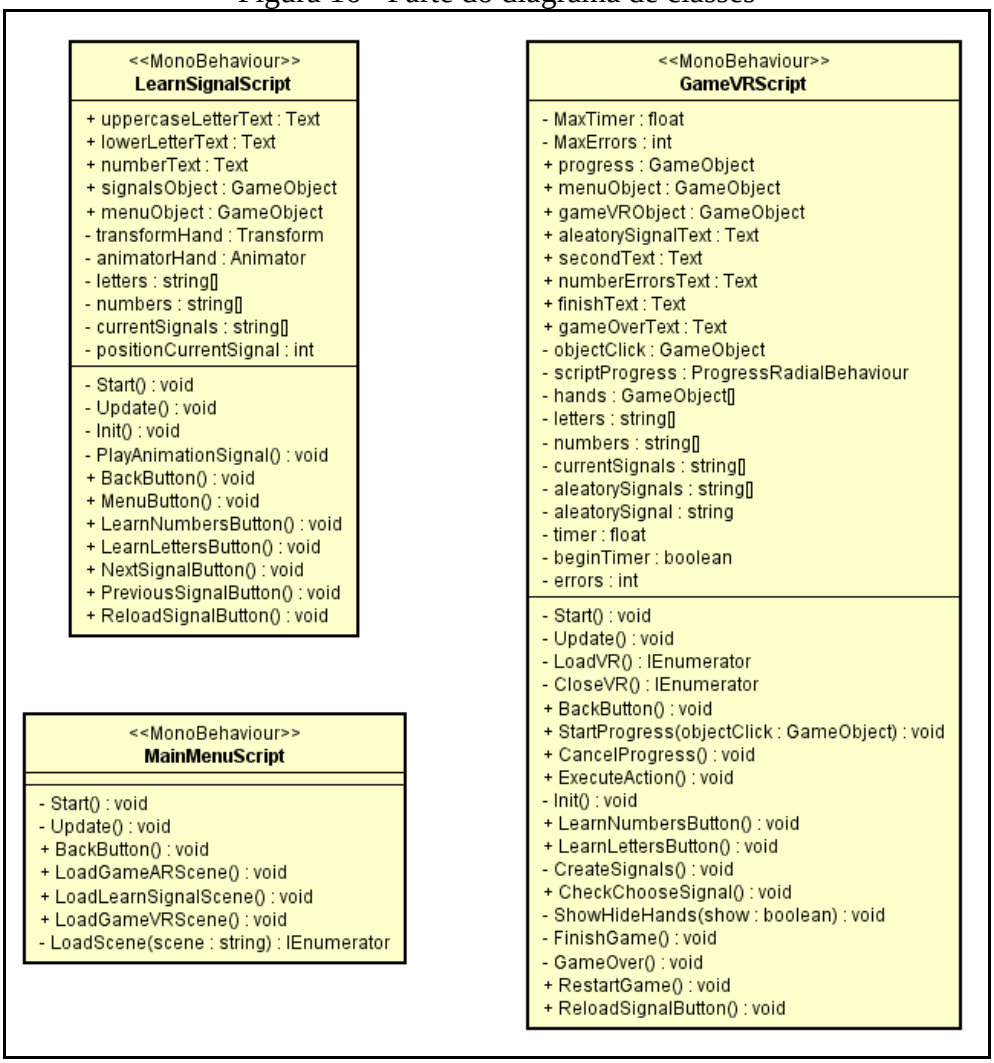
Fonte: elaborado pelo autor.

3.3.2 Diagrama de Classes

A Figura 16 e a Figura 17 apresentam o diagrama de classes desenvolvido para o sistema, divido em partes. Você irá perceber que todos os *scripts* herdam de `MonoBehaviour`, porque essa é a classe base para todos os *scripts* do Unity. Você irá notar também duas funções em todos os *scripts*, a `Start` e `Update`, pois são obrigatórias para todos os *scripts* que herdam de `MonoBehaviour`. A função `Start` é chamada uma única vez logo quando o *script* onde ela está é carregado para a cena do Unity, sendo chamada antes da função `Update`. Agora a função `Update` é chamada a cada *frame* da cena onde o *script* está localizado.

Pode-se notar também outras duas funções comuns a quase todos os *scripts*, as funções `Init` e `BackButton`, essas funções são específicas do sistema. A função `Init` é responsável por realizar as ações necessárias para a cena do Unity funcionar, como inicializar variáveis, carregar objetos da cena, etc. Já a função `BackButton` tem o objetivo de voltar a tela anterior do sistema, no caso se for o menu inicial, irá fechar o sistema.

Figura 16 - Parte do diagrama de classes



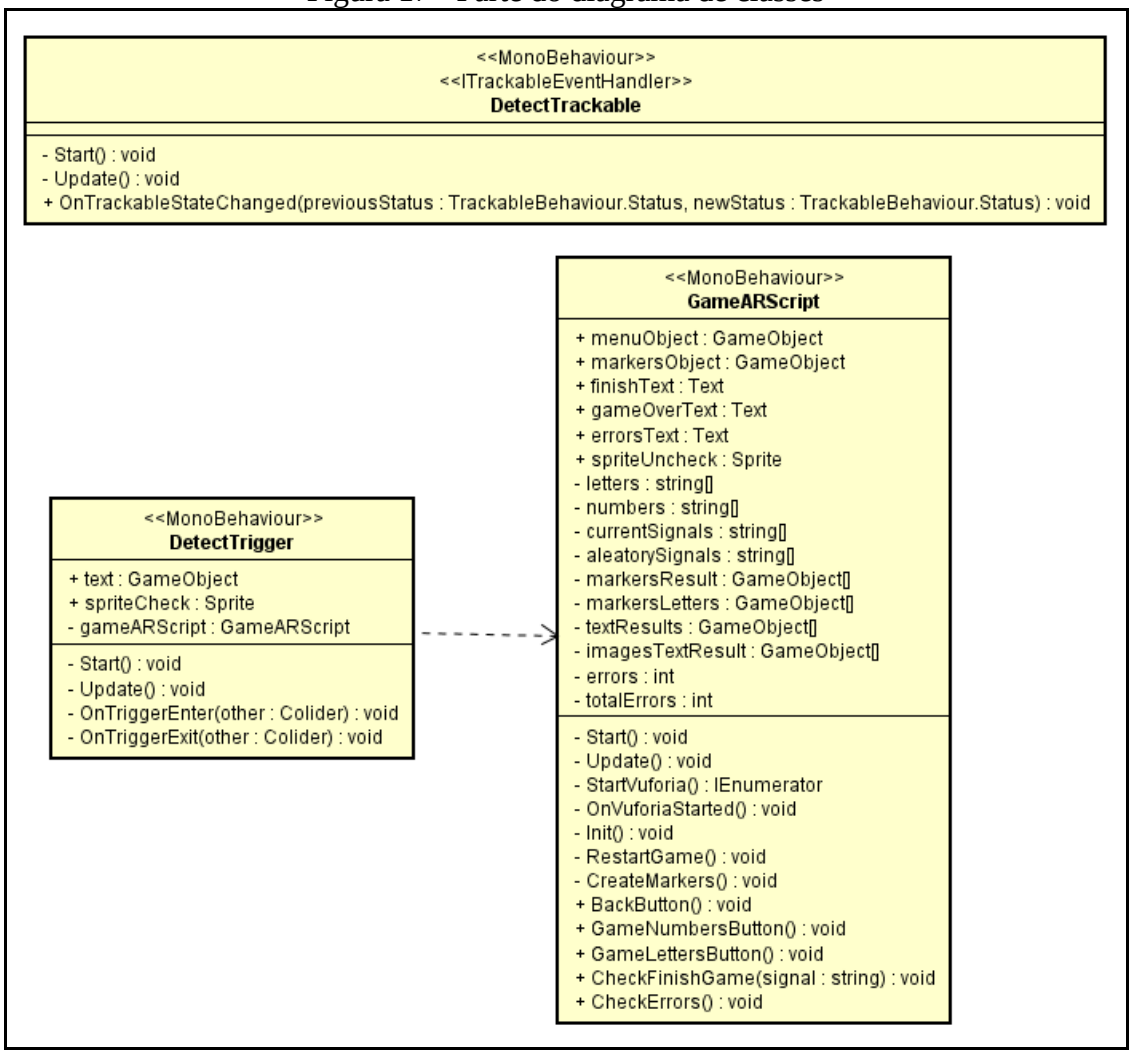
Fonte: elaborado pelo autor.

A Figura 16 mostra os *scripts* de dois módulos do sistema e um da tela inicial. O *script* `MainMenuScript` é referente ao menu do sistema que fornece acesso aos módulos, sendo a tela inicial do sistema. Esse *script* disponibiliza as funções para carregar cada módulo do sistema, que são `LoadLearnSignalScene`, `LoadGameARScene` e `LoadGameVRScene`.

Já o *script* `LearnSignalScript` é responsável pelo módulo inicial do sistema, tendo a função de exibir os sinais na LIBRAS das letras do alfabeto ou algarismos numéricos. Nesse *script* temos a função `LearnNumbersButton` e `LearnLettersButton`, que são chamadas quando o usuário escolhe se quer aprender os algarismos numéricos ou as letras do alfabeto respectivamente. Também temos a função `NextSignalButton` que avança para o próximo sinal, a `PreviousSignalButton` que volta ao sinal anterior, e `ReloadSignalButton` que deixa a mão 3D no seu estado inicial do sinal. Por fim, temos a função `PlayAnimationSignal` que executa a animação na mão 3D do respectivo sinal na LIBRAS, e a `MenuButton`, que exibe o painel inicial do módulo para escolher entre letras ou algarismos numéricos.

Por fim, temos o *script* `GameVRScript`, que é responsável pelo módulo do jogo de raciocínio rápido usando RV. Nele temos as funções `LoadVR` e `CloseVR`, para iniciar e encerrar respectivamente a RV dentro do módulo, sendo chamadas ao entrar e sair do módulo. As funções `LearnNumbersButton` e `LearnLettersButton` são chamadas quando o usuário escolhe se quer jogar com os algarismos numéricos ou as letras do alfabeto respectivamente. A função `StartProgress` é chamada quando o usuário aponta a sua cabeça para um objeto da cena que possui alguma ação, iniciando o progresso com duração de dois segundos. A função `CancelProgress` cancela o progresso caso esteja em execução, e a `ExecuteAction` executa uma função atrelada ao objeto quando o tempo do progresso acaba. A função `CreateSignals` cria de forma aleatória as mãos 3D para o jogo, e a `CheckChooseSignal` valida se o usuário escolheu o sinal correto que foi pedido. A função `FinishGame` é chamada quando o usuário acerta na escolha do sinal, já a `GameOver` é chamada quando o usuário esgotou as suas chances de escolher o sinal correto, e a `RestartGame` é para reiniciar o jogo. Por fim, a função `ReloadSignalButton` pode ser chamada pelo usuário para executar as animações dos sinais em todas as mãos 3D.

Figura 17 - Parte do diagrama de classes



Fonte: elaborado pelo autor.

A Figura 17 mostra os *scripts* referente ao módulo do `Jogo Associativo` usando RA. Os *scripts* `DetectTrackable` e `DetectTrigger` são referentes a RA do módulo. O `DetectTrackable` possui uma função chamada `OnTrackableStateChanged`, que é chamada sempre quando um marcador sofre uma alteração no seu estado, por exemplo quando é detectado ou não está mais no alcance da câmera do Vuforia. O *script* `DetectTrigger` tem a função `OnTriggerEnter` que é chamada quando acontece uma colisão entre dois marcadores, já a `OnTriggerExit` é chamada quando a colisão entre dois marcadores é terminada.

O *script* `GameARScript` é responsável por todo esse módulo. Ele possui as funções `StartVuforia` e `OnVuforiaStarted`, que são chamadas para iniciar o Vuforia e informar que ele foi completamente carregado e iniciado respectivamente. As funções `GameNumbersButton` e `GameLettersButton` são chamadas quando o usuário escolhe se quer jogar com os algarismos numéricos ou as letras do alfabeto respectivamente. A função `CreateMarkers` é responsável por criar os marcadores com seus sinais e significados aleatoriamente. A função

`CheckFinishGame` valida se o usuário terminou todas as associações do jogo, a `RestartGame` é chamada para reiniciar o jogo e a `CheckErrors` valida se o usuário esgotou suas tentativas de associar os sinais.

3.4 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

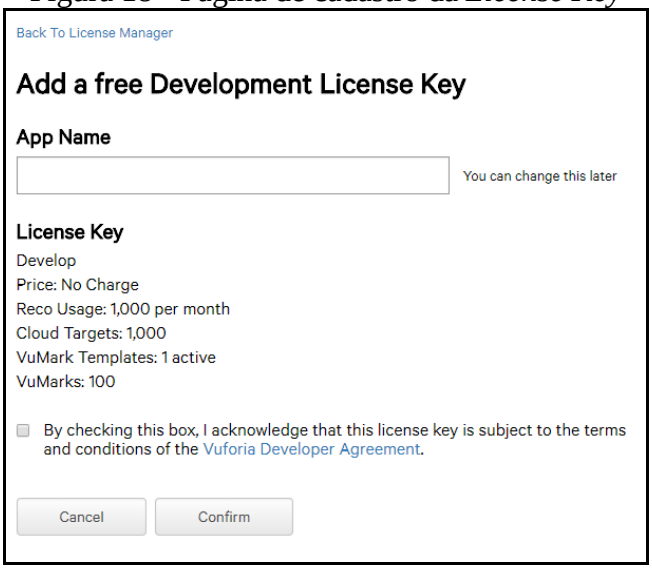
3.4.1 Técnicas e ferramentas utilizadas

Para a implementação do sistema foi utilizado o motor de jogos Unity 3D na versão 2017.3.0f3 Personal. Para o uso da RA, foi utilizado o SDK Vuforia na versão 7.0.36. Já no uso da RV, foi utilizado o SDK Google VR para Unity na versão 1.120.0. O ambiente de desenvolvimento utilizado foi o Visual Studio Code com a linguagem de programação C#. Para a criação dos marcadores foi utilizado a ferramenta Brosvision (2013), gerador de marcadores otimizados para RA junto com o Photoshop CC 2018. Para a mão 3D, o molde foi baixado do site Cadnav (2018), o esqueleto foi criado no software Blender na versão 2.79, e as animações foram feitas dentro do Unity.

3.4.1.1 Início do projeto

Para dar início ao projeto, é necessário criar uma conta na página de desenvolvedores do Vuforia para poder usar a SDK de RA. Após isso é necessário criar uma chave de licença para ser usada no Unity. Para gerar essa chave, basta ir na guia `Develop` e depois em `License Manager`, e agora clicar no botão `Get Development Key`, depois é só digitar o nome da sua aplicação e clicar em `Confirm` (Figura 18).

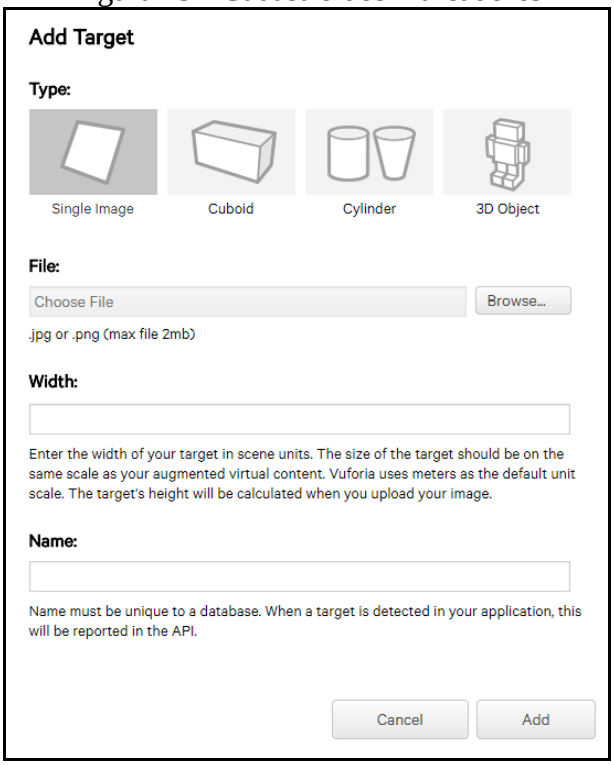
Figura 18 - Página de cadastro da *License Key*



Fonte: elaborado pelo autor.

Agora é necessário criar um *database* dos marcadores para ser importado no Unity. Para isso, acesse `Target Manager` e depois clicar no botão `Add Database`. Após criar o *database*, é necessário adicionar as imagens dos marcadores. Para isso entre no *database* e clique no botão `Add Target`. Agora é necessário escolher o tipo do marcador (no sistema desenvolvido foi utilizado o tipo *Single Image*), o arquivo de imagem do seu marcador, a largura da imagem e um nome (Figura 19). Depois de cadastro o marcador, ele receberá uma avaliação, quanto maior essa avaliação, melhor o seu marcador será detectado pelo Vuforia.

Figura 19 - Cadastro dos marcadores

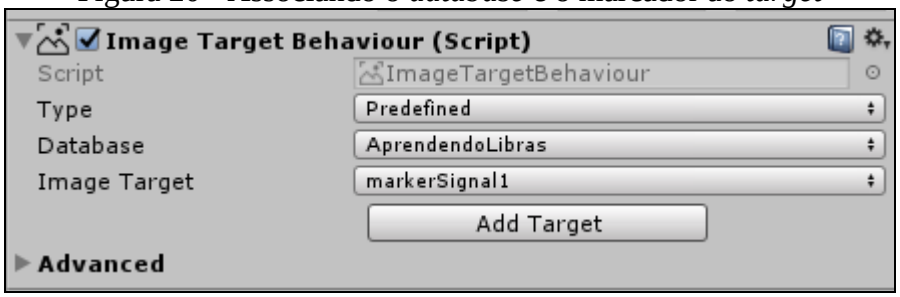


Fonte: elaborado pelo autor.

Agora basta baixar o *database* clicando no botão `Download Database (All)` e escolher a plataforma onde será usado (no sistema desenvolvido foi utilizado `Unity Editor`). Agora é necessário importar o arquivo para o Unity utilizando a opção `Custom Package` que se encontra na opção `Import Package` na aba `Assets`.

Agora no projeto Unity é necessário habilitar a RA, para isso acesse a aba `File` e depois `Build Settings`. Depois clique no botão `Player Settings` e abra a seção `XR Settings` para habilitar a opção `Vuforia Augmented Reality`. Após isso, abra a cena do Unity que irá conter a RA, remova o objeto `Main Camera` e adicione o `prefab ARCamera`, responsável pela câmera da RA. Agora, clique no objeto `ARCamera` e abra as configurações do Vuforia clicando no botão `Open Vuforia Configuration`. Aqui é necessário a chave de licença do Vuforia gerada anteriormente no campo `App License Key`, e também selecionar e ativar o *database* que foi importado na seção `Databases`. Agora é só inserir os *targets* (no sistema desenvolvido foi utilizado o `prefab Image`) na cena e associar o *database* e o respectivo marcador para cada *target* (Figura 20).

Figura 20 - Associando o *database* e o marcador ao *target*



Fonte: elaborado pelo autor.

No sistema desenvolvido foi realizado algumas configurações diferentes no Vuforia, porque o sistema possui somente uma cena que usa o Vuforia, com isso é preciso carrega-lo somente ao abrir essa cena. Para isso foi necessário desabilitar o *script* `Vuforia Behaviour` associado ao `prefab ARCamera`, e habilitar a opção `Delayed Initialization` nas configurações do Vuforia (Figura 21).

Figura 21- Configurações do Vuforia

Fonte: elaborado pelo autor.

Com isso toda a inicialização do Vuforia foi realizada no *script*, como pode ser visto no Quadro 4. A função `Init` tem a responsabilidade de inicializar o Vuforia, primeiramente validando se ele já não foi inicializado, caso sim, somente habilita e atribui a configuração de foco para a câmera. Caso ainda não tenha inicializado, é chamado a função `StartVuforia` que inicializa o Vuforia e habilita o *script* `Vuforia Behaviour` associado ao `prefab ARCamera`.

Quadro 4 - Métodos para inicializar o Vuforia

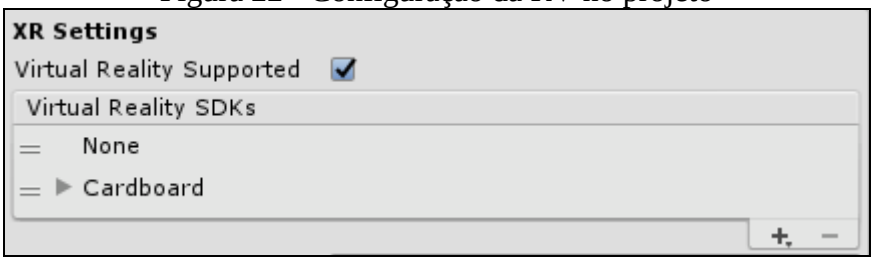
```
...
    IEnumerator StartVuforia()
    {
        VuforiaRuntime.Instance.InitVuforia();
        while (!VuforiaRuntime.Instance.HasInitialized)
            yield return null;

        VuforiaARController.Instance
           .RegisterVuforiaStartedCallback(OnVuforiaStarted);
        VuforiaARController.Instance
           .RegisterOnPauseCallback(OnPausedVuforia);
        GetComponent<VuforiaBehaviour>().enabled = true;
    }

    void OnVuforiaStarted()
    {
        CameraDevice.Instance.SetFocusMode(FocusModeAR);
    }

    void OnPausedVuforia(bool paused)
    {
        if (!paused)
        {
            CameraDevice.Instance.SetFocusMode(FocusModeAR);
        }
    }
...
    IEnumerator Init()
    {
        ...

        if (VuforiaRuntime.Instance.HasInitialized)
        {
            VuforiaBehaviour.Instance.enabled = true;
            CameraDevice.Instance.SetFocusMode(FocusModeAR);
        }
        else
            StartCoroutine(StartVuforia());

        ...
    }
...
```

Fonte: elaborado pelo autor.

Para a RV, é necessário acessar a página de desenvolvedores do Google VR e baixar a SDK para Unity. No projeto, é necessário importar o arquivo SDK utilizando a opção `Custom Package` que se encontra na opção `Import Package` na aba `Assets`. Agora é necessário habilitar a RV no projeto, para isso acesse a aba `File` e depois `Build Settings`, depois clique no botão `Player Settings` e abra a seção `XR Settings` para habilitar a opção `Virtual Reality Supported`. Agora é só ir em `Virtual Reality SDKs` e escolher a SDK que será usado (no sistema desenvolvido foi utilizado `None` e `Cardboard`), conforme a Figura 22. A *SDK* `None` foi utilizada porque esse sistema utiliza RV somente em uma cena, e ao

habilitar a RV no projeto e não escolher também a *SDK* `None`, a RV fica habilitada em todas as cenas.

Figura 22 - Configuração da RV no projeto



Fonte: elaborado pelo autor.

Com isso, toda a inicialização da RV para a cena foi realizada via *script*, conforme o Quadro 5. Na função `Start` é chamado a função `LoadVR` responsável por inicializar a RV. Na função `LoadVR` é carregado a SDK `cardboard` e habilitado a RV. A função `CloseVR` é chamada ao sair da cena, carregando a SDK `none` e desabilitando a RV.

Quadro 5 - Métodos para habilitar e desabilitar a RV no projeto

```
...
    void Start()
    {
        ...

        StartCoroutine(LoadVR());
    }

    IEnumerator LoadVR()
    {
        UnityEngine.XR.XRSettings.LoadDeviceByName("cardboard");
        yield return null;
        UnityEngine.XR.XRSettings.enabled = true;
    }

    IEnumerator CloseVR()
    {
        UnityEngine.XR.XRSettings.LoadDeviceByName("none");
        yield return null;
        UnityEngine.XR.XRSettings.enabled = false;
        SceneManager.LoadSceneAsync("MainMenuScene");
    }
...
```

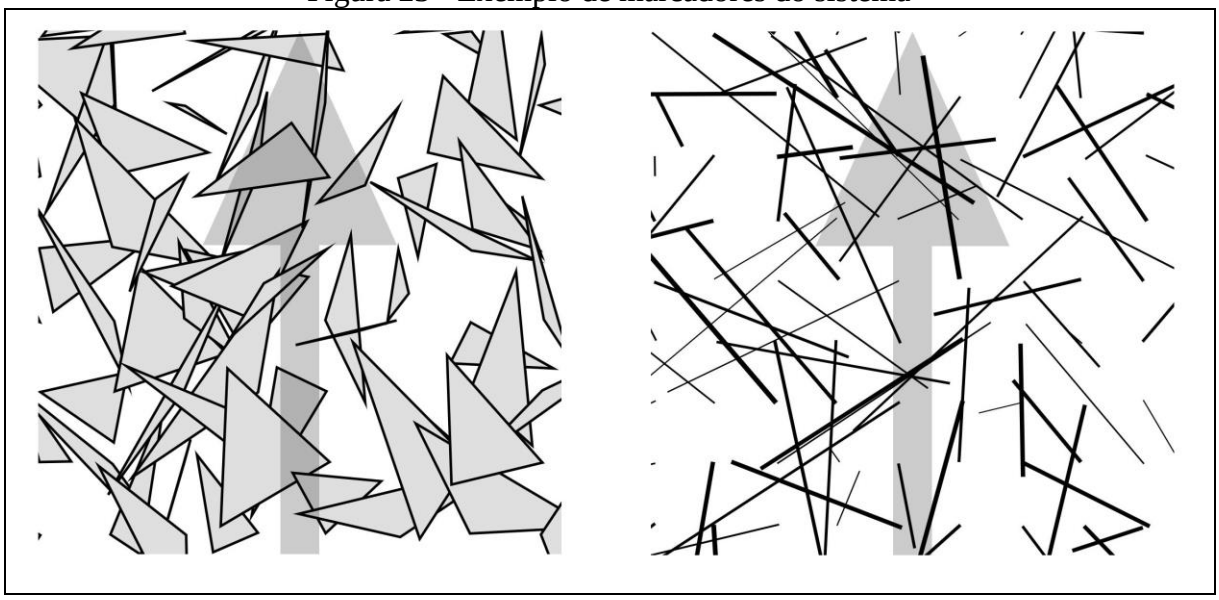
Fonte: elaborado pelo autor.

3.4.1.2 Marcadores

Esse sistema possui oito marcadores que são usados para o jogo de RA. Todos os marcadores foram gerados pela ferramenta Brosvision (2013), um gerador de marcadores otimizados para RA. Após a geração dos marcadores, foi usado o Photoshop CC 2018 para posicionar uma seta no centro do marcador, indicando a posição correta quando está apontando para frente. Metade dos marcadores são responsáveis por exibir o sinal na LIBRAS, esses marcadores são compostos por triângulos de quantidade e tamanhos

aleatórios. A outra metade é responsável por exibir os significados dos respectivos sinais na LIBRAS, esses marcadores são compostos por arestas de tamanhos aleatórios. A Figura 23 mostra um exemplo desses dois marcadores.

Figura 23 - Exemplo de marcadores do sistema

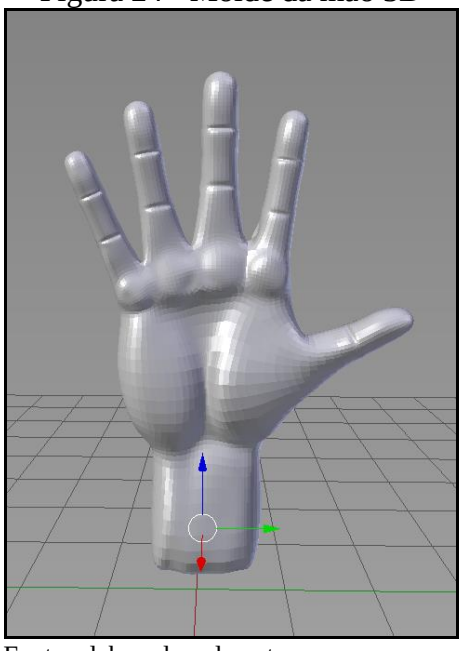


Fonte: elaborado pelo autor.

3.4.1.3 Mão 3D

O sistema desenvolvido possui uma mão 3D para representar os sinais na LIBRAS. Para isso foi encontrado um molde gratuito com uma textura de uma mão real do site Cadnav (2018), esse site disponibiliza modelos 3D de vários formatos, a maioria sendo de forma gratuita. Esse molde pode ser visto na Figura 24.

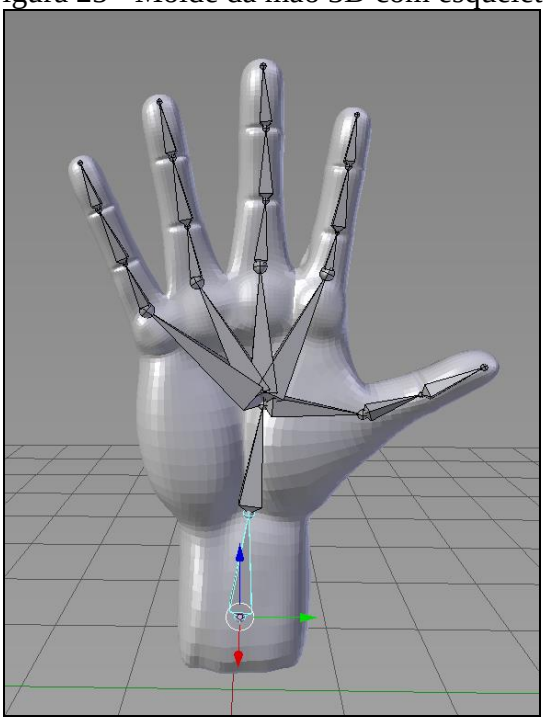
Figura 24 - Molde da mão 3D



Fonte: elaborado pelo autor.

Mas para conseguir fazer essa mão 3D representar um sinal na LIBRAS, foi necessário criar um esqueleto em cima dela, e para isso foi usado software Blender. Um esqueleto de uma mão 3D segue o mesmo princípio de uma mão real, com isso, foi criado um esqueleto com articulações muito semelhantes de uma mão real. Como podemos ver na Figura 25, os objetos em forma de pirâmide representam os ossos, e as esferas que se encontram nas pontas representam as articulações do corpo humano. Agora com base nas articulações é possível movimentar todo o molde como se fosse uma mão real.

Figura 25 - Molde da mão 3D com esqueleto

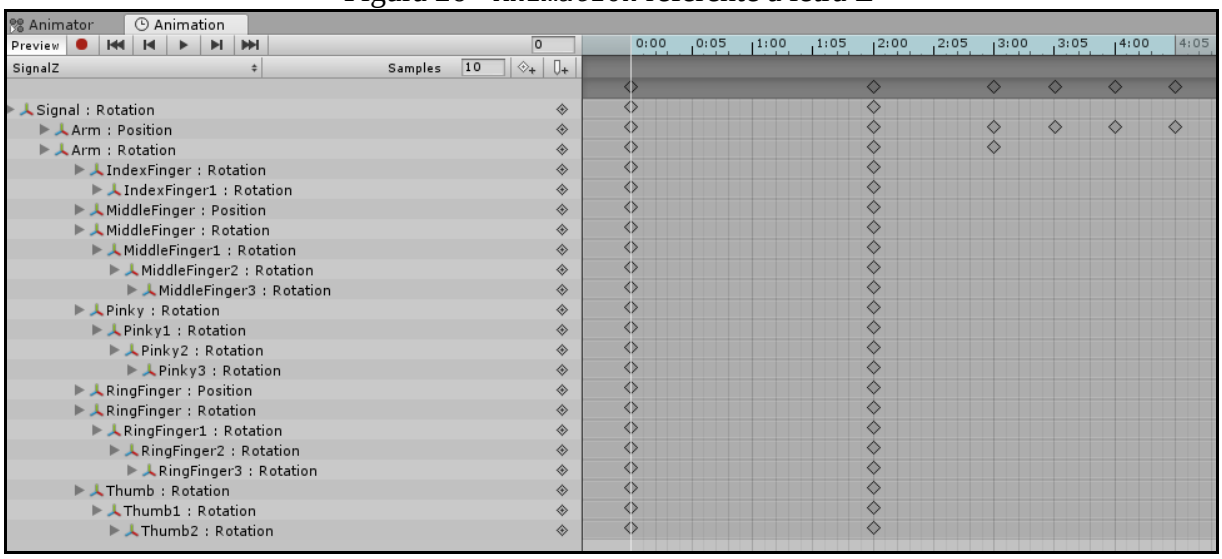


Fonte: elaborado pelo autor.

Depois de montado o esqueleto no molde da mão 3D, foi necessário criar as animações dos sinais na LIBRAS referente as letras do alfabeto e os algarismos numéricos, e para essa função foi usado o motor de animação do Unity. No Unity temos dois conceitos bem importantes relacionados a animações, o `Animation` e `Animator Controller`. Quando é falado `Animation` estamos falando de uma animação específica, por exemplo, a animação da letra `A` na LIBRAS. E quando é falado `Animator Controller`, está se referindo a máquina de estados de animações do Unity, que é responsável por reproduzir cada animação em seu determinado fluxo. Para o sistema desenvolvido, foi criado um `Animation` para cada letra do alfabeto e algarismo numérico. Para que cada `Animation` consiga executar corretamente o sinal na LIBRAS, é alterado as propriedades de rotação e posição de cada articulação do esqueleto montado para o modelo da mão 3D. A Figura 26 mostra o `Animation` da letra Z.

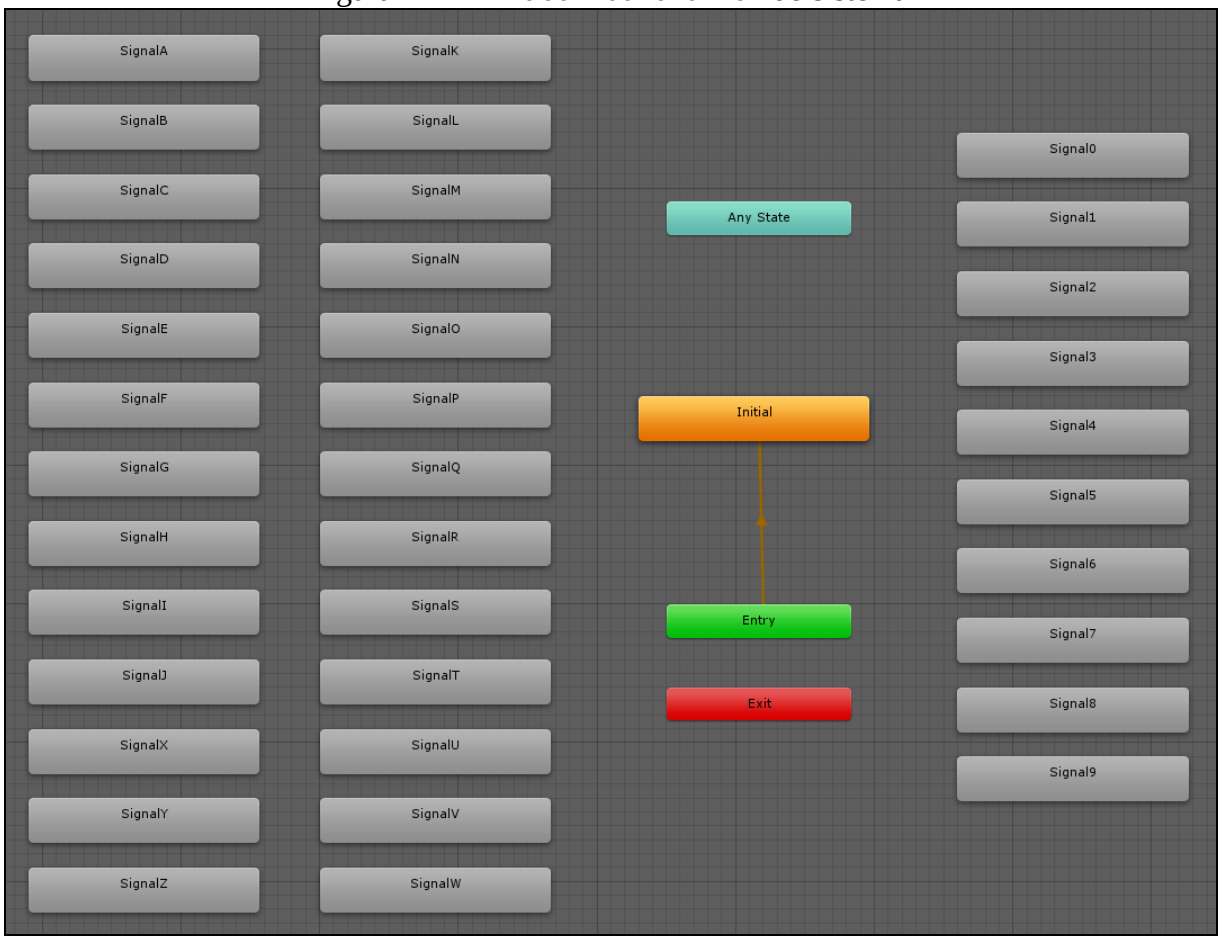
Também foi criado um `Animator Controller` que contém todos os `Animation` desenvolvidos para o sistema (Figura 27).

Figura 26 - `Animation` referente a letra Z



Fonte: elaborado pelo autor.

Figura 27 - `Animator Controller` do sistema

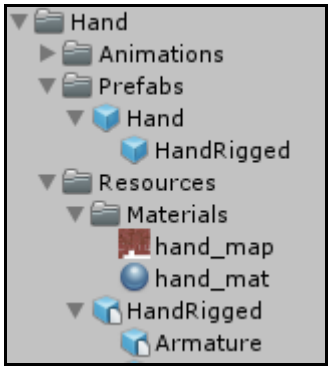


Fonte: elaborado pelo autor.

Depois de ter criado as animações, foi criado a estrutura mostrada na Figura 28 para a mão 3D dentro do Unity. A pasta `Animations` é onde fica todas as animações das letras do

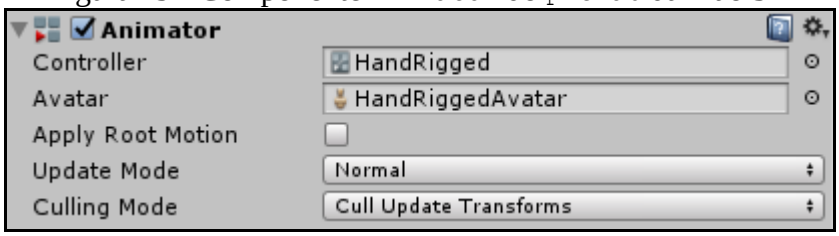
alfabeto e algarismos numéricos, e também o `Animation Controller` da Figura 27. A pasta `Resources` é onde fica a mão 3D que foi extraída do Blender, o arquivo `HandRigged` tem a extensão .fbx gerada pelo Blender para ser usado dentro do Unity, e dentro dele temos o `Armature` referente ao esqueleto que foi desenhado. A pasta `Resources` possui também a pasta `Materials`, referente ao mapa de textura da mão 3D. E por último, a pasta `Prefabs` contém o `prefab Hand` que já possui o componente `Animator` com o `Animator Controller` criado para o sistema (Figura 29). Esse `prefab` foi criado para facilitar o uso da mão 3D dentro do sistema.

Figura 28 - Estrutura da mão 3D



Fonte: elaborado pelo autor.

Figura 29 - Componente `Animator` do `prefab` da mão 3D

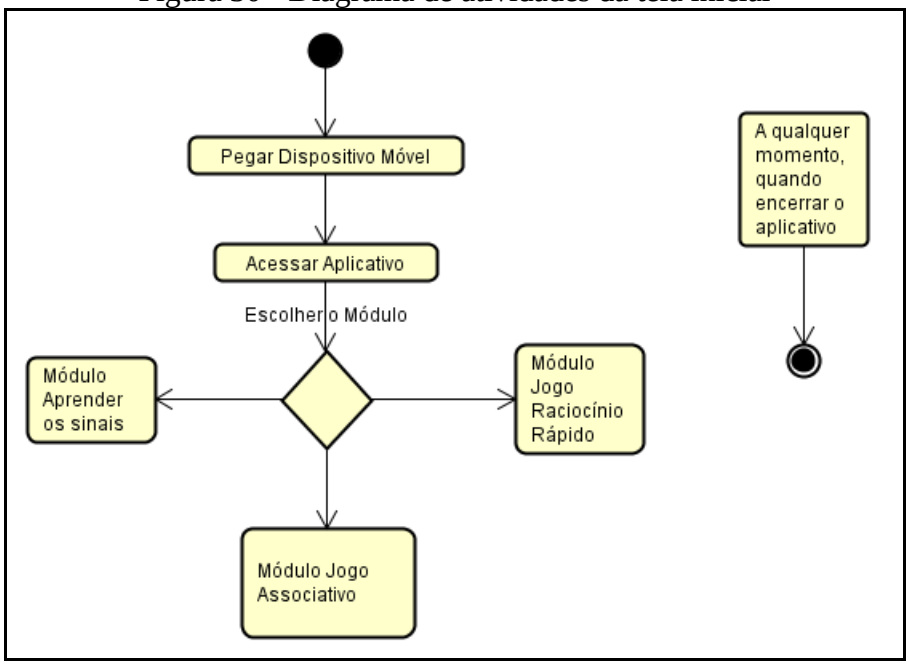


Fonte: elaborado pelo autor.

3.4.1.4 Ferramenta e operacionalidade da implementação

Para a explicação sobre o desenvolvimento do sistema, optou-se pelo uso de diagramas de atividades para mostrar o que o sistema pode fazer em determinados módulos, junto a isso, também são mostradas telas do sistema para visualizar as funcionalidades em execução. Também será mostrado parte dos códigos do sistema para explicar as principais funcionalidades. A Figura 30 mostra o diagrama de atividades da tela inicial do sistema.

Figura 30 - Diagrama de atividades da tela inicial



Fonte: elaborado pelo autor.

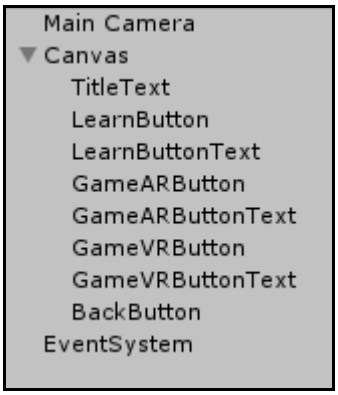
Como podemos ver na Figura 30, o sistema é composto por três módulos completamente independentes, com isso é possível usar somente os módulos mais apropriados a situação em que o sistema for utilizado. O primeiro módulo é chamado Aprender os Sinais, e tem como objetivo mostrar ao usuário a representação dos sinais na LIBRAS. O segundo e o terceiro módulo, chamados respectivamente de Jogo Associativo e Jogo de Raciocínio Rápido, disponibilizam ao usuário uma forma diferente e divertida de praticar os seus conhecimentos na LIBRAS a partir de jogos. A Figura 31 mostra a tela inicial do sistema.

Figura 31 - Tela inicial do sistema

Fonte: elaborado pelo autor.

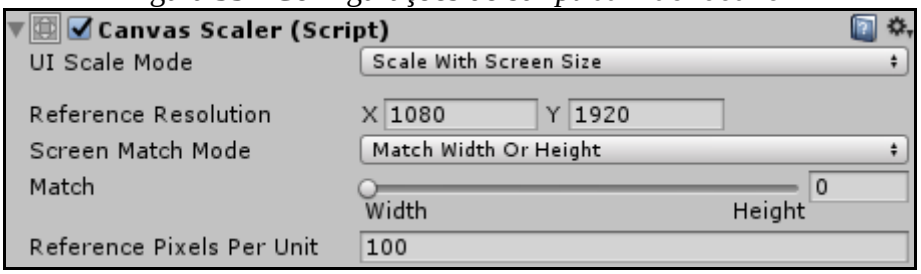
A estrutura da tela inicial é composta por uma `Main Camera`, que possui o *script* `MainMenuScript` responsável pelas ações dessa tela. E um `GameObject Canvas` com os componentes UI abaixo dele. Essa estrutura pode ser visualizada na Figura 32. O `Canvas` possui adicionado a ele o *script* `Canvas Scaler`, esse *script* tem o objetivo de redimensionar de acordo com o tamanho da tela todos os componentes que estão abaixo do `Canvas`, fazendo com que o visual dos componentes fique igual em todos os tamanhos de tela. Para o sistema desenvolvido, todos os `Canvas` possuem a mesma configuração do *script* `Canvas Scaler`. As principais mudanças foram na propriedade `UI Scale Mode`, que ficou com a opção `Scale With Screen Size`, indicando que os componentes serão ajustados baseado na largura da tela. E na propriedade `Reference Resolution`, com 1080 para X e 1920 para Y. A configuração completa pode ser vista na Figura 33.

Figura 32 - Estrutura da tela inicial



Fonte: elaborador pelo autor.

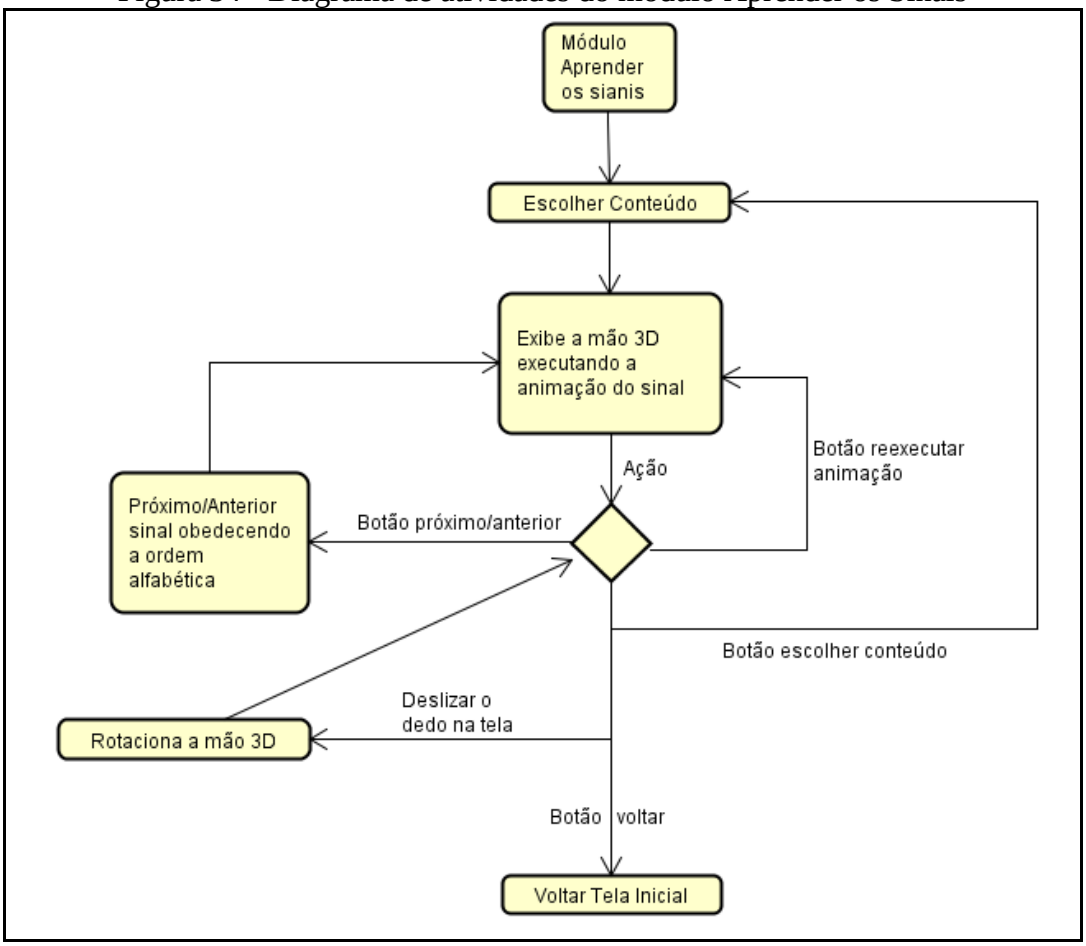
Figura 33 - Configurações do *script* `Canvas Scaler`



Fonte: elaborado pelo autor.

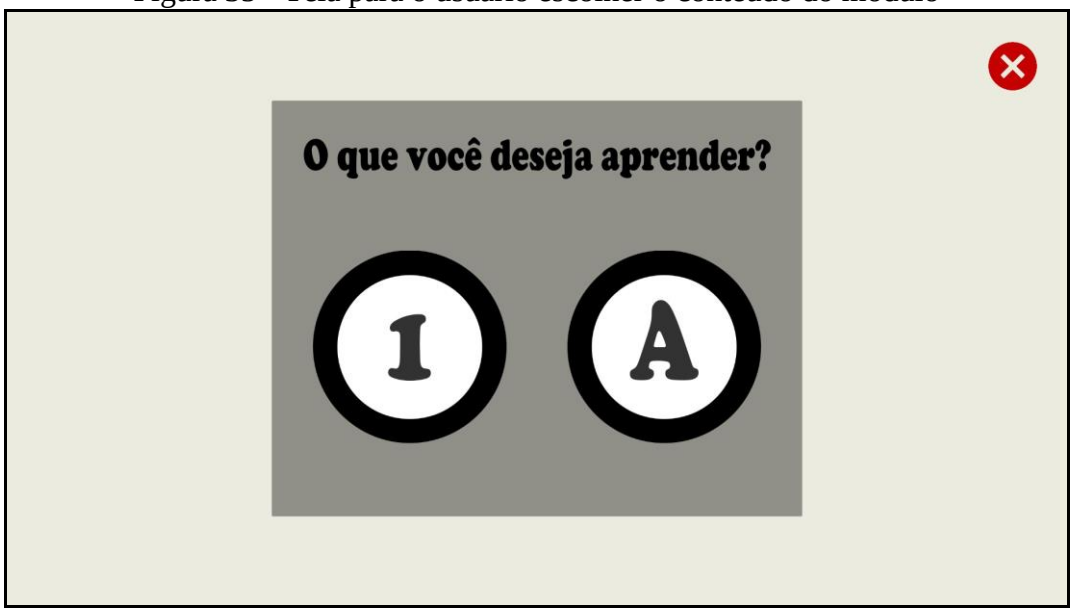
A Figura 34 exibe o diagrama de atividades do módulo Aprender os Sinais. Esse módulo tem o objetivo de mostrar ao usuário os sinais na LIBRAS das letras do alfabeto ou algarismos numéricos em ordem alfabética. Ao entrar no módulo, o usuário precisa escolher o conteúdo que ele irá visualizar, podendo escolher entre algarismos numéricos ou letras do alfabeto (Figura 35). Essa tela é comum a todos os módulos do sistema, portanto na explicação dos próximos módulos irá ser feito somente uma referência para a Figura 35.

Figura 34 - Diagrama de atividades do módulo Aprender os Sinais



Fonte: elaborado pelo autor.

Figura 35 - Tela para o usuário escolher o conteúdo do módulo

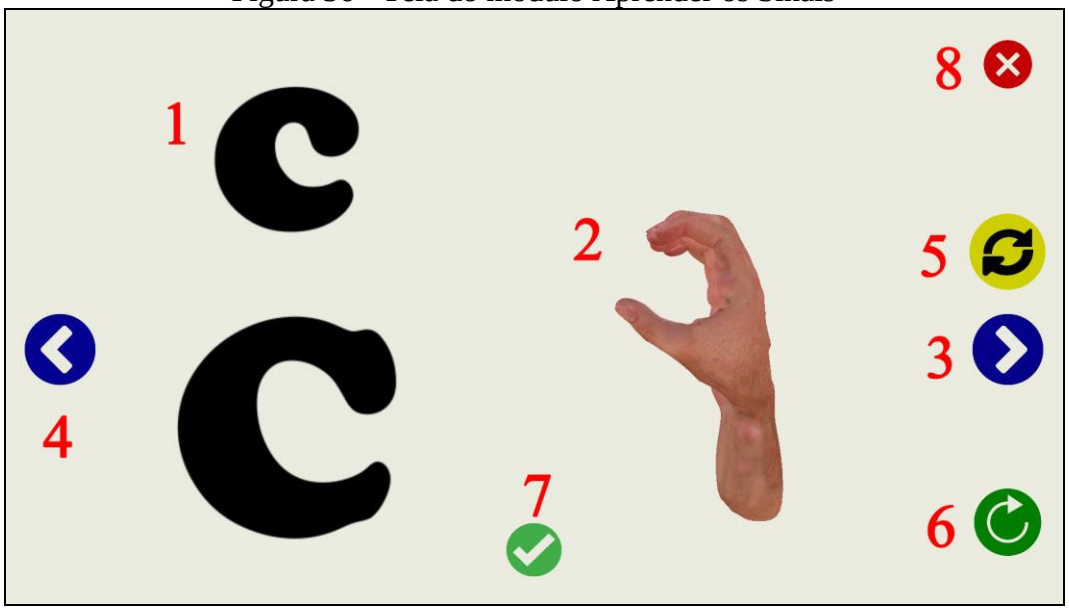


Fonte: elaborado pelo autor.

Como podemos ver na Figura 34, após o usuário escolher o conteúdo do módulo, é direcionado para a tela do módulo, conforme a Figura 36. No `item 1` da Figura 36, é exibido o significado do sinal atual. O `item 2` da Figura 36 mostra a mão 3D realizando a animação

do sinal na LIBRAS. Os `itens 3` e `4` da Figura 36 são botões para o usuário visualizar o próximo/anterior sinal na LIBRAS, respeitando a ordem alfabética. O `item 5` da Figura 36 é um botão com o objetivo de reexecutar a animação da mão 3D do sinal atual. O `item 6` da Figura 36 é um botão para o usuário voltar a tela para escolher um novo conteúdo para o módulo (Figura 35). O `item 7` da Figura 36 indica quando a animação do sinal foi finalizada, e o `item 8` é um botão para o usuário voltar ao menu inicial do sistema (Figura 31). O usuário pode rotacionar a mão 3D para poder visualizar todos os seus lados, para isso, é necessário pressionar o dedo na tela do dispositivo móvel e deslizá-lo para direita ou esquerda para realizar a rotação da mão 3D.

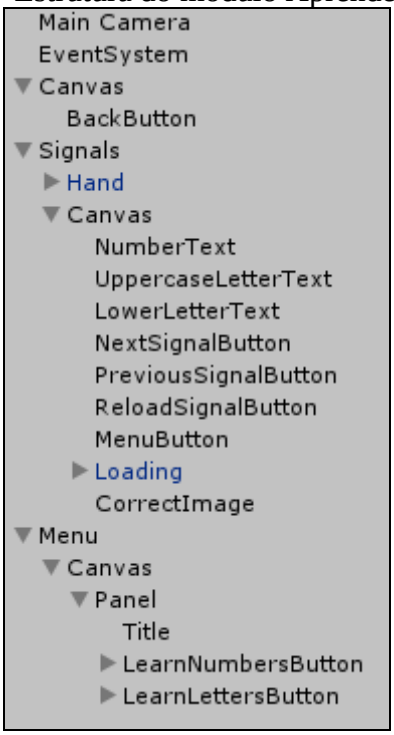
Figura 36 - Tela do módulo Aprender os Sinais



Fonte: elaborado pelo autor.

A estrutura do módulo Aprender os Sinais pode ser visualizada na Figura 37. É composto por uma `Main Camera`, que possui o *script* `LearnSignalsScript` responsável pelas ações dessa tela. Um `GameObject Canvas` com um botão `BackButton` responsável por voltar a tela inicial do sistema. Um `GameObject Menu` que possui abaixo dele os componentes para exibir a tela inicial do módulo (Figura 35) e outro `GameObject Signals` que possui abaixo dele todos os componentes mostrados na Figura 36. A estrutura da tela foi separada por esses dois componentes para facilitar a exibição da tela inicial do módulo (Figura 35) e a tela para o usuário visualizar os sinais (Figura 36), sendo necessário somente ativar o `GameObject` que deseja visualizar.

Figura 37 - Estrutura do módulo Aprender os Sinais



Fonte: elaborado pelo autor.

Ao iniciar o módulo, o usuário escolhe o conteúdo que será apresentado. Para isso, o Quadro 6 mostra como é feito para exibir os sinais de acordo com a escolha do usuário. Inicialmente são declaradas três variáveis chamadas `letters`, `numbers` e `currentSignals`. As variáveis `letters` e `numbers` somente armazenam as letras do alfabeto e os algarismos numéricos respectivamente. A variável `currentSignals` irá armazenar as letras do alfabeto ou algarismos numéricos, como podemos ver nas funções `LearnLettersButton` e `LearnNumbersButton`, que serão chamadas dependendo da escolha do usuário na tela inicial do módulo (Figura 35). Com isso, todas as operações serão realizadas em cima da variável `currentSignals`. Esse padrão é igual para todos os módulos do sistema, portanto na explicação dos próximos módulos essa parte será abstraída.

Quadro 6 - Métodos para escolher o conteúdo do módulo

```
...
    string[] letters = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J",
                        "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T",
                        "U", "V", "W", "X", "Y", "Z"};

    string[] numbers = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"};

    string[] currentSignals;
...
    public void LearnNumbersButton()
    {
        currentSignals = numbers;
        ...
    }

    public void LearnLettersButton()
    {
        currentSignals = letters;
        ...
    }
...
```

Fonte: elaborado pelo auto.

O Quadro 7 mostra como é feito a troca do sinal quando o usuário aperta nos botões `3` e `4` da Figura 36. A variável `positionCurrentSignal` contém o índice do sinal atual referente a variável `currentSignals`. A função `NextSignalButton` tem o objetivo de exibir o próximo sinal. Primeiramente, valida se não está exibindo o último sinal da variável `currentSignals`, caso for verdade incrementa o índice da variável `postionCurrentSginal` e chama a função `PlayAnimationSignal`. A função `PreviousSignalButton` tem o objetivo de exibir o sinal anterior. Primeiramente, valida se não está exibindo o primeiro sinal da variável `currentSignals`, caso for verdade decrementa o índice da variável `positionCurrentSignal` e chama a função `PlayAnimationSignal`. A função `PlayAnimationSignal` tem o objetivo de executar a animação do sinal atual. Primeiramente, rotaciona a mão 3D para sua posição inicial, isso é necessário porque o usuário pode rotacionar a mão 3D para visualizar o sinal. Depois recupera o sinal da variável `currentSignals`, atualiza os textos da tela e chama a animação a partir do componente `Animator` da mão 3D.

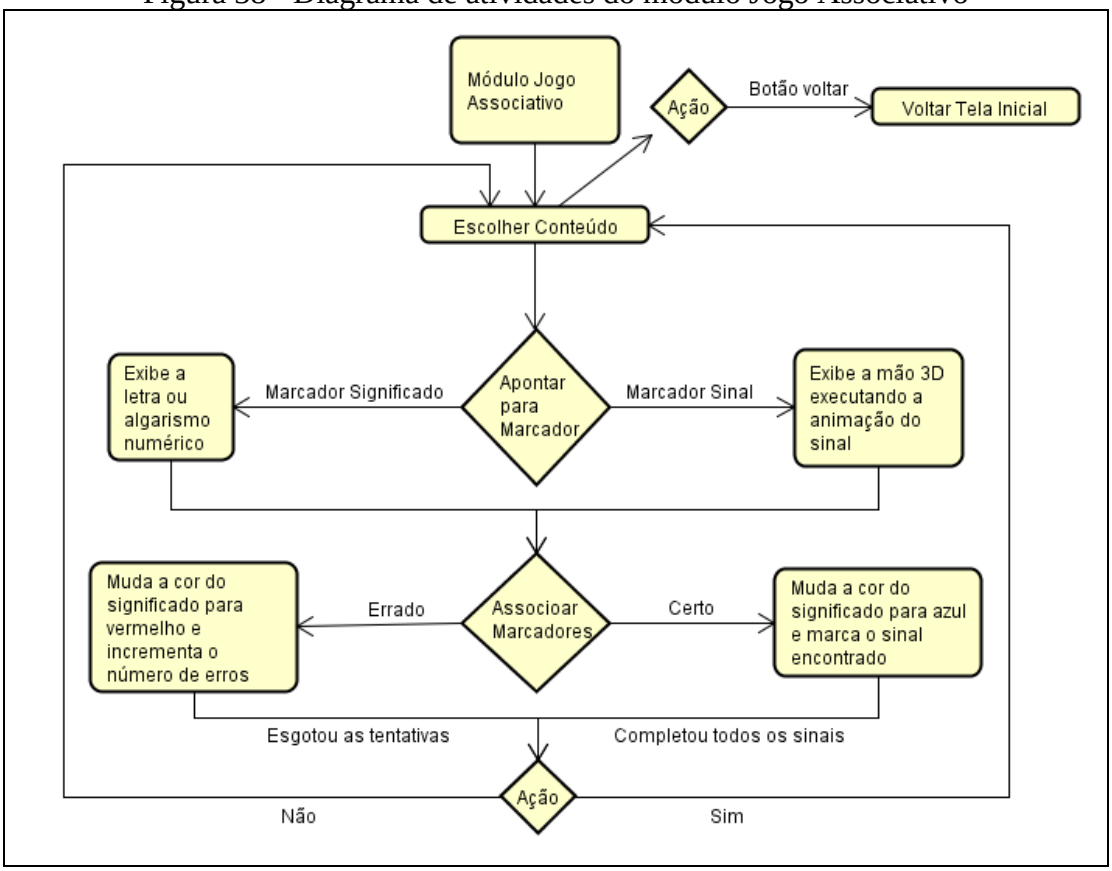
Quadro 7 - Métodos para avançar, retornar e executar a animação do sinal

```
...
    public void NextSignalButton()
    {
        if (positionCurrentSignal != currentSignals.Length - 1)
        {
            positionCurrentSignal++;
            PlayAnimationSignal();
        }
    }

    public void PreviusSignalButton()
    {
        if (positionCurrentSignal != 0)
        {
            positionCurrentSignal--;
            PlayAnimationSignal();
        }
    }

    void PlayAnimationSignal()
    {
        transformHand.rotation = Quaternion.Euler(0, -80, 0);

        string signal = currentSignals[positionCurrentSignal];
        uppercaseLetterText.text = signal;
        lowerLetterText.text = signal.ToLower();
        numberText.text = signal;
        animatorHand.Play("Signal" + signal);
    }
...
```

Fonte: elaborado pelo autor.

A Figura 38 mostra o diagrama de atividades do módulo Jogo Associativo. Esse módulo disponibiliza ao usuário um jogo associativo para treinar os seus conhecimentos na LIBRAS usando RA. O objetivo do jogo é associar o marcador do sinal na LIBRAS ao marcador que contenha o significado do sinal. Para o uso desse módulo é necessário imprimir os marcadores que podem ser encontrados em <http://tecedu.inf.furb.br/librar/marcadores>. É necessário também estar em um ambiente com uma boa iluminação e posicionar os marcadores em uma superfície plana. Ao entrar no módulo, o usuário precisa escolher o conteúdo que ele irá visualizar, podendo escolher entre algarismos numéricos ou letras do alfabeto (Figura 35).

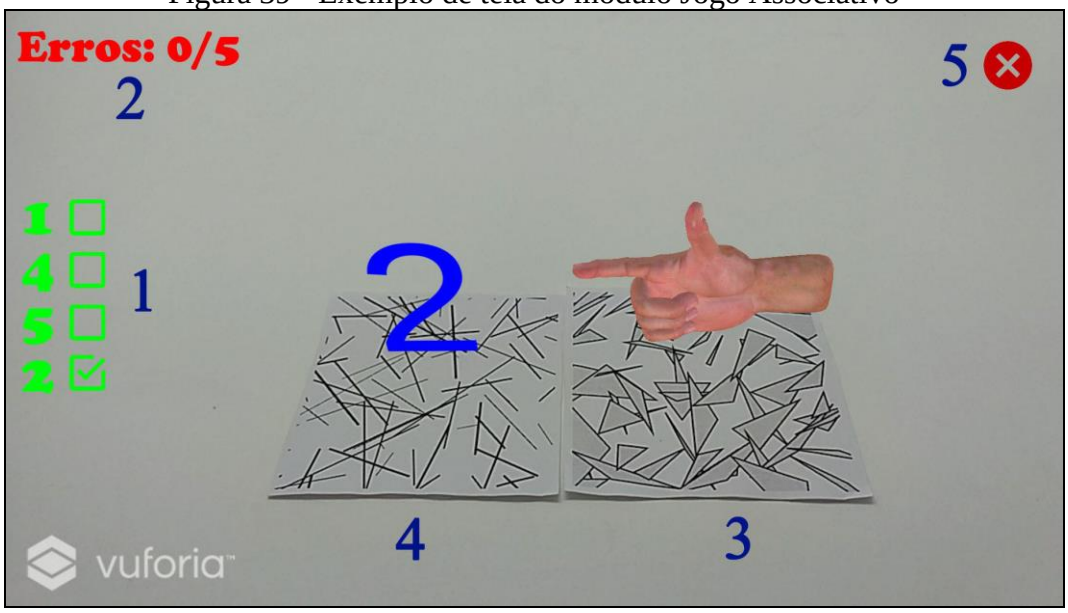
Figura 38 - Diagrama de atividades do módulo Jogo Associativo



Fonte: elaborado pelo autor.

Após escolher o conteúdo do módulo, o usuário irá visualizar a câmera do seu celular com algumas informações na tela. Agora é necessário apontar a câmera para os marcadores para visualizar os sinais e os significados dos marcadores. Depois é necessário pegar um marcador que exibe o sinal na LIBRAS e encontrar o marcador correspondente com o seu significado. Agora, basta colocar os dois marcadores um ao lado do outro para visualizar o resultado. Agora é necessário repetir esses passos até associar corretamente todos marcadores ou até esgotar as suas tentativas. Após isso, será direcionado a tela de escolher o conteúdo para poder começar o jogo novamente. A Figura 39 mostra um exemplo desse módulo.

Figura 39 - Exemplo de tela do módulo Jogo Associativo

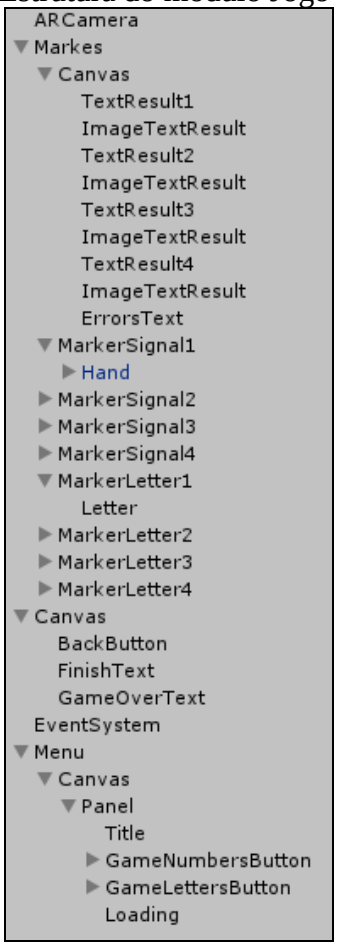


Fonte: elaborado pelo autor.

O `item 1` da Figura 39 exibe ao usuário todos os sinais que ele precisa associar para ganhar o jogo. Nesse mesmo item, é mostrado também quais já foram associados de forma correta, por exemplo, o número dois é o único sinal que já foi associado, por isso está marcado com um sinal de correto ao lado dele. O `item 2` da Figura 39 exibe ao usuário a quantidade de erros cometidos e o total de erros que pode acontecer durante o jogo. O `item 3` da Figura 39 é o marcador responsável por exibir o sinal na LIBRAS. O `item 4` da Figura 39 é referente ao marcador responsável por exibir o significado do sinal. Por fim, o `item 5` da Figura 39 é um botão para o usuário voltar a tela inicial do sistema (Figura 31).

A estrutura do módulo Jogo Associativo poder ser visualizada na Figura 40. É composto por uma `ARCamera` do Vuforia, responsável por exibir a câmera do celular e detectar os marcadores. A `ARCamera` possui o *script* `GameARScript` responsável pelas ações da tela. Um `GameObject Canvas` que possui um botão `BackButton` responsável por voltar a tela inicial do sistema, e dois `GameObject Text`, para exibir a mensagem que usuário ganhou ou perdeu o jogo. Um `GameObject Menu` que possui abaixo dele os componentes para exibir a tela inicial do módulo (Figura 35). E por fim um `GameObject Markers`, que possui todos os componentes mostrados na Figura 39.

Figura 40 - Estrutura do módulo Jogo Associativo



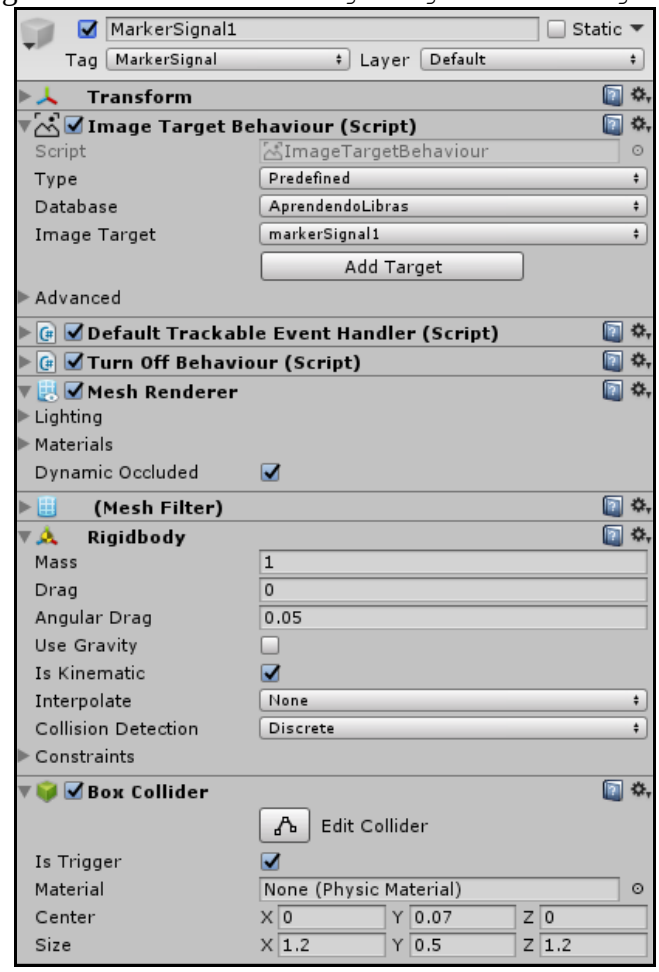
Fonte: elaborado pelo autor.

Como pode ser visto na Figura 40, todos os `GameObject` que iniciam com o nome `Marker` são `ImageTarget` para representar os marcadores. Os `ImageTarget` que iniciam com o nome `MarkerSignal` representam os marcadores que exibem os sinais, eles possuem a `tag MarkerSignal` para poder resgatá-los de forma mais fácil no código. Os `ImageTarget` que iniciam com o nome `MarkerLetter` representam os marcadores que exibem os significados dos sinais, eles possuem a `tag MarkerLetter` para poder resgatá-los de forma mais fácil no código.

Para detectar a associação entre os marcadores, foi usado o sistema de colisões do Unity para cada `ImageTarget` do sistema. O sistema de colisões do Unity obriga que todos os `GameObject` que iram sofrer colisões tenham pelo menos o componente `Rigdbody` e algum `Collider` (no sistema desenvolvido foi usado `Box Collider`). O componente `Rigdbody` disponibiliza ao `GameObject` os controles de física do Unity. O componente `Box Collider` gera a área de colisão do `GameObject` associado a ele no formato de um paralelepípedo. Para os `ImageTarget` do sistema desenvolvido, no componente `Rigdbody` foi desabilitado a opção `Use Gravity` para não sofrer o efeito da gravidade e habilitado a opção `Is Kinematic` para

não ser acionado o sistema de física. Para o componente `Box Collider` foi habilitado a opção `Is Trigger` para quando o `GameObject` sofrer uma colisão, somente disparar os eventos de colisão e não sofrer visualmente os efeitos da colisão, como ser empurrado por exemplo. A Figura 41 mostra os atributos do `ImageTarget MarkerSignal1`.

Figura 41 - Atributos do `ImageTarget MarkerSignal1`



Fonte: elaborado pelo autor.

Depois do usuário escolher o conteúdo que será usado no módulo (Figura 35), todos os sinais são escolhidos aleatoriamente de acordo com o conteúdo. O Quadro 8 mostro todo esse processo. Primeiramente, na função `CreateMarkers` são usadas 5 variáveis. A variável `aleatorySignals` irá guardar os sinais aleatórios escolhidos por essa função. A variável `markersSignals` guarda todos marcadores que contém os sinais. A variável `markersLetters` guarda todos os marcadores que contém os significados dos sinais. E as varáveis `textResults` e `imagesTextResults` guardam os textos e as imagens que aparecem ao lado dos textos do `item 1` da Figura 39. Depois de escolher um sinal aleatório que ainda não foi escolhido da variável `currentSignals`, armazena o valor na variável `aleatorySignals`, atualiza o valor do `GameObject Text` do marcador que exibe o significado do sinal e

concatena o valor no final do nome do marcador e do texto de resultados (`item 1` da Figura 39). A ação de concatenar o valor no nome dos marcadores e dos textos de resultados é necessário porque na hora de validar se a colisão entre dois marcadores foi realizada corretamente, é validado pelo final do nome dos marcadores que colidiram.

Quadro 8 - Método para escolher os sinais aleatoriamente

```
...
    void CreateMarkers()
    {
        aleatorySignals = new string[] { "", "", "", "" };
        int[] positionMarkers = { 0, 1, 2, 3 };

        for (int i = 0; i < 4; i++)
        {
            string signal = currentSignals[Random
                                .Range(0, currentSignals.Length)];

            while (System.Array.IndexOf(aleatorySignals, signal) != -1)
            {
                signal = currentSignals[Random
                            .Range(0, currentSignals.Length)];
            }

            aleatorySignals[i] = signal;
            markersSignals[i].name += signal;
            textResults[i].GetComponent<Text>().text = signal;
            imagesTextResult[i].name += signal;

            int position = positionMarkers[Random
                               .Range(0, positionMarkers.Length)];
            positionMarkers = positionMarkers.ToList()
                                .Where(x => x != position).ToArray();
            markersLetters[position].name += signal;
            markersLetters[position]
                    .GetComponentInChildren<TextMesh>().text = signal;
        }
    }
...
```

Fonte: elaborado pelo autor.

Para validar se a associação entre os marcadores foi realizada corretamente, os marcadores que exibem os significados dos sinais têm o *script* `DetectTrigger`, que possui as funções `OnTriggerEnter` e `OnTriggerExit` (Quadro 9). A função `OnTriggerEnter` é chamada sempre quando o sistema de colisão do Unity detecta que dois `GameObject` se tocaram, nesse caso dois marcadores. Essa função irá validar primeiramente se não foi uma colisão entre dois marcadores que exibem os sinais ou os significados, nesse caso não realiza nenhuma ação. Caso for entre um marcador de sinal e outro que exibe o significado, valida se a última letra do nome dos marcadores são iguais. Caso for, muda a cor da letra para azul, marca que foi associado corretamente o sinal nos textos de resultados a chama a função `CheckFinishGame` do *script* `GameARScript` para verificar se o usuário ganhou. Se for

associado de forma errada, muda a cor da letra para vermelho e chama a função `CheckErrors` do *script* `GameARScript` para verificar se o usuário perdeu.

Quadro 9 - Métodos para detectar colisão dos marcadores

```
...
    void OnTriggerEnter(Collider other)
    {
        if (gameObject.tag.Equals(other.tag))
        {
            return;
        }

        char signal = gameObject.name[gameObject.name.Length - 1];

        if (signal.Equals(other.gameObject
                                .name[other.gameObject.name.Length - 1]))
        {
            text.GetComponent<TextMesh>().color = Color.blue;
            GameObject.Find("ImageTextResult" + signal)
                        .GetComponent<Image>().sprite = spriteCheck;

            gameARScript.CheckFinishGame(signal.ToString());
        }
        else
        {
            text.GetComponent<TextMesh>().color = Color.red;

            gameARScript.CheckErrors();
        }
    }

    void OnTriggerExit(Collider other)
    {
        text.GetComponent<TextMesh>().color = new Color32(0, 212, 1, 255);
    }
```

Fonte: elaborado pelo autor.

Para os marcadores exibem os sinais na LIBRAS, toda vez que o Vuforia detecta o marcador é disparada a animação do sinal na LIBRAS para o usuário saber qual é o sinal do marcador. Para isso, esses marcadores possuem o *script* `DetectTrackable`, que implementa a interface do Vuforia `ITrackableEventHandler`. Essa interface obriga o *script* a implementar o método `OnTrackableStateChanged`, que é chamado sempre quando é trocado o estado do marcador, por exemplo quando ele for detectado. Para fazer essa função executar a animação do sinal na LIBRAS, é pego a última letra do nome do marcador, que indica qual é o sinal desse marcador, encontra o componente `Animator` da mão 3D e executa a animação. O Quadro 10 mostra essa função.

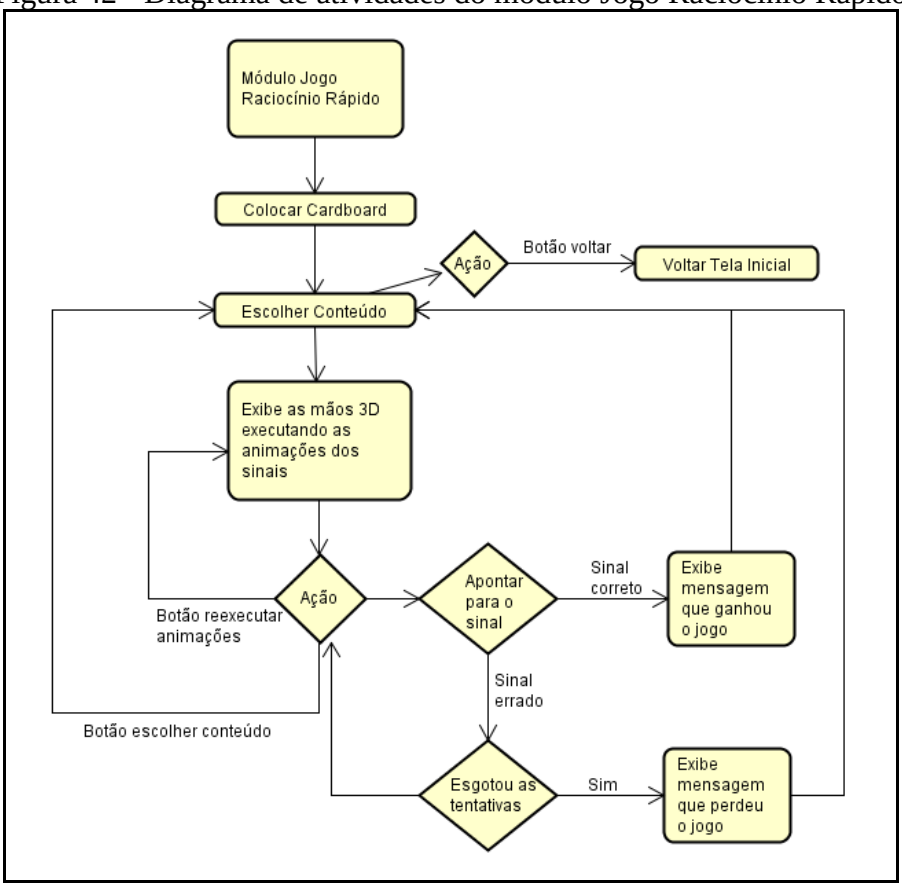
Quadro 10 - Método para executar a animação ao detectar o marcador

```
...
    public void OnTrackableStateChanged(
                        TrackableBehaviour.Status previousStatus,
                        TrackableBehaviour.Status newStatus)
    {
        if (newStatus == TrackableBehaviour.Status.DETECTED ||
            newStatus == TrackableBehaviour.Status.TRACKED ||
            newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
        {
            char lastLetter = gameObject.name[gameObject.name.Length - 1];

            GetComponentInChildren<Animator>().Rebind();
            GetComponentInChildren<Animator>()
                     .Play("Signal" + lastLetter);
        }
    }
```

Fonte: elaborado pelo autor.

A Figura 42 mostra o diagrama de atividades do módulo Jogo Raciocínio Rápido. O objetivo desse módulo é disponibilizar ao usuário uma forma mais interativa para treinar seus conhecimentos na LIBRAS usando RV. Para o uso desse módulo, é necessário ter algum modelo de Cardboard. Depois de entrar no módulo, é necessária colocar o celular no Cardboard e escolher o conteúdo que ele irá visualizar, podendo ser letras do alfabeto ou algarismos numéricos (Figura 35).

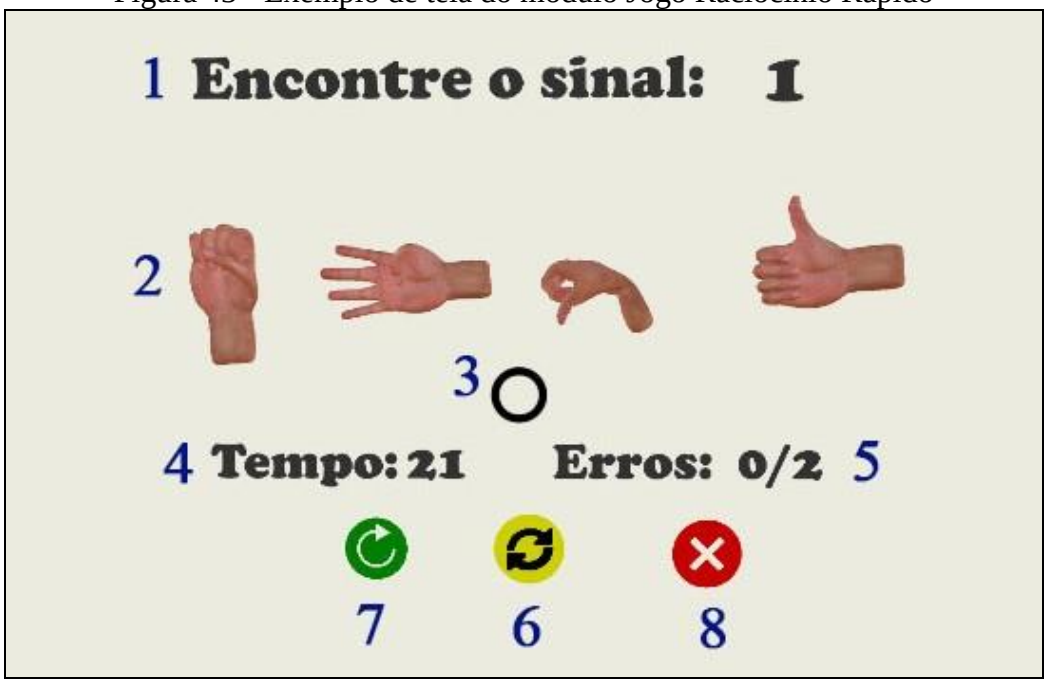
Figura 42 - Diagrama de atividades do módulo Jogo Raciocínio Rápido



Fonte: elaborado pelo autor.

Após escolher o conteúdo do módulo, o usuário irá ver uma tela parecida com a da Figura 43. Agora o objetivo é mexer a cabeça de uma forma que aponte para o sinal que o jogo está pedindo. O `item 1` da Figura 43 é o objetivo do usuário, indica qual sinal deve ser encontrado. O `item 2` da Figura 43 são os sinais disponíveis para serem escolhidos, sendo que só um deles representa o significado do `item 1`. O `item 3` da Figura 43 é um círculo que indica para onde a sua cabeça está apontando no jogo. Para selecionar um dos sinais, basta manter o círculo em cima de uma das mãos 3D e esperar por aproximadamente dois segundos. Para indicar esse tempo, o círculo ficará complementarmente branco. O `item 4` da Figura 43 mostra ao usuário o tempo que falta para o jogo acabar. O `item 5` da Figura 43 indica a quantidade de erros cometidas e a quantidade total. O `item 6` da Figura 43 é um botão que irá executar novamente a animação de todos os sinais para o usuário. O `item 7` da Figura 43 é um botão para o usuário voltar a tela para escolher um novo conteúdo para o módulo (Figura 35). O `item 8` da Figura 43 é um botão para o usuário voltar ao menu inicial do sistema (Figura 31).

Figura 43 - Exemplo de tela do módulo Jogo Raciocínio Rápido

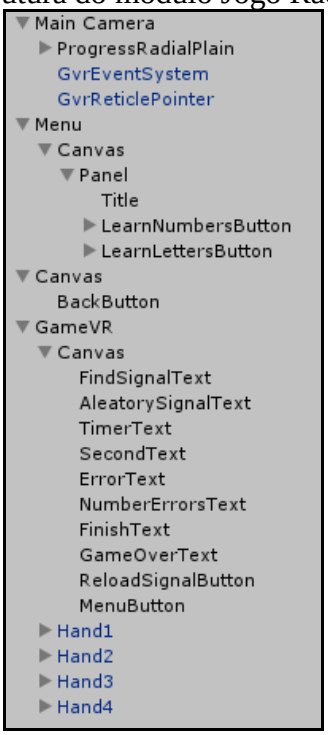


Fonte: elaborado pelo autor.

A estrutura do módulo Jogo Raciocínio Rápido por ser visualizada na Figura 44. É composto por uma `Main Camera` que possui o *script* `GameVRScript` responsável pelas ações da tela. Abaixo dela é encontrado os `prefabs GvrEventSystem` e `GvrReticlePointer` do Google VR, e o `prefab ProgressRadialPlain`. Esse `prefab` foi baixado de um *asset* gratuito do Unity chamdo Progress Bar Scripts, que disponibiliza vários `prefabs` para exibir a porcentagem de um carregamento. O uso desses três `prefabs` será explicado mais para frente.

Um `GameObject Menu` que possui abaixo dele os componentes para exibir a tela inicial do módulo (Figura 35). Um `GameObject Canvas` que possui um botão `BackButton` responsável por voltar a tela inicial do sistema. Um GameObject GameVR que possui todos os componentes mostrados na Figura 43.

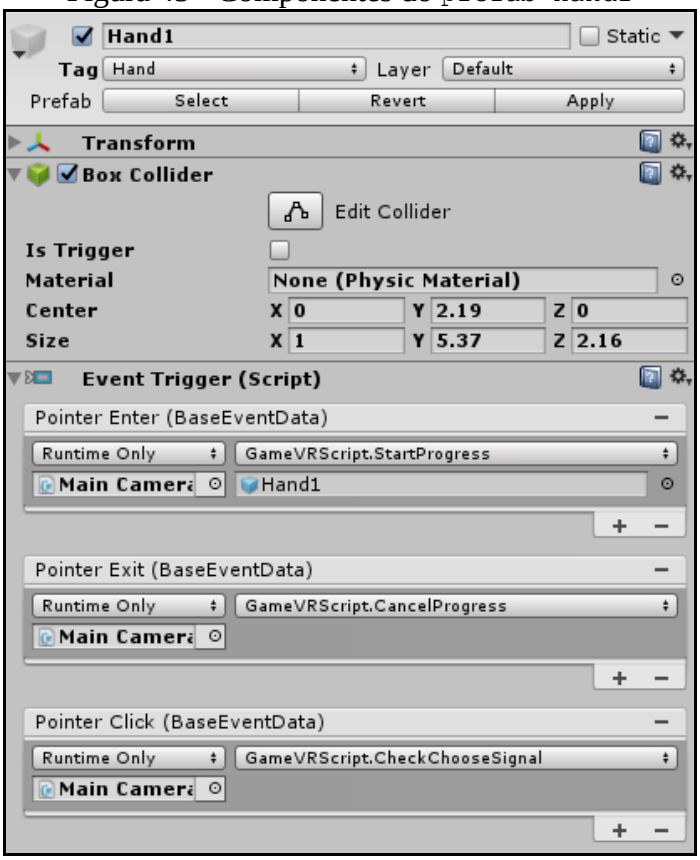
Figura 44 - Estrutura do módulo Jogo Raciocínio Rápido



Fonte: elaborado pelo autor.

Esse módulo disponibiliza ao usuário uma forma diferente de interação, usando o Cardboard e movimentando a cabeça para selecionar os objetos. Para isso foi usado o `prefab GvrReticlePointer` abaixo da `Main Camera`. Esse `prefab` exibe um círculo no meio da tela que acompanha a movimentação da cabeça do usuário junto com a câmera. Ele gera também eventos de colisão ao ser passado por cima de `GameObjects` que possuam algum componente `Collider`. E junto ao `prefab GvrReticlePointer`, é usado o `prefab ProgressRadialPlain`, para mostrar ao usuário o tempo que irá levar para executar a ação ao posicionar o `GvrReticlePointer` sobre um `GameObject`. Então quando o usuário posiciona o `GrvReticlePointer` sobre um `GameObject`, é disparado uma ação para iniciar o progresso de tempo, que ao ser finalizado, executa uma função que foi atribuída ao `GameObject`. Com isso, todos os `GameObjects` da tela que podem sofrer alguma ação possuem um componente `Box Collider` e o *script* `Event Trigger` configurado com os eventos `Pointer Enter`, `Pointer Exit` e `Pointer Click`. A Figura 45 exibe os componentes do `prefab Hand1`.

Figura 45 - Componentes do `prefab Hand1`



Fonte: elaborado pelo autor.

Como pode ser observado na Figura 45, o evento `Pointer Event` é chamado quando o usuário aponta o `GvrReticlePointer` para cima do `GameObject`. O evento `Pointer Exit` é chamado quando o usuário retira o `GvrReticlePointer` de cima do `GameObject`. Todos os `GameObjects` chamam a função `StartProgress` e `CancelProgress` nos eventos `Pointer Enter` e `Pointer Exit` respectivamente. O que muda é o parâmetro passado para a função `StartProgress`, que sempre será o `GameObject` que está sofrendo a ação. A função `StartProgress` guarda o objeto passado por parâmetro na variável `objectClick` e inicia o tempo do progresso. E a função `CancelProgress` encerra o tempo do progresso. O `prefab ProgressRadialPlain` possui um *script* `ProgressRadialBehaviour`, esse *script* recebe a função `ExecuteAction` que será executada após todo o progresso for completado (Figura 46). A função `ExecuteAction` tem o objetivo de simular um toque na tela do usuário em cima do `GameObject` que está na variável `objectClick`. Assim, será chamada à função configurada no `Pointer Click` do `GameObject`. O Quadro 11 mostra as funções `StartProgress`, `CancelProgress` e `ExecuteAction`.

Figura 46 - *Script* `ProgressRadialBehaviour` do `prefab ProgressRadialPlain`

Fonte: elaborado pelo autor.

Quadro 11 - Métodos referentes as ações do `GvrReticlePointer`

```
...
    public void StartProgress(GameObject objectClick)
    {
        this.objectClick = objectClick;
        scriptProgress.IncrementValue(100);
    }

    public void CancelProgress()
    {
        scriptProgress.Value = 0.01f;
        scriptProgress.TransitoryValue = 0;
    }

    public void ExecuteAction()
    {
        CancelProgress();
        ExecuteEvents.Execute<IPointerClickHandler>(
              objectClick,
              new PointerEventData(EventSystem.current),
              ExecuteEvents.pointerClickHandler);
    }
...
```

Fonte: elaborado pelo autor.

3.5 ANÁLISE DOS RESULTADOS

Esta seção é dedicada a mostrar as discussões e experimentos realizados com o sistema. Na seção 3.5.1 é apresentado a ideia inicial do projeto junto com alterações que foram realizadas. Na seção 3.5.2 é mostrado os experimentos realizados no sistema. E a seção 3.5.3 apresenta uma comparação entre os trabalhos correlatos e o sistema desenvolvido.

3.5.1 Ideia inicial do sistema

Na ideia inicial do sistema, era previsto que o segundo módulo do sistema fosse usar marcadores fiduciais para identificar a associação correta entre os marcadores. Porém, nos testes foi encontrado alguns problemas com essa abordagem. O primeiro problema foi na criação dos marcadores. Pois era necessário criar um marcador sem a parte do meio, e outro com somente a parte do meio, para que quando colocasse um em cima do outro fosse gerado

outro marcador. E ao juntar os dois marcadores, não poderia gerar um marcador igual a outro, sendo necessário gerar marcadores com desenhos específicos para não gerar esse conflito. Após esse processo, foi verificado também que ao tentar realizar a associação colocando um marcador em cima do outro, na maioria das vezes, o Vuforia não identificava de forma rápida o novo marcador. Pois dependendo de como era colocado o marcador um em cima do outro, não ficava totalmente correto, sendo necessário sempre fazer ajustes. E também foi notado que nessa etapa de colocar um marcador em cima do outro para realizar a associação, o Vuforia acabava perdendo o vínculo com os marcadores. Pois a mão acabava ficando por cima do marcador. Com isso, foi usado o sistema de colisão do Unity para realizar a associação entre os marcadores, que ao encostar um ao outro, é realizado a associação.

Para o terceiro módulo do sistema, a ideia inicial era usar o Vuforia junto com RV. O objetivo era disponibilizar ao usuário um marcador no formato de um cubo que mostraria a mão 3D executando o sinal na LIBRAS. Com isso, após o usuário colocar o HMD, ele iria pegar com uma mão o marcador, e com a outra mão, ele tentaria repetir o sinal na LIBRAS, podendo colocar sua mão ao lado da mão 3D. Porém, ao realizar os testes, ao colocar o HMD, acabou não ficando muito nítida a imagem junto com o Vuforia, causando um cansaço na visão. Com isso, foi implementado um jogo de raciocínio rápido usando o Google VR para interação com o HMD.

3.5.2 Experimento e resultado do sistema

No decorrer do desenvolvimento do sistema foram realizadas duas reuniões com a professora de LIBRAS Pelence (2018). Na primeira reunião foi mostrado a animação dos sinais na LIBRAS já implementados para o sistema e uma demonstração de como seria os módulos. Nessa reunião, a professora destacou alguns sinais que estavam sendo representados incorretamente. Com isso, foi gravado a animação de todos os sinais (Apêndice A) e compartilhado com a professora para análise. Após a análise feita pela professora, ela gravou vídeos demonstrando a forma correta de representar os sinais na LIBRAS que estavam incorretos (Anexo A) para serem ajustados.

Agora na segunda reunião, foi para mostrar o sistema completamente desenvolvido. No primeiro módulo, a professora destacou que não estava muito claro quando a animação do sinal na LIBRAS executada pela mão 3D tinha finalizado ou não. Com isso, enquanto a mão 3D está executando a animação, foi inserido uma imagem para mostrar ao usuário que a animação está em andamento. E quando a animação termina, foi inserido uma outra imagem indicando que foi finalizada. Fotos sobre o dia desta reunião são mostradas no Apêndice B.

O experimento do sistema foi realizado no mês de junho de 2017 na Associação Blumenauense de amigos dos Deficientes Auditivos (ABADA), no período vespertino. Fotos sobre o dia desta reunião são mostradas no Apêndice C. A turma tinha um total de cinco alunos, sendo todos deficientes auditivos, e alguns com outras deficiências psicológicas. Antes dos alunos usarem o sistema, foi realizado uma explicação geral sobre o sistema, mostrando a funcionalidade de cada módulo. Após isso, foi separado os alunos em três grupos para testar o sistema, fornecendo dois tablets e um smartphone. Cada grupo tinha uma pessoa para auxiliar no uso do sistema, sendo o professor Dalton Solano dos Reis (orientador deste trabalho), professora Fernanda Pelence (coorientadora deste trabalho), e eu @@É ASSIM QUE ME REFIRO?@@. Depois que os grupos foram formados, o professor Dalton foi passando orientações para o uso do primeiro módulo, e os alunos tinham o objetivo de acompanhar usando o sistema. Após terminado as orientações do primeiro módulo, os alunos ficavam livres para interagir com todas as funcionalidades do primeiro módulo. Depois foi realizado os mesmos passos para o uso do segundo módulo do sistema. E para o terceiro módulo, foi passado orientações individualmente para cada aluno e, logo em seguida, o aluno interagia com o terceiro módulo. Essa abordagem no terceiro módulo foi necessária pois só tínhamos um HMD para realizar os testes.

No primeiro módulo, foi observado que alguns alunos não ficaram muito interessados, pois todos já sabiam LIBRAS, e o primeiro módulo tem o objetivo de mostrar as letras e os algarismos numéricos na LIBRAS. E como era o primeiro módulo que eles estavam usando, no início pode ser observado um pouco de dificuldade no uso, porém após alguns minutos todas já estavam usando sozinhos. No segundo módulo, todos ficaram muito interessados em usar, por ser um jogo que usa uma tecnologia que é nova nos dias de hoje. Nesse módulo, foi observado um pouco de dificuldade no começo na hora de associar os marcadores e segurar o celular ao mesmo tempo. Porém depois de alguns minutos todos já estavam usando sozinhos porque já estavam familiarizados com o uso do primeiro módulo. No terceiro módulo foi observado duas dificuldades. A primeira foi no primeiro momento em que se usava o HMD, os alunos demoravam um pouco para entender que o movimento da cabeça era necessário para interagir com o sistema. Mas depois de alguns minutos já estavam interagindo corretamente com o sistema. A segunda dificuldade foi para alunos que usavam óculos, porque era necessário tirar o óculos para usar o HMD, e isso acabava dificultando na identificação dos sinais no jogo.

No final, todos ficaram livres para usar o sistema, e pode observar que todos já estavam conseguindo interagir com o sistema sozinhos. O que tornou mais fácil o uso de

todos os módulos foi o padrão de botões utilizados. Pode observar que depois dos alunos terem passado pelo primeiro módulo, nos outros dois módulos, eles já sabiam o que cada botão realizava no sistema. Por fim, pode observar que o segundo módulo que usa RA ocasionou alguns travamentos em smartphones ou tablets que tenham hardware mais limitado.

3.5.3 Comparação com os trabalhos correlatos

Está seção faz uma comparação entre o sistema desenvolvido com os trabalhos correlatos apresentados na seção 2.4. O Quadro 12 apresenta a comparação entre as principais características do sistema.

Quadro 12 - Comparativo entre os trabalhos correlatos

Características / Trabalhos correlatos	Ferramenta desenvolvida	Freire et al. (2015)	Santos e Souza et al. (2013)	Santos e Lobo et al. (2013)
Plataforma	Smartphone	Computador	Computador	Computador
Exibir os sinais em 3D	X	Sim	Parcialmente	Parcialmente
Realidade Aumentada	X	X	X	X
Realidade Virtual	X			
Associar os objetos virtuais	X	X	X	X
Consultar o sinal de um algarismo numérico ou letra	X			
Forma de associação dos marcadores	Sistema de colisão	Marcador fiducial	Marcador fiducial	Não informado
Ferramentas utilizadas	Unity, SDK Vuforia e Google VR	ARToolKit	ARToolKit	ARToolKit

Fonte: elaborado pelo autor

Com base no Quadro 12, podemos notar que apenas o sistema desenvolvido foi feito para smartphones, sendo os outros trabalhos para computador. Podemos observar também que apenas o sistema desenvolvido e o trabalho de Freire et al. (2015) exibem os sinais em 3D. Os outros trabalhos exibem um cubo que contém em todos os seus lados o sinal na LIBRAS. Todos os trabalhos usam realidade aumentada. Porém, somente o sistema desenvolvido faz uso da realidade virtual. É possível observar também que todos os sistemas disponibilizam a associação de objetos virtuais para o aprendizado da LIBRAS. Entretanto, somente o sistema desenvolvido possui uma tela para poder visualizar cada sinal na LIBRAS de forma individual. Podemos analisar também que somente o sistema desenvolvido usou um sistema de colisão para realizar a associação dos marcadores. O trabalho de Freire et al. (2015) e Santos e Souza et al. (2013) usaram marcadores fiduciais, e no trabalho de Santos e Lobo et

al. (2013) não foi informado. Por fim, somente o sistema desenvolvido utilizou Unity junto ao Vuforia e Google VR para o desenvolvimento do sistema. Os outros trabalhos utilizaram a ferramenta ARToolKit.

4 CONCLUSÕES

Este trabalho mostrou o desenvolvimento de um sistema para auxiliar no ensino de Libras utilizando Realidade Aumentada e ~~Realidade~~ Virtual. O objetivo de disponibilizar um sistema que ajude no ensino dos conceitos básicos de Libras foi atingido, de acordo com os testes realizados. O objetivo de disponibilizar uma mão 3D executando os sinais foi atingido, pois todos os módulos do sistema possuem essa característica. E os objetivos de disponibilizar jogos usando a Realidade Aumentada e Virtual foi atingido com o segundo e terceiro módulo do sistema. Com os testes, comprovou-se também que o uso da Realidade Aumentada e Virtual tornou o sistema muito mais divertido e envolvente.

As ferramentas usadas durante o desenvolvimento do sistema se mostraram adequadas para os fins que foram utilizadas. Os marcadores gerados pela ferramenta Brosvision (2013) foram muito bem detectados pelo Vuforia. O Blender facilitou muito a criação do esqueleto da mão em cima do molde. O Vuforia ficou responsável por toda a parte de Realidade Aumentada do sistema, tornando muito mais fácil a identificação dos marcadores e interação entre eles. O Google VR cuidou de toda a parte de Realidade Virtual no sistema, tornando mais fácil a seleção de objetos virtuais com o movimento da cabeça. O Unity facilitou muito o desenvolvimento da ferramenta, principalmente com o motor de animação, onde possibilitou criar todas as animações dos sinais de Libras em cima da mão 3D. E também com o sistema de colisões, que facilitou muito para detectar a associação entre os marcadores.

Por fim, este trabalho deixa uma contribuição social, onde o sistema desenvolvido pode ser usado para auxiliar no ensino dos conceitos básicos de Libras por professores ou até mesmos pelos próprios alunos. Os kits de marcadores e o HMD usados para o desenvolvimento deste trabalho encontram-se no laboratório LIFE da Universidade Regional de Blumenau com o intuito de serem utilizados para este fim. Quanto a contribuição científica, este trabalho deixa as soluções encontradas para trabalhar em um mesmo sistema com o Vuforia e Google VR. E também a forma como foi desenvolvido o esqueleto da mão 3D junto com as animações no Unity.

4.1 EXTENSÕES

Para os trabalhos futuros são sugeridos:

a) implementar no segundo módulo a possibilidade de o usuário usar o HMD para poder ficar com as duas mãos livres para manusear os marcadores;
b) criar um sistema de pontuação por usuário para o segundo e terceiro módulo do sistema;

c) implementar uma interface na LIBRAS para o sistema;
d) melhorar o desempenho do segundo módulo de RA, para funcionar melhor em smartphones e tablets com hardware mais limitado.

REFERÊNCIAS

BRASIL. Lei n° 10.436, de 24 de abril de 2002. **Dispõe sobre a Língua Brasileira de Sinais – Libras e dá outras providências**. Brasília, 24 abr. 2002.

BRITO, Lucinda. **Por uma gramática de língua de sinais.** Rio de Janeiro: Tempo Brasileiro, 1995. 273 p.

BROSVISION. **Augmented Reality Marker Generator**. 2013. Disponível em: <http://www.brosvision.com/ar-marker-generator/>. Acesso em: 04 mar. 2018.

CADNAV. **Free 3D Models, CAD Models and Textures Download.** 2018. Disponível em: <http://www.cadnav.com>. Acesso em: 10 mar. 2018.

CARDOSO, Alexandre et al. **Tecnologias para o desenvolvimento de sistemas de realidade virtual e aumentada.** Recife: Editora Universitária UFPE, 2007. 210 p.

CECHINEL, Lenita. **Inclusão do aluno surdo no ensino superior:** um estudo do uso de língua brasileira de sinais (LIBRAS) como meio de acesso ao conhecimento científico. 2005. 71 f. Dissertação (Mestrado) – Curso de Pós-Graduação Stricto Sensu, Centro de Educação de Ciências Humanas e da Comunicação, Universidade do Vale do Itajaí, Itajaí, 2005.

FORTE, Cleberson E.; KIRNER, Cláudio. **Usando Realidade Aumentada no Desenvolvimento de Ferramenta para Aprendizagem de Física e Matemática**. 2009. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wrva/2009/008.pdf>. Acesso em 05 set. 2017.

FREIRE, Matheus et al. Realidade aumentada como ferramenta de apoio na alfabetização de crianças com surdez usuárias da Língua Brasileira de Sinais. In: CONGRESSO NACIONAL DE AMBIENTES HIPERMÍDIA PARA APRENDIZAGEM, 7., 2015, São Luiz. **Anais...** São Luiz, 2015, p. 1-10.

KIRNER, Claudio; KIRNER, Tereza G. Evolução e Tendências da Realidade Virtual e da Realidade Aumentada. In: RIBEIRO, Marcos; ZORZAL, Ezequiel (Org.). **Realidade Virtual e Aumentada:** Aplicações e Tendências. Uberlândia: SBC – Sociedade Brasileira de Computação, 2011. Cap. 1. p. 10-25. Disponível em: < http://www.de.ufpb.br/~labteve/publi/2011_svrps.pdf>. Acesso em: 21 out. 2017.

KIRNER, Claudio; SISCOUTTO, Robson. **Realidade Virtual e Aumentada:** Conceitos, Projeto e Aplicações. Porto Alegre: Editora SBC, 2007. 292 p.

LESTON, Jean. Virtual reality: the IT perspective. **ITNOW**, Oxford, v. 38, p. 12-13, jun. 1996. Disponível em: <https://doi.org/10.1093/combul/38.3.12>. Acesso em: 12 maio 2018.

LOPES, Raquel A. **Um olhar sobre o ensino de Libras na formação inicial em pedagogia:** utopia ou realidade? 2013. 89 f. Dissertação (Mestrado em Psicologia) – Universidade Presbiteriana Mackenzie, São Paulo, 2013.

MARQUES, Hivi; BARROCO, Sonia; SILVA, Santos. O ensino da língua Brasileira de sinais na educação infantil para crianças ouvintes e surdas: considerações com base na psicologia histórico-cultural. **Revista Brasileira de Educação Especial**, Marília, v. 19, n. 4, p. 503-517, 2013. Trimestral. Disponível em: < http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-65382013000400003>. Acesso em: 30 ago. 2017.

MONTEIRO, Myrna S. História dos movimentos dos surdos e o reconhecimento da Libras no Brasil. **ETD – Educação Temática Digital**. p. 295-305. 2006. Disponível em: <http://nbn-resolving.de/urn:nbn:de:0168-ssoar-101789>. Acesso em: 29 ago. 2017.

PELENCE, Fernanda. **Entrevista sobre validação de requisitos**. Entrevistadores: Luan Ribeiro da Silva e Dalton Solano dos Reis. Blumenau. 2018. Entrevista feita através de conversação – não publicada.

QUADROS, Ronice; KARNOPP, Lodenir. **Língua de sinais brasileira:** estudos linguísticos. Porto Alegre: ARTMED, 2004. 221 p.

REVISTA DA FENEIS. Rio de Janeiro: FA Editoração Eletrônica Ltda., Ano 1, n. 2, 1999. Disponível em: <https://issuu.com/feneisbr/docs/revista_feneis_02>. Acesso em: 30 out. 2017.

RIOS, Ailton. **LIBRAS – Alfabeto e Números**. [2012?]. Disponível em: <http://www.ebah.com.br/content/ABAAAA9skAJ/libras-alfabeto-numeros>. Acesso em: 19 out. 2017.

SANTOS, Luiz; LOBO, Tonykley et al. Jogando com a Realidade Aumentada e Aprendendo LIBRAS. In: NUEVAS IDEAS EM INFORMÁTICA EDUCATIVA, 9., 2013, Porto Alegre. **Anais...** Porto Alegre, 2013, p. 455-458. Disponível em: <http://www.tise.cl/volumen9/TISE2013/455-458.pdf>. Acesso em: 27 ago. 2017.

SANTOS, Luiz; SOUZA, Antonio et al. Aprendendo números em LIBRAS com a tecnologia da realidade aumentada. In: SBGAMES, 9., 2013, São Paulo. **Anais...** São Paulo, 2013, p. 21-24. Disponível em: <http://www.sbgames.org/sbgames2013/proceedings/workshop/WorkshopVAR-7_Full.pdf>. Acesso em: 26 ago. 2017.

SCHLÜNZEN, Elisa; BENEDETTO, Laís; SANTOS, Danielle. O que é Libras? **Conteúdos e Didática de Libras.** Presidente Prudente, v. 11, n. 24, 2012. p. 45-48. Disponível em: <http://acervodigital.unesp.br/handle/123456789/47933>. Acesso em: 19 out. 2017.

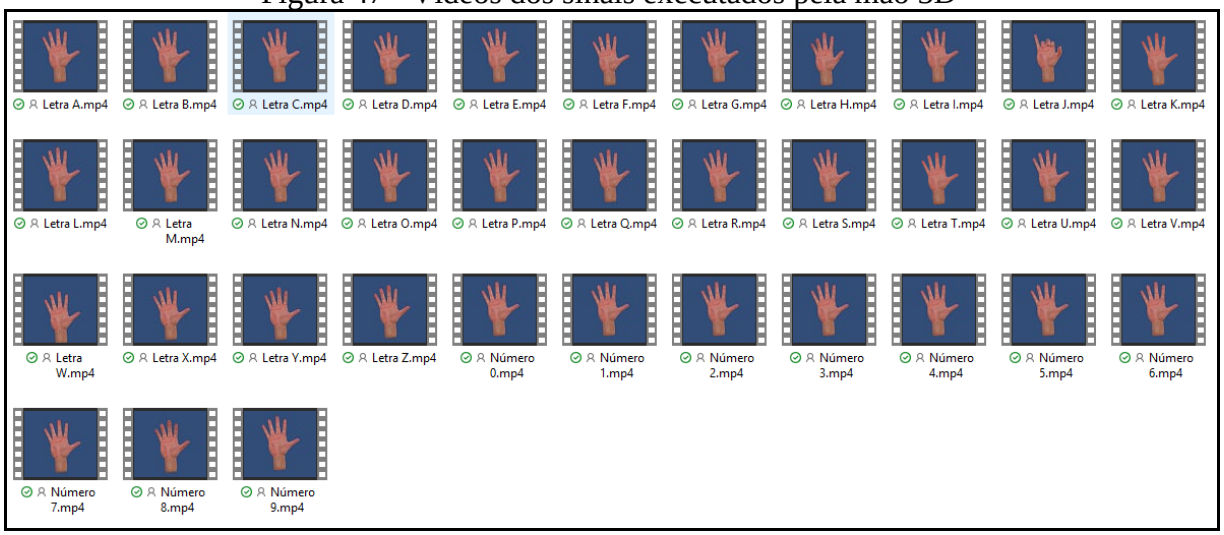
SILVA, Giselli M. **Parâmetros da Libras**. [2011?]. 10 p. Disponível em: <http://webletras01.letras.ufmg.br/dialogosdeinclusao/data1/arquivos/Parametros_da_Libras.pdf>. Acesso em: 18 out. 2017.

TORI, Romero; KIRNER, Claudio; SISCOUTTO, Robson. **Fundamentos e Tecnologia de Realidade Virtual e Aumentada.** Porto Alegre: Editora SBC, 2006. 412 p.

APÊNDICE A – Vídeos das animações feitas na mão 3D

Neste apêndice é mostrado os vídeos gravados da mão 3D. A Figura 47 mostra todos os vídeos gravados para auxiliar a professora de LIBRAS Pelece (2018) na correção das animações dos sinais.

Figura 47 - Vídeos dos sinais executados pela mão 3D



Fonte: elaborado pelo autor.

APÊNDICE B – Reunião sobre o sistema desenvolvido

Neste apêndice é mostrado as fotos da reunião realizada com a professora Pelece (2018) na Universidade Regional de Blumenau. A Figura 48 mostra eu apresentando o sistema completo. A Figura 49 mostra a professora e sua aluna usando o segundo módulo do sistema. E a Figura 50 mostra a aluna da professora usando o terceiro módulo do sistema.

Figura 48 - Mostrando o sistema

Fonte: elaborado pelo autor.

Figura 49 - Professora e sua aluna usando o segundo módulo

Fonte: elaborado pelo autor.

Figura 50 - Aluna usando o terceiro módulo

Fonte: elaborado pelo autor.

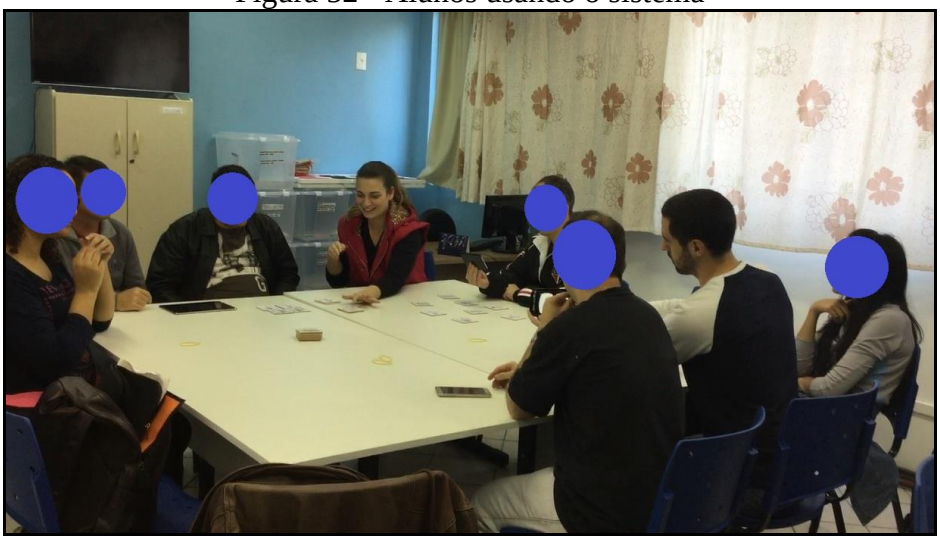
APÊNDICE C – Teste do sistema na ABADA

Neste apêndice são apresentadas fotos do teste do sistema realizado na Associação Blumenauense de amigos dos Deficientes Auditivos (ABADA) no mês de junho de 2018. A Figura 51 mostra eu apresentado as funcionalidades do sistema. A Figura 52 mostra os alunos usando o sistema.

Figura 51 - Mostrando o sistema para os alunos

Fonte: elaborado pelo autor.

Figura 52 - Alunos usando o sistema

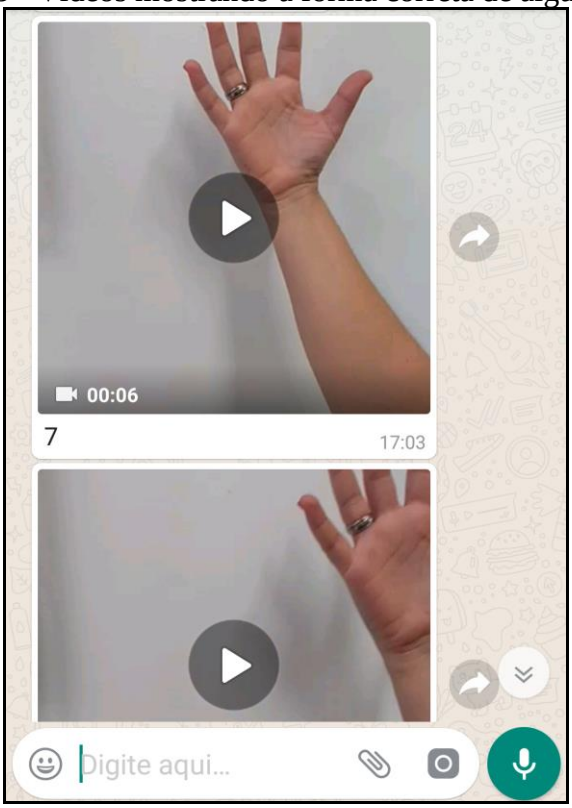


Fonte: elaborado pelo autor.

ANEXO A – Vídeos dos sinais na LIBRAS

Este anexo exibe uma parte da conversar com a professora de LIBRAS Pelece (2018), onde foi enviado vídeos dos sinais que estavam representados incorretamente no sistema (Figura 53).

Figura 53 - Vídeos mostrando a forma correta de alguns sinais



Fonte: Pelece (2018).