UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS APLICADAS A EDUCAÇÃO
DEPARTAMENTO DE CIÊNCIAS EXATAS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANIMADOR: UM SERVIÇO DE GERAÇÃO SEMIAUTOMÁTICA DE SINAIS EM LIBRAS

JOSÉ FRANCO NETO

Rio Tinto - PB
2015

JOSÉ FRANCO NETO

ANIMADOR: UM SERVIÇO DE GERAÇÃO SEMIAUTOMÁTICA DE SINAIS EM LIBRAS

Monografia apresentada para obtenção do título de Bacharel à banca examinadora no Curso de Bacharelado em Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE), Campus IV da Universidade Federal da Paraíba.

Orientador: Prof. Dr. Tiago Maritan U. Araújo

Rio Tinto - PB
2015

F825a Franco Neto, José.
Animador: um serviço de geração de semiautomática de sinais em LIBRAS. / José Franco Neto. – Rio Tinto: [s.n.], 2015.
64 f. : il.-

Orientador (a): Prof. Dr. Tiago Maritan U. Araújo.
Monografia (Graduação) – UFPB/CCAE.

1. Software - desenvolvimento. 2. Libras - software. 3. Língua Brasileira de Sinais.

UFPB/BS-CCAE CDU: 004.41 (043.2)

JOSÉ FRANCO NETO

ANIMADOR: UM SERVIÇO DE GERAÇÃO SEMIAUTOMÁTICA DE SINAIS EM LIBRAS

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal da Paraíba, Campus IV, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Assinatura do autor: _____

APROVADO POR:

_____

Orientador: Prof. Dr. Tiago Maritan U. Araújo

Universidade Federal da Paraíba – Campus I

_____

Prof. Ms. Danilo Assis Silva

Universidade Federal da Paraíba – Campus I

_____

Prof. Dr. Lincoln David Nery e Silva

Universidade Federal da Paraíba – Campus I

Rio Tinto - PB

2015

Aos amigos, colegas e professores, obrigado por fazer parte da minha vida e desse momento especial.

AGRADECIMENTOS

Gostaria primeiramente de agradecer a pessoa mais importante da minha vida que sem Ele não estaria onde estou, sem sua benção e força não seria capaz de realizar esse sonho. Deus agradeço pelo seu amor e carinho e por me conceder essa honra.

Aos meus pais, Francisca de Assis e Carlos Roberto, meu irmão Carlos Vinicius e minha irmã Shoyama Nadja por estarem comigo todo o tempo me apoiando e me dando forças, sem eles não conseguiria chegar ao final dessa caminhada.

Ao restante da minha família, minha tia Gorete Franco, meus avós Maria de Lurdes e José Franco, minhas primas Simone Santos e Suzane Santos que mesmo estando longe torceram por mim todo o tempo e isso me ajudou a prosseguir.

Aos amigos que fiz durante o curso, por que sem eles essa jornada seria mais chata e árdua, pois quem tem amigos tem tudo. Obrigado, Junior Galdino, Renan Soares, Erickson Silva, Carlo Chianca, Williby, Cryspo e tantos outros que sabem que fazem parte desse momento.

As pessoas que me ajudaram de certa forma, Aline Ferreira que revisou comigo o trabalho de conclusão de curso, André Araújo que esteve junto em projetos do LAVID, a Bruna Ochotorena por me suportar todas as vezes que eu estava irritado ou abusado, ela sempre foi minha força nos momentos difíceis.

Ao meu orientador Tiago Maritan por ter confiado em mim e ter me orientado, pelo exemplo de professor e ser humano dedicado no que faz.

Aos examinadores Danilo Assis e Lincoln David por me ajudarem a tornar esse trabalho melhor.

Em especial agradecer e dedicar essa honra a minha vó Lídia Alves que não está mais entre nós, mas que sempre sonhou em ver seu neto formado, vó essa é para você.

RESUMO

Os surdos passam por vários problemas no que diz respeito ao acesso a informações, principalmente as que circulam nas Tecnologias da Informação e Comunicação (TICs). Essas tecnologias normalmente são projetadas para pessoas que compreendem as línguas orais, ao passo que, para os surdos a língua oral é considerado uma espécie de “segunda língua”. Por outro lado, empresas responsáveis pela disponibilização dessas informações sentem grandes dificuldades em tornar acessível os conteúdos disponibilizados por elas, especialmente porque esses conteúdos são geralmente dinâmicos, e a produção das janelas em línguas de sinais (exemplo, Língua Brasileira de Sinais – LIBRAS) tem usualmente um custo operacional alto. Para minimizar esses problemas, uma alternativa é o uso de tradutores automáticos para línguas de sinais. Com esses tradutores automáticos, é possível gerar conteúdos digitais para Libras de forma dinâmica, com baixo custo, e segundo alguns trabalhos, com um nível de inteligibilidade razoável. Essas soluções, no entanto, geralmente necessitam de uma base de dados (dicionário de LIBRAS) contendo milhares de vídeos ou animações representadas por um avatar animado. O processo de construção dessas bases, no entanto, é um processo trabalhoso, e depende de uma equipe grande e especializada formada por modeladores, animadores e designers 3D, além de especialistas em Libras e usuários surdos. Para minimizar esses problemas, a proposta deste trabalho é desenvolver uma solução para geração semiautomática dessas animações (vídeos). A ideia é que essas animações possam ser criadas por usuários a partir da configuração de parâmetros (fonemas) da Língua Brasileira de Sinais (LIBRAS), reduzindo o tempo e o custo necessário para produzir esse tipo de base de dados.

Palavras chave: libras, animação, dicionário de Libras, avatar.

ABSTRACT

The deaf suffer several problems in regard to access of information, especially those information circulating in Information and Communication Technologies (ICTs). These technologies are usually designed for people who understand the spoken language, whereas for the deaf, the oral language is considered something like a "second language". On the other hand, companies responsible for making such information available feel great difficulty in making accessible the content provided by them, especially since these contents are generally dynamic, and the production of windows in sign languages (eg, Brazilian Sign Language - Libras) has usually a high operating cost. To minimize these problems, an alternative is the use of automatic translators to sign languages. With these automatic translators, it can generate digital content dynamically, with low cost, and according to some studies, with a reasonable standard of intelligibility. These solutions, however, generally require a database (dictionary LBS) containing thousands of videos or animations represented by an animated avatar. The process of building these databases, however, is a arduous process, and depends on a large and specialized team of modelers, animators and 3D designers, as well as experts on sign language and deaf users. To minimize these problems, the purpose of this work is to develop a solution for semi-automatic generation of these animations (movies). The idea is that these animations can be created by users from the parameter setting (phonemes) of the Brazilian Sign Language (Libras), reducing the time and cost required to produce this kind of database.

Keywords: pounds, animation, pounds dictionary, avatar

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

DA - Deficiência Auditiva.

CSS - *Cascading Style Sheets.*

JSON - *JavaScript Object Notation*.

HTML - *HyperText Markup Language.*

IBGE - Instituto Brasileiro de Geografia e Estatística.

IBOPE - Instituto Brasileiro de Opinião Pública e Estatística.

LIBRAS - Língua Brasileira de Sinais.

SECOM - Secretaria de Comunicação Social da Presidência da República.

USP - Universidade de São Paulo.

XML - *eXtensible Markup Language*

WEB - *World Wide Web*.

SUMÁRIO

1 INTRODUÇÃO

Atualmente, muitas pessoas acessam informações através de meios de comunicação como a televisão, jornal ou a internet. Segundo pesquisa realizada em 2014 pela Secretaria de Comunicação Social (SECOM) em parceria com o Instituto Brasileiro de Opinião Pública e Estatística (IBOPE), a televisão e a internet estão entre os principais meios de comunicação utilizados pela população em geral.

Entretanto, muito dos conteúdos existentes nesses canais de comunicação não são acessíveis a todos, principalmente para pessoas com deficiência sensorial (visual ou auditiva). Uma das razões é que as tecnologias envolvidas nesses meios de comunicação, quando são projetadas, raramente levam em conta os requisitos especiais dessas pessoas (VIEIRA, 2010).

Conforme Censo realizado pelo IBGE (2010), existem cerca de 9,7 milhões de pessoas com algum tipo de Deficiência Auditiva (DA) no Brasil, o que representava cerca de 5% da população. Esse número vem crescendo durante os anos devido ao aumento da população, em especial na população idosa, cujo número de deficiência nesse aspecto é maior.

Para tentar incluir esse público nessas tecnologias e meios de comunicação alguns recursos de acessibilidade foram criados e uma legislação específica foi desenvolvida para forçar que os responsáveis pela distribuição desse tipo de tecnologia ofereçam esses recursos adicionais. Para pessoas surdas, os principais recursos de acessibilidade definidos são os seguintes:

- *Closed caption* (legenda oculta);
- Janela de LIBRAS (linguagem de sinal).

O *closed caption* consiste em uma descrição simultânea do que está se falando em uma transmissão, o que não é muito eficiente, segundo Ferreira Brito (1995), pois o surdo não tem a língua escrita (por exemplo, a língua portuguesa) como primeira língua. Como consequência, muitos deles têm dificuldade em aprender e se comunicar usando textos.

Outro recurso de acessibilidade importante são as janelas em línguas de sinais (por exemplo, janelas de LIBRAS), que consistem na sobreposição de uma janela de tradução sobre o vídeo, geralmente posicionada no canto da tela. Embora seja a primeira opção para as pessoas surdas, esse recurso, no entanto, é bastante dispendioso, necessitando um intérprete de LIBRAS disponível durante todos os dias da semana, 24 horas por dia, além de possuir altos custos operacionais de produção (câmera, estúdio, equipamentos de produção, edição, equipe técnica, dentre outros).

Para reduzir esses problemas, uma alternativa é a utilização de tradutores automáticos para línguas de sinais. Esse tipo de solução geralmente faz tradução de textos e fluxos de áudio para língua de sinais utilizando janelas com intérpretes virtuais (avatares) para representar a informação na língua de sinais. Com isso, é possível gerar conteúdos digitais para Libras de forma dinâmica, com baixo custo, e segundo alguns trabalhos, com um nível de inteligibilidade razoável. Existem soluções que oferecem ferramentas que possibilitam ao usuário traduzir textos em português para língua de sinais, bem como também a possibilidade de criar novos sinais. Dentre elas as principais são: Poli-Libras uma ferramenta multiplataforma que oferece serviços de tradução de conteúdos da língua natural para LIBRAS e um módulo para adicionar novos sinais a uma base de dados, ProDeaf outra ferramenta multiplataforma que semelhante ao Poli-Libras disponibiliza ferramentas para tradução de Português para LIBRAS e também o serviço de criação de novos sinais, existe ainda o F-LIBRAS que também permite a criação de novos sinais por meio de uma interface. As três soluções possuem abordagens semelhantes e serão melhor detalhadas no capítulo 3.

Essas soluções, no entanto, geralmente necessitam de uma base de dados (dicionário de LIBRAS) contendo milhares de vídeos ou animações representadas por um avatar animado. Os principais dicionários de Libras que existem atualmente no mercado, por exemplo, contém entre 6000 (seis mil) e 10.000 (dez mil) sinais (CAPOVILLA, 2001; INSTITUTO..., 2008).

O processo de construção dessas bases, no entanto, é um processo trabalhoso, e depende de uma equipe grande e especializada de modeladores,

animadores e designers 3D, além de especialistas em Libras e usuários surdos, o que acaba dificultando que esse tipo de tecnologia seja viabilizado e implantado.

Para reduzir esse problema, a proposta deste trabalho é desenvolver uma solução para geração semiautomática dessas animações (vídeos). A ideia é que essas animações possam ser criadas por usuários a partir da configuração de parâmetros (fonemas) da Língua Brasileira de Sinais (LIBRAS) que segundo Quadros e Karnoop (2004) existem 5 (cinco) parâmetros: configuração de mão (posição que a mão assume durante a execução de um sinal), movimento (uma vasta rede de formas e direções que o pulso, mãos e braços assumem e variam durante toda a execução do sinal), locação (espaço que a mão ocupa quando o sinal é realizado), orientação manual (direção que a palma da mão aponta quando o sinal é executado) e expressões não-manuais (movimentos faciais e corporais). Utilizando esses parâmetros pode-se criar qualquer tipo de sinal em LIBRAS. O propósito é reduzir o tempo e o custo necessário para produzir uma base de dados. Além disso, a proposta da solução é permitir que pessoas que possuam algum tipo de conhecimento em LIBRAS possam construir esses sinais de forma colaborativa.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver uma solução que permita a geração semiautomática de sinais da Língua Brasileira por pessoas que entendam de LIBRAS e que não possuam conhecimentos em animação, de forma intuitiva e rápida, a partir da configuração de alguns parâmetros (fonemas) que compõem os sinais em Libras.

1.2 OBJETIVOS ESPECIFICOS

Para atingir o objetivo geral deste trabalho serão necessários os seguintes objetivos específicos:

- Fazer um levantamento dos principais requisitos para criação de animações de sinais em Libras.

- Identificar a definição e quantidade de parâmetros da Libras mais relevantes de acordo com os principais autores, e que serão utilizados para construir os sinais de forma semiautomática;
- Desenvolver uma ferramenta que possibilite aplicações a gerarem sinais de forma rápida.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte forma:

O capítulo I apresenta uma breve introdução ao tema, o problema e a motivação para o desenvolvimento do trabalho, bem como, os objetivos que se pretende atingir ao final deste trabalho.

No capítulo II são apresentados de forma detalhada o conceito de Libras, quais os parâmetros necessários para a definição e construção de um sinal e, as principais ferramentas utilizadas em animações 3D.

No capítulo III são apresentados trabalhos com soluções semelhantes ao desde trabalho e quais as vantagens e desvantagens em relação à solução deste estudo.

O capítulo IV apresenta como está estruturada e quais foram às tecnologias utilizadas na solução proposta por este trabalho, além de explicar como os parâmetros (fonemas) escolhidos pelo usuário devem ser enviados, bem como, o modelo avatar utilizado neste trabalho.

O capítulo V mostra os resultados de testes realizados em cima da solução apresentada neste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os principais conceitos que fundamentam este trabalho. Inicialmente, os conceitos e características da Língua Brasileira de Sinais (LIBRAS) serão explanados. Posteriormente, as principais ferramentas utilizadas para modelagem e animações 3D. A ferramenta Blender será especificada, pois foi a utilizada neste trabalho.

2.1 LÍNGUA DE SINAIS - LIBRAS

A Língua Brasileira de Sinais é uma língua natural de modalidade espaço-visual, a qual utiliza gestos que são percebidos visualmente (Revista da FENEIS, número 2:16). A comunidade surda se comunica por meio de gestos das mãos, da face e do corpo, permitindo expressar conceitos, sejam eles descritivo, emotivo, racional, literal, metafórico e outros decorrentes da necessidade da comunicação com outra pessoa. Observa-se também que essa língua possui gramática bem estruturada, fonologia, morfologia, sintaxe e semântica.

Na língua de sinais, as sentenças são articuladas de forma diferente das línguas orais, que utilizam fonemas. Na gestual, as sentenças são compostas por sinais que, por sua vez, são formados por um conjunto de parâmetros (queremas ou parâmetros de configuração). Esses parâmetros serão apresentados na seção 2.1.1.

2.1.1 Sinal

Em LIBRAS, um sinal é formado pela combinação do movimento, formato e localização das mãos, localizadas em contato com o corpo ou em frente a ele. Estas articulações das mãos, que podem ser comparadas aos fonemas e às vezes aos morfemas, são denominadas de parâmetros (Revista da FENEIS, número 2:16).

Quadros e Karnoop (2004) propõem cinco parâmetros que compõem um sinal em Libras (ver Figura 1), os quais são definidos como:

- Configuração de mão;
- Movimento;
- Locação ou ponto de articulação;
- Orientação manual e
- Expressões não-manuais ou facial.

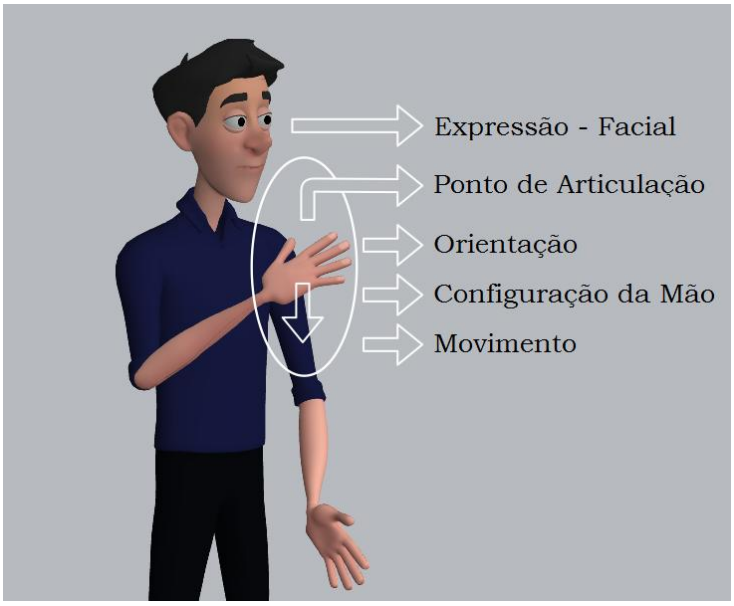


Figura 1 - Parâmetros constituintes da LIBRAS.
Fonte: Elaborada pelo autor.

A **configuração da mão** é a forma que a mão predominante (mão esquerda para os canhotos e direita para os destros) do sinalizador ou emissor assume durante a execução do sinal. Entretanto, pode-se utilizar as duas mãos na execução do sinal. Essas configurações variam em diferentes formas, seja variando o número de dedos estendidos, se a mão está aberta ou fechada ou pela abertura e contato dos dedos.

O *Sign Writing*[1] define que existem 74 (setenta e quatro) configurações da mão para as línguas de sinais. Entretanto, Felipe e Monteiro (2005) afirmam que a LIBRAS utiliza apenas 60 (sessenta) dessas configurações da mão. Há deste modo, grande variação na quantidade deste parâmetro de configuração. Neste trabalho,

[1] Um sistema de escrita para escrever línguas de sinais.

adotaremos a definição de Felipe e Monteiro (2005), que afirmam existirem 60 configurações da mão, conforme ilustrado na Figura 2.

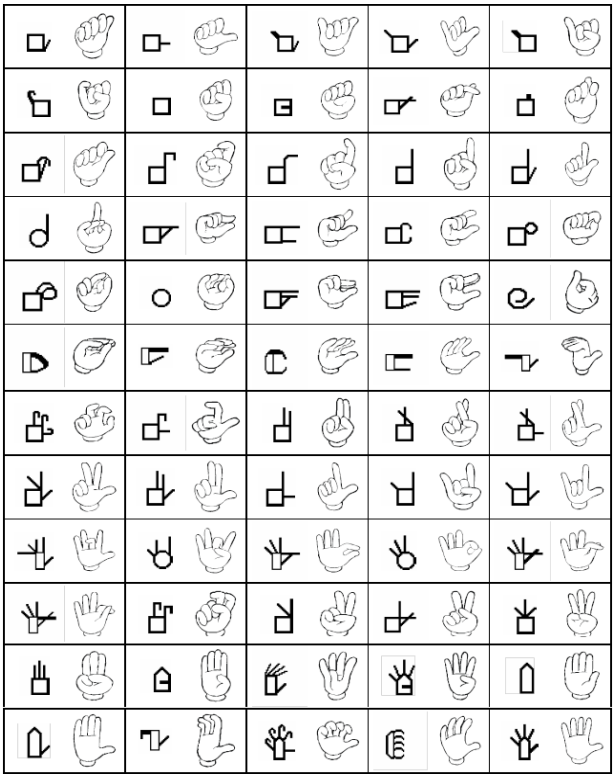
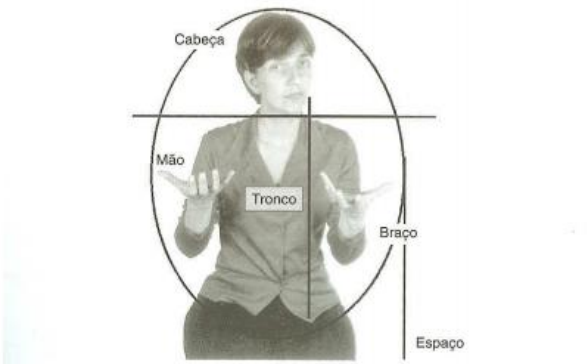


Figura 2 - Configurações de mão da LIBRAS.

O **ponto de articulação** é o espaço ou a localização que a mão predominante do emissor ocupa quando o sinal é realizado. A mão pode estar localizada em uma parte do corpo ou próximo a ele, em um espaço neutro vertical (meio do corpo até a cabeça) ou horizontal (à frente do emissor), conforme visualizamos na Figura 3.

De acordo com a Figura 3, o sinal poderá ser realizado à frente do corpo e poderá também tocar a pele do sinalizador, efetuando o contato com a superfície corporal.

Na Figura 4, dois sinais são apresentados. A diferença entre eles consiste apenas no ponto de articulação, pois a palavra APRENDER é sinalizada em contato com a testa e a palavra SÁBADO à frente da boca.



Espaço de realização dos sinais e as quatro áreas principais de articulação dos sinais (baseado em Battison, 1978, p. 49)

Figura 3 - Espaço de realização dos Sinais.
(QUADROS; KARNOPP, 2004).

Significado do sinal: aprender, aprendizagem.

Significado do sinal: sábado

Figura 4 - Sinais localizados em pontos de articulação diferente.
Fonte: http://150.164.100.248/dialogosdeinclusao/data1/arquivos/Parametros_da_Libras.pdf

A **orientação da mão**, de acordo com Quadros e Karnopp (2004) indica a direção que a palma da mão aponta quando o sinal é executado. Os autores indicam a existência de seis tipos de orientação da mão: palma da mão para cima, para baixo, para o corpo, para frente, para o lado esquerdo ou lado direito. Em um sinal a mão pode mudar muitas vezes sua orientação.

O **parâmetro movimento,** segundo Ferreira Brito (1995), “é um parâmetro complexo que pode envolver uma vasta rede de formas e direções, desde os movimentos internos da mão, os movimentos do pulso, os movimentos direcionais no espaço até conjuntos de movimentos no mesmo sinal”.

Por fim, o **parâmetro expressões não-manuais ou facial** referem-se aos movimentos faciais e corporais, movimentos do corpo, movimentos da face, dos olhos e do tronco. Muitos sinais podem requerer características adicionais para expressar alegria, tristeza, uma pergunta ou exclamação. Portanto, expressões não-manuais podem assumir tanto uma função léxica (expressar alegria e tristeza, por exemplo) quanto uma função sintática (expressar uma pergunta ou uma exclamação, por exemplo) na estrutura de um sinal.

2.2 FERRAMENTAS DE ANIMAÇÃO

Existem muitas ferramentas específicas para modelagem, animação e criação de efeitos em 3D, como, por exemplo, o 3Ds Max, Maya, Houdini, Blender, etc. Algumas dessas ferramentas serão detalhadas na Tabela 1.

Tabela 1 - Principais ferramentas de animação utilizadas.

FERRAMENTA	DESCRIÇÃO
3Ds Max	Ferramenta utilizada para criação de cenários e animações 3D, vasta e completa em controles para o usuário familiarizado em animações.
Maya	É um software que oferece um conjunto de recursos criativo abrangente para a animação 3D computador, modelagem, simulação, renderização e composição em uma plataforma de produção altamente extensível. Maya tem tecnologia de próxima geração de exibição, os fluxos de trabalho de modelagem acelerados, e ferramentas para manipulação de dados complexos.
Udini	É uma ferramenta que oferece recursos avançados para a criação de efeitos de partículas usadas para criar poeira, detritos e explosões. Também poderosos recursos de animação, renderização e bibliotecas de desenvolvimento para extensibilidade do usuário.

Nesse trabalho, a ferramenta Blender será utilizada como apoio para construção das animações de forma semiautomática. O motivo da escolha do Blender é que além de ser livre, ela possui uma API aberta, que permite que programadores e animadores possam configurar alguns aspectos da animação de forma automatizada. Mais informações sobre o Blender serão apresentadas na seção a seguir.

2.2.1 Blender

O Blender é um software de código aberto escrito em C, C++ e Python. Ele está disponível para as plataformas *Windows*, Linux e Mac OS, foi desenvolvido pela Blender Foudation e é conduzido sob a licença *GNU General Public License* (*GPL*), o que permite aos usuários realizar mudanças na base do código do software, criação de novos recursos, correção de *bugs* e melhoria na usabilidade da ferramenta.

Segundo Clua e Bittencourt (2005), a ferramenta Blender é uma ferramenta bastante completa, com muitas funcionalidades tanto para modelagem de objetos quanto para animação, pós-produção, renderização e criação 3D. Oferece suporte a diferentes tipos de formatos para importação e exportação como 3DS, Cal3D, MDL, OBJ, VRML, DirectX e outros.

Ela é composta por um conjunto complexo de recursos (Andrade, 2008, p. 18). São eles:

- Dinâmica de corpos rígidos e macios;
- Dinâmica de fluídos e partículas;
- Modelagem por subdivisão e por curvas;
- Módulos para animação, *inverse kinematics*, esqueletos e animação não-linear;
- *Game-engine*;
- Módulo de edição de vídeo e áudio;

- *Shaders*, NURBS, escultura digital (como o *Z-Brush*);
- Radiosidade, *Ray Tracing*;
- Compatível com renderizadores externos (*YafRay, Aqsis, Indigo, PovRay,* etc.);
- Programação de interatividade (jogos e passeios virtuais 3D);
- Importação e exportação de modelos compatíveis com outros programas (3DS, VRML, X, OBJ, LWO, etc.);

Ainda é possível desenvolver *plug-ins* novos para esta ferramenta, estendendo as funcionalidades básicas disponíveis ou ainda melhorar as existentes por meio de scripts em Python (CLUA; BITTENCOURT, 2005).

Além disso, o Blender possui uma interface bastante flexível, podendo ser totalmente personalizada de acordo com as preferências do usuário. Porém, este software é pouco intuitivo e necessita de elevado grau de atenção para o aprendizado (CLUA; BITTENCOURT, 2005). Na Figura 5, a interface do Blender é apresentada com um *layout* personalizado.

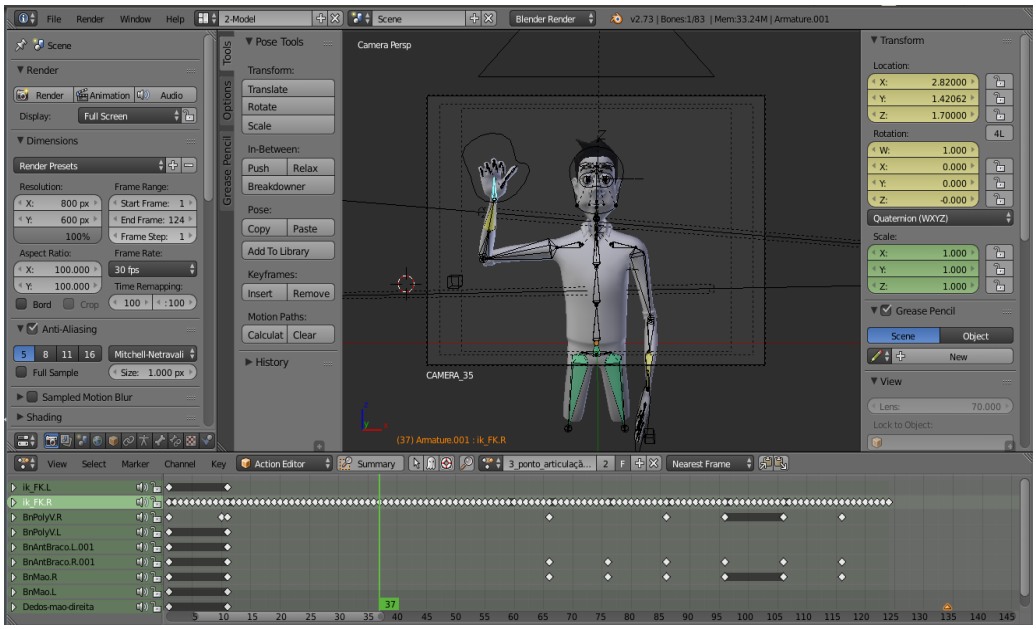


Figura 5 - Interface do Blender.
Fonte: Elaborada pelo autor.

O Blender é uma ferramenta com poderosos recursos, os quais permitem ao usuário, por exemplo, rapidamente modelar usando um conjunto grande de ferramentas disponíveis para criação, transformação e edição de modelos. Permite também renderizações mais realistas com seu motor de renderização, conforme ilustra a Figura 6.

O Blender oferece um amplo conjunto de ferramentas para animação, dando controle ao usuário sobre personagens, transformando-os em animações complexas, seja com um simples *keyframing* ou ciclos mais complexos.

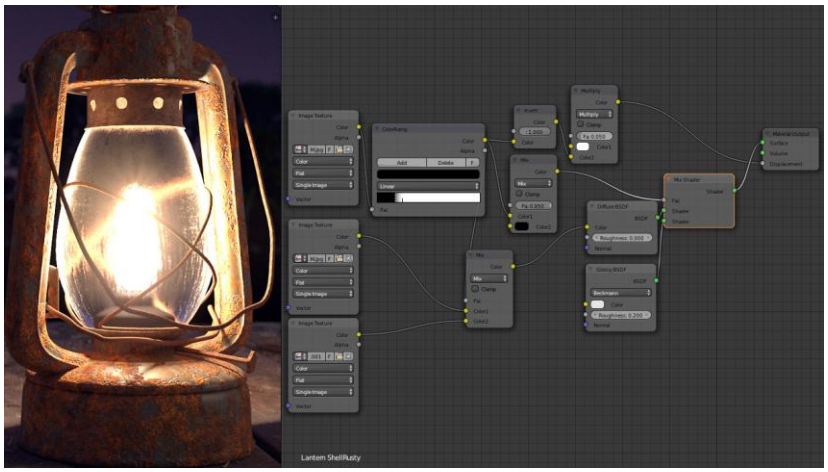


Figura 6 - Material renderizado no Blender.
Fonte: Blender.org.

Nesse capítulo foram apresentados detalhadamente conceitos sobre LIBRAS e os parâmetros que são necessários para construção de um sinal. Em seguida, apresentadas as principais ferramentas usadas em criação e edição de animações 2D e 3D e uma breve descrição de cada uma delas. A ferramenta Blender foi explicada mais detalhadamente, pois é a ferramenta utilizada na solução deste trabalho.

No próximo capítulo serão apresentados trabalhos semelhantes, principais características, vantagens e desvantagens em comparação ao apresentado neste trabalho. Será mostrado um quadro comparativo com as principais características de cada ferramenta, o que cada uma tem em relação a outra e o que faz delas serem desvantajosas ou vantajosas em relação ao da solução proposta por este trabalho.

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados trabalhos que abordam temas relacionados a LIBRAS, bem como a geração de sinais animados. As vantagens e desvantagens de cada trabalho também serão elencadas.

3.1 Poli-Libras

Poli-Libras é uma ferramenta de tradução automática de frases em português para uma sequência de sinais em LIBRAS, a qual gera como saída uma representação gráfica com base em um Avatar 3D (CARVALHO; ALEXANDRE; LI, 2010).

Este sistema foi desenvolvido por alunos do curso de Engenharia de Computação da USP (CARVALHO; ALEXANDRE; LI, 2010) e consiste em um conjunto de projetos *open-source*, sendo um desses o módulo chamado WikiLibras que permite aos usuários de forma colaborativa gerar sinais animados, adicionando novos verbetes da LIBRAS (ver Figura 7). Este sistema também tem como objetivo ampliar seu dicionário Português-Libras e contribuir com a expansão da língua.

Conforme a Figura 7Figura 7, o módulo do WikiLibras não é muito intuitivo, caracterizando uma desvantagem, pois uma pessoa com poucos conhecimentos sobre quais parâmetros são utilizados na construção de um sinal em LIBRAS terá dificuldades em criar novos sinais ou editar algum já existente, podendo vir a deixar de contribuir de forma efetiva e eficiente com a evolução do dicionário Português-Libras.

A quantidade de textos no software também acaba se tornando um problema para muitos surdos. Segundo Ferreira Brito (1995), a língua materna dos surdos é a Língua de Sinais. O português, deste modo, é considerado uma segunda língua para os surdos. Sabe-se também que mesmo após muitos anos de estudos, a compreensão do português é pequena (STUMPF, 2000).

Outra desvantagem da ferramenta é a necessidade de conhecimentos matemáticos (Geometria Espacial) para o preenchimento dos campos, conforme ilustrado na Figura 7.

Uma vantagem deste módulo é a possibilidade de criar novos sinais ou acessar sinais existentes no dicionário por meio de um *WebService*, possibilitando a qualquer pessoa que deseje ajudar a comunidade surda desenvolvendo aplicações que utilizam desse serviço.

Figura 7 - Poli-Libras módulo do WikiLibras.
Fonte: https://dl.dropboxusercontent.com/u/18183156/polilibras/wikilibras-small.png

3.4 F-LIBRAS

O F-LIBRAS é uma ferramenta que agrupa em uma interface diversas funcionalidades dentre elas a gravação, editoração e visualização de movimentos e sinais da Libras (BAPTISTA, 2007). Uma de suas principais funções, além da tradução de textos para LIBRAS é a possibilidade de criação e inclusão de sinais em um dicionário. O módulo Editor desta ferramenta possui uma abordagem semelhante ao deste trabalho. Um avatar é visualmente configurado (ver Figura 8) em um ambiente virtual 3D.

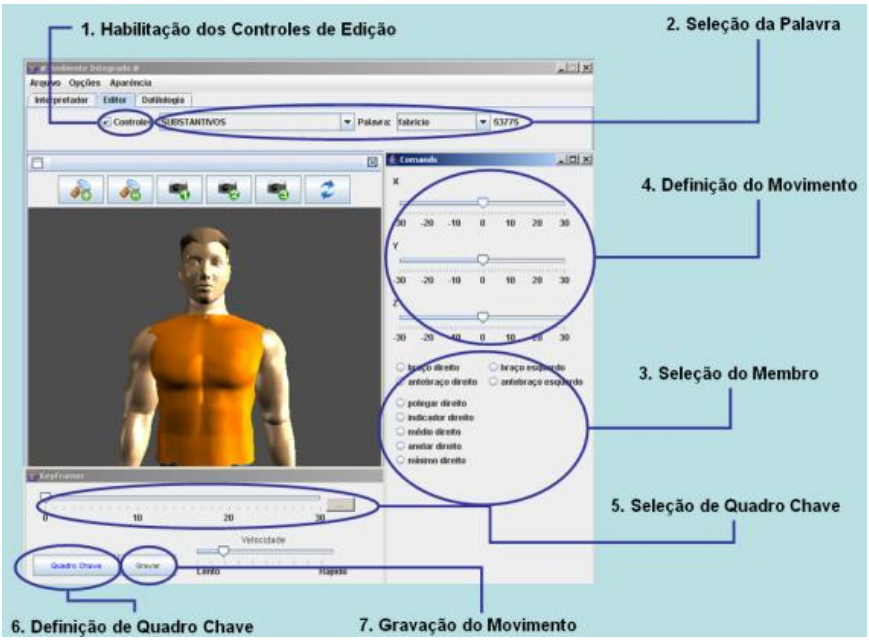


Figura 8 – Interface do módulo Editor.
(Baptista, 2007)

Ao analisar a Figura 8 percebe-se que para um usuário criar um sinal precisa-se configurar vários campos como: Configurar o ângulo de rotação de cada eixo X, Y, Z, além de ter que selecionar qual membro ele está configurando no momento. O usuário deve selecionar qual quadro (frame) que será refletida por essas configurações. Isso pode ser uma desvantagem no ponto de vista de que os usuários precisam ter conhecimentos em geometria, podendo assim levar muito tempo para configurar o sinal. Levando ao usuário uma sensação de incapacidade.

Uma vantagem desta ferramenta é a possibilidade de visualizar a execução de uma configuração em tempo real uma vez que o sinal já esteja configurado.

3.5 ProDeaf

Proativa (2013) explica o ProDeaf como “um conjunto de soluções de software capazes de traduzir texto e voz de português para LIBRAS com o objetivo de permitir a comunicação entre surdos e ouvintes”. O ProDeaf pode ser considerada um suíte de softwares onde oferece diferentes soluções para as plataformas *Web* e *mobile.* Na *Web* é disponibilizado um tradutor que semelhante ao

Poli-Libras faz a tradução Português -> Libras de textos e áudios. Além de um dicionário Português – Libras outra ferramenta disponibilizada é a de criação de sinais onde o usuário pode criar seus próprios sinais e compartilha-los (ver Figura 9).

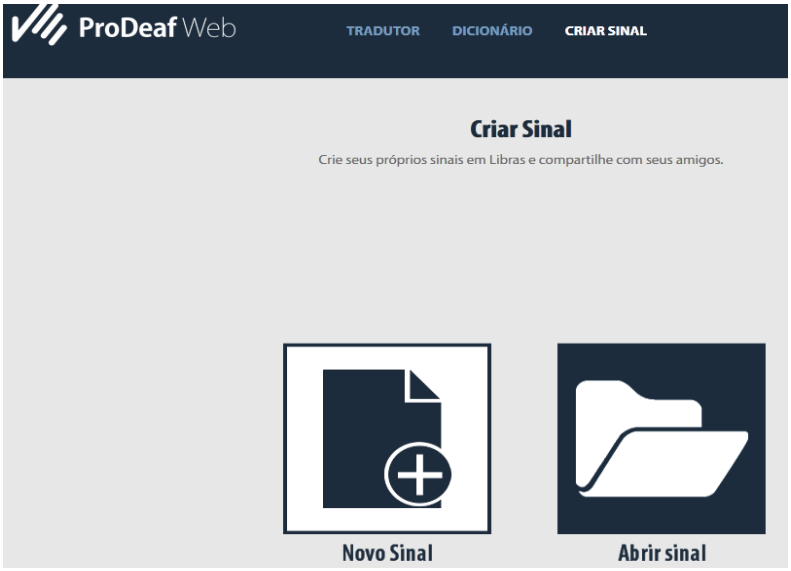
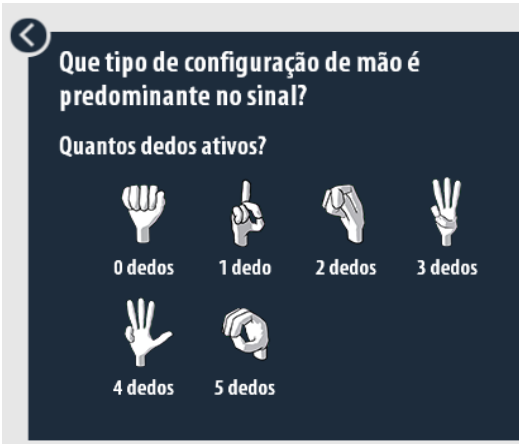


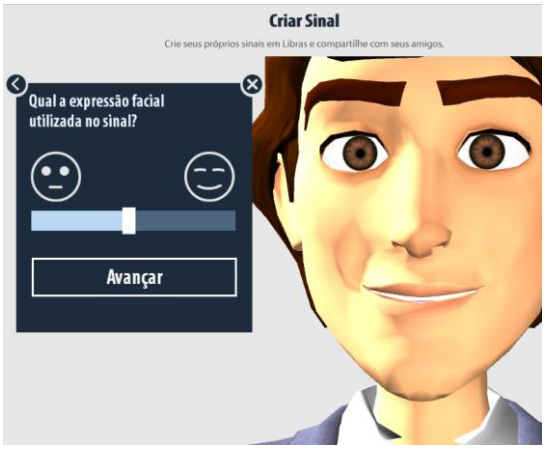
Figura 9 - Captura da tela de ferramenta de criação de sinal da ProDeaf.
Fonte: http://web.prodeaf.net/CriarSinal

A Figura 10a mostra a seção de configuração da mão. A Figura 10b, 10c e 10d mostra as opções oferecidas pela ferramenta para configurar a expressão facial, corporal, ponto de articulação e movimento.

(a)



(b)



(c) (d)

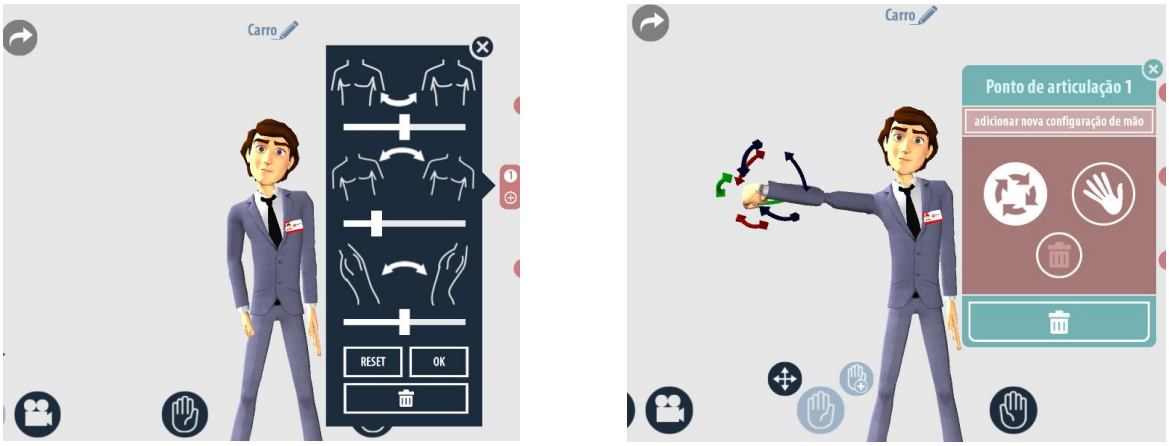


Figura 10 - (a) Captura da tela de configuração da mão predominante, (b) facial, (c) corporal, (d) movimento e ponto de articulação.
Fonte: http://web.prodeaf.net/CriarSinal

A ferramenta é bastante intuitiva, porém exige do usuário um mínimo de prática para utilizar todos os recursos que esse módulo oferece. Uma limitação desta ferramenta é que ela executa apenas no portal do ProDeaf, diferente da ideia deste trabalho que permite a outras soluções a possibilidade de desenvolver softwares voltados a utilização da API.

A vantagem desta solução é a possibilidade de o usuário visualizar em tempo real se as configurações escolhidas por ele estão realmente de acordo, e posteriormente visualizar a execução do sinal, caso as configurações não estejam de acordo, a possibilidade de mudar qualquer uma das configurações e posteriormente executar novamente o sinal.

Neste capítulo foram apresentadas as soluções que geram sinais em Libras e quais são suas principais características, bem como, quais são as vantagens e desvantagens em relação a solução deste trabalho.

No próximo capitulo será apresentado a solução deste trabalho e suas características.

4 SOLUÇÃO PROPOSTA

Existem inúmeras ferramentas de tradução automática de textos/áudios de Português para LIBRAS, e para realizar tal tarefa geralmente essas ferramentas dependem de uma base de dados (dicionário) de sinais animados. Porém muitas delas ainda não oferecem a possibilidade de criação de sinais por qualquer pessoa, o que auxiliaria no crescimento e na alimentação dessa base de dados. O que se ver ainda é a total dependência de uma equipe especializada de modeladores, animadores e designers 3D, além de especialistas em Libras e usuários surdos para realizar a expansão dessa base de dados (dicionário).

Os dicionários de LIBRAS são utilizados para evitar a renderização de sinais em tempo real, no qual possui um alto nível de processamento para isso. Uma boa característica desta solução é oferecer um serviço por meio de uma API que facilite a interação de pessoas com algum conhecimento em LIBRAS a gerarem sinais. Através de uma interface que ilustre os parâmetros de LIBRAS, os usuários poderão escolher quais parâmetros iram compor o sinal.

Outro ponto importante da solução é que qualquer usuário poderá alimentar o dicionário de sinais, gerando-os de forma mais rápida do que se estivesse usando alguma ferramenta de animação, pois a partir da escolha de alguns parâmetros e acesso à API do sistema, o usuário consegue criar um novo sinal. Além disso, o dicionário fica disponível para que os usuários tenham acesso aos sinais gerados.

Na Figura 11 é apresentado uma visão arquitetural do sistema baseado em uma arquitetura cliente-servidor. A ideia é que uma aplicação com uma interface apresente as posturas salvas no avatar e possibilite a escolha dos parâmetros pelo usuário, e logo após a escolha do usuário a requisição seja enviada a API, passando os parâmetros escolhidos na aplicação, possibilitando no termino da renderização a visualização do sinal.

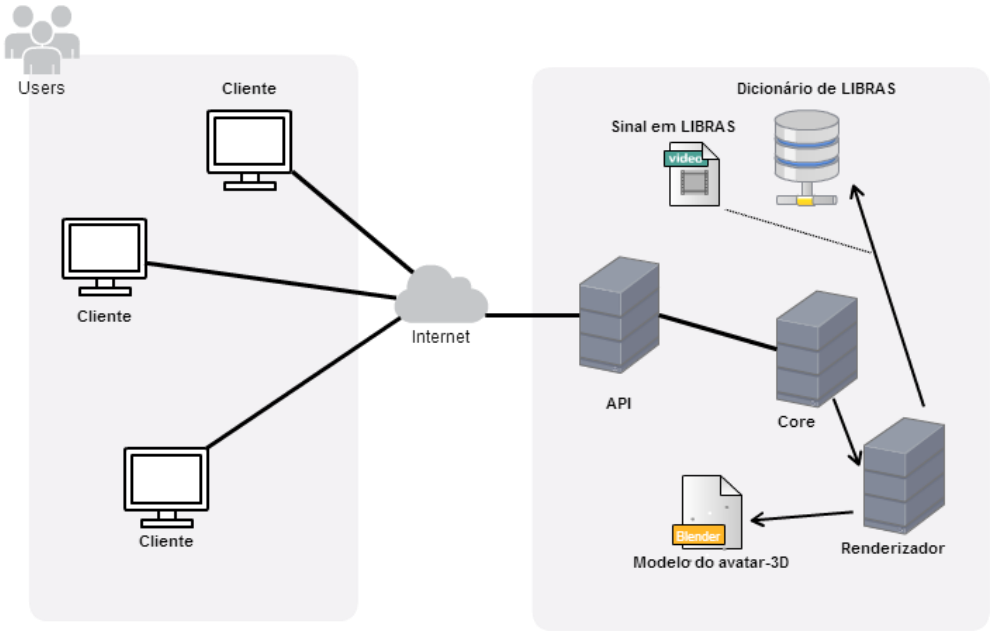


Figura 11 - Visão arquitetural da solução.
Fonte: Elaborada pelo autor.

A API será melhor detalhada na seção 4.1 onde será explicado qual a tecnologia usada para desenvolver a API, em seguida será apresentado uma descrição detalhada dos parâmetros passados no corpo do JSON que é a forma que a API se comunica, e em seguida como está estruturado o modelo avatar 3D usado na renderização de acordo com os parâmetros passados no JSON.

4.1 API

A API é o componente responsável por receber todas as requisições enviadas pelos usuários por meio de uma aplicação. A tecnologia utilizada na construção deste componente foi o Node.js explicado na seção 4.1.1. Na API foi utilizado o módulo EXPRESS para utilizar os recursos que uma aplicação web oferece, este módulo é um framework Node.js de aplicações *web*, o qual provê um vasto e robusto conjunto de recursos para o desenvolvimento web. Esse módulo possui vários

métodos HTTPs (Hypertext Transfer Protocol) e *middlewares,* o que faz com que a API seja bastante robusta.

Conforme ilustrado na Figura 12, a API possui um método (*Endpoint*) do tipo POST do protocolo HTTP, ele é responsável por receber a requisição feita sobre o caminho “/sign”. O JSON enviado no corpo da mensagem é processado e repassado para o script Python. Um sinal é gerado e armazenado no dicionário de sinais no final de todo processo.

```
app.post('/sign', function(req, res) {
    options.args = JSON.stringify(req.body);
    PythonShell.run('controller.py', options, function (err, results) {
        if (err) { console.log(err); endpoint_sinal.init(req, res, Sign,"Falhou"); res.send(400); return; }
        // results is an array consisting of messages collected during execution
        endpoint_sinal.init(req, res, Sign,"Sucesso");
        res.send(200);
    });
});
```

Figura 12 - Método responsável por receber as requisições envidas pelo usuário.
Fonte: Elaborada pelo autor.

A estrutura e a descrição dos campos enviados na requisição serão explicadas no tópico 4.2.

4.1.1 Node.js

Node.js é uma plataforma cujo objetivo é a fácil construção de rápidas e escaláveis aplicações de rede, que emprega um modelo baseado em eventos e *non-blocking* I/O[2].

O Node.js é composto por vários módulos. Alguns fazem parte do núcleo da plataforma e outros são desenvolvidos pela comunidade de desenvolvedores (JUNIOR, 2012).

O Node.js foi escolhido por ser uma plataforma que facilita o desenvolvimento de aplicações escaláveis e não bloqueantes importantes características quando se visa o desenvolvimento de aplicações como a proposta por esse trabalho, visando ser colaborativa. Possivelmente com uma alta taxa de requisições.

[2] Informações contidas no site https://nodejs.org/en/

Por ser uma tecnologia JavaScript *server-side* facilita na transição dos dados no formato JSON utilizados nesta solução. O formato JSON é sintaticamente idêntico ao código para a criação de objetos no JavaScript .

Devido a isso, um JSON pode ser facilmente convertido em um objeto JavaScript. Para melhor explicar será detalhado no próximo tópico o formato JSON e os parâmentros enviados para a API.

4.2 JSON

JSON (JavaScript Object Notation) é um formato de intercâmbio de dados leve e é baseado em um subconjunto da linguagem de programação JavaScript. JSON é totalmente independente de linguagem específica.

Uma das vantagens em se utilizar JSON é que é muito fácil para uma pessoa ler e escrever no formato JSON, assim como é muito fácil para uma máquina processar os dados nesse formato.

Segundo o site oficial do JSON[3] ela é construída em duas estruturas, que são:

- Uma coleção de pares com nome/valor.
- Uma lista ordenada de valores.

A Figura 13 ilustra as duas formas que o JSON pode ser construído e os tipos que cada valor pode assumir, que são string, number, object ou array. Por ser bastante simples, o JSON é uma linguagem de intercâmbio ideal e é utilizada neste trabalho para intercambiar os dados.

[3] Disponível em http://json.org.

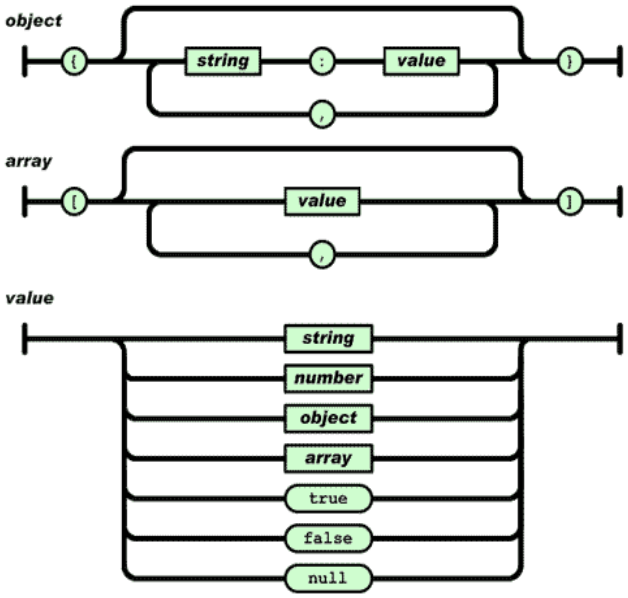


Figura 13 - Estruturas do JSON.
Fonte: http://www.json.org.

Os dados recebidos pela API da aplicação devem seguir alguns padrões de nomeação, bem como, a estrutura do objeto JSON. Esses padrões e os campos necessários serão explicados nos tópicos a seguir.

4.2.1 Estrutura do objeto JSON

A Figura 14 mostra como deve ficar o cabeçalho do JSON que será enviado para API.

```
{
    "userId": "nome_do_usuario",
    "sinal": "nome_do_sinal",
    "interpolacao": "normal",
    "movimentos": [
        (...)
    ]
}
```

Figura 14 - Cabeçalho do objeto JSON.
Fonte: Elaborada pelo autor.

“userId”: Tipo String. Identifica a pasta pessoal do colaborador. Esta identificação deve ser única, pode ser alfanumérica e não é case sensitivo, ou seja, sempre será convertido para caracteres minúsculos.

“sinal”: Tipo String. Especifica o nome do vídeo gerado (não é necessário especificar a extensão). Não é case sensitivo.

“interpolacao”: Tipo String. Assume valores entre **lento**, **normal** ou **rapido**. Serve para controlar o tempo na composição de movimentos.

“movimentos”: Tipo Vetor de Objetos.

4.2.2 Vetor de Objetos (Movimentos)

O objeto **"movimentos"**, é um vetor de objetos, onde cada sub-objeto contido nele, contém dados referentes a expressão facial, e para cada uma das mãos os parâmetros LIBRAS (Ponto de Articulação, Configuração de Mão, Orientação da palma da Mão) tanto para a mão direita, quanto para a esquerda e o movimento respectivo para cada mão (Circular, Contato, Helicoidal, Pontual, Retilíneo, Semi-Circular, Senoidal).

A quantidade de objetos que se pode ter nesse vetor é dinâmica, ele pode ter itens vazios que são representados apenas com "{}". A Figura 15 apresenta 3 objetos.

```
{
    (...)
    "movimentos": [
        {
            "facial": {
                (...)
            },
            "mao_esquerda": {
                (...)
            },
            "mao_direita": {
                (...)
            }
        }
        {},
        {
            "facial": {},
            "mao_esquerda": {},
            "mao_direita": {}
        }
    ]
    (...)
}
```

Figura 15 - Vetor de Objetos (movimentos).
Fonte: Elaborada pelo autor.

4.2.3 Objeto Facial

A Figura 16 mostra quais parâmetros são necessários para configurar a expressão facial.

```
{
    (...)
    "movimentos": [
        {
            "facial": {
                "expressao": 21,
                "transicao": "normal",
                "duracao": "normal"
            }
            "mao_esquerda": {
                (...)
            }
            "mao_direita": {
                (...)
            }
        }
    ]
    (...)
}
```

Figura 16 – Exemplo de objeto para expressão facial.
Fonte: Elaborada pelo autor.

“facial”: Tipo Objeto. Contém o índice da expressão facial, a velocidade de transição e de execução.

“expressao”: Tipo Inteiro. Os valores devem ser entre **0** até **21**. Especifica o índice da expressão facial na biblioteca de poses do Blender. O valor padrão é **0**. Estes índices podem ser alterados de acordo com a inclusão de novas expressões, caso o índice passado esteja fora do intervalo, o avatar permanecerá na expressão padrão.

“transicao”: Tipo String. Os valores entre “**lento”**, “**normal”** ou “**rapido”**. Especifica a velocidade de transição entre o repouso ao início da expressão facial escolhida.

“duracao”: Tipo String. Os valores entre “**lento”**, “**normal”** ou “**rapido”**. Especifica o tempo de permanência na expressão facial escolhida.

4.2.4 Objeto Mãos

Como mostra a Figura 17, cada mão atua de forma independente da outra, onde elas simultaneamente podem realizar movimentos diferentes uma da outra durante a execução do sinal. Uma vez que, o objeto esteja vazio a mão que estiver vazia permanecerá na posição padrão.

```
{
    (...)
    "movimentos": [
        {
            "facial": {
                (...)
            },
            "mao_esquerda": {
                "circular": {
                    (...)
                }
            },
            "mao_direita": {
                "pontual": {
                    (...)
                }
            }
        }
    ]
    (...)
}
```

Figura 17 - Exemplo de objeto para configuração das mãos.
Fonte: Elaborada pelo autor.

O objeto **“mao_esquerda”** ou **“mao_direita”** caso seja configurado deve definir um objeto que represente o movimento que a respectiva mão assumirá durante a execução do sinal. Esses movimentos serão melhor detalhados nos próximos tópicos.

4.2.5 Tipos de Movimento

Assumindo-se que os objetos **“mao_esquerda”** ou **“mao_direita”** serão configurados, o objeto que identifique o tipo de movimento deve estar contido dentro

de cada mão. Esse parâmetro faz referência aos tipos de ações (movimentos) que se consegue realizar sobre o avatar 3D e ao queresma movimento da LIBRAS.

Atualmente é possível realizar 7 (sete) tipos de movimentos, são eles:

- Movimento Circular;
- Movimento Semi-Circular;
- Movimento de Contato;
- Movimento Helicoidal (Espiral);
- Movimento Pontual;
- Movimento Retilíneo e;
- Movimento Senoidal.

O objeto pode ser diferente um do outro dependendo do tipo de movimento. Cada tipo possui argumentos específicos e comuns para a execução do movimento.

Os argumentos comuns a todos são: “**articulacao**”, “**configuracao**” e “**orientacao**”, esses argumentos devem ser do tipo inteiro. Os valores passados são usados para acessar os posicionamentos salvos no avatar o que pode variar dependendo das posições salvas na biblioteca do avatar.

Atualmente para o modelo de avatar utilizado nesta solução o argumento “**articulação**” deve ter valores entre: **0** a **124,** o de "**configuracao**" valores entre: **0** a **59** e "**orientacao**" valores entre: **0** a **142**.

4.2.5.1 Movimento circular

Os argumentos para a execução do movimento “**circular**” são apresentados na Figura 18.

.

```
"circular": {
    "plano": "esquerda-cima",
    "raio": "medio",
    "velocidade": "normal",
    "lado_oposto": false,
    "sentido_inverso": false,
    "articulacao": 124,
    "configuracao": 59,
    "orientacao": 142
}
```

Figura 18 - Exemplo de objeto para o movimento circular.
Fonte: Elaborada pelo autor.

“plano”: Tipo String. Assume valores entre “**frente-esquerda”**, “**frente-cima”** e “**esquerda-cima”**. Identifica o plano de deslocamento da mão.

"raio": Tipo String. Assume valores entre **"pequeno"**, **"medio"** e **"grande".** Identifica o tamanho do raio do movimento circular.

"velocidade": Tipo String. Assume valores entre **"lento"**, **"normal"** e **"rapido"**. Identifica a velocidade do movimento.

"lado_oposto": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, inicia o movimento pelo lado oposto.

"sentido_inverso": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, faz com que o movimento gire no sentido contrário.

4.2.5.2 Movimento semi-circular

Os argumentos necessários para a execução do movimento **“semi-circular”** são apresentados na Figura 19.

```
"semicircular": {
    "plano": "esquerda-cima",
    "raio": "medio",
    "velocidade": "normal",
    "lado_oposto": false,
    "sentido_inverso": false,
    "articulacao": 124,
    "configuracao": 59,
    "orientacao": 142
}
```

Figura 19 - Exemplo de objeto para o movimento semi-circular.
Fonte: Elaborada pelo autor.

"plano": Tipo String. Assume valores entre **"frente-esquerda"**, **"frente-cima"** e **"esquerda-cima"**. Identifica o plano de deslocamento da mão.

"raio": Tipo String. Assume valores entre **"pequeno"**, **"medio"** e **"grande"**. Identifica o tamanho do raio do movimento semi-circular.

"velocidade": Tipo String. Assume valores entre **"lento"**, **"normal"** e **"rapido"**. Identifica a velocidade do movimento.

"lado_oposto": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, inicia o movimento pelo lado oposto.

"sentido_inverso": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, faz com que o movimento gire no sentido contrário.

4.2.5.3 Movimento contato

O movimento **“contato”** deve conter um sub-objeto identificando o tipo de contato. Os tipos de contatos são:

- Alisar **(“alisar”)**;
- Coçar **(“cocar”)**;
- Tocar **(“tocar”)** e;
- Riscar **(“riscar”)**;

Os argumentos necessários para a execução do movimento **“contato”** são apresentados na Figura 20.

.

```
"contato": {
    "alisar": {
        "movimento_orientacao": "paralelo",
        "articulacao": 124,
        "configuracao": 59,
        "orientacao": 142
    },
    "cocar": {
        "articulacao": 124,
        "configuracao": 59,
        "orientacao": 142
    },
    "tocar": {
        "articulacao": 124,
        "configuracao": 59,
        "orientacao": 142
    },
    "riscar": {
        "articulacao": 124,
        "configuracao": 59,
        "orientacao": 142
    }
}
```

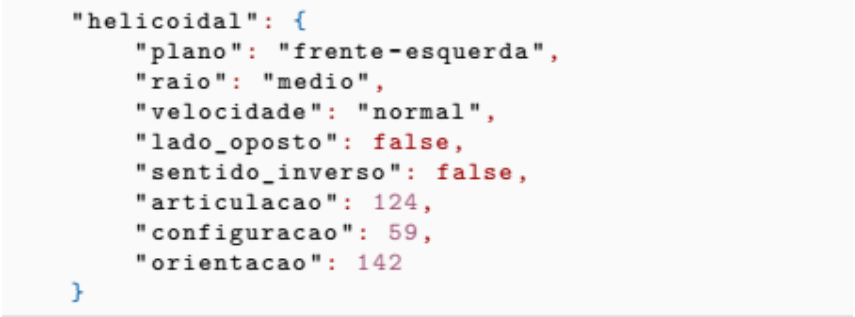

Figura 20 - Exemplo de objeto para o movimento contato.

Fonte: Elaborada pelo autor.

Todos os tipos de contatos possuem os argumentos: **"articulacao"**, **“configuracao"** e **"orientacao"**. Apenas o tipo **“alisar”** contém um argumento **"movimento_orientacao"**, que recebe valores entre: **"perpendicular"**, **paralelo"**, **diagonal-direita"** e **"diagonal-esquerda"**.

4.2.5.4 Movimento helicoidal (espiral)

Os argumentos necessários para a execução do movimento **“helicoidal”** são apresentados na Figura 21.



```
"helicoidal": {
    "plano": "frente-esquerda",
    "raio": "medio",
    "velocidade": "normal",
    "lado_oposto": false,
    "sentido_inverso": false,
    "articulacao": 124,
    "configuracao": 59,
    "orientacao": 142
}
```


Figura 21 Exemplo de objeto para o movimento helicoidal.

Fonte: Elaborada pelo autor.

"plano": Tipo String. Assume valores entre **"frente-esquerda"**, **"frente-cima"** e **"esquerda-cima"**. Identifica o plano de deslocamento da mão, por padrão o sentido do deslocamento padrão segue a regra da mão direita.

"raio": Tipo String. Assume valores entre **"pequeno"**, **"medio"** e **"grande"**. Identifica o tamanho do raio do movimento helicoidal.

"velocidade": Tipo String. Assume valores entre **"lento"**, **"normal"** e **"rapido"**. Identifica a velocidade do movimento.

"lado_oposto": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, inicia o movimento no sentido oposto a regra da mão direita.

"sentido_inverso": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, faz com que o movimento seja no sentido contrário.

4.2.5.5 Movimento pontual

É o movimento mais simples dentre os movimentos e utiliza os argumentos que são comuns a todos.

Os argumentos necessários para a execução do movimento **“pontual”** são apresentados na Figura 22.

```
"pontual": {
    "articulacao": 124,
    "configuracao": 59,
    "orientacao": 142
}
```

Figura 22 - Exemplo de objeto para o movimento pontual.
Fonte: Elaborada pelo autor.

4.2.4.6 Movimento retilíneo

O movimento retilíneo é uma variação do movimento pontual onde se define um ponto de início e fim para o ponto de articulação. Na prática isso faz com que a mão percorra do ponto inicial até o ponto final. Os argumentos são **“articulacao_inicial”** e **“articulacao_final”** (ver Figura 23).

```
"retilineo": {
    "articulacao_inicial": 124,
    "articulacao_final": 124,
    "configuracao": 59,
    "orientacao": 142
}
```

Figura 23 - Exemplo de objeto para o movimento retilíneo.
Fonte: Elaborada pelo autor.

4.2.4.7 Movimento senoidal

O movimento **“senoidal”** faz com que a mão percorra na forma como se estivesse em curvas sinuosas.

Os argumentos necessários para a execução do movimento **“senoidal”** são apresentados na Figura 24.

```
"senoidal": {
    "direcao": "frente",
    "desloca_eixo": "esquerda-direita",
    "direcao_oposta": false,
    "articulacao": 124,
    "configuracao": 59,
    "orientacao": 142
}
```

Figura 24 - Exemplo de objeto para o movimento senoidal.
Fonte: Elaborada pelo autor.

"direcao": Tipo String. Assume valores entre **"frente"**, **“esquerda”** e **"cima"**.

Uma vez configurado o campo **“direcao”**, o eixo de deslocamento **("desloca_eixo")** fica condicionado da seguinte forma:

Se o valor de **"direcao"** for igual a **"frente"** o argumento **"desloca_eixo"** pode assumir os seguintes valores: **"esquerda-direita"** e **"cima-baixo"**.

Se o valor de **"direcao"** for igual a **"esquerda"** o argumento **"desloca_eixo"** pode assumir os seguintes valores: **"frente-tras"** e **"cima-baixo"**.

Se o valor de **"direcao"** for igual a **"cima"** o argumento **"desloca_eixo"** pode assumir os seguintes valores: **"frente-tras"** e **"esquerda-direita"**.

"direcao_oposta": Tipo Booleano. Assume valores entre **true** e **false**. Caso seja **true**, indica que a direção deve ser contrária, por exemplo, se a direção for escolhida para frente, passa a ser para trás, esquerda passa a ser direita, cima passa a ser baixo.

4.2.6 Exemplo de sinais

A Figura 25 mostra os argumentos e os valores necessários para gerar o sinal panela. Esses valores serão transparentes ao usuário, uma vez que esses valores serão vinculados a uma interface de uma aplicação. Para testar a utilização da API foi feita uma requisição para o endereço “http://150.165.204.35:5001/sign” (ver Figura 26) passando os dados mostrado na Figura 25. Os valores e o que cada argumento faz foram explicados anteriormente.

Figura 25 - Exemplo de objeto para gerar o sinal panela.
Fonte: Elaborada pelo autor.

```
curl -X POST -d @panela.json http://150.165.204.35:5001/sign
--header "Content-Type:application/json"
```

Figura 26 - Exemplo de uma requisição HTTP POST a API.
Fonte: Elaborada pelo autor.

Após feita a requisição o é sinal gerado e armazenado ao dicionário. O sinal fica disponível na pasta pessoa do colaborador passado no argumento **“userId”**. A Figura 27 mostra o sinal gerado após a requisição feita demostrado anteriormente nesse tópico.

Figura 27 – Vídeo gerado: sinal panela.
Fonte: Elaborada pelo autor.

4.3 CORE

Este componente contém os scripts Python que são responsáveis por receber as informações da API e inicializar o Blender. A Figura 28 ilustra a função que define, por exemplo, as configurações da face.

Para cada parâmetro da língua de sinais há no script uma função responsável por configurar no Blender os parâmetros escolhidos pelo o usuário para que seja posteriormente renderizado o sinal.

```
# Função que inicia a configuração da face
def configureFace():
    global endFrame
    if(json_input["facialExp"] != []):
        # Set face
        setFaceConfiguration(json_input["facialExp"], [endFrame/4], util.faceBonesConf)
```

Figura 28 - Função que configura a face.
Fonte: Elaborada pelo autor.

4.3.1 WikiLIBRAS

O WikiLibras é uma ferramenta Web de construção colaborativa de sinais em Libras (SILVA, 2012), desenvolvida pelos pesquisadores do Núcleo de Pesquisa e Extensão em Aplicações de Vídeo Digital (LAVID) da Universidade Federal da Paraíba (UFPB). A proposta do WikiLibras é permitir que colaboradores, especialmente pessoas surdas e especialistas em Libras possam participar do processo de desenvolvimento do dicionário de sinais do VLibras[4] a partir da adição de novos sinais ou da edição dos sinais existentes, tornando o seu desenvolvimento mais produtivo.

Nessa seção, o objetivo é mostrar a integração do WikiLibras com o Animador proposto neste trabalho, para gerar os sinais configurados pelos colaboradores. No entanto, para realizar essa integração, algumas adaptações tiveram que ser realizadas no WikiLibras para ajustar com os dados trafegados com a API do Animador.

Os dados eram recebidos por meio de um XML e as funções de manipulação do avatar eram mapeados através de um dicionário onde se passava a chave no XML e o valor era recuperado no core. A manipulação das posições no avatar eram feitas programaticamente, ou seja, o core era responsável por configurar, a rotação, ângulo e o deslocamento dos ossos do avatar. Atualmente algumas dessas posições já são pré-definidas e salvas no próprio avatar sem a necessidade de manipular a rotação, ângulo ou deslocamento dos ossos do avatar. Apenas os movimentos são inseridos pelo core, pois são dinâmicos.

Atualmente a interface web do WikiLIBRAS é construída com a linguagem de programação JavaScript e as tecnologias necessárias para a *WEB* (HTML, CSS).

Uma vez que, segundo Stumpf (2000) em geral os surdos têm muitas dificuldades para ler e escrever em português, a interface foi projetada para conter o mínimo de textos, tendo como foco a utilização de elementos gráficos e animações.

---

[4] O VLibras consiste em um conjunto de ferramentas computacionais de código aberto, responsável por traduzir automaticamente conteúdos digitais (texto, áudio e vídeo) para Língua Brasileira de Sinais (LIBRAS), tornando computadores, dispositivos móveis e plataformas Web acessíveis para pessoas surdas.

Dessa forma, o usuário colaborador pode configurar o sinal passando por várias etapas, onde em cada etapa um parâmetro do sinal é configurado. As Figuras 29, 30, 31 e 32 ilustram essas etapas para configuração de um sinal. Após o colaborador configurar todos os parâmetros tem como resultado final o vídeo que é ilustrado na Figura 33.

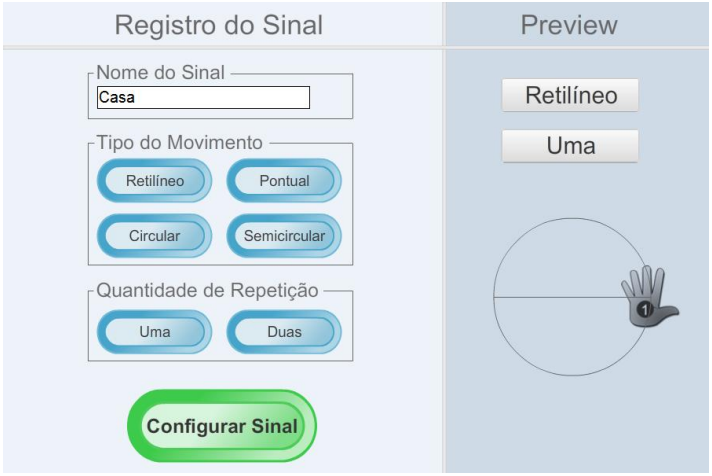


Figura 29 - Captura da tela principal de cadastro de um sinal do WikiLIBRAS.
Fonte: Elaborada pelo autor.

Figura 30 - Tela de configuração de mão.
Fonte: Elaborada pelo autor.

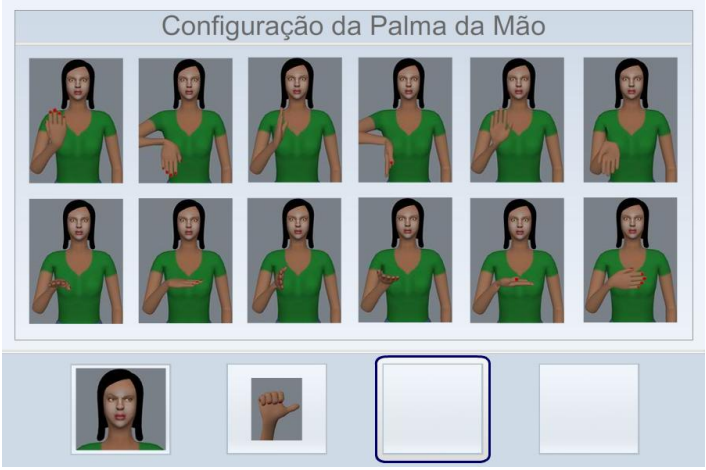


Figura 31 - Tela de configuração da orientação da mão.
Fonte: Elaborada pelo autor.

Figura 32 - Tela de configuração do ponto de articulação.
Fonte: Elaborada pelo autor.

Figura 33 - Sinal gerado pelo WikiLibras.
Fonte: Elaborada pelo autor.

Nesta seção foram apresentados a forma como o *core* está implementado e quais foram as principais mudanças realizadas para integrar o WikiLibras com o Animador solução proposta neste trabalho, e os passos necessários para se configurar o sinal no WikiLibras.

Na próxima seção será apresentado o modelo avatar usado nesta solução, suas principais características e os principais recursos da ferramenta Blender utilizados no desenvolvimento do avatar.

4.4 AVATAR

Para o desenvolvimento deste trabalho, optou-se por utilizar o modelo animado (avatar-cartoon 3D) de um agente para representar os sinais gerados. Esse modelo foi desenvolvido também pelos pesquisadores do LAVID/UFPB e também compõe a ferramenta de tradução automática para Libras, denominado VLibras.

O Avatar foi desenvolvido na ferramenta *Blender* e dispõe de uma armadura com 82 ossos, distribuídos da seguinte forma:

- 23 ossos para configurar a face;
- 22 ossos para configurar o corpo e braços;
- 15 ossos para cada mão e;
- 7 ossos auxiliares.

Na definição de uma configuração de mão, por exemplo, os parâmetros de localização e rotação de todos os 15 ossos da mão são definidos. Na configuração da face alguns ossos são bloqueados, ou seja, os parâmetros de localização e rotação não podem ser modificados. A movimentação dos braços é realizada modificando apenas dois ossos através da cinemática inversa. Um osso está localizado no pulso do Avatar e outro localizado fora da malha do Avatar esse responsável pela movimentação do cotovelo e antebraço.

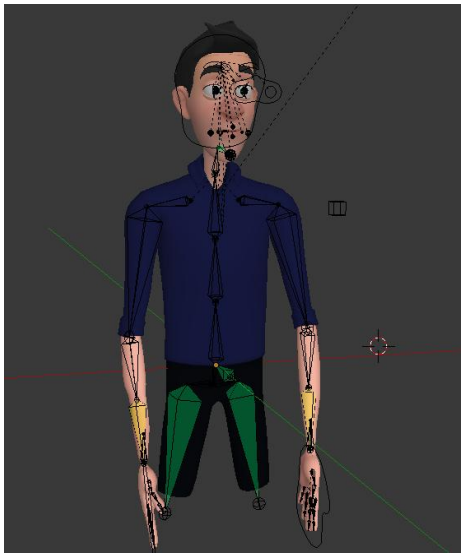
No Avatar é utilizado a *Inverse Kinematics* (cinemática inversa), no qual define - se alguns elementos de controle e objetos relacionados a esses elementos.

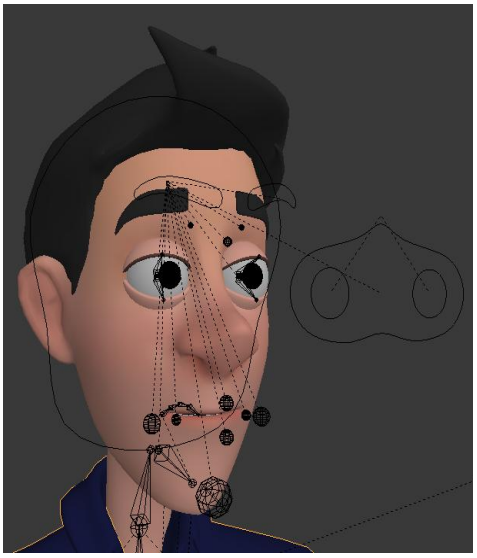
Na medida em que um elemento de controle se movimenta, as ações perpetuam-se para os demais objetos relacionados a ele, por exemplo, o osso do pulso é um elemento de controle (iK_Fk) e possui os ossos do braço relacionados a ele.

A fim de facilitar a configuração dos ossos do Avatar, utilizam-se bibliotecas de poses construídas previamente. No Blender, há um recurso *Pose Library* (biblioteca de pose) que possibilita armazenar posturas pré-definidas. Algumas configurações de face são armazenadas, por exemplo, as configurações de FELIZ, TRISTE e outros. São armazenados também algumas localizações e orientações da mão que serão utilizados para construção de sinais. Todas essas poses podem ser acessadas por índices, de forma eficiente e rápida.

A Figura 34a ilustra o modelo Avatar 3D *cartoon*. A Figura 34b, 34c e 34d ilustram o modelo com ênfase nos ossos da face, das mãos, auxiliares e do corpo, respectivamente.

(a) (b)





(c) (d)

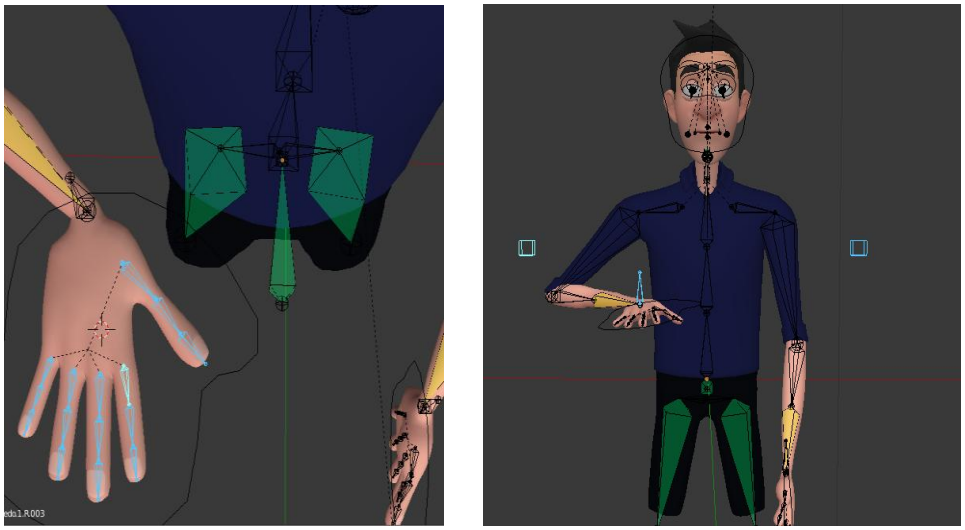


Figura 34 - (a) Modelo do avatar 3D cartoon. (b) Ossos da face, (c) das mãos, (d) auxiliares e do corpo.
Fonte: Elaborada pelo autor.

Nesse capítulo foram apresentados a arquitetura e as principais características da solução proposta. Os principais componentes: API, Core e a estrutura do objeto JSON foram apresentados mais detalhadamente. Por fim, uma explicação sobre as principais características do modelo avatar 3D, como quantos ossos o avatar é composto e como esses ossos se comportam dentro do modelo.

No próximo capítulo serão apresentados os resultados de alguns testes computacionais realizados com a solução proposta.

5 RESULTADOS

Neste capítulo serão apresentados alguns testes computacionais realizados com a solução proposta. Os testes foram desenvolvidos com o objetivo de avaliar o desempenho e a carga máxima que o sistema suporta.

Para realização do teste foi utilizada um servidor com as seguintes configurações:

- Sistema Operacional: Linux (Ubuntu 12.04);
- 6 Gigabytes de memória RAM e 512MB de cache L2;
- Processador Intel (R) Core (TM) i5-2430M CPU @ 2.40GHz;
- Disco Rígido (HD) 7200 RPM 32MB de cache SATA 3.0Gb/s com NCQ, 250 GB de SSD.

Os testes foram estruturados em dois conjuntos e realizados localmente, e sem a utilização da internet. São eles:

1. Realização de um conjunto de requisições simultâneas do mesmo tipo de movimento e um sinal simples com a configuração de apenas uma mão para o Animador, considerando cada um dos tipos de movimento suportados pelo Animador;
2. Realização de requisições simultâneas contendo diferentes tipos de movimentos, a fim de verificar o desempenho para requisições aleatórias.

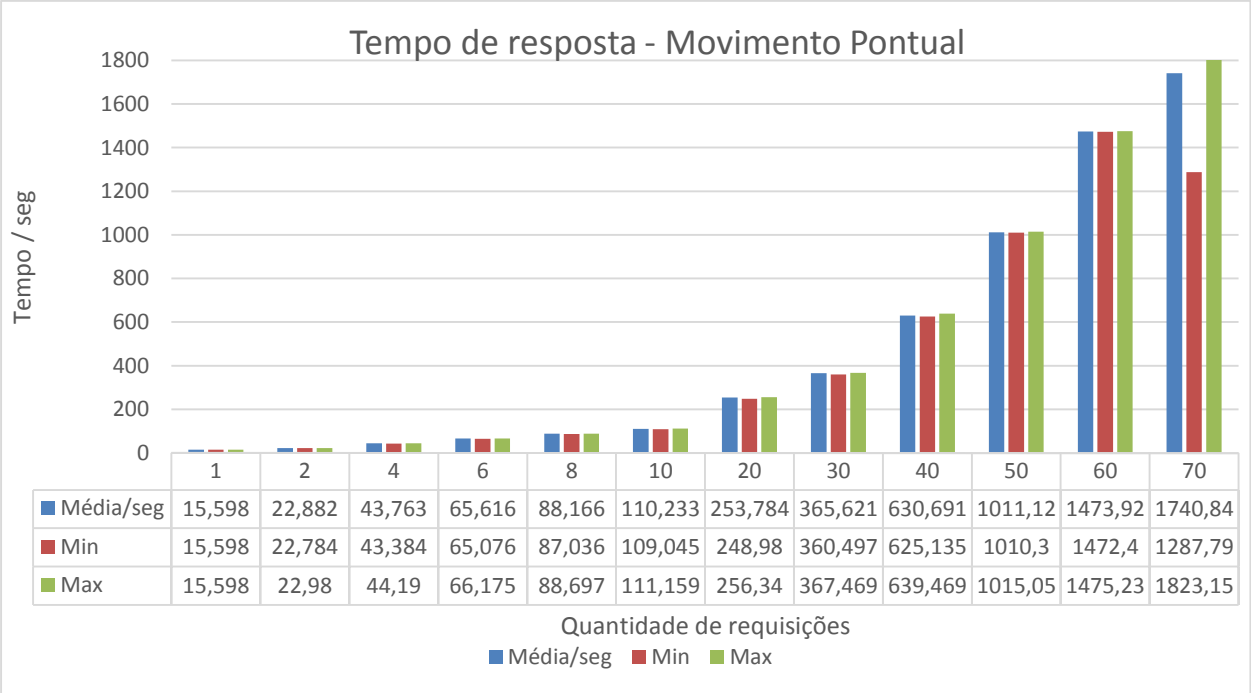
Para os dois conjuntos de testes, foram avaliados o tempo médio de resposta para todas a requisições feitas a API foi avaliado quais foram as requisições que obtiveram o menor e maior tempo de resposta. Em seguida, a vazão do sistema foi calculada considerado a seguinte definição:

$$Vazão = \frac{Quantidade\ de\ requisições}{Tempo\ de\ observação\ (minuto)}$$

Na Figura 35, é ilustrado o tempo médio de resposta para requisições de geração de sinais para o movimento pontual, enquanto que na Figura 36, é ilustrada

a vazão da solução, de acordo com a quantidade de requisições feitas à API. Nesse teste, foram feitas apenas requisições enviando o seguinte JSON contendo apenas o movimento pontual em uma das mãos, conforme pode ser observado abaixo:

{"userId": "Neto", "sinal": "modelo_pontual", "interpolacao": "normal", "movimentos": [{"facial": {}, "mao_direita": {}, "mao_esquerda": {"pontual": {"articulacao": 71, "configuracao": 19, "orientacao": 11}}}]}



	1	2	4	6	8	10	20	30	40	50	60	70
Média/seg	15,598	22,882	43,763	65,616	88,166	110,233	253,784	365,621	630,691	1011,12	1473,92	1740,84
Min	15,598	22,784	43,384	65,076	87,036	109,045	248,98	360,497	625,135	1010,3	1472,4	1287,79
Max	15,598	22,98	44,19	66,175	88,697	111,159	256,34	367,469	639,469	1015,05	1475,23	1823,15

Figura 35 - Gráfico de tempo médio de resposta - movimento pontual.
Fonte: Elaborada pelo autor.

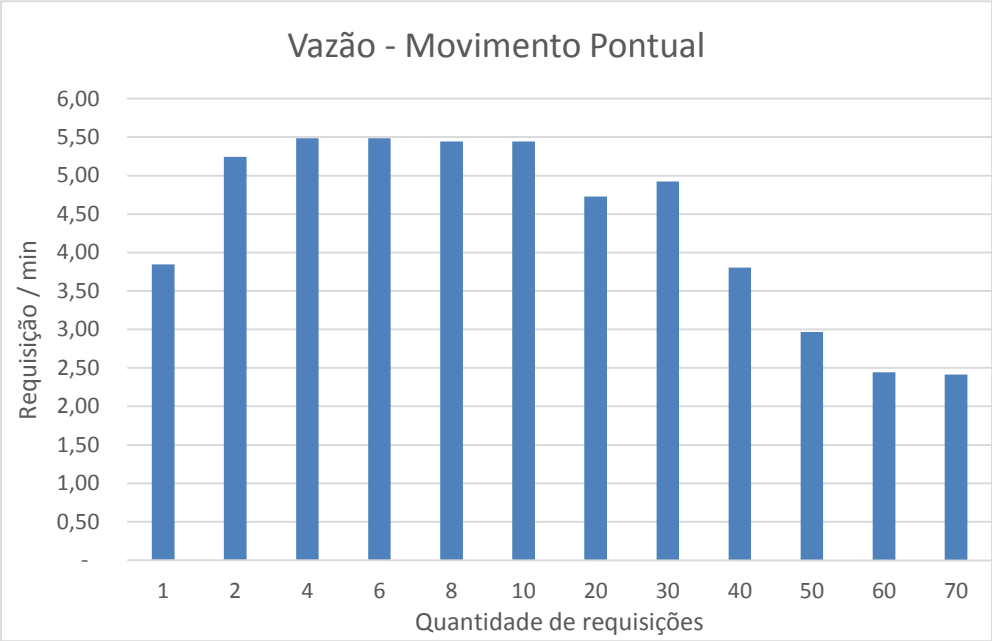


Figura 36 - Gráfico de vazão - movimento pontual.
Fonte: Elaborada pelo autor.

Considerando as requisições cujo movimento do sinal é apenas o Pontual, a vazão média do sistema observada foi de **4,35** requisições por minuto.

Nas Figuras 37 e 38, são apresentados o tempo médio de resposta e a vazão da solução para o movimento retilíneo, de acordo com a quantidade de requisições feitas à API. Nesse teste, foram feitas requisições enviando o JSON contendo apenas o movimento retilíneo em umas das mãos, conforme pode ser observado abaixo:

{"userId": "Neto", "sinal": "modelo_retilineo", "interpolacao": "normal", "movimentos": [{"facial": {},"mao_direita": {},"mao_esquerda": {"retilineo": {"articulacao_inicial": 71, "configuracao_inicial": 19, "orientacao_inicial": 11, "articulacao_final": 78, "configuracao_final": 19, "orientacao_final": 11}}}]}

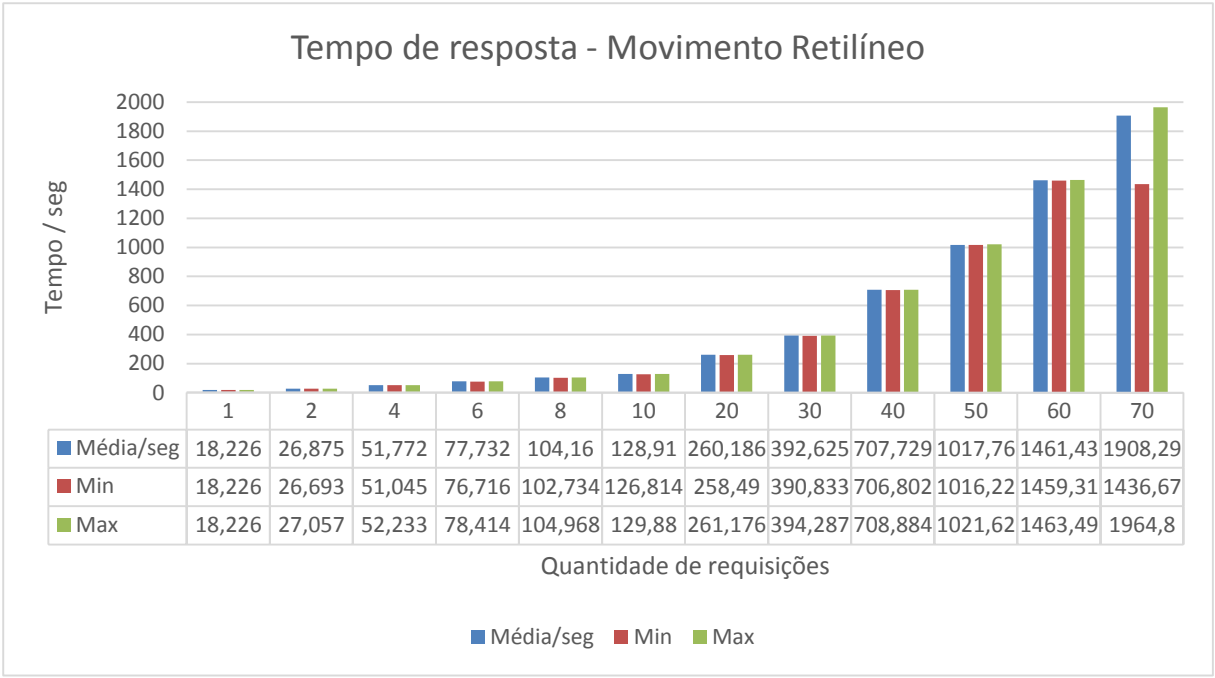


Figura 37 - Gráfico de tempo médio de resposta - movimento retilíneo.
Fonte: Elaborada pelo autor.

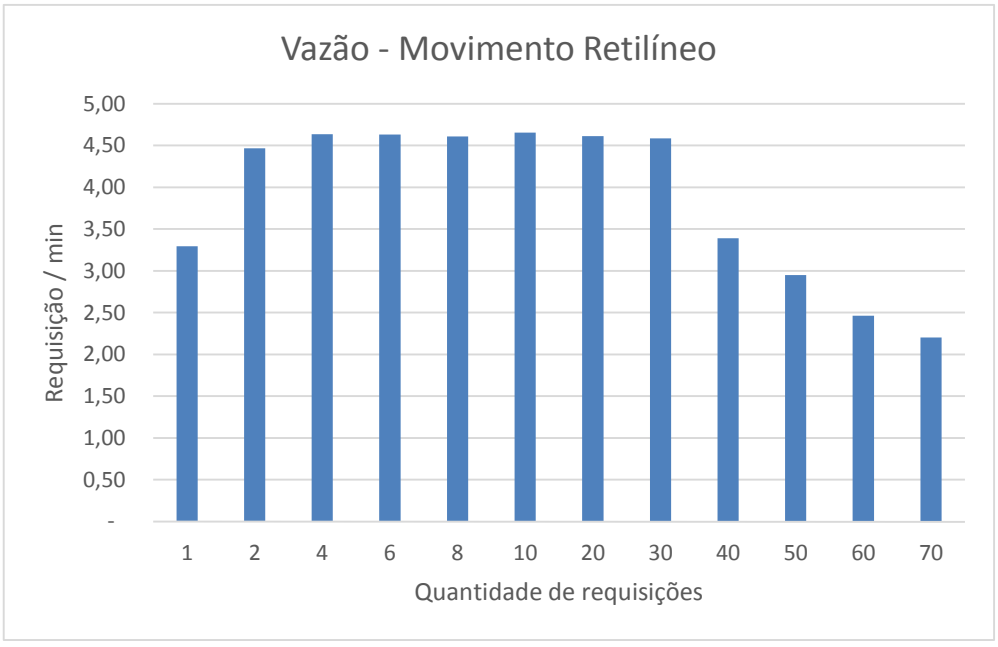
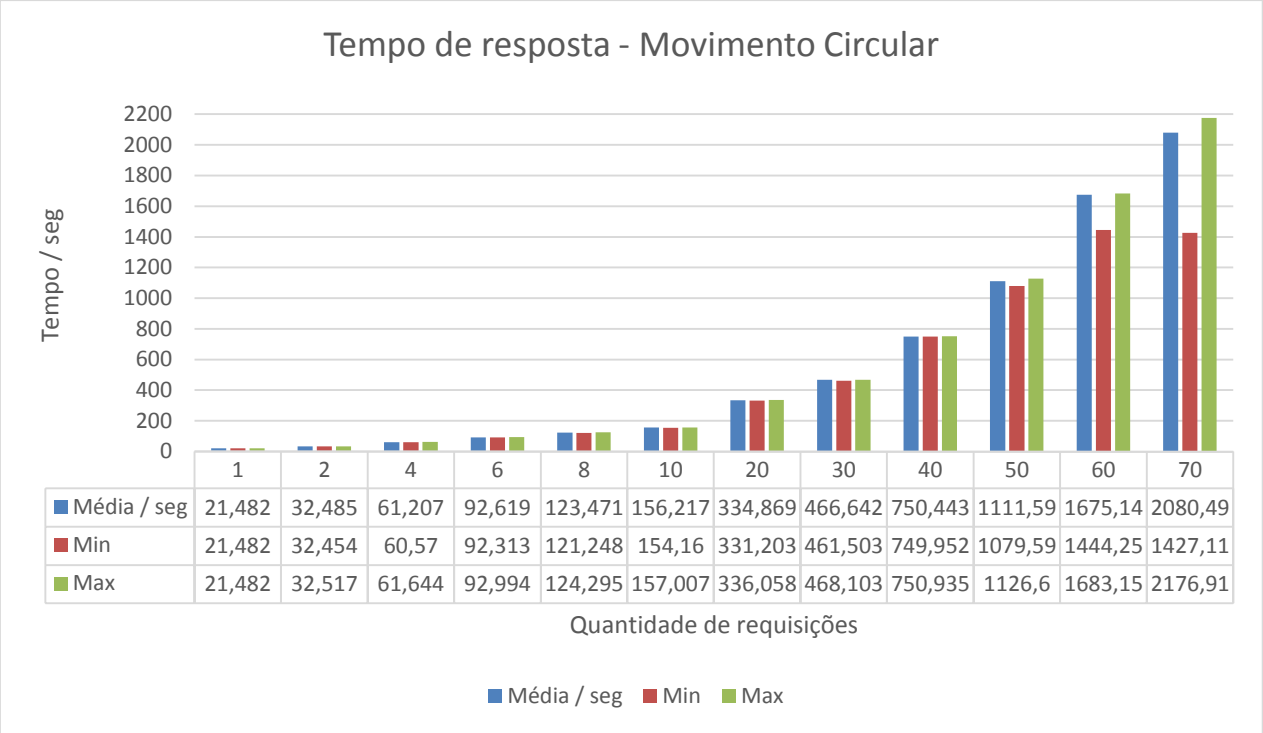


Figura 38 - Gráfico de vazão - movimento retilíneo.
Fonte: Elaborada pelo autor.

Considerando requisições cujo movimento do sinal é apenas o Retilíneo, a vazão média do sistema obtida foi de **3,87** requisições por minuto.

Nas Figuras 39 e 40, 41 e 42, 43 e 44, 45 e 46, são ilustrados os mesmos testes aplicados para calcular o tempo médio de resposta e a vazão do Animador para os movimentos circular, semicircular, senoidal e helicoidal, respectivamente.

De acordo com essas Figuras, é possível observar que para o movimento Circular a vazão média do sistema foi de **3,29** requisições por minuto. Para os movimentos Semicircular, Senoidal e Helicoidal, a vazão média obtida foi de **3,84, 1,81 e 1,70** requisições por minuto, respectivamente.



	1	2	4	6	8	10	20	30	40	50	60	70
Média / seg	21,482	32,485	61,207	92,619	123,471	156,217	334,869	466,642	750,443	1111,59	1675,14	2080,49
Min	21,482	32,454	60,57	92,313	121,248	154,16	331,203	461,503	749,952	1079,59	1444,25	1427,11
Max	21,482	32,517	61,644	92,994	124,295	157,007	336,058	468,103	750,935	1126,6	1683,15	2176,91

Figura 39 – Gráfico de tempo médio de resposta - movimento circular.
Fonte: Elaborada pelo autor.

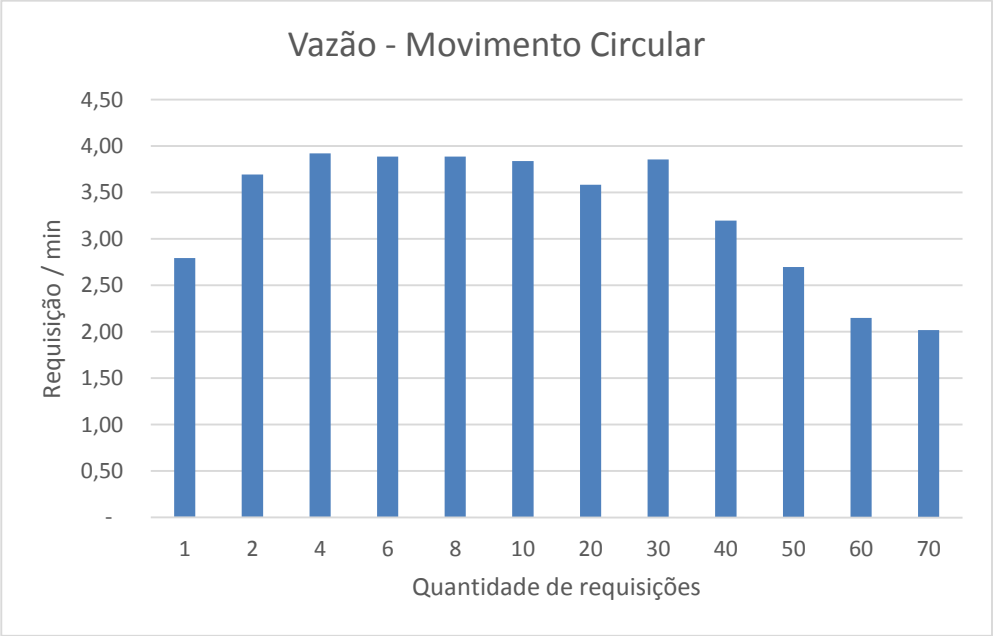
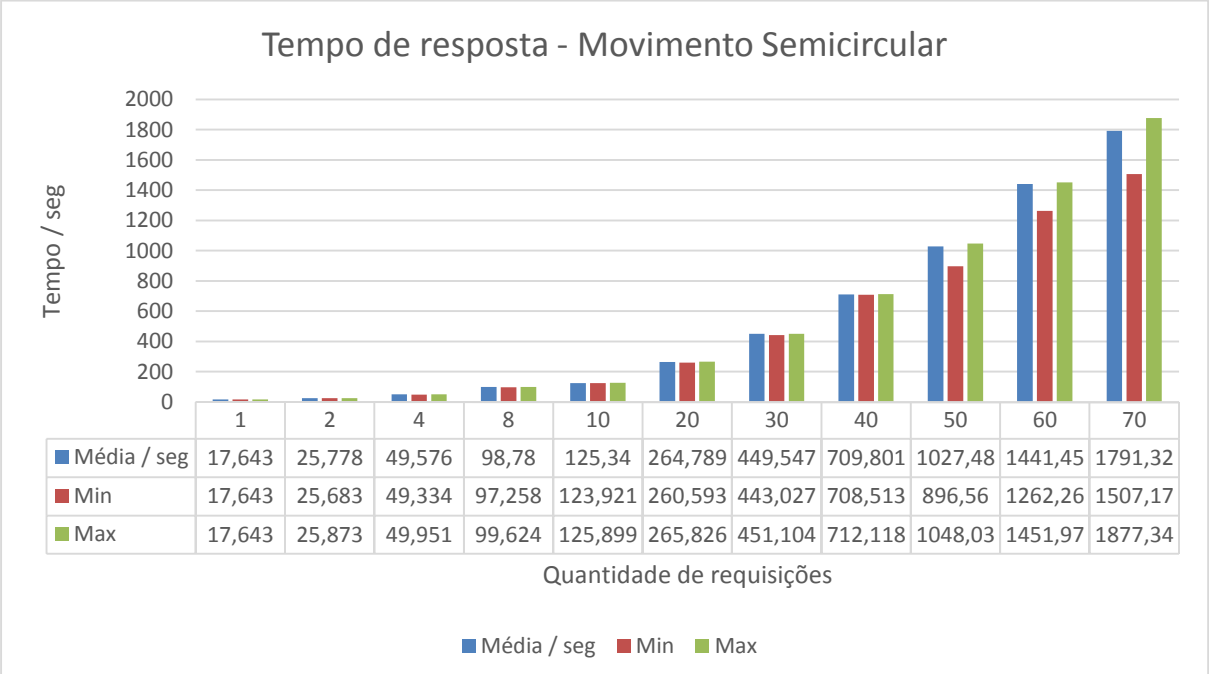


Figura 40 - Gráfico de vazão - movimento circular.
Fonte: Elaborada pelo autor.



	1	2	4	8	10	20	30	40	50	60	70
Média / seg	17,643	25,778	49,576	98,78	125,34	264,789	449,547	709,801	1027,48	1441,45	1791,32
Min	17,643	25,683	49,334	97,258	123,921	260,593	443,027	708,513	896,56	1262,26	1507,17
Max	17,643	25,873	49,951	99,624	125,899	265,826	451,104	712,118	1048,03	1451,97	1877,34

Figura 41 - Gráfico de tempo médio de resposta - movimento semicircular.
Fonte: Elaborada pelo autor.

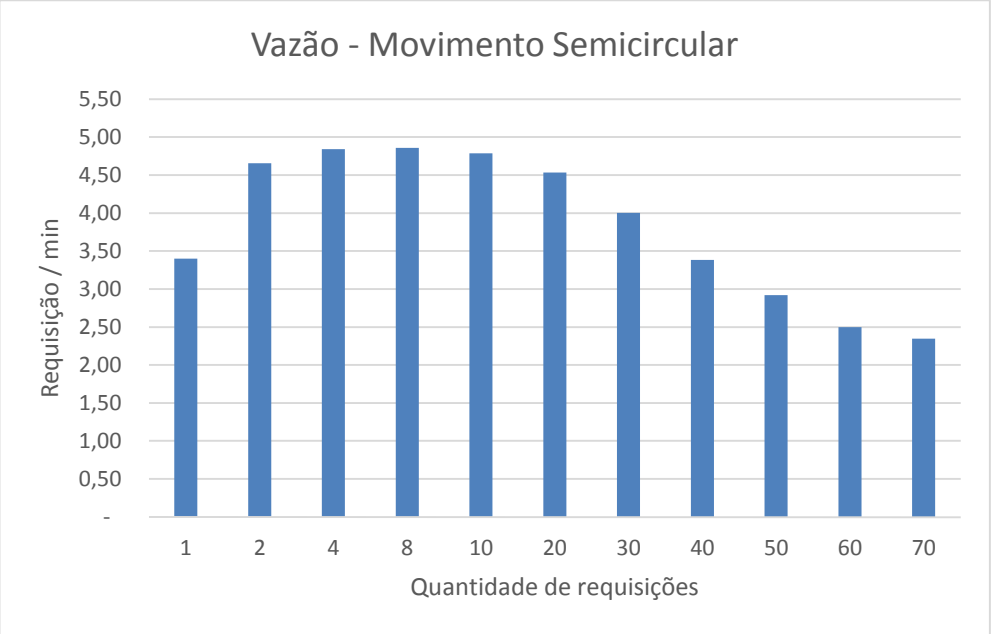
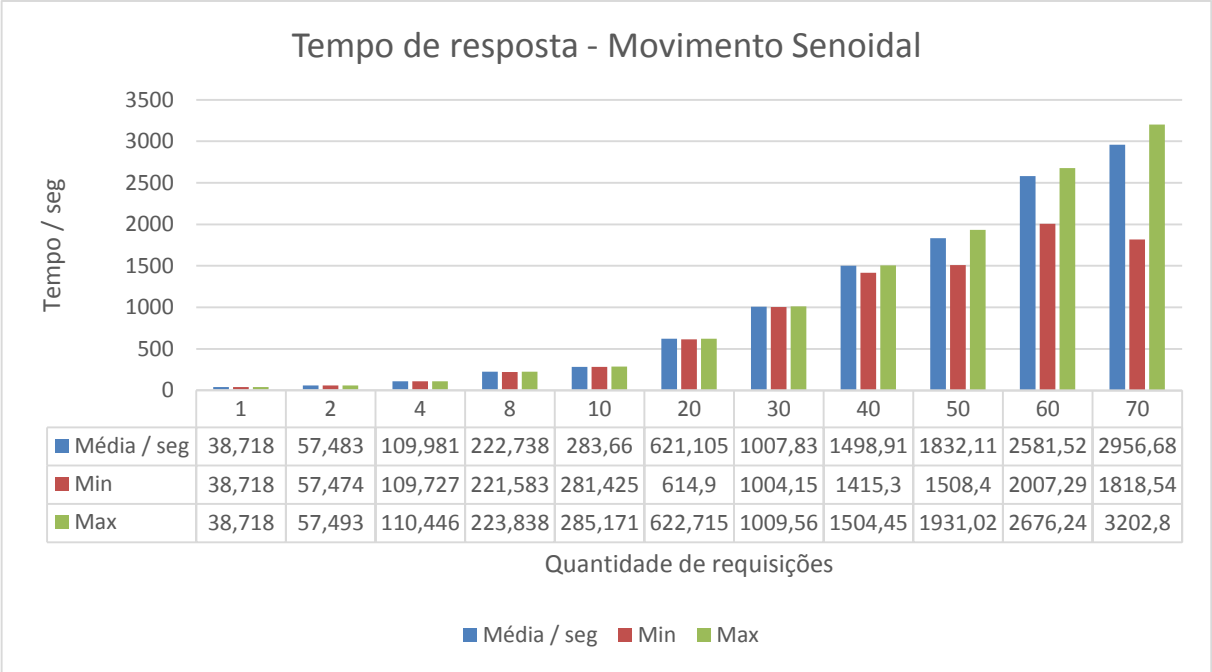


Figura 42 - Gráfico de vazão - movimento semicircular.
Fonte: Elaborada pelo autor.



	1	2	4	8	10	20	30	40	50	60	70
Média / seg	38,718	57,483	109,981	222,738	283,66	621,105	1007,83	1498,91	1832,11	2581,52	2956,68
Min	38,718	57,474	109,727	221,583	281,425	614,9	1004,15	1415,3	1508,4	2007,29	1818,54
Max	38,718	57,493	110,446	223,838	285,171	622,715	1009,56	1504,45	1931,02	2676,24	3202,8

Figura 43 – Gráfico de tempo médio de resposta - movimento senoidal.
Fonte: Elaborada pelo autor.

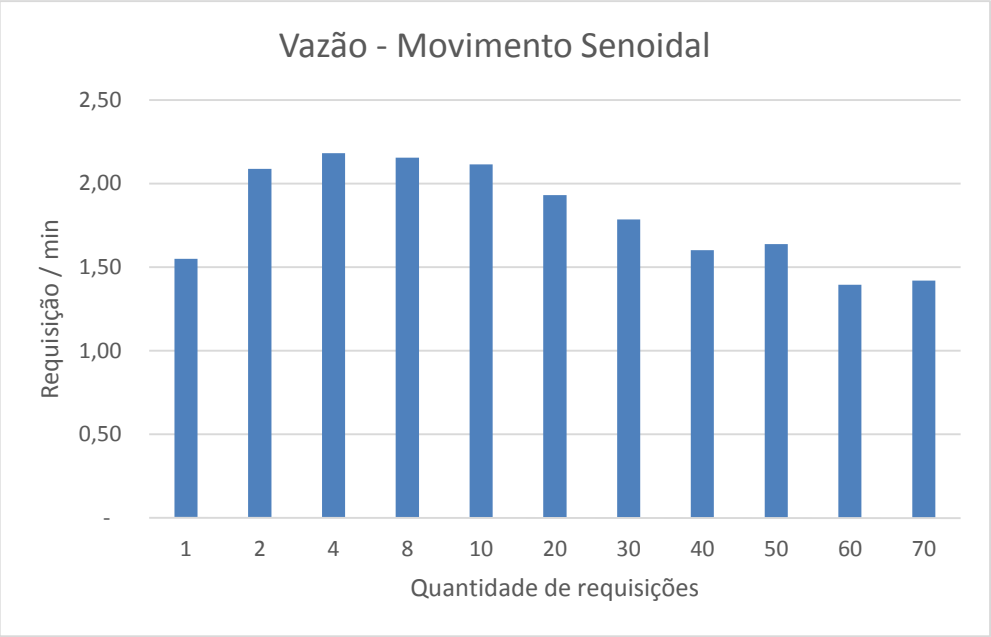
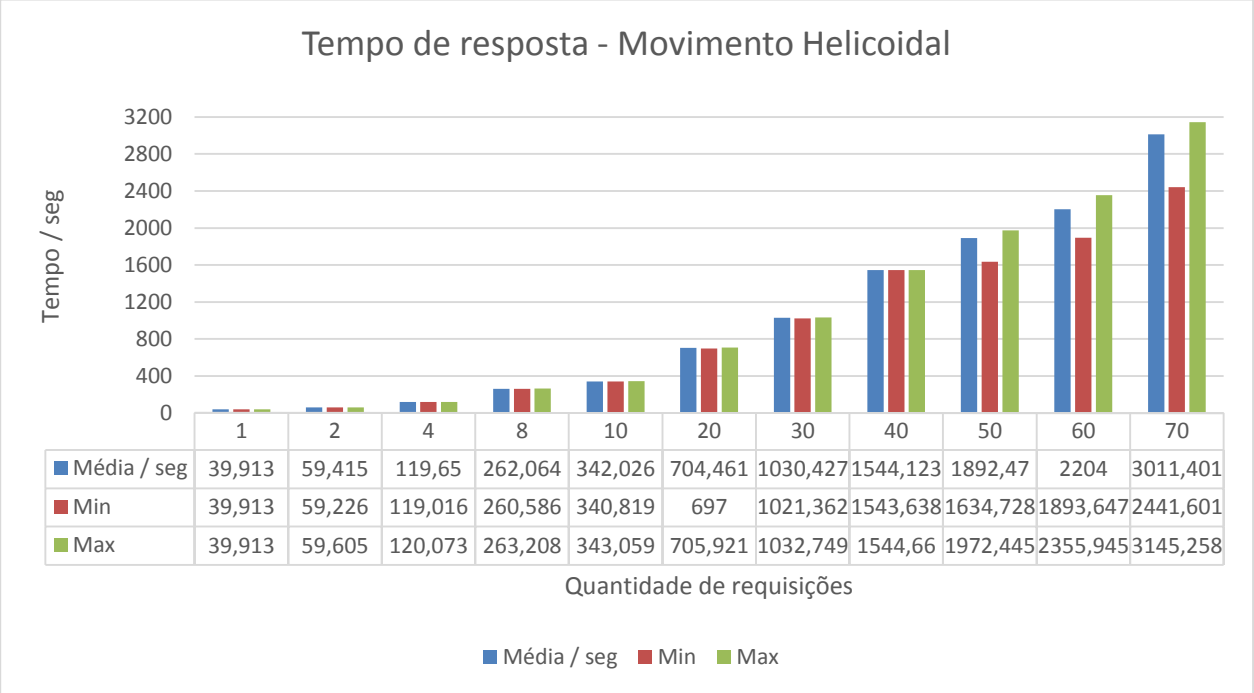


Figura 44 - Gráfico de vazão - movimento senoidal.
Fonte: Elaborada pelo autor.



	1	2	4	8	10	20	30	40	50	60	70
Média / seg	39,913	59,415	119,65	262,064	342,026	704,461	1030,427	1544,123	1892,47	2204	3011,401
Min	39,913	59,226	119,016	260,586	340,819	697	1021,362	1543,638	1634,728	1893,647	2441,601
Max	39,913	59,605	120,073	263,208	343,059	705,921	1032,749	1544,66	1972,445	2355,945	3145,258

Figura 45 - Gráfico de tempo médio de resposta - movimento helicoidal.
Fonte: Elaborada pelo autor.

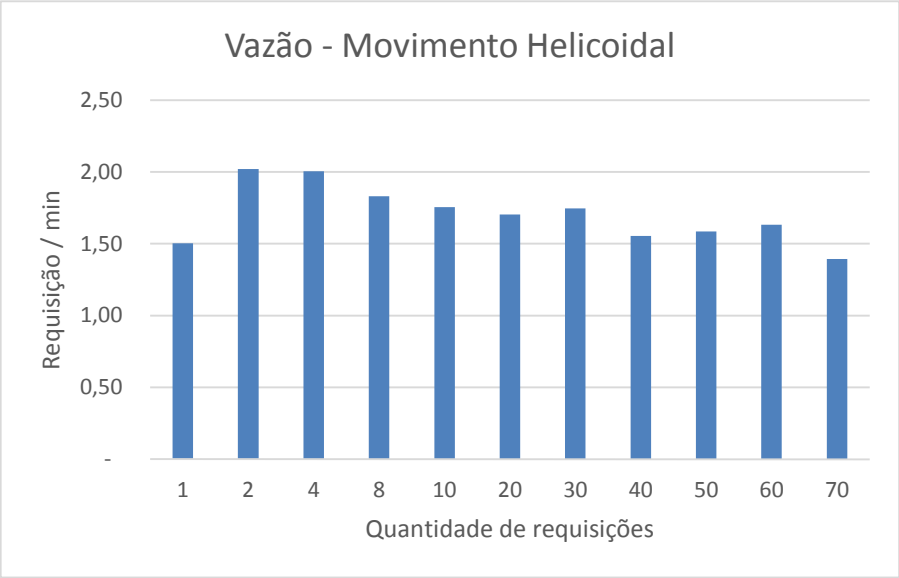
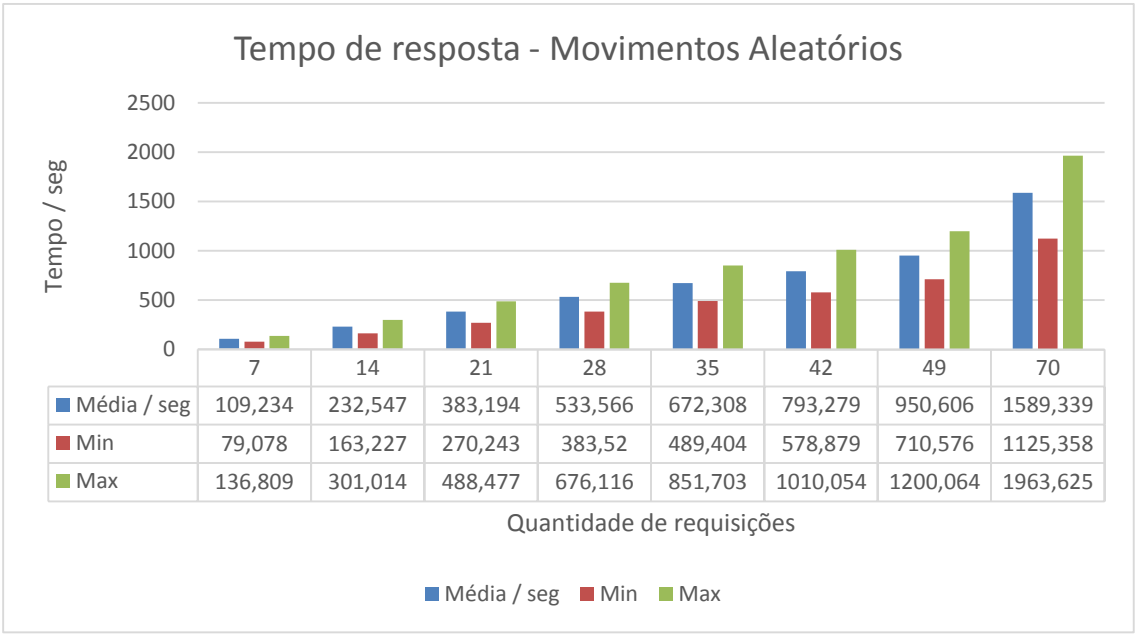


Figura 46 - Gráfico de vazão - movimento helicoidal.
Fonte: Elaborada pelo autor.

Em seguida, alguns testes foram realizados combinando grupos de requisições feitas com tipos de movimentos escolhidos aleatoriamente. Os resultados desses testes são apresentados nas Figuras 47 e 48. Essas requisições foram feitas simultaneamente usando os JSONs utilizados pelos testes de cada movimento citados anteriormente.



	7	14	21	28	35	42	49	70
Média / seg	109,234	232,547	383,194	533,566	672,308	793,279	950,606	1589,339
Min	79,078	163,227	270,243	383,52	489,404	578,879	710,576	1125,358
Max	136,809	301,014	488,477	676,116	851,703	1010,054	1200,064	1963,625

Figura 47 – Gráfico de tempo médio de resposta – movimentos aleatórios.
Fonte: Elaborada pelo autor.

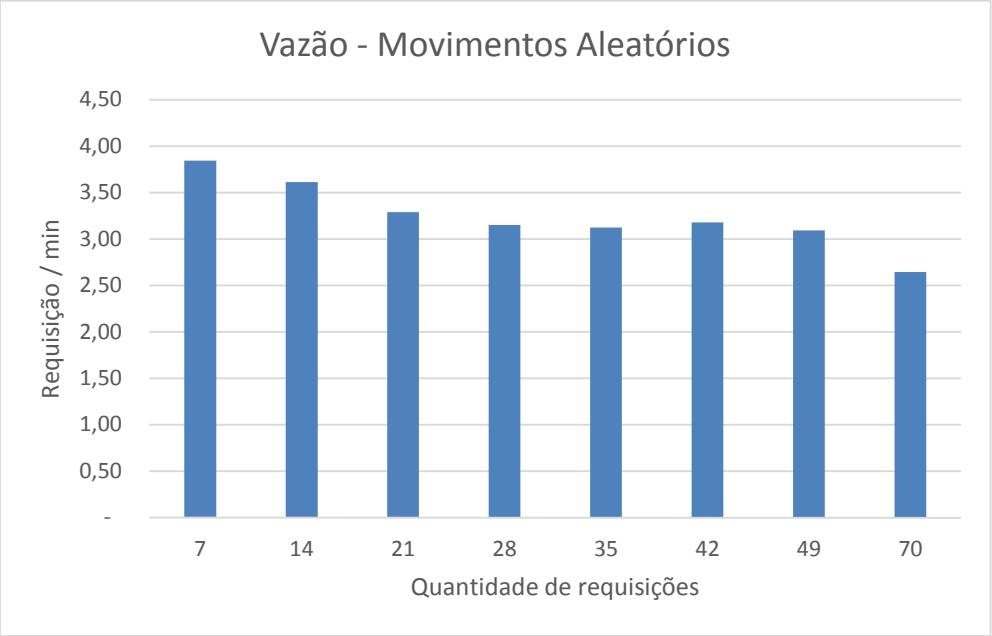


Figura 48 - Gráfico de vazão - movimentos aleatórios
. Fonte: Elaborada pelo autor.

Considerando requisições aleatórias combinando diferentes tipos de movimento, a vazão média do sistema foi de **3,24** requisições por minuto.

5.1 Discussões

Nos testes realizados, foi possível observar que o sistema utiliza um alto processamento, e que à medida que novas requisições são feitas simultaneamente, o tempo médio de resposta aumenta razoavelmente, levando-se em consideração as configurações da máquina em que os testes foram realizados.

Para requisições de movimento pontual, por exemplo, a solução, em média, levava 15,598 segundos para que uma das requisições fosse concluída. Quando 70 (setenta) são feitas de forma simultânea, o tempo para atender todas as requisições sobe para 1.740,84 segundos, o que significa que são necessários cerca de 29 minutos para que todas as requisições sejam concluídas. Para outros movimentos, o processamento ainda pode ser maior, pois o número de *frames* necessários para a geração do sinal é maior.

Por exemplo, para o movimento circular 1 (uma) requisição levou 21,482 segundos, pois o sinal é composto por mais frames. Para o mesmo movimento 70 (setenta) requisições levou 2080.487 segundos o que significa uma média de 34 minutos para concluir todas as requisições. Um tempo bem maior em comparação ao movimento Pontual.

Para o movimento senoidal e helicoidal, que são uma variação do movimento circular, é possível observar que o tempo de resposta aumenta significativamente. Para 1 (uma) requisição levou 38,718 e 39,913 segundos respectivamente e com 70 (setenta) requisições levou 2956,679 e 3011,401 segundos, aproximadamente 49 minutos para o movimento senoidal e 50 minutos para o movimento helicoidal.

Por outro lado, quando foram realizadas mais de 70 requisições simultâneas, para qualquer movimento a máquina que foi utilizada nos testes não conseguiu atender as requisições e parou de funcionar.

No próximo capítulo serão apresentadas as considerações finais sobre o trabalho desenvolvido, e possíveis trabalhos futuros e que não pode ser contemplado neste trabalho, afim de melhorar e evoluir a proposta desde trabalho.

6 CONSIDERAÇÕES FINAIS

Este trabalho propõe o desenvolvimento de um serviço para facilitar e possibilitar o desenvolvimento de aplicações que visem a geração de sinais de LIBRAS. A ideia é tornar cada vez mais amigável a interação do usuário com esses tipos de serviços, e estimular pessoas a criarem novos sinais.

O objetivo é que a ferramenta desenvolvida seja colaborativa, disponibilizando uma API para que novas aplicações sejam desenvolvidas, possam consumir os serviços da API e venham a alimentar uma base de dados de sinais.

Observa-se que os principais objetivos foram alcançados, que é fornecer levantar os principais requisitos para gerar um sinal animado, bem como os principais parâmetros da Libras para isso e também um serviço que se comunique com um core e crie o sinal alimentando uma base de dados. Um teste foi realizado usando o serviço e foram enviados os parâmetros para a construção do sinal “panela” e pode-se verificar que o vídeo foi gerado com sucesso [Capítulo 4.2.6].

Um comparativo foi realizado com outras soluções existentes, conforme ilustrado na tabela abaixo:

Ferramentas de criação de sinais.

Ferramenta	**Serviço[1]**			**Plataformas[2]**			**Dicionário. Expansível**
	WS	F	NP	W	M	PC	
Poli-Libras	X			X			X
F-LIBRAS			X			X	X
ProDeaf		X		X			
Animador	X			X			X

[1] WS = *Web Service*, F = Fechado, NP= Não Possui; [2] W = Web, M = Mobile.

Foram comparadas as características importantes relacionadas a criação de sinais. A Solução Poli-Libras disponibiliza um *Web Service*, como também a possibilidade de expansão do dicionário. A modelagem do avatar foi feita no 3D Studio Max. O Poli-Libras é bastante limitado em relação a criação de sinais com

poucas possibilidades de configurações para representação de um sinal. Já o F-LIBRAS não disponibiliza um serviço web para criação de sinais, apenas localmente.

Dentre as ferramentas avaliadas, a melhor solução é a ProDeaf a qual disponibiliza uma ferramenta web para criação de novos sinais. Porém, ela também não disponibiliza uma API para que outras aplicações possam se comunicar com o seu core e crie novos sinais, além de não ter uma base de dados expansível, já que os sinais criados pelos seus usuários apenas ficam salvos para visualização dos próprios usuários sem que esses sinais sejam utilizados por outras aplicações.

O Animador que é a solução proposta por esse trabalho disponibiliza uma API com o intuito de facilitar que outras aplicações possam ser desenvolvidas e possam consumir seus serviços, criando novos sinais, a fim de expandir ainda mais a biblioteca de sinais. A ideia é que essa base de dados seja construída e alimentada de forma colaborativa, o que torna essa solução mais abrangente do que as outras.

Embora este trabalho proponha uma solução abrangente para geração de sinais de forma semiautomática, alguns desafios podem ser identificados. Como trabalhos futuros, pretende-se implementar uma aplicação com uma interface que mapeie os índices das posições salvas na biblioteca de posições utilizadas no avatar.

Um esforço também deve ser atribuído a um estudo de usabilidade para tornar a configuração do sinal mais intuitiva e fácil para as pessoas especialistas em LIBRAS. Dessa forma, outra proposta de trabalho futuro seria investigar outras alternativas ou formas de melhor apresentar as posições do avatar para o usuário.

Outras propostas de trabalhos futuros que podem ser elencadas:

- Incorporar mais movimentos ao script de geração de sinais.
- Aumentar a variação dos movimentos existentes atualmente.
- Desenvolvimento de uma arquitetura escalável, tolerante a falhas e distribuída.
- Realizar outros testes com mais configurações e com mais sinais, por exemplo sinais compostos a fim de avaliar o impacto no tempo de resposta.

REFERÊNCIAS

ALVES, William Pereira. Modelagem e animação com Blender. [São Paulo]: Érica, 2006.

ANDRADE, B. M. Guia do Usuário Para Uso do Blender 3D Orientado a Design. 2008. Trabalho de Conclusão (Graduação em Design), Universidade Federal de Pernambuco.

BAPTISTA, F. F-Libras - Ambiente Integrado De Ensino-Aprendizagem Para Língua Brasileira De Sinais. Marília: [s.n.], 2007. 35

BRASIL. Presidência da República. Secretaria de Comunicação Social. Pesquisa brasileira de mídia 2015: hábitos de consumo de mídia pela população brasileira– Brasília: Secom, 2014. Disponível em: <http:/http://www.secom.gov.br/atuacao/pesquisa/lista-de-pesquisas-quantitativas-e-qualitativas-de-contratos-atuais/pesquisa-brasileira-de-midia-pbm-2015.pdf>. Acesso em: 14 abr. 2015.

CAPOVILLA, Fernando César; RAPHAEL, Walkíria Duarte (Ed). Dicionário encidopédico ilustrado trilingiie da língua de sinais brasileira. 2. ed. Ilustrações de Silvana Marques. São Paulo: USP/Imprensa Oficial do Estado 2001.v. I: sinais de A L e v. 11: sinais de M a Z.

CARVALHO, Guilherme Januário; ALEXANDRE, Leonardo Ferreira Leite; LI, Marcelo Koga. Poli-Libras um tradutor de português para Libras. 2010. 104 f. Trabalho de Conclusão de Curso (Graduação) apresentado à Escola Politécnica da USP, 2010.

CLUA, E. W. G.; BITTENCOURT, J. R. Desenvolvimento de Jogos 3D: concepção, Design e Programação. Anais da XXIV Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, pp. 1313-1356, São Leopoldo, Brazil, julho de 2005.

FENEIS. FELIPE, Tanya A.; MONTEIRO, Myrna S. Libras em Contexto: curso básico. Brasília: Ministério da Educação, Secretaria de Educação Especial. 2005, p.26.

FERREIRA BRITO, L.: LANGEVIN, R. Por uma gramática de língua de sinais. Rio de Janeiro: Tempo Brasileiro,1995.

IBGE. Censo Demográfico 2010 – Características gerais da população, religião e pessoas com deficiência. Resultados da Amostra. IBGE, 2010. Disponível em: <ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/tab1_3.pdf>. Acesso em 14 de abril de 2015.
JUNIOR, Francisco de Assis Ribeiro. Programação Orientada a Eventos no lado do servidor utilizando Node.js. 2012.

INSTITUTO NACIONAL DE EDUCAÇÃO DE SURDOS (INES). Dicionário da língua brasileira de sinais. 2008. disponível em: <http://www.acessibilidadebrasil.org.br/libras/>. Acesso em: 30 de out. 2015.

JSON, Disponível em: < http://www.json.org/> Acesso em: 09/10/2015.

MAYA, Disponível em: < http://www.autodesk.com/education/free-software/maya /> Acesso em: 09/10/2015

NODE.JS, Disponível em: < nodejs.org/>. Acesso em: 02/10/2015.

PROATIVA, S. . N. ProDeaf. 2013. Disponível em: http://www.prodeaf.net/. 36

QUADROS, R.M.; KARNOPP, L.B. Língua de Sinais Brasileira: estudos lingüísticos. Porto Alegre: Artmed, 2004.

REVISTA DA FENEIS. Números 1 ao 13. R.J. 1999/2002.

STUMPF, M. R. Língua de Sinais: escrita dos surdos na Internet. V Congresso Ibero-Americano de Informática na Educação – RIBIE. Viñadelmar, Chile: [s.n.]. 2000. p. 1-8.

VIEIRA, Ana Lúcia dos Anjos. Acessibilidade de surdo na Televisão: surdo Assiste Tv. 2010. 51 p. Trabalho de Conclusão de Curso (Graduação) - Universidade Católica de Brasília, Brasília - DF, 2010.

3DS MAX, Disponível em: < http://www.autodesk.com/products/3ds-max/overview /> Acesso em: 09/10/2015.

SILVA, D. A. N. S. Uma Linguagem Expansível para Descrição da Língua Brasileira de Sinais**.** 69 f. Dissertação (Mestrado) - Departamento de Informática, Universidade Federal da Paraíba, João Pessoa, 2012.