Pós-Graduação em Ciência da Computação

JOSÉ THIAGO PEREIRA DA SILVA

UMA LINGUAGEM DE DOMÍNIO ESPECÍFICO PARA AUXILIAR O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM GESTOS DAS MÃOS

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

RECIFE

2017

JOSÉ THIAGO PEREIRA DA SILVA

UMA LINGUAGEM DE DOMÍNIO ESPECÍFICO PARA AUXILIAR O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM GESTOS DAS MÃOS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Andre Luis de Medeiros Santos

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S586l Silva, José Thiago Pereira da
Uma linguagem de domínio específico para auxilixar o desenvolvimento de aplicações baseadas em gestos das mãos / José Thiago Pereira da Silva. – 2017.
80 f.: il., fig., tab.

Orientador: André Luis de Medeiros Santos.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências.

1. Linguagem de programação. 2. Reconhecimento de gestos. I. Santos, André Luis de Medeiros (orientador). II. Título.

005.13 CDD (23. ed.) UFPE- MEI 2018-032

Jose Thiago Pereira da Silva

Uma Linguagem de Domínio Específico para Auxiliar o Desenvolvimento de Aplicações Baseadas em Gestos das Mãos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 28/08/2017

BANCA EXAMINADORA

_____

Prof. Dr. Geber Lisboa Ramalho
Centro de Informática / UFPE

_____

Prof. Wlater Franklin Marques Correia
Departamento de Design / UFPE

_____

Prof. Dr. André Luis de Medeiros Santos
Centro de Informática
(Orientador)

Dedico este trabalho a minha família e aos meus amigos e professores que foram essenciais para superar as dificuldades durante este percurso.

Agradecimentos

À minha esposa Drielly Evelin, pela paciência, cuidado e insistência para que eu pudesse fazer este trabalho.

À meu orientador, professor Dr. Andre Luis de Medeiros Santos, que me aceitou como aluno e me orientou durante este trabalho. Meus sinceros agradecimentos.

À professora Nancy Lyra responsável pelos estágios iniciais dessa jornada.

Aos meus amigos Oberdan Alves, Flávio Neves e Raisa Brito que foram grandes companheiros durante essa jornada.

Agradeço a todos os amigos que tive a oportunidade de conhecer no Centro de Informática da UFPE.

Agradeço também a todos os professores da UFPE pela competência com que transmitiram os conteúdos e ensinamentos.

À CAPES e ao CIn – UFPE pelo apoio financeiro para realização desta pesquisa.

A todos, os meus mais sinceros agradecimentos.

"Não é o muito saber que sacia e satisfaz a alma, mas o sentir e saborear internamente as coisas."

(Inácio de Loyola)

Resumo

Temos observado o crescimento da utilização da interação natural nos mais diversos ambientes computacionais devido a facilidade de aprendizado e naturalidade de interação oferecidas ao usuário. Dispositivos capazes de reconhecer os gestos humanos são utilizados juntamente com técnicas de reconhecimento de gestos para alcançar tal finalidade. O reconhecimento de gestos na maioria das vezes envolve aprendizagem de má-quina ou algoritmos avançados de reconhecimento adicionando mais uma complexidade no desenvolvimento de uma aplicação baseada em interação natural. O nosso trabalho tem por objetivo a construção de uma Linguagem Específica de Domínio (DSL) capaz de descrever os gestos das mãos, reduzindo consideravelmente os esforços no desenvolvi-mento. Para o desenvolvimento da DSL, foi utilizada a linguagem de programação C#, uma linguagem de propósito geral amplamente utilizada, a plataforma de desenvolvimento de jogos Unity3D e o Leap Motion, um sensor capaz de detectar movimentos das mãos e dedos e os utilizar como comandos de entrada. O principal objetivo desta dissertação é disponibilizar uma DSL que auxilie o desenvolvimento de aplicações que utilizem gestos naturais das mãos como interação. A proposta da DSL é tornar a especificação de gestos mais simples através de uma notação capaz de abstrair tais complexidades.

Palavras-chaves: Interação Humano-Computador. Interação Natural. Leap Motion. Reconhecimento de Gestos. Linguagens Específicas de Domínio.

Abstract

We have observed the growth of the use of natural interaction in the most diverse computational environments due to the ease of learning and the naturalness of interaction offered to the user. Devices capable of recognizing human gestures are used in conjunction with gesture recognition techniques to achieve this purpose. Gesture recognition most often involves machine learning or advanced recognition algorithms adding more complexity to the development of an application based on natural interaction. Our work aims to construct a Domain Specific Language (DSL) capable of describing hand gestures, considerably reducing development efforts. For the development of this DLS, we used the programming language C#, a widely used general purpose language, the Unity3D gaming development platform and Leap Motion, a sensor capable of detecting hand and finger movements and using them as input. The main objective of this work is to provide a DSL that helps the development of applications that use natural hand gestures as interaction. The aim of this DSL is to facilitate the specification of gestures through a notation capable of abstracting such complexities.

Key-words: Human–computer interaction. Natural Interaction. Leap Motion. Gesture recognition. Domain-Specific Language.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API *Application Programming Interface.*

AVC Acidente Vascular Cerebral.

BNF *Backus-Naur Form.*

DSL *Domain-Specific Language.*

FPS *Frames Per Second.*

HTML *HyperText Markup Language.*

LED *Light Emitting Diode.*

LPPG Linguagem de Programação de Propósito Geral.

NUI *Natural User Interface.*

SDK *Software Development Kit.*

SQL *Structured Query Language.*

WIMP *Windows, Icons, Menus, and Pointer.*

Sumário

1 INTRODUÇÃO

A diversidade de formas de interação entre homem e máquina proporcionada pela evolução dos dispositivos possibilita que os sistemas computacionais estejam presentes em diversas áreas de atuação. Isto é possibilitado pelo fato dos usuários ganharem novas possibilidades de mecânicas, antes não possíveis como o mouse, teclado e joystick, por exemplo.

Dentro desta esfera de possibilidades de interação a NUI ou Interface Natural do Usuário faz com que o próprio usuário, por meio de suas ações, seja o controle. Por descartar a utilização e aprendizado de dispositivos físicos de controle, esta interface tem ganhado popularidade e interesse dos usuários. A utilização desse paradigma tem sido explorada em diversos domínios como jogos (KIM et al., 2012), ambientes de ensino/aprendizagem, modelagens gráficas (KIM et al., 2005) e saúde (KHADEMI et al., 2014).

Dentre os subgrupos presentes nesse paradigma temos o *non-touch*, no qual estão os dispositivos e métodos que permitem aos usuários interagirem como o sistema sem que estes vistam algum dispositivo ou tenham de tocar em qualquer interface de hardware. Através de sensoriamento remoto, estes dispositivos são capazes de realizar o rastreamento do corpo do indivíduo em três dimensões, possibilitando a identificação, movimentação e gestos do usuário. Um exemplo deste tipo de dispositvo é o sensor *Leap Motion*.

O *Leap Motion* é um sensor capaz de identificar pontos das mãos e dos dedos captando movimentos como entrada, permitindo que gestos realizados pelos usuários sejam identificados (Figura 1). Este dispositivo possui um baixo custo e se adequa aos domínios que necessitem apenas das mãos como objeto a ser trabalhado. Apesar das constantes evoluções da *Application Programming Interface* (API)[1] ou Interface de Programação de Aplicativos, os dados ainda continuam num baixo nível de abstração, exigindo assim, que os desenvolvedores empreguem esforços na construção de algoritmos que identifiquem configurações de mão e gestos realizados pelo usuário a partir dos dados obtidos.

O presente trabalho deter-se-á na construção de abstrações de modelos de representação de configurações e reconhecimento de gestos das mãos, através de uma linguagem que seja capaz de especificá-los e reconhecê-los.

1.1 Motivação

Como destacado anteriormente há uma vasta quantidade de aplicações, das mais diversas categorias, desenvolvidas tanto para fins recreativos quanto para domínios mais específicos como, por exemplo, a reabilitação. Essas aplicações se beneficiam do fato de que a

[1] Em programação de computadores, uma Interface de Programação de Aplicação (Application Programming Interface - API) é um conjunto de definições de sub-rotinas, protocolos e ferramentas para construção de aplicações de *softwares*.

interação natural faz mais sentido, já que os gestos utilizados na aplicação são análogos (total ou parcialmente) aos executados no mundo real pelo usuário.

Para que haja comunicação entre homem e computador é necessário que o computador perceba as informações enviadas pelo usuário. Ao receber estas informações o computador irá interpretá-la e de acordo com o que estiver programado, realizará alguma tarefa em resposta ao comando. A evolução da computação possibilitou o surgimento de vários domínios de aplicações e consigo a necessidade de interagir de maneira objetiva, fácil, rápida e confortável. Temos a exemplo disto a constante evolução dos *joypads* (dispositivos de entradas para vídeo games), que têm sofrido mudanças constantes em seus formatos e no número de funções a fim de se adequarem aos comandos exigidos pelos jogos. Posteriormente controles que utilizam mais movimentos corporais dos usuários vão surgindo e trazendo consigo novas possibilidades, podemos citar aqui o *Wii Remote* ou *Wiimote* e o *PSMove*. Estes dispositivos que são utilizados anexados ao corpo do usuário (O usuário o segura com as mãos enquanto o utiliza e executa os gestos), realizam o rastreamento através de giroscópios e acelerômetros contidos nestes dispositivos. Apesar de ganharmos mais liberdade para a utilização de gestos mais naturais, a utilização de dispositivos anexos ainda causam alguns desconfortos, pois vão contra o conceito de liberdade em interação natural. Ademais os usuários precisam dividir a atenção e manter constante cuidado com estes dispositivos para controlá-los e não arremessá-los enquanto o utilizam.

A alternativa para capturar os movimentos do usuário sem que este método necessite ter algum dispositivo anexado ao seu corpo é a utilização do sensoriamento remoto, no qual um ou mais dispositivos não anexos são responsáveis por capturar as informações (ações realizadas) do usuário. As câmeras digitais comuns (chamadas tecnicamente de câmeras monoculares) podem ser utilizadas como dispositivos de rastreamento remoto, porém estes dispositivos precisam de um alto poder de processamento devido ao tempo de processamento que alguns algoritmos exigem, além das características singulares (dimensões do corpo que estão sendo rastreadas) de cada usuário que dificultam o reconhecimento. Há também outros sensores mais robustos que são capazes de recuperar informações de profundidade acerca do ambiente. Entre os dispositivos de sensoriamento remoto existentes, destacamos o *Leap Motion* por se tratar de um sensor que tem como proposta o rastreamento específico das mãos e dos dedos, o que não é naturalmente possível com outros sensores como *Kinect* e *EyeToy*. A utilização desses dispositivos possuem como vantagens em relação as câmeras monoculares as informações extras que elas disponibilizam, tais como um mapa das diferentes partes do corpo e seu posicionamento em três dimensões, o que proporciona maior precisão e facilidade no processo de reconhecimento de gestos.

Apesar dos sensores juntamente com suas APIs apresentarem informações extremamente importantes para a detecção de gestos, estas ainda não são suficientes para que o dispositivo reconheça um conjunto específico de gestos que faça sentido para uma determinada aplicação, por exemplo. Este processo de reconhecimento de gestos deve ser

realizado pelo desenvolvedor através de uma técnica de reconhecimento de gestos para que sua aplicação seja capaz de interpretar os gestos realizados pelo usuário da aplicação, contudo tais técnicas apresentam algumas dificuldades.

As abordagens mais utilizadas para construção de reconhecedores de gestos são: abordagem baseada em comparação e abordagem baseada em regras. A abordagem baseada em comparação é baseada nas técnicas de aprendizagem de máquina. Seu principal objetivo é "ensinar"o computador para que "aprenda"regras que o permita reconhecer gestos. Esta abordagem consiste no armazanamento de sequências de imagens de um determinado gesto realizado várias vezes por diferentes pessoas. Estes dados armazenados e classificados são posteriormente comparados com as imagens obtidas pelo dispositivo. No momento em que acontece uma combinação entre os dados obtidos com os armazenados, é um indicativo de que o gesto foi reconhecido. Porém, esta aborgem apresenta algumas desvantagens. A primeira desvantagem é que a etapa de "treinamento"implica em custos adicionais ao desenvolvimento, pois é necessário que um bom número de pessoas execute os gestos por algumas vezes, pois quanto maior a base de dados melhor será o reconhecimento. Todavia, uma base de dados muito grande torna o processamento mais custoso e demorado. Além disto, o computador utilizado precisa ter uma certa robustez para que seu poder de processameto não afete a velocidade na etapa de reconhecimento.

A outra abordagem, também bastante comum, é a abordagem baseada em regras. Nesta abordagem o desenvolvedor analisa os gestos pretendidos pela aplicação e então estabelece regras por meio de algoritmos. No algoritmo estão contidas as etapas que um determinado gesto deverá cumprir para ser detectado. Esta abordagem também possui alguns inconvenientes. Uma das maiores dificuldades é a complexidade de analisar o algoritmo resultante. Os dados disponibilizados pelas APIs dos dispositivos fornecem, em geral, dados em baixo nível. O próprio conhecimento a respeito da API se torna um barreira para o desenvolvedor, que precisará aprender sobre os detalhes, muitas vezes em baixo nível, obtidos pelo sensor. Além disto também será necessário noções de cálculos matemáticos complicados para o reconhecimentos de gestos mais complexos.

1.2 Objetivos

Este trabalho tem como principal objetivo oferecer uma abordagem que facilite a especificação computacional de gestos das mãos para dispositivos de profundidade de modo que reduza a complexidade do desenvolvimento de aplicações que utilizam gestos como forma de interação.

Para atingir tal objetivo as seguintes atividades foram executadas:

- Estudo sobre os modelos de representação do movimento humano, com o propósito de especificar um modelo computacional capaz de descrever os gestos das mãos.

- Apresentar os requisitos e as técnicas, de acordo com o estado da arte, para reconhecimento de gestos.
- Definição de uma linguagem *Domain-Specific Language* (DSL)[2] tendo em vista os modelos de representação já existentes, acrescida de uma arquitetura de reconhecimento para dispositivos de reconhecimento.
- Avaliação da solução proposta, através do reconhecimento e especificação de gestos.

1.3 Organização da dissertação

Esta dissertação, além deste capítulo de introdução, é composta pelos capítulos descritos a seguir:

- O capítulo 2 trata a respeito da utilização de gestos naturais na computação. Primeiramente é demonstrado o início das pesquisas envolvendo o paradigma de interação natural. Em seguida, é discutido a utilização de gestos realizados com as mãos. Posteriormente é tratado os dispositivos utilizados para captura dos movimentos bem como as técnicas utilizadas para a construção de reconhecedores de gestos.
- O capítulo 3 trata das abstrações existentes para descrição gestos. A princípio é exposto o conceito das DSLs e suas principais características. Em seguida é apresentado as tentativas de representação do movimento humano por diversas áreas até chegar na computação, área que se encontra este trabalho.
- O capítulo 4 aborda a solução proposta por este trabalho. É descrito em detalhes o funcionamento, arquitetura e as etapas da implementação da linguagem (DSL) desenvolvida para construção de reconhecedores de gestos. É apresentado também seus recursos juntamente com alguns exemplos de uso.
- O capítulo 5 trata da avaliação da utilização da DSL proposta a fim de avaliar sua eficiência para construção de reconhecedores de gestos.
- O capítulo 6 apresenta as considerações finais deste trabalho e as sugestões de trabalhos futuros.

[2] Uma DSL (em português, Linguagem Específica de Domínio) é uma linguagem de programação com um elevado nível de abstração, cujo objetivo é modelar e especificar os conceitos de um dado domínio.

2 INTERAÇÃO NATURAL E RASTREAMENTO DAS MÃOS

Neste capítulo, é descrito a utilização da interação natural na computação. É realizado um pequeno apanhado cronológico da aparição e evolução de tecnologias cujo propósito é o favorecimento deste modo de interação. A interação através dos movimentos das mãos é detalhado de forma particular, apresentando sua importância a partir dos domínios em que se encontra sua utilização. O capítulo exibe também uma série de dispositivos que são atualmente utilizados para fins de capturar as mãos de um indivíduo numa determinada cena. Por fim, o capítulo encerra discutindo as abordagens utilizadas para reconhecimento de gestos, abordagem baseada em regras e a abordagem baseada em comparação.

2.1 Utilização de gestos naturais como modelo de interação

A utilização de interação baseada em gestos não é uma novidade em computação. (MYERS, 1998) em sua revisão, realizada em 1998, já descreve trabalhos realizados na década de 60. Em 1963 foi demonstrado o software de edição gráfica *Sketchpad* (JOHNSON, 1963), criado por Ivan Sutherland. O Sketchpad já utilizava a interação baseada em gestos através de uma caneta óptica[1]. Em 1980 Bolt (1980) apresentava uma das primeiras interfaces a utilizar a combinação de gestos, da fala e do movimento dos olhos, o *Media Room*. Com o *Media Room* o usuário poderia interagir com os objetos projetados a sua frente, pelo computador, através da fala, gestos e movimento dos olhos ou a combinação de ambos. Em 1982 temos o trabalho da tese de mestrado de Mehta (1982), um dos primeiros sistemas *multi-touch*[2] projetado para interface humano-computador (BUXTON et al., 2007). Sensores especializados em detecção da localização e dos movimentos humanos já vêm sendo pesquisado há algum tempo. Na figura 1 temos exemplos da utilização do uso de gestos para interação com computador.

Com o surgimento de novos dispositivos de interação comerciais capazes de capturar os movimentos do corpo (*Kinect*, *Leap Motion*, *Wii Remote*, *MYO*), mais aplicações com interação baseada em gestos foram criadas. Graças ao aumento da oferta de dispositivos de captura de movimentos comerciais o acesso a esses dispositivos por parte dos usuários aumentaram e desta forma, abriu-se uma janela de oportunidades para a criação de novas aplicações baseadas em gestos. Essa nova demanda de aplicações trouxe à tona os

[1] Uma caneta óptica é um dispositivo de entrada de computador no formato de um bastão sensível à luz usado em conjunto com um monitor CRT.

[2] Em computação, 'multi-toque' refere-se a uma superfície de detecção de toque (*trackball* ou *touchscreen*) a capacidade de reconhecer a presença de dois ou mais pontos de contato com a superfície.

Figura 1 – Exemplos de utilização de gestos naturais para interação com computadores

problemas intrínsecos a interação baseada em gestos. Ademais, com a crescente demanda de aplicações baseadas em gestos, surgiu nos desenvolvedores a necessidade de dinamizar cada vez mais suas aplicações, procurando formas de interação mais criativas e com isso a exploração de novos gestos.

A utilização de gestos na interação entre homens e computadores traz consigo problemas inerentes. De acordo com o domínio da aplicação, os usuários precisam obter um *feedback* quase imediato (domínio de jogos), o que traz ao desenvolvedor a necessidade de criar gestos mais simples, que possam ser assimilados e memorizados de forma rápida pelo usuário final. Estes gestos precisam ser detectados o mais rápido possível, para evitar um atraso entre um gesto realizado perfeitamente pelo usuário e a resposta do sistema. O sistema precisa ainda ser tolerante a possíveis erros dos usuários, não permitindo falsos negativos, nem falsos positivos. Estas característica garantem uma fluidez do sistema e evitam frustrações por parte dos usuários. Em outros domínios, como aplicações de saúde e educação, os problemas ganham maiores proporções quando o leque de gestos aumentam em número e complexidade. Um sistema com com muitos gestos exige dos desenvolvedores um maior tempo no desenvolvimento de reconhecedores de gestos, o que nem sempre é uma tarefa fácil. Tomando por exemplo o domínio de linguagem de sinais, onde há uma variedade muito grande de configuração de mãos, os reconhecedores de gestos precisam ser ainda mais sofisticados para lidar com gestos que possam parecer ambíguos, em um curto espaço de tempo. Estes problemas exigem dos desenvolvedores técnicas sofisticadas para construção de reconhecedores de gestos.

2.2 Utilização das mãos no paradigma NUI

Uma revisão feita por Murthy e Jadon (2009) apresenta alguns domínios de aplicação que utilizam os gestos das mãos como interação. É destacado a seguir alguns desses trabalhos, de acordo com o domínio da aplicação.

Realidade virtual e aumentada: é uma das áreas com mais pesquisas. Diversas aplicações têm sido criadas, tanto em três dimensões quanto em duas. Estas aplicações possibilitam a manipulação de objetos virtuais utilizando as mãos (SHARMA et al., 1996; STARNER et al., 2000). Esta interface é ainda utilizada em conjunto com outros domínios, como por exemplo a reabilitação fisica (MERIANS et al., 2002).

Róbotica e telepresença: este domínio é muito utilizado para fins de exploração espacial e pesquisas militares. Os gestos utilizados para interagir com os robôs são semelhantes aos realizados em realidade virtual, no qual os indivíduos se encontram totalmente imersos, contudo o mundo em que o usuário interage geralmente é real e as imagens do mundo real são apresentadas através de uma camera localizada no robô (GOZA et al., 2004).

Aplicações desktop e tablet: em aplicações *desktop* os gestos podem ser utilizados em alternativa aos dipositivos comumente utilizados, como *mouse* e *teclado* (STOTTS; SMITH; GYLLSTROM, 2004). Diversos gestos para tarefas de aplicações *desktop* envolvem a manipulação de gráficos e edição de documentos através do uso de gestos baseados em caneta (SMITH et al., 2004).

Jogos: Neste domínio temos presente várias aplicações comerciais e não comerciais. As aplicações comerciais passaram a surgir em maior quantidade devido a aparição dos dispositivos de rastreamento comerciais, tornando estas aplicações mais acessiveis aos usuários. Em *MIND-WARPING*, um jogo de luta em realidade aumentada, os jogadores interagem com oponentes virtuais através da detecção de gestos realizados com as mãos (STARNER et al., 2000). De forma semelhante, *QuiQui's*, um jogo de ação infantil, também baseado em gestos das mãos, utiliza uma camera *USB*[3] de baixo custo para detectar os movimentos do jogador (HÖYSNIEMI et al., 2005).

Língua de sinais: A linguagem de sinais é uma língua que utiliza de gestos e expressões faciais, em vez de sons para se comunicar. Trata-se de um caso importante de gestos comunicativos. As mãos são bastante utilizadas na execução do gestos, durante o processo de comunicação. Por ser altamente estruturada ela oferece uma boa base de gestos para serem utilizados, ao mesmo passo que podem ser uma boa forma de auxiliar na interação entre pessoas com deficiência auditiva e os computadores. A língua de sinais vêm recebendo grande atenção de trabalhos que lidam com gestos (KAK, 2002; VOGLER; METAXAS, 2001; STARNER; WEAVER; PENTLAND, 1998).

Aplicações médicas e assistivas: A interação a partir de gestos também é utilizado em áreas como a saúde. Os gestos podem ser utilizados para controlar equipamentos

[3] *Universal Serial Bus* (USB) é um tipo de conexão "ligar e usar"que permite a fácil conexão de periféricos sem a necessidade de desligar o computador.

médicos, assim como auxiliar na reabilitação de pessoas com deficiência (KHADEMI et al., 2014; WACHS et al., 2008). Wachs et al. (2008) desenvolveu uma interface gestual chamada *Gestix*, um sistema que permite ao cirugião manipular imagens em uma sala de cirugia, em tempo real e de forma estéril, através das mãos, tornando a tarefa mais intuitiva e mais rápida. Khademi et al. (2014) propõe a utilização de um jogo, onde os usuários praticam a dissociação dos dedos, auxiliando na reabilitação de pacientes com Acidente Vascular Cerebral (AVC)[4].

2.3 Rastreamento de gestos humanos

Para que os movimentos corporais possam ser utilizados como forma de interação entre o homem e o computador é necessário que antes esses movimentos sejam reconhecidos pelo computador. Há um campo bastante estudado na área de interação natural, chamado Reconhecimento de Gestos, que se propõe a aperfeiçoar as formas de interpretar os movimentos corporais. A área de Reconhecimento de Gestos refere-se a todo o processo necessário para converter um gesto humano (movimento corporal) em comandos semanticamente relevantes (RAUTARAY; AGRAWAL, 2015).

A importância do estudo de Reconhecimento de Gestos é de fundamental importância para a área de interação natural. Há trabalhos na área de rastreamento do corpo humano datando o início da década de 1980 (AGGARWAL; CAI, 1997). Deste então, os métodos para captar o corpo humano tem variado de muitos modos. O rastreamento de gestos humanos permite capturar partes específicas, como expressões faciais (LI et al., 2013; GRAFSGAARD et al., 2013), dos olhos (HOLMQVIST et al., 2011; DUCHOWSKI, 2007), movimentos das mãos e dos dedos (RAUTARAY; AGRAWAL, 2015). Como é observado, os esforços para rastrear o corpo humano têm se dividido a partir de partes específicas do corpo. Isto significa dizer que as técnicas empregadas para rastrear as mãos diferem das técnicas utilizadas para rastrear os olhos.

Uma das formas de rastrear o corpo (ou parte do corpo) é através da utilização de dispositivos ou acessórios anexados ao corpo. Estes dispositivos estão em constante contato com o usuário e é através deles que os dados acerca dos movimentos realizados são obtidos. Há um bom número de dispositivos para tal fim. Existe por exemplo, dispositivos com sensores inerciais, capazes de medir deslocamento e rotação, como acelerômetros e giroscópios (LI et al., 2009). De outro modo, existem acessórios que utilizam suas cores ou mesmo a emissão de luzes para fornecer os dados acerca da parte do corpo que está sendo rastreada (WANG; POPOVIĆ, 2009; LEE, 2008). Estes dispositivos podem ainda ser classificados como, mecânicos, hápticos, ultra-sônicos, inerciais e magnéticos (KAÂNICHE, 2009)

[4] Um acidente vascular cerebral (AVC) ocorre quando problemas na irrigação sanguínea cérebro causam a morte das células, o que faz com que partes do cérebro deixem de funcionar devidamente.

Por causa da restrição que os dispositivos anexos oferecem aos usuários, devido ao incômodo ou mesmo a necessidade de uma experiência prévia na sua utilização, as técnicas que permitam ao usuário se movimentar livremente têm sido empregadas para capturar os movimentos realizados. As técnicas para tal finalidade utilizam uma ou mais câmeras que capturam os movimentos a partir de uma sequência de vídeo para uma posterior análise e interpretação (MITRA; ACHARYA, 2007). Sensores de profunidade também vêm sendo utilizados mais recentemente (MICROSOFT, 2017).

2.3.1 Dispositivos de rastreamento anexo

Um dos modos de capturar os movimentos corporais é através da utilização de dispositivos anexos ao corpo. Estes dispositivos na maioria das vezes estão presos ao corpo do usuário ou encontram-se segurados pelas mãos. Além disso, são capazes de fornecer informações a respeito do corpo do usuário, como localização e orientação utilizando para tanto, pistas visuais ou sensores inerciais. Essas informações são enviadas a uma central de processamento onde será realizada a intepretação e reconhecimento dos movimentos. Esta comunicação (transmissão) entre dispositivo e central de processamento pode ocorrer através de fios ou de tecnologias sem fio. Uma das características destes dispositivos é a alta precisão dos dados capturados alinhado a alta taxa de atualização.

Há dispositivos que capturam os movimentos do corpo através de seu *Hardware*, componentes sensíveis com circuitos específicos responsáveis por obter os dados acerca dos movimentos do usuário (SYSTEMS, 2017) (HQ, 2016). A outra abordagem para estes acessórios anexos é por meio do uso de pistas visuais detectáveis por câmeras comuns ou infravermelhas. Em tal caso, os sensores anexos ao corpo são substituídos por sinais visuais que facilitam o processo de rastreamento do corpo. Nesta abordagem, existem desde roupas com luzes infravermelhas (RASKAR et al., 2007), desde luvas coloridas (WANG; POPOVIĆ, 2009).

Há ainda a utilização de uma abordagem mista, como é possível verificar nos dispositivos comerciais *WiiRemote* (NINTENDO, 2017) (ver Figura 2-D) e *PlayStation Move* (SONY, 2017) (ver Figura 2-B). Estes dois dispositivos funcionam de modo semelhante. Ambos, possuem um segundo dispositivo fixo, encarregado de capturar os sinais enviados pelo *Joystick*. O *Joystick WiiRemote* é equipado com sensores de movimento, tal como acelerômetro, giroscópio e câmera infravermelha (LEE, 2008). Sua câmera capta apenas sinais infravermelhos num baixo tempo de resposta com alta precisão. De forma semelhante, o *Joystick PlayStation Move* faz uso de sinais luminosos, sensores inerciais e câmera. O *PlayStation Move* possui uma esfera em sua ponta que contém três *Light Emitting Diode* (LED)s capazes de reproduzir vinte e quatro bits de cores RGB, desta forma, a cor escolhida é aquela que mais se destaca no ambiente, dentro do espaço visto pela câmera (SONY, 2017).

Figura 2 –

(A) M.Y.O. (THALMICLABS, 2016) (B) PlayStation Move (SONY, 2017) (C) Cyber Glove III (SYSTEMS, 2017) (D) WiiRemote (NINTENDO, 2017).

2.3.2 Dispositivos de rastreamento livre

Para fazer uso de um dispositivo anexo, o usuário precisa estar segurando um dispositivo ou de alguma forma tê-lo junto ao seu corpo. Embora sua utilização associada a algum *Videogame* seja comum, isto vai de encontro com o conceito de liberdade em interação natural. Com isto, a utilização de dispositivos anexos ocasiona alguns efeitos indesejáveis ao usuário. É necessário que o usuário dedique constantemente sua atenção para segurar o aparelho, se o aponta na direção correta e até preocupar-se em não soltar o dispositivo, causando uma série de danos. Ainda a depender do dispositivo, seu peso pode gerar um desconforto ao usuário. Mesmo que estes dispositivos tenham sido bem projetados, ainda assim, poderão limitar as ações do usuário durante a interação, desde o aspecto cognitivo quando o físico.

Existem alternativas que dispensam o emprego de acessórios adicionais ligados ao usuário. As técnicas de rastreamento para este fim, utilizam câmeras comuns, chamadas tecnicamente de câmeras monoculares, comumente encontradas em computadores portateis ou dispositivos móveis, ou sensores de profunidade, sensores capazes de capturar

informações de profundidade em relação ao ambiente.

Câmeras monoculares

Uma das formas de capturar os movimentos é a partir de uma sequência de imagens obtidas por uma câmera. O conjunto de imagens funciona como um fluxo de entrada de dados, para que os dados do tal conjunto de imagens seja processado por um algoritmo baseado em técnicas de reconhecimento de gestos, para que, de forma rápida, possa interpretar e reconhecer os gestos realizados pelo usuário. A complexidade desta tarefa se dá pela extração de informações das imagens obtidas, para que sejam interpretadas à um modelo pré-determinado (VATAVU; PENTIUC, 2012). As técnicas de rastreamento com câmeras monoculares podem apresentar problemas adicionais. Problemas como variação de iluminação no ambiente, atrasos no tempo de resposta devido ao tempo de processamento dos algoritmos, oclusão, movimentação do usuário ocasionando borramento na imagem e outros. Ademais, a própria singularidade do corpo de cada indivíduo traz problemas extras ao rastreamento. Ainda que a anatomia do corpo humano seja bem conhecida e respeite padrões de proporcionalidade, não é possivel rastreá-lo com técnicas usuais (LIMA et al., 2010).

A nova geração de sensores de movimento, como *Kinect*; *Leap Motion*, trazem consigo a capacidade de capturar informações extras do ambiente, tornando possível obter informações do usuário em três dimensões. De outra maneira, trazendo mais informações a respeito do ambiente, diminuindo asssim, o trabalho posterior de interpretar e reconhecer os gestos. Portanto, estes novos dispositivos visam facilitar o reconhecimento e melhorar a qualida de captura.

Sensores de profundidade

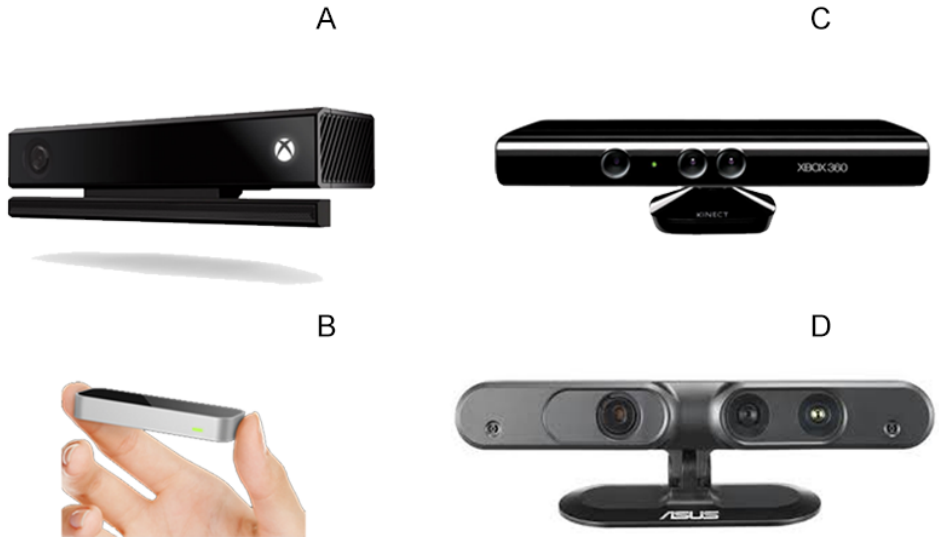
Os sensores de profunidade são dispositivos capazes de precisar a distância entre os objetos na cena capturada. Isto torna-se possível graças a sua capacidade de determinar a posição no eixo z para cada pixel da cena. Estes sensores, em geral, possuem um conjunto de emissores (projetor infravermelho) e receptores (câmera infravermelho) de iluminação infravermelho. Através da projeção de um padrão de luz infravermelha associado a captura por uma câmera, também, infravermelho, os sensores permitem identificar e rastrear o usuário quando este se posiciona em frente ao sensor. Assim, é possível obter a posição em três dimensões do mesmo, destacando-o em relação ao resto do ambiente.

Dispositivos como o *Kinect* (MICROSOFT, 2017), *Xtion PRO* e *Xtion PRO LIVE* (ASUS, 2017), foram desenvolvidos para rastrear movimentos de corpo inteiro. Ambos os dispositivos utilizam uma padrão de luz infravermelho, para obtenção do mapa de profundidade. Enquanto o *Xtion PRO* é mais leve e é alimentado por uma entrada *USB*, o *Kinect* necessita de uma fonte de alimentação extra, porém possui uma maior qualidade nas imagens capturadas que o *Xtion PRO* (SONG; LICHTENBERG; XIAO, 2015). O *Kinect*

V2, segunda versão do dispositivo (MICROSOFT, 2017), é baseado no princípio *Time-of-flight* (TOF), onde luzes infravermelho são enviadas tendo o atraso entre sua emissão e reflexão calculados, possibilitando assim, obter informações tridimensionais do ambiente (AMON; FUHRMANN; GRAF, 2014). As imagens de profundidade capturadas são mais precisas, com alta taxa de fidelidade, contudo tem dificuldades ao se deparar com objetos pretos e superficies ligeiramente reflexivas (SONG; LICHTENBERG; XIAO, 2015). A Figura 3 ilustra alguns dos principais modelos do mercado.

Figura 3 –

(A) Kinect v2 (MICROSOFT, 2017) (B) Leap Motion (LEAPMOTION, 2017b) (C) Kinect v1 (MICROSOFT, 2017) (D) Asus Xtion pro Asus (2017)



Diferente dos dipositivos mencionados anteriormente, há outros dipositivos que se propõem a detectar partes específicas do corpo, como mãos e dedos. Devido ao foco deste trabalho se concentrar em gestos realizados por mãos e dedos, foi escolhido um dipositivo comercial desenvolvido para este fim.

Dispositivo escolhido

Os dispositivos expostos até aqui, adéquam-se bem as tarefas para o qual foram desenvolvidos mesmo considerando suas limitações. A escolha por um sensor de profundidade, foi realizada graças as informações extras que estes dispositivos são capazes de fornecer em relação, por exemplo, as câmeras de vídeo (monoculares). Desta, maneira, a tarefa de reconhecer gestos mais complexos em três dimensões torna-se mais fácil.

No foco destre trabalho está presente o rastreamento das mãos, portanto o dispositivo *Leap Motion* possui as características desejáveis. Até o presente momento, o *Leap Motion* é o dispositivo mais acessado, possui um custo acessível, integração com diversas linguagens de programação e *Engines*[5], e conta com constantes atualizações em sua API

[5] São *softwares* e/ou conjunto de bibliotecas, para simplificar e abstrair o desenvolvimento de jogos eletrônicos.

gratuitos. Além do que, sua utilização conjunta com *Head-Mounted Display*[6] vêm promovendo um aumento no número de aplicações de realidade virtual desenvolvidas. Logo, estas propriedades encontradas no *Leap Motion* colaboram para sua escolha. Na seção seguinte, o dispositivo será discutido em maiores detalhes.

Leap Motion

Em meados de 2008, David Holz, durante seu doutorado, deu início ao desenvolvimento do que viria a ser o dispositivo Leap Motion. Após o financiamento de um investidor-anjo, David cria, em parceria com seu amigo Michael Buckwald, a empresa no ano de 2010. Em 2012 a empresa lança um programa de desenvolvimento de *software* para o dispositivo, revelando seu primeiro produto chamado *Leap*, oferecendo ferramentas de suporte ao desenvolvimento para os desenvolvedores. Seguidamente em 2014, a empresa lança a segunda versão do dispositivo. A segunda versão do dispositivo passou a contar com suporte para realidade virtual. Com esta versão, já era possível que as mãos dos usuários fossem rastreadas enquanto o dispositivo estivesse anexado a um *Head-Mounted Display*. Posteriormente em 2016, a empresa anunciou o seu novo *sofwtare*, chamado *Orion*, criado especificamente para *realidade virtual*, a fim de torna-lo mais rápido, leve e mais confiável enquanto trabalha com aplicações desta natureza.

O dispositivo *Leap Motion* apresenta algumas limitações, como a limitação do espaço sensorial e a variação das amostras obtidos, levando a uma inconsistência dos dados de posicionamento das mãos e dos dedos (GUNA et al., 2014). Apesar disto, o dispositvo se mostra um avanço no rastreamento de mãos e dedos, possibilitando por exemplo, obter posição e direção de cada falange da mão, o que não seria trivial ao utilizar câmeras de vídeo digital.

Especificações

O dispositivo *Leap Motion* é um sensor de profunidade comercial, desenvolvido pela empresa Leap. O dispositivo faz parte da categoria dos controladores *in-air motion*, isto é, a capacidade de utilizar as mãos para interagir com objetos virtuais sem a necessidade de vestir ou segurar algum periférico (DO, 2016). O sensor é um pequeno periférico *USB* (aproximadamente do tamanho de uma caixa de fósforo) projetado para ser utilizado junto a um computador *desktop* ou associado a um *headset* de realidade virtual. O *Leap Motion* foi criado primeiramente para rastrear o movimento das mãos e dos dedos com precisão sub-milimétrica e fluidez (GUNA et al., 2014).

Para realizar o rastreamento, o dispositivo conta com um conjunto de duas câmeras infravermelho monocromáticas e três LEDs emissores infravermelho. O dispositivo apresenta uma capacidade de rastrear mãos e dedos (orientação, posição, velocidade e etc.) a

[6] Um head-mounted display (ou helmet-mounted display, para aplicações na aviação), ambos abreviados HMD, é um dispositivo de display, usado na cabeça ou como parte de um capacete, que possui um pequeno display óptico em frente de um (HMD Monocular) ou de cada olho (HMD Binocular).

uma taxa de amostragem que pode variar de 30 *Frames Per Second* (FPS)[7] a 115 FPS. As imagens capturadas são enviados para um computador através de uma conexão *USB* para então serem avaliados pelo *software* de rastreamento *Leap Motion*. O software *Leap Motion* processa as imagens para gerar uma representação em três dimensões do que é visualizado pelo dispositivo. A "reconstrução"tridimensional é alcançada por meio de cálculos complexos realizados por algoritmos avançados. Devido a questões de patente, não é possível obter maiores detalhes a respeito do software que os disponíveis pela API. O campo de visão do dispositivo é de aproximadamente 150 graus numa distância que compreende aproximadamente de 25 a 600 milímetros(0,6 metros), assumindo uma forma semelhante a uma pirâmide invertida. Com o lançamento do *software Orion* a visão foi expandida para aproximadamente 0,8 metros (COLGAN, 2014).

Uma das limitações da utilização do *Leap Motion* é devido a interferência causada pela iluminação natural. A incidência de luz solar no ambiente, através de espaços abertos, como janelas, pode prejudicar a caputra dos movimentos, já que a luz infravermelho emitida pelo sol se sobrepõe a luz infravermelho projetada pelo *Leap Motion* afetando negativamente o rastreamento.

Apesar das limitações que envolvem o dispositivo, sua utilização ainda é interessante devido ao seu apelo comercial (baixo custo, número de aplicações já desenvolvidas, número de unidades vendidas e atualizações recorrentes no projeto do dispositvo, *software* e *hardware*. A comunidade tem se esforçado para aperfeiçoar a sua utilização, tanto por meio de construção de ferramentas que visam facilitar o desenvolvimento de aplicações para o dispositivo, quanto pela construção de novas aplicações, estimulando que a empresa mantenha o aperfeiçoamento de seu dispositivo.

O *Leap Motion* possui uma loja virtual que disponibiliza, para usuários do dispositivo, uma série de aplicativos de diversos domínios que vão desde aplicativos educacionais a jogos (LEAPMOTION, 2017a). Há também um bom número de aplicações sendo desenvolvidas em pesquisas científicas, como nos domínios de reabilitação motora (KHADEMI et al., 2014), jogos educacionais (LIPP; MOSSMANN; BEZ, 2015), jogos e realidade virtual (LEE et al., 2015), intervenção em terapia ocupacional (COSTA, 2014) (SOUSA; SILVEIRA, 2016), tecnologias assistiva (BASSILY et al., 2014). A variedade de aplicações e domínios envolvidos mostra o potencial do dispositivo e interesse da comunidade, mesmo tendo em conta suas limitações técnicas.

2.4 Reconhecimento de gestos com *Leap Motion*

O sensor de movimento *Leap Motion* possibilita aos usuários interagir com um computador através das mãos e dos dedos. Isto torna o dispositivo *Leap Motion* uma poderosa

[7] *Frames Per Second* (ou quadros por segundo), é o número de imagens que um dispositivo audiovisual registra, processa ou exibe por segundo.

ferramenta de comunicação entre homem e computador. Os desenvolvedores têm buscado maneiras de criar reconhecedores de gestos que sejam robustos, computacionalmente eficientes, tolerante a erros do usuário e escalável (MURTHY; JADON, 2009). Basicamente há duas abordagens para construção de reconhecedores de gestos: baseada em regras ou baseada em aprendizagem de máquina. As duas abordagens utilizam o fluxo de dados fornecido pela API do *Leap Motion*, que permite o acesso as informações capturadas e processadas pelo sensor.

Cada abordagem possui suas vantagens e desvantagens. A seguir, será discutido em maiores detalhes, como trabalha cada uma dessas abordagens bem como as consequências provocadas.

2.4.1 Abordagem baseada em comparação (*Machine Learning*)

A abordagem baseada em aprendizagem de máquina consiste basicamente em classificar o gesto realizado pelo usuário de acordo com uma base de dados (*training set*) construída previamente. Esta base de dados é composta por vídeos de gestos realizados por pessoas. Através de um algoritmo de aprendizado de máquina, a sequência de dados recebida através do sensor é comparada com as imagens gravadas. Caso seja encontrado um padrão correspondente, o gesto será reconhecido.

O primeiro passo desta abordagem é a construção de uma base de dados contendo exemplos dos gestos a ser especificados. No processo de construção, os gestos são repetidos várias vezes por diversos usuários. Mesmo um único gesto, será executado de forma particular por cada pessoa (um movimento mais acentuado no início ou no fim do gesto, omissão de alguma características, mais rápido ou mais lento e etc.). Apesar disto, querem realizar o mesmo gesto e o computador a partir de uma ampla base de dados poderá encontrar um padrão que consiga reconhecer a intenção dos usuários ao realizar um determinado movimento. Esta abordagem busca "ensinar"ao computador o padrão de movimento realizado pelo gesto a ser reconhecido. O tamanho (número de repetições de um mesmo gesto) e a diversidade (diferentes usuários) da base de dados garante maior eficiência na etapa de classificação do gestos.

Implementar um sistema baseado nesta abordagem não é uma tarefa simples. Para cada gesto contido no sistema, será preciso capturar vários exemplos, por diferentes pessoas e executados de modo distinto, acarretando num maior gasto de tempo e de recursos. Isto cresce ainda mais na medida em que o número de gestos e sua complexidade aumentam. O aumento do número dos gestos provoca também um maior consumo dos recursos computacionais. Quanto mais gestos o sistema possui, maior será a base de dados. Isto resulta também num maior ciclo de processamento para procurar uma combinação correspondente.

Ao adotar a abordagem baseada em aprendizagem de máquina, o desenvolvedor precisa estar bem familiarizado com a técnica utilizada a fim de realizar de forma adequada os

ajustes convenientes durante a etapa de treinamento. O desenvolvedor precisa também estar familizarizado com a API do sensor, pois as informações conseguidas através dela são utilizadas como parâmetro no treinamento. Ademais, adições de novos gestos e ajustes torna-se difícil devido também ao código final resultante que não é trivial de ser analisado (FIGUEIREDO et al., 2016).

2.4.2 Abordagem baseada em regras (*Algoritmos*)

Uma outra forma utilizada para construir reconhecedores de gestos é através da escrita manual de algoritmos. Esta abordagem se baseia nos dados do esqueleto das mãos obtidos pelo *Software Development Kit* (SDK)[8] do *Leap Motion*, juntamente com outras informações, como velocidade, direção e outros.

Após a captura e processamento dos dados pelo dispositivo, as informações são disponibilizadas atráves do *software Leap Motion*. Este processo permite o rastreamento das mãos e dedos, possibilitando a obtenção de informações em três dimensões. É com base nestas informações que os desenvolvedores podem escrever algoritmos contendo as regras para a construção do reconhecedor de gestos.

Ainda que o *Leap Motion* seja capaz de reconhecer mãos e dedos, com cada uma de suas articulações em três dimensões, não significa que os gestos possam ser diretamente interpretados. Para construir reconhecedores de gestos com esta abordagem, o desenvolvedor terá de pensar primeiramente no conjunto de movimentos que compõe o gesto. O conjunto de movimentos precisa ser abstraido de modo que seja possível implementá-los a partir das informações dispostas pelo SDK. O algoritmo utiliza as informações tridimensionais do esqueleto (coordenadas das articulações capturadas) para determinar as regras e situações que devem ser obedecidas para que um determinado gesto seja reconhecido.

A escrita manual de códigos não necessita de ferramentas especiais ou de uma base de dados de treinamento, o que torna esta abordagem prática inicialmente. Embora tenha tal praticidade, há também nesta abordagem, problemas que precisam ser superados no momento de sua utilização.

Primeiramente temos o problema da legibilidade do código gerado. O código resultante é difícil de ser lido devido a utilização de informações de baixo nível advindas do sensor. Estas informações são basicamente números que representam informações, como localização tridimensional (coordenadas cartesianas). Devido a particularidade anatômica de cada usuário, pode ser necessário realizar ajustes manuais nos parâmetros do algoritmo. Isto requer dos desenvolvedores um conhecimento específico para uma abstração adequada. Além disto, os códigos resultantes por esta abordagem, tendem a ser extensos e aumentam proporcionalmente a complexidade do gesto implementado. A abordagem baseada em regras, geralmente, torna o código de difícil compreensão e manutenção.

[8] *Software Development Kit*, ou Kit de desenvolvimento de *Software* é um conjunto de ferramentas que podem ser usadas para desenvolver aplicativos de software visando uma plataforma específica.

A eficiência do algoritmo depende de parâmetros como velocidade e duração. Por conta da complexidade do código final, compete somente aos desenvolvedores definir os parâmetros para cada algoritmo. Em virtude desta complexidade, a análise estática do código torna-se difícil, o que dificulta, por exemplo, a detecção de gestos ambíguos (conflito de gestos), cabendo esta tarefa, também, aos desenvolvedores.

Finalmente, a necessidade de aprender a manipular a API do dispositivo acrescenta um custo a mais no desenvolvimento do sistema. Isto se torna ainda mais custoso, devido ao baixo nível de abstração dos dados disponiveis provenientes API, o que torna difícil seu processo de aprendizagem.

2.4.3 Considerações finais

A utilização dos movimentos das mãos como interação, mais precisamente a partir de gestos, tem sido empregada em vários domínios. Como consequência disto, surgem necessidades de melhorias tanto na escolha de um repertório de gestos adequado (confortáveis, intuitivos, fáceis de lembrar e aprender) quanto na melhoria das tecnologias e dispositivos aplicados. Dos dispositivos apresentados o *Leap Motion* foi considerado o mais adequado para utilização no projeto desta pesquisa, tendo em consideração seu custo e eficiência para capturar os movimentos das mãos. Finalmente, é apresentada as limitações correspondente a cada uma das abordagens para reconhecimento de gestos. Estas limitações são objetos de interesse do presente trabalho que tem por finalidade solucioná-los

3 ABSTRAÇÃO DO MOVIMENTO DAS MÃOS

3.1 Introdução

Este capítulo trata de métodos empregados para facilitar o processo de construção de reconhecedores de gestos. Primeiramente é apresentado o conceito de abstração aplicados no desenvolvimento de *software* e acrescenta a abordagem de Linguagens Específicas de Domínio. Na sequência são exibidos alguns dos modelos existentes que abordam gestos humanos. No final, são apresentados alguns trabalhos que possuem como proposta facilitar a construção de reconhecedores de gestos, demonstrando suas limitações.

3.2 Abstração de Reconhecedores de Gestos

No Capítulo 2 foram expostos alguns dos problemas relacionados aos reconhecedores de gestos. Diante das limitações encontradas nas abordagens de aprendizagem de máquina e na abordagem baseada em regras surge a necessidade de uma solução em que o usuário ou um especialista de domínio pudesse especificar um gesto, em alto nível, utilizando um conjunto de recursos limitado expressos de forma comum, eliminando toda a complexidade de lidar com cálculos matemáticos ou algoritmos complexos de aprendizagem de máquina.

A abstração está entre os principais motivos para o progresso da engenharia de *software*. Através da inserção de camadas de alto nível, em outras palavras, num grau de abstração longe do código de máquina e mais próximo da linguagem humana, os desenvolvedores podem raciocinar por meio de conceitos de alto nível para construir soluções mais complexas. O sucesso para abstração é o tipo de detalhe que está sendo omitido. Detalhes mais importantes, que são inerentes ao domínio, são escolhidos em detrimento de outros, por exemplo os detalhes de estruturas de dados do sistema.

Tratando de reconhecedores de gestos, as Linguagem de Programação de Propósito Geral (LPPG), API e ferramentas baseadas na abordagem de aprendizagem de máquina permitem novos níveis de abstração. Porém, ainda oferece limitações no que diz respeito a produtividade, complexidade na expressão de sentenças e entendimento do código gerado.

Uma abordagem comumente utilizada em engenharia de *software* para construção e manutenção de sistemas são as Domain Specific Language, ou linguagens específicas de domínio (DSL). As DSLs visam a construção de novas camadas de abstração no desenvolvimento de *software*. Na seção seguinte é descrito maiores detalhes a respeito das DSLs.

3.2.1 Linguagens Específicas de Domínio

Durante o curso da história das linguagens de programação surgiram aquelas criadas com um propósito específico, em outras palavras desenvolvidas para tratar de uma natureza específica de tarefas, ao contrário de outras que foram projetadas para tratar de problemas relacionados a domínios mais restritos. As linguagens orientadas a domínios mais específicos se apoiaram no fato de que abordagens específicas tendem a oferecer uma solução melhor por dedicar-se a um conjunto menor de problemas, em oposição as abordagens genéricas que oferecem uma solução geral para muitos problemas. Linguagens como Cobol, Fortran e Lisp surgiram com propósitos específicos e gradualmente progrediram para LPPG. Portanto, as linguagens específicas de domínio surgem a partir da necessidade de tratar de problemas atrelados a pequenos domínios.

Ainda que muitas DSLs tenham sido construídas e muito utilizadas ao longo dos anos, o estudo sistemático a seu respeito só começou mais recentemente. Devido a este fato, o que viria a ser uma DSL ainda carece de uma definição que seja amplamente aceita. Neste trabalho, empregaremos o conceito de DSL proposto por Deursen et al. (2000):

> *Uma linguagem específica de domínio (DSL) é uma linguagem de programação ou linguagem de especificação executável que oferece, através de notações e abstrações apropriadas, o poder de expressão focado, e geralmente restrito, em um problema particular de um domínio.*

A principal característica de uma DSL segundo esta definição é sua expressividade limitada a uma área. Deste modo, é possível ainda analisar alguns elementos compreendidos nesta definição:

Linguagem de programação de computadores: As DSLs são linguagens destinadas a instruir tarefas a um computador. Seu principal diferencial neste aspecto é o objetivo de facilitar seu entendimento por humanos, assim como em qualquer outra linguagem de programação moderna. São geralmente declarativas e devem possuir uma coleção de sentenças, de notação textual ou visual e uma boa definição formal acerca de sintaxe e semântica.

DSLs são *pequenas*: As DSLs são geralmente pequenas, dispondo apenas de um pequeno conjunto de notações e abstrações. As DSLs suportam a menor quantidade de recursos possível necessárias para utilização em seu domínio. O uso da DSL é para um aspecto específico. Com menor número de recursos, como estruturas de dados, de controle e de abstração variadas, facilita e agiliza o aprendizado e o uso.

Tendo em consideração esta definição, é possível encontrar dezenas de DSLs existentes. Muitas DSLs são construídas como soluções particulares para empresas. Porém, muitas se tornaram conhecidas ao longo dos anos e seguem sendo utilizadas em larga escala. Exemplos bem conhecidos são:

SQL: *Structured Query Language* (SQL) (CHAMBERLIN; BOYCE, 1974), ou Linguagem de Consulta Estruturada. SQL é a linguagem declarativa padrão para banco de dados relacional. Por meio de elementos sintáticos e semântica própria possibilita o desenvolvimento de soluções para problemas de dados através da utilização de cálculo relacional.

BNF: *Backus-Naur Form* (BNF), ou Formalismo de Backus-Naur. BNF é uma notação formal utilizada para descrever gramáticas que definem a sintaxe de uma linguagem. A BNF trata de especificar um conjunto de regras de produção.

HTML: *HyperText Markup Language* (HTML), ou Linguagem de Marcação de Hipertexto. HTML é uma linguagem de marcação que permite a construção de páginas Web. Os documentos HTML são geralmente interpretados por navegadores Web.

O que estas linguagens tem em comum é a forma em que abstraem o problema. As DSLs omitem os detalhes do problema de domínio. Isso possibilita ao desenvolvedor de *software* se concentrar nos problemas intrínsecos ao domínio e não em detalhes de implementação, tornado-o mais produtivo.

Uma das vantagens ao se adotar o uso das DSLs é a abstração fornecida para que o desenvolvedor possa pensar sobre o problema, de maneira que seja fácil especificar uma solução e que seja compreendida naturalmente. As DSLs alcançam isto ao prover um modo mais expressivo de ler e manipular esta abstração (FOWLER, 2010). Outro benefício na adoção de DSLs é o aumento de produtividade, cofiabilidade e manutenção (DEURSEN; KLINT et al., 1998; KIEBURTZ et al., 1996). As DSLs permitem ainda a validação e otimização a nível de domínio (BRUCE, 1997). Muito disto se deve ao fato de que sua expressividade limitada dificulta executar as tarefas de modo errado e facilita a visualização do que está sendo feito.

Em linhas gerais, DSLs são relevantes para o desenvolvimento de *software* porque apresentam uma maneira mais natural de codificar a solução de um determinado problema do que os mecanismos de abstração tradicionais, tais como as LPPG. Desta forma, ao aumentar a abstração dentro do domínio, as DSLs podem vir a se tornar boas ferramentas no auxílio ao desenvolvimento de reconhecedores de gestos.

Deursen et al. (2000) identificam três passos para o desenvolvimento de uma DSL:

Análise:(1) Identificar o domínio do problema; (2) Reunir todos os conhecimentos relevantes ao domínio; (3) Agrupar o conhecimento obtido em noções semânticas e operações acerca das mesmas; (4) Construir uma DSL que descreva as aplicações de forma concisa no domínio.

Implementação: (5) Construir uma biblioteca que implemente as noções semânticas; (6) Desenvolver e implementar um compilador que traduz os programas na DSL para uma sequência de chamadas a biblioteca.

Uso: (7) Escrever programas com DSL para todos os aplicativos desejados e compile-os.

Furtado e Santos (2006) incrementa dois passos a etapa implementação, logo após o

passo (6):

(1) Criação de um editor visual para permitir aos desenvolvedores manipular graficamente a DSL;

(2) Criar validações semânticas para identificar erros de modelagem em tempo de desenvolvimento.

Depois de analisar se há benefícios na construção de uma DSL para o domínio em questão, deverá ser realizada uma análise sobre o domínio. O domínio que o presente trabalho busca abstrair é dos gestos das mãos e dedos, tendo como proposta melhorar a descrição computacional para construção de reconhecedores de gestos de forma que esta tarefa tenha sua complexidade reduzida.

A seguir é apresentado um estudo realizado acerca das representaçãoes dos movimentos das mãos e dedos a fim de obter regras que pudessem ser abstraídas e modeladas para transformá-las em elementos semânticos e sintáticos dentro de uma linguagem apropriada. Este estudo inclui ainda trabalhos que propõem reconhecedores de gestos para o dispositivo *Leap Motion* utilizado neste trabalho.

3.3 Um Estudo das Representações do Movimento Humano

Estudos anteriores em NUI têm basicamente se concentrado na melhoria de técnicas de análise e reconhecimento de gestos. Enquanto estudos no domínio de reconhecimento de gestos vinham sendo conduzidos, percebeu-se a importância de não só detectar eficazmente um gesto, mas de escolher gestos intuitivos para o sistema, direcionando desta forma pesquisas para esta finalidade. Os estudos anteriores no domínio de reconhecimento de gestos utilizaram ou arquivos de vídeos como fonte para sua base de dados ou extensos e complexos algoritmos contendo regras para definição de reconhecedores de gestos. A utilização de gravações de vídeo têm sido utilizadas também pelos estudos que avaliam o quão intuitivo os gestos são para os usuários.

Devido as limitações existentes em cada uma dessas áreas, a representação dos gestos através de uma forma textual nasce como uma proposta vantajosa quando levado em consideração as limitações apontadas. Quando comparada as outras abordagens, os registros textuais ajudam a construir reconhecedores de gestos de maneira direta sem a necessidade de um reprodutor de vídeo para análise de gestos ou na utilização de terceiros para realizar a captura dos gestos. Uma representação textual oferece ainda uma vantagem se comaparada a técnica de reconhecimento baseada em regras, pois propõe-se a descrever os movimentos num nível mais alto de abstração.

Portanto, no presente estudo, é apresentado formas de se representar os movimentos das mãos e dedos de forma textual, de maneira que possa ser utilzida para a construção de reconhecedores de gestos.

3.3.1 Biomecânica

Um dos modos de se descrever os movimentos das mãos é através dos descritores oferecidos pela ciência da biomecânica. A biomecânica é o estudo das leis da mecânica e seus efeitos, aplicados aos sistemas biológicos. Por isto, a biomecânica provê um importante conjunto de conhecimentos para o estudo dos movimentos humano. Por meio de sua área de conhecimento é possível analisar diversas causas e fenômenos relacionados ao movimento humano.

Por volta dos anos 70, a disciplina de biomecânica passou a ter uma ligação com a ergonomia, que por sua vez estuda a relação entre homem e trabalho, de forma, a ser possível mediante os recursos a príncipios biomecânicos reconstruir ferramentas de trabalho tendo como objetivo garantir melhores condições de trabalho e minimizar riscos a saúde dos trabalhadores. Diversos profisionais como fisioterapeutas, educadores físicos e terapeutas ocupacionais, entre outros, estudam a biomecânica para compreender o funcionamento dos músculos e ligamentos durante o movimento.

O estudo biomecânico envolve uma vasta gama de conhecimentos específicos próprio de sua área de atuação, como os estudos do sistema muscular, sistema esquelético, tecidos moles entre outros. Contudo o presente estudo se atém a um pequeno domínio que contribui mais especificamente com a descrição de gestos. É possível saber através da biomecânica quais músculos são recrutados, quais os comportamentos das articulações, durante um certo movimento. Todavia, essas informações não são importantes para se construir reconhecedores de gestos.

Termos anatômicos

Para descrever informações das diversas partes do corpo humano, como a sua localização, regiões e sua identificação, é preciso utilizar uma terminologia adequada. O conhecimento e utilização desta terminologia possibilita uma comunicação eficaz. A descrição de uma parte específica do corpo, bem como um determinado movimento de uma articulação qualquer, é expressa em relação a uma posição referencial. Tal posição referencial é chamada de *posição anatômica* e possibiliza uma referência precisa e padrão para a descrição anatômica. Um inidivíduo se encontra em *posição anatômica* quando está com o corpo ereto com o rosto em direção a sua frente, os braços ao lado do corpo e com as palmas das mãos apontando para frente, pés paralelos e juntos.

A estrutura da mão é basicamente um sistema de ossos interligados, arranjados numa cadeia de arcos transversais e longitudinais. Essencialmente, há dois arcos tranversos (o arco transverso proximal, composto pelos ossos do carpo, e o distal, composto pelas cabeças dos ossos do metacarpo) e arcos longitudinais, formados pelos raios dos dedos. O carpo é constituído por um conjunto de oito ossos, arranjados em duas fileiras horizontais: proximal e distal. A fileira proximal (mais próxima ao corpo) é constituída por escafoíde, semilunar, piramidal e pisiforme. A fileira distal (mais afastada do corpo) é formada por

trapézio, trapezóide, capitato e hamato. O metacarpo é formado por cinco ossos que podem ser numerados de I (um) a V (cinco) e correspondem aos dedos da mão (iniciando com o polegar e terminando no dedo mínimo). Cada um destes ossos apresenta uma base proximal, um corpo e uma cabeça distal, que articula com a base da falange proximal adjacente). Os dedos são formados por catorze ossos denominados falanges. A falange é o nome dado a cada uma das partes do dedo. As falanges estão organizadas em três linhas, denominadas, linha proximal, linha média e linha distal, com exceção do polegar que não possui a linha média. Os dedos são enumerados em sequência de I (um) até V (cinco), iniciando a contagem a partir do polegar. Na Figura 4 estão presentes os descritores dos movimentos do polegar, dedos e punho.

Além dos termos utilizados para descrever partes do corpo, existem também descritores próprios para descrever a forma mecânica do movimento. A nomeclatura utilizada, especifica o movimento conforme a direção e posição das articulações.

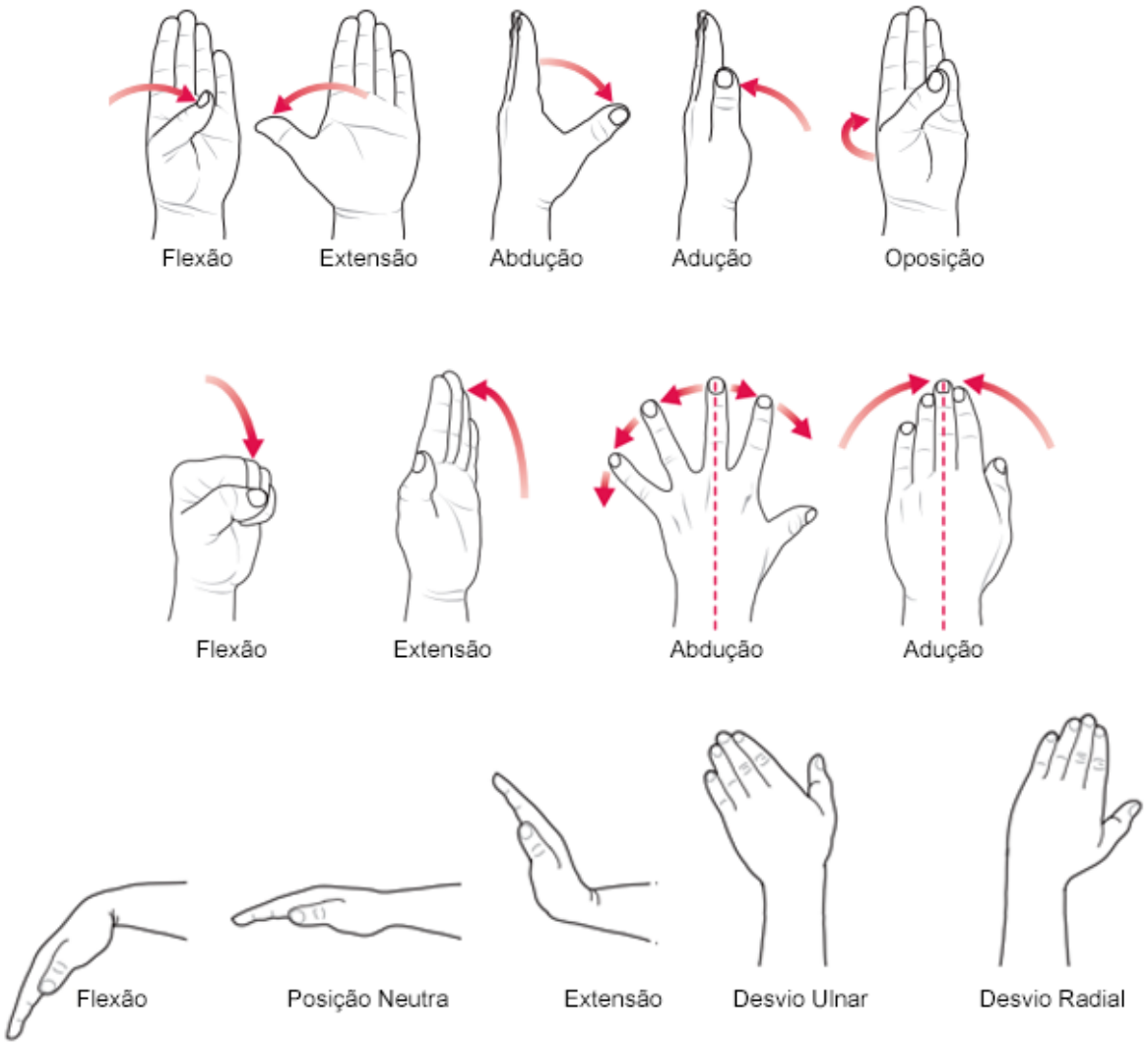
Flexão é o movimento de inclinação de um osso sobre o outro, causando uma diminuição no ângulo da articulação. Comumente isto ocorre entre as faces anteriores dos ossos articulares. De modo inverso, extensão descreve o movimento de endireitamento de um osso em relação ao outro, causando um aumento no ângulo da articulação. Abdução é o movimento que distancia uma estrutura ou parte da linha média do corpo. Adução refere-se a um movimento de retorno do segmento para a linha média do corpo. As exceções a esta definição de linha média são os dedos da mão e do pé, onde o ponto de referência para os dedos é o dedo médio.

Por conta do grande número de ossos presentes na mão, existe um alto número de ligações e tipos de movimentos realizáveis. Em relação a mão é possível definir quatro articulações: articulação do punho, articulação carpometacarpal (CMC), articulação metacarpofalangeana (MCF) e articulação interfalangeana (IF).

A articulação do punho na realidade é formada de duas articulações: a articulação radiocarpal e a articulação mediocarpal. A articulação radiocarpal consiste na extremidade distal do rádio e no disco radioulnar, estruturas ósseas do antebraço, proximalmente, e no escafóide, semilunar e piramidal, distalmente. Esta articulação permite os movimentos de flexão e extensão, abdução e adução, conhecidos como desvios radial e ulnar do punho. A articulação mediocarpal ocorre entre as duas fileiras de ossos carpais, permitem movimentos de deslizamento entre os ossos do carpo e contribuem para o movimento do punho. Na Figura 5 é possível observar as articulações e estruturas da mão discutidas nos próximos parágrafos.

As articulações carpometacarpais ocorrem entre os ossos carpais da fileira distal e a extremidade proximal dos ossos metacarpais. A articulação carpometacarpal do primeiro dedo, o polegar, é composta do trapézio, que se articula com a base do primeiro metacarpal, permite a flexão e extensão, abdução e adução, oposição e reposição. Os movimentos do polegar diferem da nomeação convencional dos demais movimentos articulares, já que

Figura 4 – Descritores dos movimentos do polegar, dedos e punho

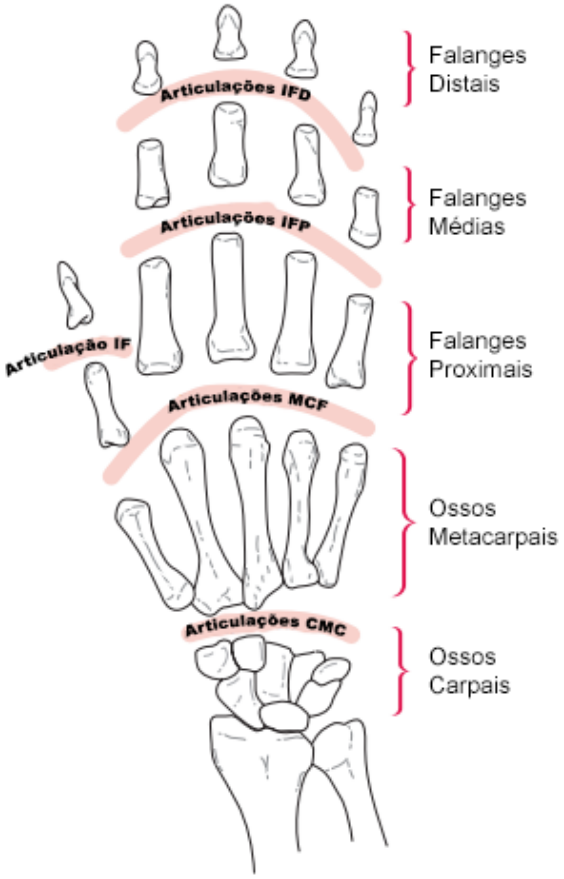


a flexão e a extensão ocorrem em um plano paralelo à palma, enquanto que, a abdução e adução ocorrem em um plano perpendicular à palma. A oposição é uma combinação de flexão e abdução, com um movimento acessório, e reposição é o retorno à posição anatômica.

As articulações carpometacarpais dos demais dedos conferem mais estabilidade do que mobilidade. O trapezóide articula-se com o segundo metacarpal, o capitato com o terceiro metacarpal e o hamato com o quarto e quinto metacarpais. A quinta articulação metacarpal é a mais móvel dos dedos, com exceção da primeira e permite uma pequena quantidade de oposição do quinto dedo. A quarta articulação carpometacarpal é ligeiramente móvel, mas a segunda e a terceira não o é.

As articulações metacarpofalangeanas são formadas pela extremidade distal dos metacarpos em contato com a base das falanges proximais. Os movimentos permitidos nestas

Figura 5 – Articulações e estruturas da mão



articulações são flexão, extensão. abdução e adução. A abdução ocorre quando o segundo, quarto e quinto dedos se afastam do dedo médio e quando o dedo médio se move em qualquer direção. A adução é o retorno da abdução e ocorre com o segundo, quarto e quinto dedos. Não há adução do dedo médio.

Há duas articulações interfalangeanas nos dedos. A articulação interfalangeana proximal ocorre entre a falange proximal e a média; e a articulação interfalangeana distal ocorre entre as falanges média e distal, ambas permitem apenas os movimentos de flexão e extensão.

3.3.2 Língua de sinais

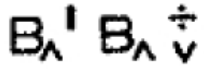
A língua de sinais é uma linguagem que transmite conteúdos diversos, dos mais simples aos mais complexos, através de uma comunicação manual bem definida e movimentos corporais sem a necessidade de sons (QUEK et al., 2002). Já na década de 1960, havia tentativas em definir um sistema de notação para transcrever a língua de sinais. Embora estes métodos de transcrição tenham sido desenvolvidos para lidar particularmente com a língua de sinais, estes podem oferecer pistas de como gestos corporais se encontram estruturados.

Em 1960, Stokoe estabeleceu a existência de três unidades (queremas) bases para transcrever a ASL (Língua de Sinais Americana) (STOKOE, 2005). Estes queremas, utilizadas conjuntamente, seriam capazes de descrever qualquer sinal da língua. Os três queremas definidos pelo autor são: configuração de mão, localização e movimento. O querema configuração de mão, define a forma em que a mão se encontra (disposição dos dedos) durante a realização de um sinal qualquer. Localização corresponde ao local em que o sinal é executado, podendo se referir ao corpo ou a um espaço escolhido. O movimento se refere ao deslocamento das mãos no espaço. Mais tarde, num trabalho posterior, Battison (1978) incrementou um quarto parâmetro (querema) ao método, a orientação da(s) mão(s). Stokoe (2005) incluiu em sua notação um conjunto totalizando 55 símbolos, utilizados associadamente, para compor um sinal. Para representar as configurações de mão há 19 símbolos, 12 para indicar a localização e 24 para especificar o movimento. Devido a sua natureza sequencial Stokoe não permite uma descrição clara acerca de eventos que ocorram simultaneamente, deste modo, a notação de Stokoe acaba por deixar de lado os aspectos de simultaneidade tão presentes na língua de sinais. Além da restrição referente a especificação de simultaneidade, Stokoe acaba por utilizar um conjunto limitado de configurações de mão. Assim sendo, para representar uma nova configuração de mão, será utilizado um símbolo já existente que mais se assemelhe a nova configuração. Considerando o grande número exequível de configurações da mão e o caráter dinâmico da língua de sinais (aparição de novos sinais), a limitação das configurações torna-se um problema. Na Figura 6, as mãos em configuração de B, pontas dos dedos das duas mãos em contato. As mãos movem-se de maneira espelhada, afastando-se uma da outra com um movimento para baixo e na diagonal. Após um tempo, as mãos continuam a descer, num movimento reto, não mais na diagonal, Lembrando o contorno do telhado e das paredes de uma casa (AMARAL, 2012).

Figura 6 – Sinal CASA na Língua de Sinais Americana.

O SignWriting (SUTTON, 1995), notação para escrita visual, foi criado em 1974, projetado para capturar os movimentos realizados na língua de sinais e representá-lo. A notação proposta por SignWriting consta de três unidades básicas: configuração de mão, contato e movimento. É possível ainda especificar outros parâmetros como localização, orientação e

Figura 7 – Sinal CASA escrito no sistema de transcrição Stokoe.



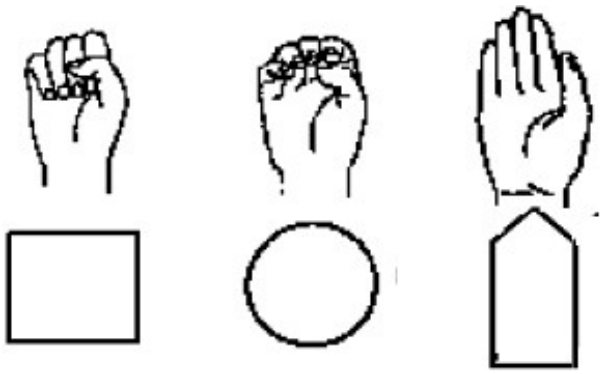
expressões faciais (STUART, 2011). SignWriting representa os sinais de forma visual, através de ícones visuais, ao invés de uma representação baseada em características fonêmicas ou semânticas da língua de sinais. Em SignWriting a configuração de mão é representada por diversos glifos que variam de cinco elementos básicos: quadrado, círculo, pentágono, forma 'C' e forma em ângulo.

Um quadrado representa uma mão fechada, com os dedos flexionados (dobrados) de modo que seja possível tocar a palma da mão. Um quadrado simples, sem a presença de qualquer outro traço, representa uma mão fechada tal qual a letra S do alfabeto manual. Quando modificado, sinaliza que ao menos um dos dedos permanece em contato com a palma da mão.

Um círculo simboliza uma mão aberta, de modo que a ponta do polegar e os demais dedos estejam em contato. Um círculo sem adorno simboliza uma mão aberta tal qual a letra O do alfabeto manual. Quando modificado, indica que ao menos um dos dedos faz contato com o polegar.

Um pentágono representa uma mão plana, onde os dedos encontram-se retos e agrupados (realizando contato lateralmente). Um pentágono simboliza uma configuração de mão semelhante a letra B do alfabeto manual, exceto pelo polegar que não cruza a palma da mão, mas mantém-se em contato com o dedo indicador. As posições mencionadas anteriormente estão ilustradas na Figura 8.

Figura 8 – Representação de mãos em SignWriting



A forma C representa uma mão onde polegar e dedo encontram-se curvados, mas não o suficiente para que entrem em contato. Esta forma simboliza uma configuração tal qual a letra C do alfabeto manual, e pode ser modificado para mostrar variação na configuração dos dedos.

Uma forma em ângulo, semelhante a um L, representa uma mão onde os quatro dedos estão planos (retos e agrupados), mas flexionados em 90° em relação ao plano da palma

da mão. Este glifo sempre vem acompanhado de uma modificação que denota a posição em que o polegar se encontra.

Todas estas formas sofrem alterações através de traços que representam a disposição dos dedos que não estão posicionados tal como as formas básicas. Apesar de contar com um determinado padrão, cada uma destas alterações é um pouco particular e precisa de memorização.

Para indicar o movimento, que as mãos realizam durante a execução de um determinado sinal, é utilizado um conjunto de setas. Existem centenas de setas de diversos tipos e a notação é bastante complexa, em virtude disto diferentes pessoas podem descrever o mesmo sinal de maneiras diferentes.

Em SignWritign há um conjunto de glifos específicos para sinalizar o modo em que o contato entre as partes do corpo deva ocorrer. O glifo de contato é posicionado próximo ao glifo que indica a configuração de mão e a superfice a ser tocada. Existem seis glifos de contato e a sua escolha modifica o modo de contato.

O glifo de asterisco define um toque simples no local indicado, onde a mão gentilmente faz contato; um círculo com um ponto, define o movimento de escovar uma superfície e logo depois abandoná-la; o glifo espiral, especifica o movimento para escovar o local indicado e ali permanecer. Caso não haja uma seta adicional, o movimento deve ser realizado de forma circular; O glifo cerquilha, é utilizado para definir o contato do tipo batida, este contato é indicado com uma mão fortemente em contato com o local indicado; O sinal de adição, representa o contato do tipo pegar, utilizado para agarrar o local indicado (geralmente a outra mão ou uma parte da roupa); um símbolo com duas barras de cada lado do asterisco define o contato do tipo Entre, onde duas partes do corpo devem passar uma através da outra (geralmente entre dedos).

A Figura 9 apresenta o sinal CASA (Figura 3.4) descrito em SignWriting.

Figura 9 – sinal CASA escrito em Signwriting

Em SignWriting, na maioria das vezes, os sinais são escritos da perspectiva do signatário. Assim, os glifos podem representar a palma, dorso ou o lado da mão.

Além de possibilidade de especificar expressões faciais, SignWriting conta com a capacidade de descrever a dinâmica dos movimentos. Existem símbolos que, adicionados aos símbolos de movimento ou de expressões faciais, configuram simultaneidade, como quando ambas as mãos se movem sincronicamente, movimentos alternados, movimentos rápidos, suave, tenso e relaxado.

Apesar de se mostrar uma notação concisa e de uma grande capacidade expressiva (ROSENBERG, 1999), SignWriting conta com mais de 600 símbolos, além de outros deta-

lhes que podem ser escritos, o que traz um maior desafio e esforço ao seu aprendizado. Isto significa que a complexidade das transcrição aumenta proporcionalmente a complexidade do sinal. Além da extenso número de símbolos há várias formas de se escrever um único sinal. SignWriting não pode ser utilizado em processadores de textos convencionais, necessitando de *softwares* apropriados para sua gravação em imagem. Todas estas características dificultam sua utilização como uma descrição computacional.

3.3.3 Um método de taxonomia e notação para gestos das mãos em três dimensões

Uma outra notação encontrada para especificar gestos das mãos em três dimensões é o trabalho proposto por Choi, Kim e Chung (2014). Esta pesquisa propõe uma nova e sistemática abordagem para descrever os gestos realizados pelas mãos. Primeiramente o estudo concentra esforços em identificar os elementos envolvidos durante um gesto com as mãos, dedos, relação entre os dedos, palma da mão, relação entre as mãos, entre outros. O resultado do estudo oferece uma taxonomia robusta capaz de descrever uma ampla gama de movimentos. Para especificar os gestos o mesmo estudo disponibiliza uma notação baseada em uma de elementos associado a um código que corresponde a uma configuração possível para cada elemento.

Ainda que o poder de expressividade do método proposto pela pesquisa seja suficiente para descrever uma variedade de movimentos, a notação proposta não oferece legibilidade adequada para um indivíduo que não conhece o método, de forma que uma leitura correta seria improvável de acontecer. A figura 10 apresenta um gesto realizado pelas mãos e a notação que corresponde ao mesmo gesto.

Figura 10 – Realização de um gesto arbitrario e sua respectiva notação segundo o método de Choi, Kim e Chung (2014)

6 1-H1 → → 2(0,0,0) → 31111; 2222-222 → 5; 1 → 2; 3
⇒ 56666; 4422-333

A partir da leitura desta notação é difícil para um indivíduo que desconhece o método

inferir que gesto a notação descreve, será necessário um mínimo de raciocínio para decifrar o que está escrito. Portanto, a DSL proposta por esta dissertação toma como base a taxonomia e o sistema de notação proposto por (CHOI; KIM; CHUNG, 2014) com as modificações julgadas necessárias. O capítulo 4 que discute a solução proposta, contém maiores detalhes acerca da taxonomia, notação e alterações realizadas, bem como as justificativas para cada uma delas.

3.3.4 Descrevendo Movimento Humano na Computação

Recentemente, muitas disciplinas das áreas de multimídia e comunicação têm focado seus esforços na pesquisa de NUI. A pesquesa nesta área não é novidade, porém as pesquisas, hoje, se concentram muito mais em como descrever gestos e convencioná-los (NORMAN, 2010). Muitos desses estudo visam a substituição de paradigmas já tradicionais como o *Windows, Icons, Menus, and Pointer* (WIMP) (DAM, 2000). Os gestos realizados pelas mãos são considerados ainda mais dominantes devido aos graus de liberdade que facilita e dá maiores possibilidades na interação natural entre usuários e sistemas (KANG et al., 2013). Com os avanços na área de processamento de imagem e surgimento de dispositivos cada vez mais poderosos, vários estudos passaram a empregar esforços na pesquisa envolvendo gestos tridimensionais em interfaces humano-computador (KIM et al., 2011; KIM et al., 2005; HENZE et al., 2010).

Apesar dos esforços aplicados no estudo das interfaces naturais, os sistemas de notação utilizados têm se baseado em sua maioria nas linguagens de sinais ou em interfaces multitoque (CHOI; KIM; CHUNG, 2014). Com excessão dos estudos de Mo (2007) e Choi, Kim e Chung (2014), não foi encontrado métodos de notação para gestos das mãos em 3D interfaces humano-computador.

Com a aparição de novos dispositivos projetados especialmente para rastrear partes do corpo, como o *Leap Motion*, vários *softwares* têm sido desenvolvidos para auxiliar os desenvolvedores na tarefa de construir reconhecedores de gestos. Apesar de tais esforços, as soluções existentes são muito limitadas no que diz respeito a criação de novos gestos, quando na maioria dos casos permitem apenas pequenas costumizações.

3.3.5 Trabalhos Relacionados

No contexto de gestos livres das mãos, alguns trabalhos têm sido apresentados. Os trabalhos encontrados, desenvolvidos para customização de gestos em *Leap Motion*, não se caracterizam como uma DSL segundo as definições de Fowler (2010). Limitações como o baixo número de parâmetros para especificar os gestos ou mesmo conter um número de gestos limites já predefinidos, dificultam chegando a impossibilitar a especificação de alguns gestos. Para mais, uma característica desejada aos sistemas projetados com o fim de especificar gestos, seria a capacidade de reutilizar os gestos criados.

Apesar de limitações, quando comparada a solução do presente trabalho, estão presentes a seguir alguns trabalhos que tem por objetivo customizar gestos para o *Leap Motion* e que mais se aproximam da abordagem presente na solução deste trabalho.

Leap Motion Simple Control

O *Leap Motion Simple Control* (LMSC), um pacote disponível para a *engine Unity*, é uma ferramenta projetada para agilizar e facilitar a construção de reconhecedores de gestos em aplicações desenvolvidas no *Unity*[1]. LMSC tem por objetivo fornecer um conjunto predefinido de gestos de mão baseados no dispositivo *Leap Motion*. Há um total de doze gestos predefinidos, porém é possível ampliar este número adicionando novos gestos. É possível ainda ajustar configurações relacionadas ao reconhecimento dos gestos. Para que um gesto seja reconhecido em LMSC é preciso que o movimento das mãos se comporte de acordo com as regras especificadas e atenda a dois parâmetros ajustáveis. O primeiro parâmetro diz a respeito do tempo em que o gesto deve ser mantido. Por exemplo, para que o gesto *FaceUp* seja identificado é preciso que além da condição específica da palma da mão está voltada para cima, é preciso que esta pose seja mantida por um determinado tempo. O segundo parâmetro refere-se ao tempo de atraso necessário para que um novo gesto seja executado. Ambos os parâmetros podem ser configurados a critério do desenvolvedor.

LMSC permite que o desenvolvedor crie seu próprio gesto e o adicione a coleção de gestos predefinidos. Apesar da maior parte da interação com LMSC ser através de interface gráfica, a criação de um novo tipo de gesto é alcançado através de uma linguagem de propósito geral, mediante a especificação de regras que permitam identificar quando o gesto em questão é realizado. Criar um reconhecedor de gesto através da abordagem baseada em regras vai contra todo o príncipio pretendido pelas abordagens que buscam abstrações mais intuitivas.

GECO MIDI

O *software GECO* é uma ferramenta destinada a interagir com os formatos *MIDI*, *OSC* e *CopperLan* através de um uso livre das mãos. *GECO* foi projetado para que músicos e produtores sejam capazes de controlar, em tempo real, parâmetros relacionados a música, durante exibições ao vivo. A ferramenta funciona de maneira satisfatória, apresentando baixa carga de processamento e baixa latência. O *software* conta com um *feedback* visual em tempo real além de permitir algumas personalizações.

GECO dispõe de um conjunto particular de gestos predefinidos, onde, por intermédio de uma interface gráfica, exibe em tempo real os dados de entrada capturados pelo dispositivo *Leap Motion*. A interface gráfica utiliza um sistema de cores para mostrar quais movimentos estão sendo realizados no momento. O sistema considera ainda como dife-

[1] Unity é uma engine de jogos multi-plataforma desenvolvido pela Unity Technologies. É utilizada principalmente para desenvolver videogames e simulações para computadores, consoles e dispositivos móveis.

rentes os movimentos realizados por mão esquerda e direita, adicionando mais dinâmica para os gestos. É possível também adicionar a configuração de mão aberta e fechada para diferenciar os movimentos.

GECO é um sistema bastante específico e é capaz de interagir com alguns formatos de mídia através de um *software* ou *hardware* capaz de manipulá-los. Além de não poder ser utilizado para interagir com outras aplicações, não é possível adicionar mais parâmetros aos gestos que os já citado. Com o conjunto de movimentos presentes em *GECO* não seria possível especificar gestos mais complexos, por exemplo com variações na configuração de mão.

3.4 Considerações finais do capítulo

O estudo a respeito das formas de representação dos gestos humanos contribui para uma melhor fundamentação do domínio da solução sugerida. As diversas formas de representação estudadas apresentam inconvenientes que variam desde a complexidade dos termos empregados em sua notação, esforço na aprendizagem, limitações na representação e extensa escrita resultante para descrição de gestos mais complexos. Todavia, a taxonomia e notação sugerida por Choi, Kim e Chung (2014) se mostra adequada a solução deste trabalho, visto que oferece expressividade satisfatória, com ressalvas ao seu método de notação que não é intuítivo. Os trabalhos cujo interesse está focado na customização de gestos para o dispositivo *Leap Motion*, são demasiados limitados no que diz respeito a poder de expressividade ficando limitado ao um conjunto pequeno de gestos. Levando em consideração os pontos anteriormente citados, é proposta uma nova linguagem (DSL) baseada na notação proposta por Choi, Kim e Chung (2014) cujo principal objetivo é facilitar e agilizar o desenvolvimento de reconhecedores de gestos para o dispositivo *Leap Motion* em aplicações que utilizam os gestos das mãos como modo de interação.

4 SOLUÇÃO PROPOSTA

4.1 Introdução

O presente capítulo descreve detalhadamente a solução proposta, a DSL Leap Gesture, uma DSL interna voltada a especificação de gestos das mãos em três dimensões. Leap Gesture foi projetada para ser utilizada juntamente com a *engine* de jogos *Unity* e o sensor de movimento *Leap Motion* trazendo consigo mais agilidade e facilidade no desenvolvimento de jogos que utilizem o dispositivo como meio de interação.

As próximas sessões apresentam os detalhes da DSL juntamente com exemplos de sua utilização. Serão ainda apresentados alguns detalhes da arquitetura e implementação da DSL, suas limitações e possibilidades de melhorias.

4.1.1 LeapGesture: uma DSL interna

Leap Gesture é uma DSL interna concebida para facilitar e agilizar o desenvolvimento de jogos/aplicações, na engine *Unity*, que utilizem como interação os gestos das mãos, através do sensor *Leap Motion*. O *Unity* oferece suporte a programação de scripts através da linguagem C#, linguagem hospedeira para a Leap Gesture. Os desenvolvedores utilizam os scripts para controlar vários eventos do jogo como comportamento físico de objetos, efeitos gráficos, entradas do usuário, entre outros. É através dos scripts que os desenvolvedores manipulam os dados obtidos através do sensor *Leap Motion*, para controlar a interação com o jogo/aplicação. O *Leap Motion* oferece uma API através de um pacote asset para integração com *Unity*. Esse pacote contém vários scripts e modelos 3D que representam as mãos bem como outros recursos para auxiliar o desenvolvimento de jogos/aplicações.

Ainda que a API para o *Unity* oferecida pelo *Leap Motion* traga bastantes recursos que otimizam e facilitam a construção de jogos e aplicações, o uso da API torna o código do jogo menos legível , com expressões muito verbosas, além do custo de aprendizado que distanciam o desenvolvedor de sua principal preocupação, as características chave do seu domínio. A Leap Gesture traz consigo a possibilidade do desenvolvedor empregar menos esforços na construção da interação de seu jogo/aplicação com o dispositivo.

Por se tratar de uma DSL interna, a Leap Gesture permite que os desenvolvedores a utilizem no ambiente de desenvolvimento ao qual já estão habituados e desta maneira beneficiar-se-ão de ambos recursos, da Leap Gesture e da LPPG , neste caso C# a linguagem hospedeira da DSL. O principal benefício de uma DSL interna neste contexto é de permitir ao desenvolvedor mesclar e estender a DSL interna com a LPPG. Com isso o desenvolvedor poderá contar com os principais recursos da linguagem hospedeira e ao mesmo tempo contar com o poder de expressão oferecido pela Leap Gesture. Apesar

do ruído sintático intrínseco que as DSLs internas possuem, julgamos que os especialistas deste domínio não enfrentarão grandes dificuldades já que estão habituados com as convenções sintáticas da linguagem hospedeira.

4.2 A sintaxe de Leap Gesture

A Leap Gesture possui uma sintaxe baseada na notação de Choi, Kim e Chung (2014) com sentenças em inglês, idioma padrão no desenvolvimento de *software*, seus termos são simples abstraindo toda a complexidade dos dados obtidos através do sensor. No código 4.1, temos um exemplo de sua utilização.

```
1
//Specifying the gesture
3   gesture
      .pose () //First pose
5       .HandLocation ("1")
        .FingerPose ("POINTING_SIDE", "POINTING_UP", "POINTING_UP", "CLOSE",
            "CLOSE")
7       .FingerRelation ("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "
    SEPARATE", "SEPARATE", "GROUP")
      .pose () //Second pose
9       .HandLocation ("3")
        .FingerPose ("POINTING_SIDE", "POINTING_UP", "POINTING_UP", "CLOSE",
    "CLOSE")
11      .FingerRelation ("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE",
    "GROUP", "SEPARATE", "GROUP")
    ; //End Gesture
```

Listing 4.1 – Exemplo de utilização de Leap Gesture

O código 4.1 descreve um gesto de uma mão aberta, ele diz explicitamente a configuração de cada dedo bem como sua posição em relação aos outros. A utilização destas e outras sentenças serão mais detalhadas ao longo desta seção. Apesar dos ruídos encontrados nesta DSL (vírgulas, parênteses, pontos), devido a natureza da notação adotada por Leap Gesture, os desenvolvedores diferenciarão claramente o código construído para a realização da captura dos gestos dos demais construídos para gerenciar os eventos do jogo/aplicação.

A construção de reconhecedores de gestos em Leap Gesture é feito através de um objeto do tipo GestureBuilder. A especificação do gesto se dá pela utilização de uma ou mais poses através do objeto *GestureBuilder*. Cada pose em Leap Gesture equivale ao que chamamos de um gesto estático, em outras palavras, um gesto que poderia ser representado em um único quadro. Através da combinação das cláusulas *pose* é possível especificar os gestos a serem reconhecidos. O código 4.2 exemplifica a estrutura de um gesto implementado em Leap Gesture.

```
//Declaracao de um metodo com o tipo de retorno GestureBuilder
GestureBuilder fist(){

  //Criacao do objeto que sera retornado:
  GestureBuilder gesture = new GestureBuilder();

  //Aqui comeca a especificacao do gesto:
  gesture
    .pose()
      //... Aqui invocamos as sentencas responsaveis por especificar a pose
    .
    .pose()
    //Podemos especificar n pose para descrever meu gesto
    //pose()
    //pose()

  ;//fim do gesto

  return gesture;
}
```

Listing 4.2 – Estrutura básica de um gesto em Leap Gesture

Podemos ainda, com o uso de uma endentação apropriada, dispor de uma hierarquia visual ao código escrito, melhorando ainda mais a legibilidade do código. Esta maneira de codificar é estranha para o padrão usual de programação, mas se tratando de DSLs internas faz todo sentido já que o foco está na construção de sentenças que façam sentido em um determinado contexto, no presente caso, expressões que descrevam gestos das mãos num espaço tridimensional.

4.2.1 A cláusula *pose*

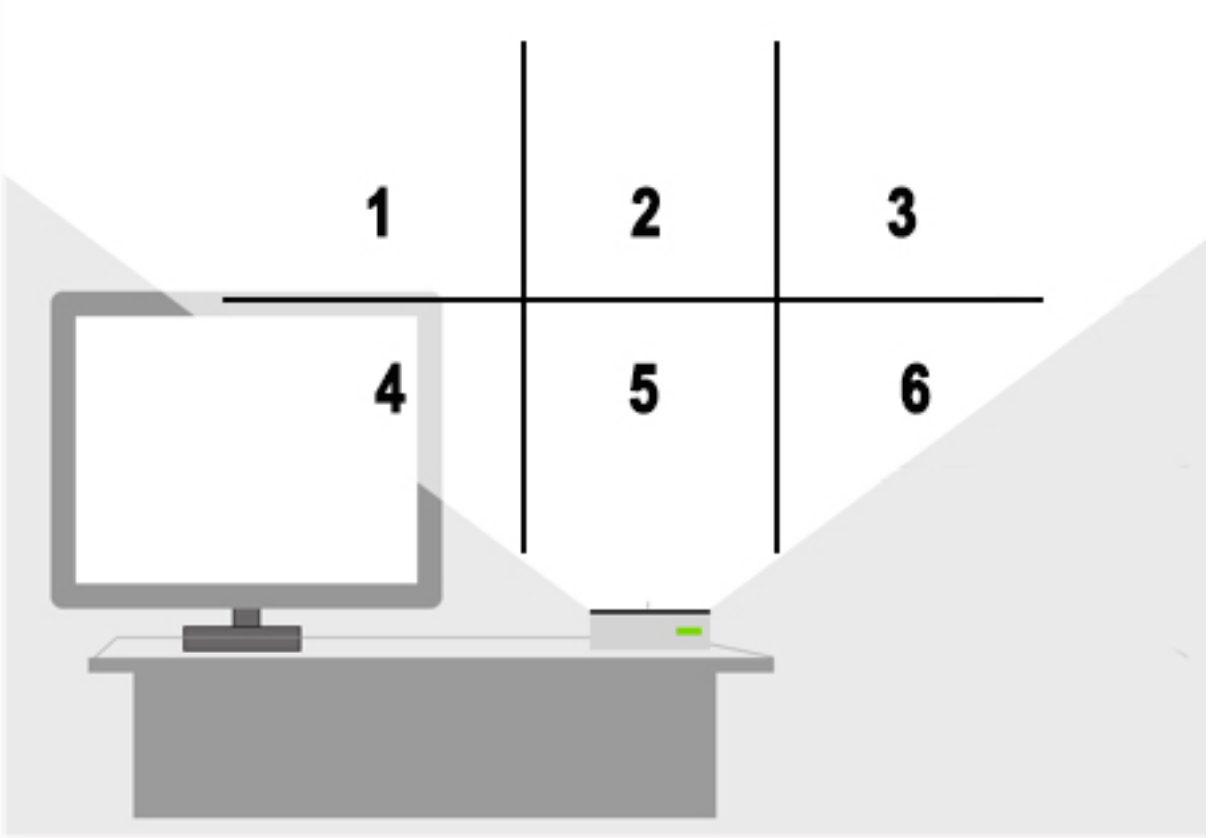
Em Leap Gesture os elementos possíveis de manipulação são os dedos e as mãos e são com estes elementos que podemos compor cada quadro. A cada quadro deste podemos chamar de gesto estático. A combinação em sequência destes quadros gera um movimento o qual podemos chamar de gesto dinâmico. A cláusula *pose* nos permite definir num único (definir um gesto estático) quadro como estarão dispostos os elementos que o compõe. Como dito anteriormente um gesto poderá ter uma ou mais cláusulas *pose* e isto será escolhido de acordo com a necessidade de cada gesto.

O uso da cláusula *pose* em Leap Gesture é realizado através da invocação do método *pose()* a partir de um objeto do tipo GestureBuilder. Com a invocação deste método é possível especificar, a partir de outras sentenças, os demais elementos que compõe cada quadro. Nas sessões a seguir veremos a descrição e utilização de cada sentença permitida numa *pose*.

4.2.2 A cláusula *handLocation*

Com cláusula *HandLocation* nós especificamos a posição da mão no espaço, Leap Gesture dispõe de 6 localizações no eixo vertical. Utilizamos a sentença *HandLocation* através do método *HandLocation (String location)* e passamos através de um parâmetro do tipo string, um número inteiro de 1 a 6 que corresponda a localização desejada para a pose que está sendo especificada, de acordo com as zonas possíveis. Devido a limitação do campo de visão do sensor *Leap Motion* utilizamos seis zonas de marcação no espaço ao invés de nove como proposto por Choi, Kim e Chung (2014), ir, além disto, impossibilitaria a detecção das mãos por parte do sensor. Diferentemente da notação de Choi, Kim e Chung (2014) em que a zona de localização das mãos se baseia no corpo do usuário, as zonas de localização das mãos possíveis em Leap Gesture estão alinhadas ao campo de visão do sensor *Leap Motion*. É possível observar isto na figura 11

Figura 11 – Zonas de localização de Leap Gesture.



Fonte: Autoria própria

Caso queiramos especificar um gesto onde uma mão se desloca de um ponto a outro num plano vertical/horizontal, poderemos utilizar várias cláusulas *Pose* contendo a cláusula *HandLocation* especificando sua posição de acordo com as zonas previstas. Temos no código 4.3 um exemplo de script para especificar um gesto de mover uma mão para a direita.

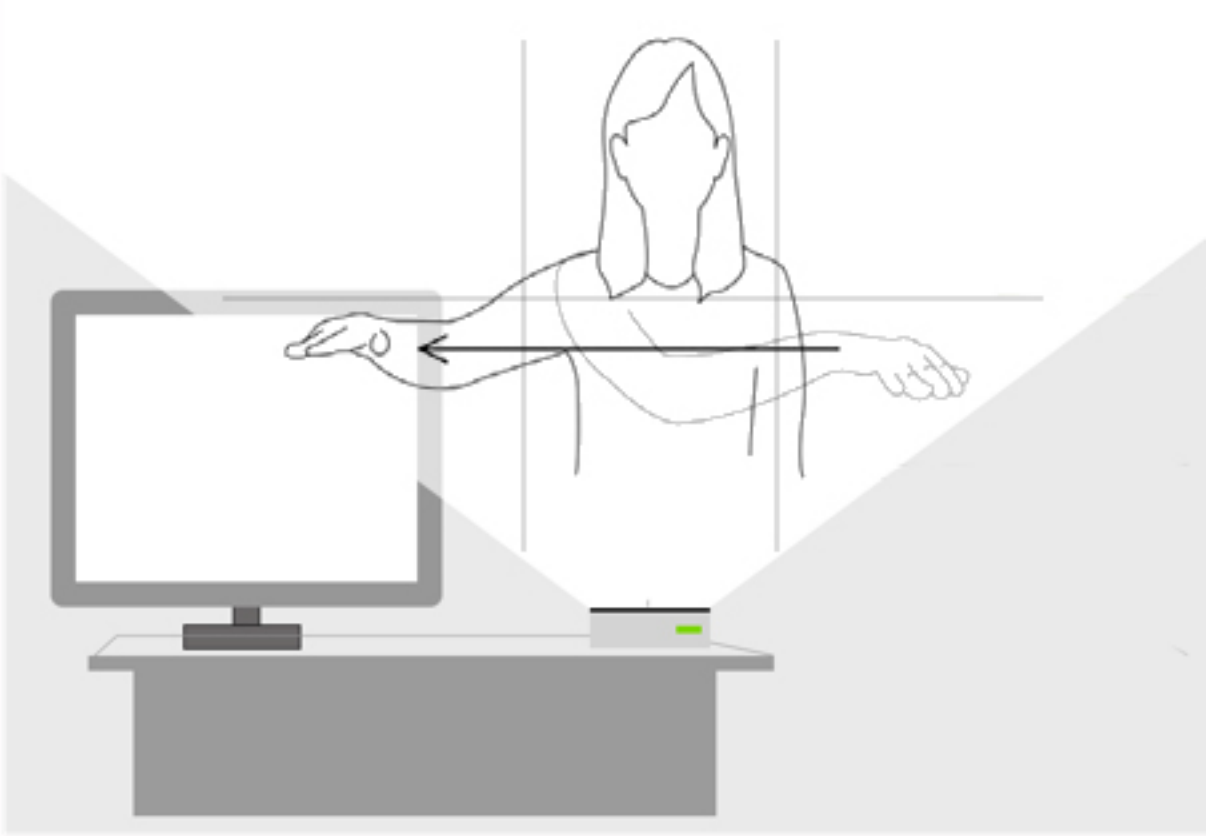
```
1 gesture
```

```
    //Primeira pose.
3   .pose ()
      .HandLocation ("6") //Zona de partida.
5
    //Segunda pose.
7   .pose()
      .HandLocation ("4") //Zona de chegada.
9 ;//fim do gesto
```

Listing 4.3 – Exemplo de utilização de Leap Gesture

Com este *script* será possível representar o gesto ilustrado pela figura 12. Para realizar o gesto especificado através do *script* o usuário deverá mover sua mão a partir da zona seis levando-a até a zona quatro.

Figura 12 – Gesto utilizando a cláusula *HandLocation*.



Fonte: Autoria própria

4.2.3 A cláusula *palmOrientation*

A cláusula *PalmOrientation* nos permite indicar a orientação da palma da mão. Através desta cláusula é possível especificar para qual direção à palma da mão está voltada. A utilização desta sentença em Leap Gesture dar-se pela utilização do método *PalmOrientation(String orientation)*. O valor que passamos na chamada do método corresponde a

direção da orientação da palma da mão na pose corrente. Dentre os valores que podemos passar na chamada do método, estão os mesmos propostos por Choi, Kim e Chung (2014) para especificar a orientação num espaço tridimensional, são eles: "TOP" (a cima), "BOTTOM" (a baixo), "RIGHT" (à direita), "LEFT" (à esquerda), "FORWARD" (à frente), "BACKWARD" (para trás).

O código 4.4 descreve dois exemplos de utilização da cláusula *PalmOrientation*.

```
Gesture wristFlexion(){
    gesture
        .pose()
            PalmOrientation("BOTTOM")
        .pose
            PalmOrientation("FORWARD")
    ;
}

Gesture forearmRotation(){
    gesture
        .pose()
            PalmOrientation("BOTTOM")
        .pose
            PalmOrientation("TOP")
    ;
}
```

Listing 4.4 – Exemplo de utilização da cláusula *PalmOrientation*

No primeiro gesto temos como ponto de partida uma pose em que a palma da mão está virada para baixo e na pose seguinte a palma da mão está orientada para frente, Esse gesto caracteriza um gesto de extensão do punho. No segundo gesto temos uma pose que se inicia com a palma da mão virada para baixo e conclui o gesto, na segunda pose, com a palma da mão virada para cima, caracterizando um gesto de rotação do antebraço.

4.2.4 A cláusula *fistFaceOrientation*

A partir da utilização de *FistFaceOrientation* podemos informar a orientação desejada para a face do punho na pose em que estamos trabalhando. A utilização desta sentença em Leap Gesture é feita através do método *FistFaceOrientation(String orientation)* dos quais os possíveis valores a serem passados na chamada do método são os mesmos propostos por Choi, Kim e Chung (2014) para especificar a orientação da mão, da mesma maneira que utilizamos na cláusula *PalmOrientation*. Com a utilização de *FistFaceOrientation* em conjunto com *PalmOrientation* podemos especificar a orientação da mão com uma precisão ainda maior deixando a interpretação do gesto menos ambígua. Para ilustrar um simples caso tomemos como exemplo o script para o gesto wristFlexion, neste gesto temos presente apenas a orientação da palma da mão para cada pose que compõe o gesto. Com

isto, o usuário poderia realizar o mesmo gesto de diferentes maneiras, pois apesar de ter de iniciar o gesto com a palma da mão para baixo e concluir com a mesma virada para cima, a pose nada diz a respeito da direção que o punho está apontando, assim para a pose final teríamos algumas diferentes possibilidades, dentre estas, as seguintes ilustradas pela figura 13:

Figura 13 – Possibilidade de gestos para a mesma orientação da palma da mão.

Fonte: Autoria própria

Para especificar a orientação, tal qual vimos na figura 13, devemos utilizar a cláusula *FistFaceOrientation* na pose desejada. O código 4.5 exemplifica sua utilização a fim de obter as diferentes *pose* ilustradas na figura 13.

```
gesture
    //Primeira pose.
    .pose ()
        .PalmOrientation ("FORWARD")
        .FistFaceOrientation("RIGHT")
    //Segunda pose.
    .pose()
        .PalmOrientation ("FORWARD")
        .FistFaceOrientation("TOP")
    //Terceira pose.
    .pose()
        .PalmOrientation ("FORWARD")
        .FistFaceOrientation("LEFT")
;//fim do gesto
```

Listing 4.5 – Exemplo de utilização da cláusula *FistFaceOrientation*

4.2.5 A cláusula *fingerPose*

Durante a realização dos gestos os dedos podem assumir diferentes formas o que possibilita a cada gesto uma maior particularidade bem como possibilita a existência de um maior número de gestos.

A cláusula *FingerPose* nos permite definir a maneira com que os dedos das mãos estarão dispostos durante uma pose, em outras palavras, podemos configurar cada dedo da mão de forma que estas combinações construam diferentes formas para a mão. Em Leap Gesture utilizamos esta cláusula através do método:

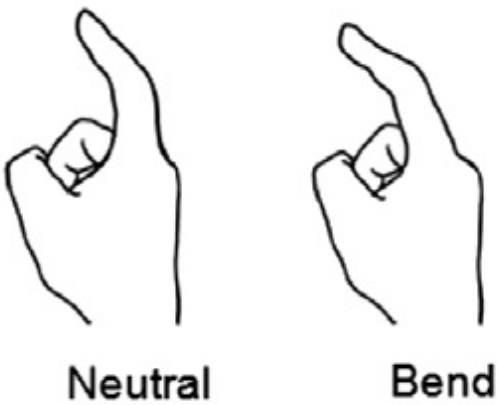
```
public PoseBuilder FingerPose(string thumb, string index, string middle,
    string ring, string pinky){}
```

Listing 4.6 – Método *FingerPose*

Esta cláusula deve ser utilizada em cada pose no qual se deseja obter uma forma específica para a mão. Os parâmetros deste método produzem uma sequência de argumentos do tipo *String* onde o primeiro argumento corresponde a configuração definida para o polegar e os demais seguem a sequência: indicador, médio, anelar e mínimo. As configurações possíveis para cada dedo são: “POINTING_UP” (apontando para cima), “POINTING_FORWARD” (apontando para frente), “POINTING_SIDE” (apontando para o lado), “BEND” (parcialmente dobrado para frente), “CLOSE” (completamente dobrado).

As configurações possíveis em Leap Gesture são baseadas na notação de Choi, Kim e Chung (2014) mas diferentemente desta não possui a configuração neutral que especifica o dedo como estando em uma posição naturalmente relaxada. A decisão de não incluir esta configuração em Leap Gesture foi devido a limitação do sensor e a similaridade com a configuração bend.

Figura 14 – Comparação entre as configurações *neutral* e *bend*.



Como é possível ver pela figura 14, as configurações são muito parecidas dificultando a definição de uma fronteira entre essas configurações e depois devido a falta de precisão do sensor em obter os pontos de referência da mão, ambas as configurações poderiam ser facilmente confundidas trazendo maiores problemas na execução da DSL. Além disto, seria difícil também para os usuários diferenciar estas configurações no momento de executar os gestos.

O código 4.7 mostra um exemplo de utilização da cláusula *FingerPose*:

```
gesture
    //Primeira pose.
    .pose ()
      .FingerPose("CLOSE", "CLOSE", "CLOSE", "CLOSE", "CLOSE")
    //Segunda pose.
    .pose()
      .FingerPose("POINTING_UP", "POINTING_UP", "POINTING_UP", "POINTING_UP
    ", "POINTING_UP")
;//fim do gesto
```

Listing 4.7 – Exemplo de utilização da cláusula *FingerPose*

Como foi visto, as configurações para cada dedo se dá de acordo com a sequência estabelecida na passagem dos parâmetros. Para dar uma maior legibilidade a Leap Gesture, propomos um segundo método para a cláusula *FingerPose*, através do método:

```
public PoseBuilder FingerPose(Thumb thumb, Index index, Middle middle, Ring
    ring, Pinky pinky){}
```

Listing 4.8 – Método *FingerPose* alternativo

Em Leap Gesture existe tipos de objetos para cada um dos dedos das mãos: *Thumb*, *Index*, *Middle*, *Ring*, *Pinky*. Esses objetos foram criados especialmente para serem usados junto ao método acima citado. Com eles é possível invocar métodos correspondentes a cada uma das configurações permitidas a cada dedo em Leap Gesture. Utilizando esse método alternativo para especificar a cláusula *FingerPose* obtemos uma maior legibilidade ao script já que vemos explicitamente a chamada para o dedo específico. Ademais com esse método o usuário é obrigado a lembrar da sequência correta da passagem de parâmetros já que a utilização de outro dedo (tipo de objeto correspondente a cada dedo) na posição errada levará a um erro de compilação. Com isto ainda podemos explorar os benefícios da IDE como o *autocomplete* que lista as opções de configurações possíveis.

O código 4.9 exemplifica a utilização da cláusula *FingerPose* com o segundo método:

```
gesture
    .pose()
        .FingerPose (
            Thumb.close(),
            Index.close(),
            Middle.close(),
            Ring.close(),
            Pinky.close()
        )//fim da clausula fingerPose
    .pose()
        .FingerPose (
            Thumb.close(),
            Index.pointingUp(),
            Middle.close(),
            Ring.close(),
```

```
            Pinky.close()
17        )//fim da clausula fingerPose
```

Listing 4.9 – Utilização da cláusula *FingerPose* com o método alternativo

Contudo, fica ao critério do usuário escolher o método que mais lhe parecer oportuno. O primeiro é menos verboso por ser mais direto se baseando na sequencialidade de seus parâmetros. O segundo é mais legível, pois contém em cada argumento o nome do dedo que se deseja configurar e evita que o usuário erre a sequência dos dedos.

4.2.6 A cláusula *fingerRelation*

Além da configuração que cada dedo possui eles podem assumir diversas condições em relação ao dedo vizinho. A utilização da cláusula *FingerRelation* permite descrever a relação entre dois dedos. A utilização desta cláusula em Leap Gesture é feita através do método:

```
1 public PoseBuilder FingerRelation(string thumbToIndex, string thumbToMiddle
    , string thumbToRing, string thumbToPinky, string indexToMiddle, string
    middleToRing, string ringToPinky){}
```

Listing 4.10 – Método *FingerRelation*

Como podemos observar esse método se baseia na sequencialidade dos argumentos de tipo string onde cada posição equivale a uma relação específica entre dois dedos. Na primeira posição especificamos a relação entre o polegar e o indicador e na sequência, polegar e médio, polegar e anelar, polegar e mínimo, indicador e médio, médio e anelar, anelar e mínimo. Para cada uma dessas relações é possível estabelecer os seguintes valores: “SEPARATE” (dedos separados), “GROUP” (dedos juntos), “CROSS” (dedos cruzados), “TOUCH” (contato entre as pontas de cada dedo).

Diferente da notação de Choi, Kim e Chung (2014) em Leap Gesture não existem os valores *neutral* e *loop*. A relação *neutral* é descrita por Choi, Kim e Chung (2014) como sendo uma relação entre dois dedos, onde estes se encontram naturalmente separados. Devido a imprecisão do quanto esses dedos estariam afastados, levamos em consideração o quão subjetivo seriam para os usuários realizar tais gestos já que poderia ser confundido com a relação separate não havendo aqui um fronteira muito bem definida entre elas. A relação *loop* é descrita por Choi, Kim e Chung (2014) como sendo uma relação entre dois dedos, onde juntos formam um círculo. Devido a limitação do nosso reconhecedor não é possível fazer a distinção entre as relações *touch* e *loop*, por este motivo temos presente em Leap Gesture apenas o valor *touch*.

No código 4.11 temos um exemplo da utilização da cláusula *FingerRelation* com o método explicado anteriormente.

```
1 gesture
      .pose()
```

```
3         .FingerRelation ("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "
      SEPARATE", "SEPARATE", "SEPARATE")
      .pose()
5         .FingerRelation ("GROUP", "SEPARATE", "SEPARATE", "SEPARATE", "
      GROUP", "GROUP", "GROUP")
```

Listing 4.11 – Exemplo de utilização da cláusula *fingerRelation*

A fim de proporcionar uma maior legibilidade ao *script*, propomos um segundo método para a cláusula *FingerRelation*:

```
1 public PoseBuilder FingerRelation(ThumbToIndex thumbToIndex, ThumbToMiddle
      thumbToMiddle, ThumbToRing thumbToRing, ThumbToPinky thumbToPinky,
      IndexToMiddle indexToMiddle, MiddleToRing middleToRing, RingToPinky
      ringToPinky){}
```

Listing 4.12 – Método *FingerRelation* alternativo

Neste método utilizamos para cada parâmetro um objeto com nome que corresponde a relação no qual queremos configurar um valor. Esses objetos foram criados para serem utilizados especialmente neste método. Cada um desses objetos contém métodos que correspondem a cada um dos valores possíveis para as relações entre dedos em Leap Gesture. Esse método possibilita uma maior legibilidade ao código, evita que o usuário erre a sequência dos parâmetros e permite a exploração dos recursos da *IDE*.

No código 4.13 temos um exemplo de utilização deste método alternativo para a cláusula *FingerRelation*.

```
1 gesture
      //Primeira pose.
3     .pose ()
          .FingerRelation(
5             ThumbToIndex.group(),
              ThumbToMiddle.separate(),
7             ThumbToRing.separate(),
              ThumbToPinky.separate(),
9             IndexToMiddle.group(),
              MiddleToRing.group(),
11            RingToPinky.group()
          )
13
      //Segunda pose.
15    .pose()
          .FingerRelation(
17            ThumbToIndex.separate(),
              ThumbToMiddle.separate(),
19            ThumbToRing.separate(),
              ThumbToPinky.separate(),
21            IndexToMiddle.separate(),
              MiddleToRing.separate(),
```

```
23                  RingToPinky.separate()
                )
25  ;//fim do gesto
```

Listing 4.13 – Utilização da cláusula *FingerRealtion* com o método alternativo

Com esses dois métodos é possível obter os mesmos resultados em Leap Gesture. O segundo método mostrado oferece uma maior legibilidade ao *script* o que pode trazer um maior benefício para quem o lê posteriormente, enquanto o primeiro é mais direto, precisando apenas dos valores desejados, poupando mais tempo na hora de sua escrita.

4.3 Implementação da DSL

Esta seção contém uma visão geral da implementação de Leap Gesture. A linguagem é até o presente momento composta por dois subprojetos. O primeiro subprojeto é o módulo responsável pelo serviço de reconhecimento e rastreamento do usuário através do sensor de profunidade. Este projeto é chamado de *GestureManager* e foi construído para exercer a função de um *modelo semântico* da linguagem, propiciando uma semântica computacional as setenças expressas pela linguagem. O segundo subprojeto refere-se à implementação do *construtor de expressões*, encarregado de estabelecer uma gramática e algo análogo a uma análise sintática e, portanto, pelo mapeamento e preenchimento do modeo semântico da DSL Leap Gesture.

Para a implementação da linguagem Leap Gesture foi utilizada a LPPG *C#* e a *engine* de jogos *Unity*. Leap Gesture poderá ser utilizada com qualquer aplicação que possa ser executada pela mesma plataforma. A engine *Unity* está entre as mais utilizadas para o desenvolvimento de jogos, além de ter uma boa integração com o dispositivo de profundidade utilizado. Por esta razão, este trabalho escolhe esta plataforma para que o maior número possível de desenvolvedores seja alcançado.

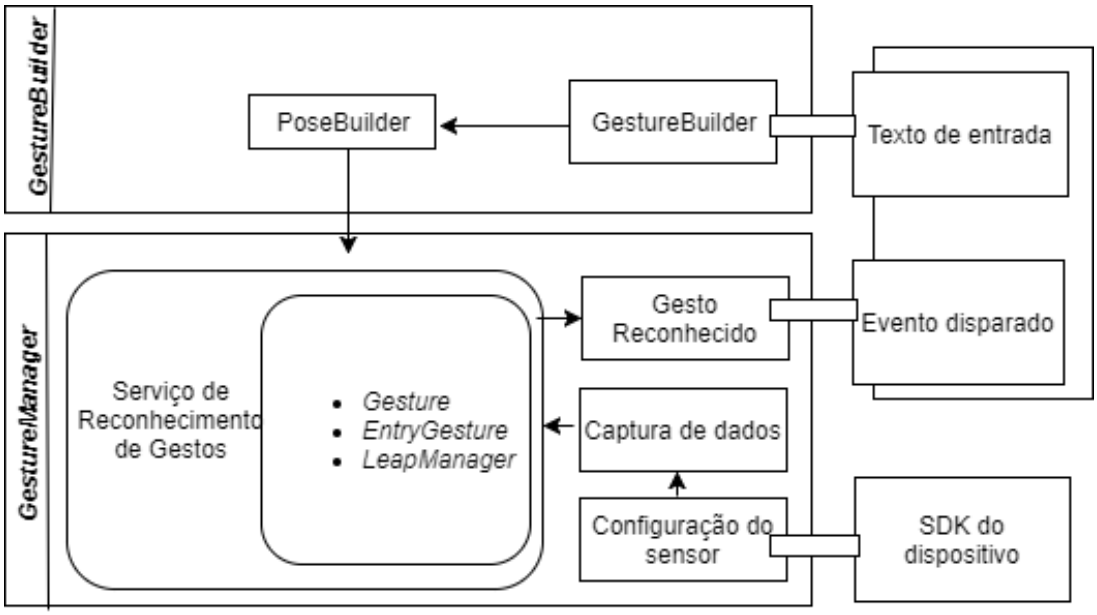
As seções seguintes descrevem em detalhes a arquitetura de cada módulo e apresenta discussões sobre as decisões de implementação.

4.3.1 Arquitetura da DSL

Leap Gesture é uma DSL interna que utiliza a infraestrutura da linguagem C# que por sua vez está acoplada a plataforma de desenvolvimento de jogos *Unity*. Esta DSL é formada por dois elementos base: o construtor de expressões, módulo responsável por converter as chamadas realizadas através das sentenças de Leap Gesture em chamadas ao módulo subjacente (análogo a uma análise sintática) e o modelo semântico, módulo responsável por avaliar os gestos especificados no momento em que aplicação está sendo executada. O construtor de expressões define quais as sentenças do domínio que são válidas durante a especificação do gesto. O construtor de expressões também é responsável pelo

preenchimento do modelo semântico. No modelo semântico temos as regras necessárias para avaliação dos gestos especificados que após serem preenchidos pelo modelo semântico faz uma série de chamadas a outras APIs utilizadas durante todo o processo de execução da DSL. A Figura 15 apresenta a arquitetura da linguagem Leap Gesture.

Figura 15 – Arquitetura da linguagem Leap Gesture



4.3.2 *GestureBuilder*: O construtor de expressões de Leap Gesture

Um construtor de expressões é um objeto que fornece uma interface fluente, que, então, se traduz em chamadas em uma API comando-consulta subjacente (FOWLER, 2010). Em Leap Gesture, o construtor de expressões é utilizado para traduzir as expressões formadas pelas diversas cláusulas existentes, organizando o fluxo de chamadas às funções do modelo semântico. Há diversas maneiras de se organizar um construtor de expressões e isto varia de acordo com as cláusulas no qual se está trabalhando. A maneira como organizamos o construtor de expressões de Leap Gesture foi através da técnica de Encadeamento de Métodos (*Method Chaining*). Como sugerido por Fowler (2010), o Encadeamento de Métodos funciona bem quando temos cláusulas que são opcionais na construção de uma determinada expressão, por esta razão optamos por utilizá-la nos contrutores de expressões.

Temos essencialmente em Leap Gesture uma forma de descrever gestos para as mãos como está descrito no código 4.14.

```
1
GestureBuilder gesture = new GestureBuilder();
3
gesture
5    .pose()
        .HandLocation("1")
```

```
7           .FingerPose ("POINTING_SIDE", "POINTING_UP", "POINTING_UP", "CLOSE"
    , "CLOSE")
            .FingerRelation ("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "
    SEPARATE", "SEPARATE", "GROUP")
9       .pose ()
            .HandLocation ("3")
11          .FingerPose ("POINTING_SIDE", "POINTING_UP", "POINTING_UP", "CLOSE"
    , "CLOSE")
            .FingerRelation ("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "
    GROUP", "SEPARATE", "GROUP")
13  ;
```

Listing 4.14 – Exemplo de utilização de Leap Gesture

Para realizar isto, foram criados dois construtores de expressões, um para *gestures* e outro para *pose*. O construtor de *gesture* guarda uma lista de construtores de *pose*, e o método pose retorna um construtor de poses, iniciando uma expressão.

```
1
class GestureBuilder...
3
    private List<PoseBuilder> poses = new List<PoseBuilder>();
5
    public PoseBuilder pose(){
7       PoseBuilder child = new PoseBuilder(this);
        poses.Add(child);
9       return child;
    }
```

Listing 4.15 – Construtor de *gesture*

Através da chamada de métodos que fazem um trabalho análogo ao que seria uma análise sintática, o construtor de poses captura os dados especificados nas cláusulas em seus próprios campos. Em cada chamada a um desses métodos, o construtor retorna a si próprio de forma a continuar a cadeia.

```
1 class PoseBuilder {

3     private GestureBuilder parent;
      private string handLocation;
5     private string palmOrientation;
      private string fistFaceOrientation;
7     private FingerPose fingerPose;
      private FingerRelation fingerRelation;
9
      public PoseBuilder(GestureBuilder parent) {
11        this.parent = parent;
      }
13
```

```
    public PoseBuilder FingerPose(Thumb thumb,...){
15      fingerPose = new FingerPose(thumb.getValue() ,...);
    return this;
17  }

19  public PoseBuilder FingerPose(string thumb, string index, ...){
        fingerPose = new FingerPose(thumb, index, ...);
21      return this;
    }
23
    public PoseBuilder FingerRelation(ThumbToIndex thumbToIndex,...){
25      fingerRelation = new FingerRelation(thumbToIndex.getValue() ,...);
        return this;
27  }

29  public PoseBuilder FingerRelation(string thumbToIndex,...){
        fingerRelation = new FingerRelation(thumbToIndex,...);
31      return this;
    }
33
    public PoseBuilder HandLocation(string location){
35      handLocation = location;
        return this;
37  }

39  public PoseBuilder PalmOrientation(string orientation){
        palmOrientation = orientation;
41      return this;
    }
43
    public PoseBuilder FistFaceOrientation(string orientation){
45      fistFaceOrientation = orientation;
        return this;
47  }
```

Listing 4.16 – Construtor de poses

O método pose indica o início da próxima *pose* a ser especificada. A chamada a este método, no construtor de poses, delega para seu pai a criação de um novo objeto construtor de poses.

```
1 class PoseBuilder...
    public PoseBuilder pose(){
3       return parent.pose();
    }
```

Listing 4.17 – Método responsável por iniciar uma nova pose

Com estes dois construtores podemos criar qualquer expressão definidas por Leap Gesture.

4.3.3 *GestureManager*: O modelo semântico de Leap Gesture

O *GestureManager* de Leap Gesture, é o modelo semântico desenvolvido para executar os gestos especificados. Nele estão contidas todas as classes utilizadas durante o processo de validação entre os gestos especificados e executados. Cada uma dessas classes contém um comportamento que formam o todo do processo de avaliação. É neste módulo que evoluções, como a construção de novos recursos bem como melhorias nos já existentes, poderão ser realizadas. Estão presente nas próximas sessões os detalhes dos recursos encontrados em *GestureManager*.

4.3.3.1 *Gesture*: o modelo do gesto

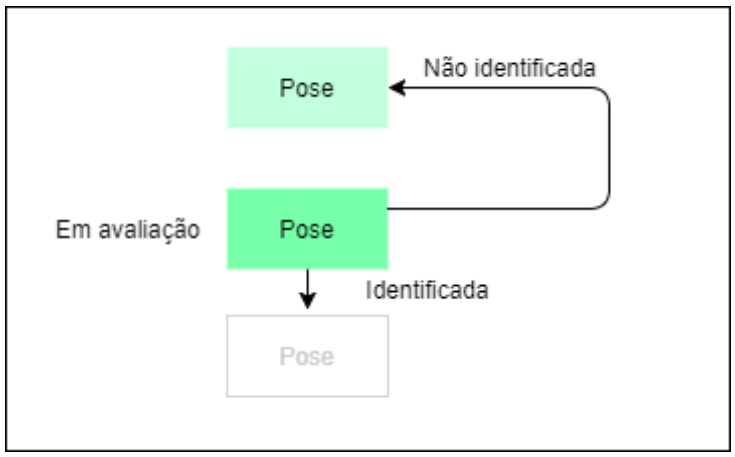
Por meio do script escrito com o construtor de expressões (*GestureBuilder*) de Leap Gesture, obtemos um objeto do tipo *Gesture* onde ficam salvos as informações de cada detalhe especificado. Neste objeto está contida uma lista de *pose* que ao serem identificadas como válidas durante a execução disparará um evento de gesto reconhecido. Leap Gesture foi desenhada de forma a utilizá-la juntamente aos componentes da plataforma de desenvolvimento *Unity*, assim, através do método *Update()* que é invocado a cada *frame*, o objeto *Gesture* verifica se as mãos do usuário se encontram de acordo com a ordem de cada *pose* especificadas na construção do gesto. Cada *pose* é tratada por ordem de especificação, assim, a primeira *pose* a ser definida através do construtor de expressões *GestureBuilder* é a primeira a ser avaliada e assim sucessivamente. Na medida em que uma *pose* é identificada como válida, a próxima na fila de verificação passa a ser avaliada. Caso no *frame* corrente nenhuma *pose* seja identificada, a avaliação no *frame* seguinte continuará exatamente do ponto anterior e todos os gestos realizados anteriormente serão considerados válidos até que a *pose* corrente seja identificada. Com a identificação da última *pose*, o objeto *Gesture* emite um evento sinalizando o reconhecimento do gesto. A figura 16 ilustra o estado de cada *pose* durante o processo de reconhecimento.

Para avaliar cada pose o objeto *Gesture* utiliza um serviço auxiliar chamado *EntryGesture*. A solicitação a esse serviço é efetuado pela invocação do método *CheckPose(bool left, bool right, Pose pose)*, um método estático disponível pela classe *EntryGesture* classe responsável pela implementação do serviço .O serviço *EntryGesture* também faz parte do módulo semântico e será detalhado na sessão a seguir.

4.3.3.2 *EntryGesture*

O serviço *EntryGesture*, pertencente ao módulo semântico, é responsável por avaliar se as definições especificadas em cada pose estão de acordo com o que é executado pelo usuário.

Figura 16 – Estados de rastreamento de uma *pose* durante avaliação de um gesto.



Aqui ele checa individualmente quais cláusulas foram especificadas e se suas definições são iguais ao que está sendo executado. Esse serviço é implementado pela classe *EntryGesture*, mediante o método *CheckPose(bool left, bool right, Pose pose)*, onde a pose é avaliada.

Primeiramente o serviço checa se o gesto especificado foi configurado para mão esquerda, direita ou ambas. Na sequência ele avalia as especificações de cada *pose*. Em Leap Gesture existem cinco cláusulas (*HandLocation*, *HandOrientation*, *FistFaceOrientation*, *FingerPose*, *FingerRelation*) possíveis para especificar uma *pose* e a utilização de cada uma destas cláusulas é opcional, desse modo, o serviço *EntryGesture* checa se há uma definição para cada uma dessas cláusulas. Havendo uma definição, o serviço compara com o que está sendo executado. Isso é realizado através da chamada a outro serviço existente no modelo semântico, o LeapManager, que será explicado em maiores detalhes na próxima seção. Ao final de sua execução, o método *CheckPose*, retornará um evento de pose válida, caso todas as definições especificadas sejam compatíveis com a *pose* (gesto estático) executada.

4.3.3.3 *LeapManager*

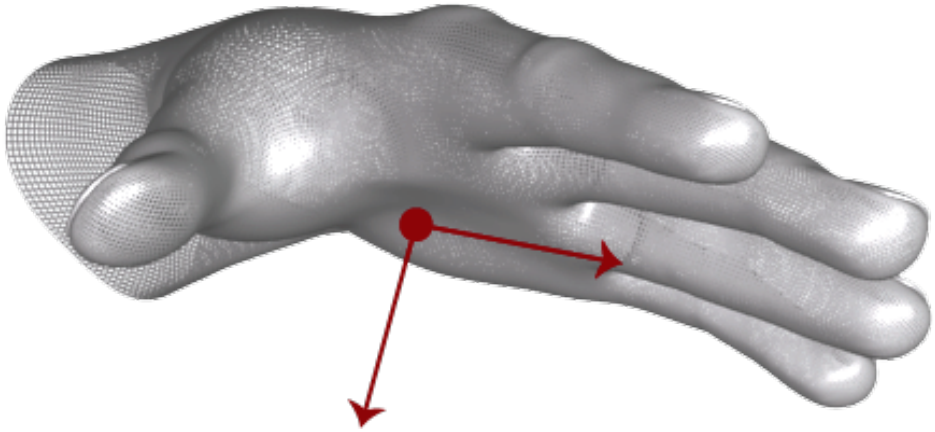
O módulo semântico dispõe de um classe responsável pela comunicação de Leap Gesture com o sensor utilizado, a *LeapManager*. Essa *classe* manipula o SDK fornecido pelo *Leap Motion*, extraindo os dados capturados pelo sensor e realizando cálculos para classificar, de acordo com a notação utilizada em Leap Gesture, quais as configurações realizadas pelo usuário no momento em que o gesto está sendo executado.

O dispositivo de controle *Leap Motion* rastreia mãos e dedos disponibilizando informações, como posição, velocidade e orientação. O sistema do *Leap Motion* é composto de um dispositivo de *hardware* (o sensor) e o componente de *software*. O componente de *software* analisa imagens geradas pelo hardware e envia informações de rastreamento (posição, velocidade e orientação) às aplicações. O SDK acessa as informações fornecidas pelo *software* para obter os dados. Através do SDK é possível ter acesso a uma série de

informações: a posição de cada falange[1] da mão em três dimensões, funções matemáticas, matrizes de direção e funções auxiliares que auxiliam na classificação dos gestos.

Na *classe LeapManager* existem um conjunto de métodos encarregados pela classificação dos gestos realizados pelo usuário. O método *getHandOrientation(Hand hand)* retorna as informações a respeito da orientação da mão. Em Leap Gesture a orientação da mão é definida pela orientação da palma da mão (*PalmOrientation*) e pela orientação do punho (*FistFaceOrientation*). Para realizar a classificação, o método *getHandOrientation(Hand hand)* faz uso de duas funções auxiliares disponibilizadas pelo SDK do *Leap Motion*: *PalmNormal* e *Direction*. A função *PalmNormal* retorna um vetor perpendicular ao plano formado pela palma da mão. O vetor aponta para a direção da palma da mão. A função *Direction* retorna um vetor que aponta do centro da palma da mão em direção aos dedos. A figura 17 ilustra os vetores retornados por essas funções.

Figura 17 – Vetores *PalmNormal* e *Direction*.



GetHandLocation(HandList myHandList) é o método responsável por classificar a localização da mão do usuário (cláusula *HandLocation*). O método estabelece uma zona dividida em seis áreas a partir do alcance máximo da visão do dispositivo *Leap Motion*. Esta zona é criada a partir dos componentes x e y (plano cartesiano) de um vetor resultante da função *PalmPosition* disponível no SDK. A função *PalmPosition* retorna um vetor de três dimensões referente a posição do centro da palma da mão em relação ao ponto de origem do *Leap Motion*.

Para classificar a cláusula *FingerPose* temos o método *GetFingerPose(Finger finger)*. As diferentes configurações dos dedos são obtidas a partir da combinação dos valores de alguns parâmetros. O método utilizado se baseia em três parâmetros para identificar a configuração que o dedo se encontra, a saber: *isExtend*, *levelBending*, *anglePalm*. Esses parâmetros são encontrados a partir de alguns cálculos e com a utilização de métodos nativos do *Leap Motion*.

[1] As falanges são os ossos que formam os dedos das mãos e pés dos vertebrados. Dicionário Michaelis

Fonte: Autoria própria

Parâmetro	**Descrição**
isExtend	Propriedade nativa do SDK. Retorna *true* caso o dedo esteja estendido.
levelBending	Nível de flexão (curvatura do dedo).
anglePalm	Ângulo da palma da mão.Retorna o ângulo formado por a direção da falange proximal do dedo em relação a direção da palma da mão.

Tabela 1 – Parâmetros para reconhecimento da cláusula *FingerPose*.

O *isExtend* é uma propriedade nativa do SDK e retorna um valor *boolean* verdadeiro quando o dedo está completamente estendido. O parâmetro *levelBending* define, em números, o quanto um dedo está flexionado. Para chegar a esse valor é realizado o seguinte cálculo:

$$levelBending] = 1.0 - (1.0 + n)/2.0$$

onde n é o produto escalar dos vetores unitários das falanges proximal.

As falanges de cada dedo podem ser obtidas a partir do método *Bone(Bone.BoneType type)* onde *type* corresponde a falange que desejamos obter. Para calcular o parâmetro *anglePalm* nós calculamos o ângulo entre dois vetores, a direção da palma da mão e a direção da falange proximal do dedo, a fórmula utilizada para encontrar esse ângulo pode ser descrita como:

$$\cos\varphi = \frac{< \upsilon, \mu >}{|\upsilon| \cdot |\mu|}$$

Na tabela 2 é exibido a combinação dos valores que determinam a configuração para cada dedo.

Por último, o método que classifica a relação entre os dedos (cláusula *InterFingerRelation*). A classificação desta cláusula fica a cargo do método *getInterRelation(Hand hand)*. Este método funciona de maneira análoga ao método *getIFingerPose(Hand hand)*, avaliando uma combinação de parâmetros para classificar as configurações possíveis de relação entre os dedos. O método utiliza os parâmetros *IsExtend*, *PinchStrength*, *GrabStrength*, *Angle*. O parâmetro *PinchStrength* é obtido por meio da propriedade homônima do SDK. O parâmetro *PinchStrength* determina a força de preensão de um gesto de pinça. Quando a mão se encontra aberta o valor da força é zero e varia até 1.0 quando um gesto de pinça for reconhecido. O gesto de pinça pode ser realizado pelo polegar e qualquer outro dedo da mesma mão. O parâmetro *GrabStrength* determina a força do gesto de agarrar (mão fechando). O valor da força é zero para uma mão aberta e varia até 1.0 quando um gesto de agarrar for reconhecido. O parâmetro *angle* define, em graus, a distância entre dois

Fonte: Autoria própria

*Estes valores são utilizados para definir a configuração do polegar.

Configuração	**Valores**
Pointing (up)	IsExtend, levelBending <= 0.10, anglePalm <= 115f
Pointing (up)*	IsExtended, angleIndexThumb <= 30f
Pointing (Forward)	IsExtended, levelBending <= 0.10f, anglePalm <= 115f
Pointing (Side)*	IsExtended, levelBending <= 0.20f, angleIndexThumb > 30f
Bend	IsExtended, levelBending >= 0.10f, levelBending <= 0.59f
Bend*	IsExtended, levelBending >= 0.10f, levelBending <= 0.59f, anglePalm < 149f
Closed	IsExtended, levelBending >= 0.40f, anglePalm <= 120f
Closed*	IsExtended, levelBending >= 0.10f, levelBending <= 0.59f, anglePalm > 149f

Tabela 2 – Combinação de valores para regra da cláusula *FingerPose*.

dedos e tem seu valor encontrado com base no cálculo utilizado para encontrar o ângulo entre dois vetores. Baseado nestes parâmetros, a tabela 3 apresenta as combinações para cada configuração de relação entre os dedos.

Fonte: Autoria própria

*Estes valores são utilizados para definir a configuração do polegar.

Configuração	**Valores**
Separate	angle > 5f ou angle < -11f
Separate*	Regra padrão, caso nenhuma das outras regras se apliquem.
Group	angle > -11f, angle < 4f
Group*	angleThumb < 0f, fingers[i+1].Type == Finger.FingerType.TYPE$_INDEX, dotThumb >= 0.89f$
Cross	angle < -11f, fingers[i].IsExtended, fingers[i+1].IsExtended
Cross*	Math.Abs(angleThumb) > 100f, PinchStrength < 0.8f, PinchStrength > 0.4f, dotThumb < 0f, GrabStrength != 1
Touch*	dotThumb < 0, PinchStrength > 0.9f

Tabela 3 – Combinação de valores para regra da cláusula *FingerRelation*

4.4 Considerações finais

Este capítulo apresentou de forma breve o processo de implementação da DSL Leap-Gesture, basicamente formada por dois módulos, o modelo semântico responsável por identificar os gestos realizados pelo usuário e o construtor de expressões responsável por interpretar o *script* especificado, mapeando as sentenças para o modelo semântico.

As limitações da DSL Leap Gesture, como ruídos sintáticos e ausência de cláusulas que acabam por limitar sua expressividade, são causadas pelas limitações encontradas no dispositivo, como exemplo a incapacidade de detectar as mãos diante de uma oclusão, erros em estimar a posição das articulações e informações em baixo nível e insuficientes provenientes de sua API. Além disto, o tempo da pesquisa não foi suficiente para encontrar uma abordagem que possa superar estes desafios encontrados.

5 AVALIAÇÃO

O presente capítulo apresenta uma avaliação experimental acerca da utilização da linguagem Leap Gesture envolvendo um conjunto de gestos realizados pelas mãos descritos na literatura, onde foi identificado a oportunidade de especificação dos mesmos através da linguagem Leap Gesture. O objetivo é avaliar o poder de expressividade da lingaugem e sua eficiência do reconhecimento considerando cada gesto em tempo de execução. Para tanto, este capítulo está organizado da seguinte forma: inicialmente, a Seção 5.1 apresenta a metodologia aplicada e o conjunto de gestos selecionados à avaliação experimental. A Seção 5.2 apresenta, em detalhes, a especificação em Leap Gesture e posterior avaliação de cada gesto. Por fim, na Seção 5.3 são discutidos os resultados obtidos, juntamente com uma análise a respeito da utilização da linguagem Leap Gesture.

5.1 Metodologia

A proposta da linguagem Leap Gesture é possibilitar aos desenvolvedores uma abordagem para especificação de gestos que seja capaz de abstrair cálculos e algoritmos complexos. Para tal fim, a linguagem conta com uma notação específica voltada ao domínio dos movimentos das mãos e recursos adicionais que auxiliam na construção de reconhecedores de gestos. Para avaliar a capacidade de Leap Gesture em especificar gestos, foi construída uma aplicação para reconhecer um conjunto de gestos previamente estabelecidos.

Para a montagem do contexto de avaliação foram selecionados 15 (quinze) gestos a partir de conjunto de gestos provenientes de um experimento realizado por Choi, Kim e Chung (2014). Os gestos escolhidos são variados e proporcionam uma boa abrangência para a avaliação proposta. A Figura 18 ilustra o conjunto de gestos escolhidos para a avaliação.

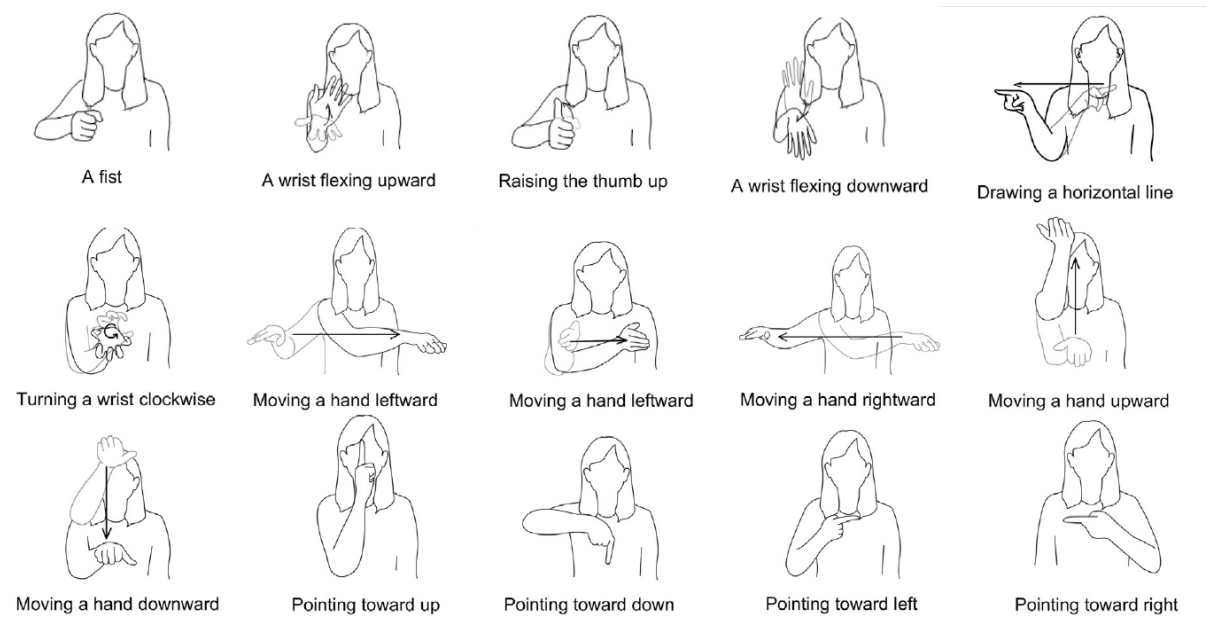
Após a escolha dos gestos, foram criadas as especificações para cada um deles com a linguagem Leap Gesture. Para avaliar a expressividade e o processo de reconhecimento de Leap Gesture foi realizado a seguinte atividade:

- Para avaliar o *script* especificado para cada gesto, foi repetido por 10 (dez) vezes o mesmo gesto a fim de verificar sua efetividade em reconhecer o gesto.

O ambiente utilizado durante o processo de avaliação foi uma sala fechada, com baixa incidência de luz natural e iluminado artificialmente por lâmpada fluorescente.

Para execução dos testes foi utilizado um *notebook* com processador Intel® Core™ 2.60GHz, com memória de 6 GB RAM e processamento efetuado no sistema operacional *Windows 8.1 Single Language x64*

Figura 18 – Gestos escolhidos para avaliação.



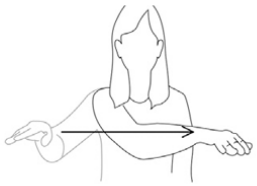
5.2 Expressividade da linguagem

A primeira parte da avaliação experimental consiste em testar o poder de expressividade da linguagem. Para cada um dos gestos escolhidos foi implementado um *script* para reconhecê-lo. O objetivo de tal atividade é testar as possibilidades e capacidade que a linguagem Leap Gesture oferece para especificar reconhecedores para determinados gestos.

Temos na figura 19 um exemplo da especificação de um gesto em Leap Gesture. É válido ressaltar que a escolha dos gestos se deve a sua variedade quanto ao modo de execução (variedade de movimentos e formas que as mãos assumem durante o gesto) e por serem criados especificamente para interação natural.

Figura 19 – Exemplo de especificação de Gesto em Leap Gesture

```
//Specifying the gesture
gesture
    .pose() //First pose
        .HandLocation("4")
        .FingerPose("POINTING_UP", "POINTING_UP", "POINTING_UP","POINTING_UP", "POINTING_UP")
        .FingerRelation("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "GROUP", "GROUP", "GROUP")
        .PalmOrientation("BOTTOM")
        .FistFaceOrientation("FORWARD")
    .pose() //Second pose
        .HandLocation("6")
        .FingerPose("POINTING_UP", "POINTING_UP", "POINTING_UP","POINTING_UP", "POINTING_UP")
        .FingerRelation("SEPARATE", "SEPARATE", "SEPARATE", "SEPARATE", "GROUP", "GROUP", "GROUP")
        .PalmOrientation("BOTTOM")
        .FistFaceOrientation("FORWARD")
;
```



A segunda parte da avaliação é estimar a capacidade de reconhecimento através das especificações obtidas. Para quantificar a performance do reconhecimento para cada gesto,

foram seguidos os passos descritos na seção 5.1, considerando o total de vezes em que foram identificados corretamente.

A tabela 4 apresenta os gestos especificados com Leap Gesture e a taxa de acerto relativa a identificação dos mesmos.

Com relação aos resultados é notável que Leap Gesture teve êxito ao reconhecer todos os gestos propostos pela avaliação, porém não em 100% das vezes em que foram executados. A baixa na taxa de acerto em alguns gestos se deve a sua complexidade, pois apresentam maior variedade de movimentos e configuração dos dedos. Toda esta variedade de movimentos que envolve o gesto resulta em problemas de oclusão que acaba por prejudicar o rastreamento por parte do dispositivo. Durante os testes o dispositivo mostrou-se muito inconsistente apresentando dados não compátiveis aos que estavam sendo executados. A linguagem Leap Gesture utiliza os dados provenientes do dipositivo que faz o rastreamento, se estes dados não estiverem iguais aos executados, logo o gesto não será reconhecido.

Outro problema relacionado ao rastreameto por parte do dispositivo utilizado é a interferência devido a incidência de luz natural. Como o dispositivo *Leap Motion* faz uso de luz infravermelha para auxiliar o rastreamento das mãos, a luz natural por emitir, também, luz infravermelha, acaba por interferir no resultado final do rastreamento. Para diminuir estas interferências os testes foram executados em ambiente fechado.

Tabela 4 – Resumo com os resultados da Avaliação

Gesto	**Taxa de acerto (%)**
A fist	80%
A wrist flexing upward	80%
Raising the thumb up	70%
A wrist flexing downward	90%
Drawing a horizontal line	60%
Turning a wrist	50%
Moving a hand leftward	70%
Moving a hand leftward	70%
Moving a hand rightward	70%
Moving a hand upward	100%
Moving a hand downward	90%
Pointing toward up	50%
Pointing toward down	80%
Pointing toward left	70%
Pointing toward right	70%

Fonte: Autoria Própria

Apesar da utilização de um ambiente fechado para execução dos testes e realizar os

movimentos de forma que a disposição dos dedos não permanecesse oclusa ao dispositivo, alguns gestos não foram rastreados conforme o que estava sendo executado.

5.3 Análise e conclusão da avaliação

A primeira parte da avaliação mostrou que a linguagem Leap Gesture, através de seus recursos, foi capaz de especificar o conjunto de gestos proposto. Os *scripts* produzidos para cada gesto resultaram em poucas linhas de código, além de favorecer um fácil entendimento do que está especificado, muito disto devido a própria natureza de uma DSL. A presente avaliação não compara o número de Linhas de Código resultante entre gestos descritos com Leap Gesture e uma LPPG (abordagem baseada em regras). Tal avaliação torna-se difícil de ser executada devido a grande variedade em que os algoritmos podem ser implementadas por meio de uma abordagem baseada em regras. Deste modo, seria difícil obter o melhor caso (menor números de linhas de código) para então comparação com o *script* gerado pela DSL para o mesmo gesto. Apesar disto, não é difícil supor que o resultado produzido pela abordagem baseada em regras teria um maior número de linhas de código em comparação a uma DSL.

Um outro questionamento que surge ao compararmos estas abordagens entre si é o desempenho de seus algoritmos, em outras palavras o custo computacional envolvido durante o processo de reconhecimento. Por conta de sua natureza declarativa é difícil empregar uma métrica para mensurar a complexidade do código. Apesar disto, foi realizado uma avaliação do processo de reconhecimento de Leap Gesture. A segunda parte da avaliação foi responsável por verificar a precisão da linguagem em identificar os gestos executados. Apesar das limitações no reconhecimento, é possível que este problema seja superado com a evolução no módulo da linguagem (modelo semântico) respónsavel por esta tarefa. Através da melhoria deste módulo seria tanto possível otimizar o desempenho da linguagem (custo computacional) quanto torná-la mais precisa em identificar corretamente os gestos capturados.

É oportuno destacar que o foco do presente trabalho não consiste em abordar elementos de otimização de algortimos de reconhecimento de gestos ou técnicas de processamento de imagens. Este trabalho tem como um de seus objetivos mostrar que através da elevação do nível de abstração é possível construir reconhecedores de gestos, para dispositivos que se proponham a rastrear movimentos, de maneira mais produtiva.

Uma possível análise para ser executa posteriormente é a latência entre a execução do gesto e sua identificação em Leap Gesture. Este baixo tempo de resposta, de modo quase imperceptível é imprescindível para domínios que precisem de uma interação em tempo real.

5.4 Conclusão

Este capítulo apresentou a avaliação experimental do uso da linguagem Leap Gesture como ferramenta para construção de reconhecedores de gestos à sensores de profundidade. Foram escolhidos 15 (quinze) gestos a partir da literatura baseados no uso das mãos para interação natural. Os gestos escolhidos foram especificados na linguagem Leap Gesture. Na avaliação, ficou evidenciado que os gestos testados foram reconhecidos com uma taxa média de acerto de 73% pelo sensor *Leap Motion*.

6 CONCLUSÕES

A presente dissertação apresentou uma linguagem específica de domínio para especificação e reconhecimento de gestos envolvendo as mãos. A linguagem é uma DSL textual, declarativa e interna, que apesar disto, possui em seu conceito uma sintaxe particular. Esta linguagem foi desenvolvida de tal forma que o usuário descreve o gesto como código de entrada, através da chamada de métodos (construtor de expressões), para então preencher o modelo semântico da linguagem que é responsável por reconhecer os gestos especificados em tempo de execução, por meio dos dados obtidos pelo sensor. Posteriormente, com o propósito de avaliar a utilização da linguagem foi executada uma avaliação experimental, através do uso da mesma para especificar gestos das mãos baseados em um conjunto de gestos para interação natural encontrados na literatura.

Durante as fases de concepção e de desenvolvimento da linguagem proposta, foram avaliadas diversas propostas quanto a expressividade da linguagem e as tecnologias responsáveis pelo processo de reconhecimento dos gestos. Portanto, neste capítulo são apresentadas algumas considerações a respeito das decisões tomadas durante este processo de conceptualização e implementação da linguagem. Além disto, são apresentados também os principais resultados obtidos durante a construção deste trabalho e também algumas propostas de trabalhos futuros.

Um dos maiores desafios encontrados durante a construção de uma DSL é a abstração realizada diante de um domínio qualquer. Durante o desenvolvimento deste trabalho foi preciso investigar os modelos de representação de movimentos das mãos existentes e de que forma estes modelos poderiam ser utilizados num modelo computacional. Este desafio torna-se ainda maior tendo em vista as limitações do dispositivo de reconhecimento utilizado, tanto de sua precisão na captura dos dados quanto a forma em que esses dados encontram-se disponíveis e no fato de que a sintaxe deva ser facilmente compreendida por usuários não-técnicos. Portanto, foi utilizada uma notação que poderia ser facilmente mapeada para um modelo computacional.

Um das primeiras decisões tomadas durante a fase inicial deste projeto foi a escolha entre a construção de uma DSL interna ou externa. A decisão pela escolha em construir uma DSL interna foi pelo tempo de implementação que é relativamente mais rápido se comparado com a construção de uma linguagem externa. Isto se deve ao fato de que uma linguagem interna se beneficia dos recursos da linguagem hospedeira, poupando assim esforços por evitar o desenvolvimento de um interpretador propriamente dito e mecanismos de controle. Apesar de ser construída em menos tempo, uma DSL interna traz consigo os ruídos causados pela sintaxe da linguagem hospedeira. Contudo, esta DSL tem como proposta ser uma ferramenta que auxilie o desenvolvimento de sistemas que

possuam interação natural, de forma que a especificação dos reconhecedores de gestos seja fácil e rápida se comparada com as abordagens tradicionais, assim o público para o qual ela se destina está de alguma forma adaptado aos ruídos sintáticos contidos na DSL proposta.

A DSL proposta por este trabalho mostrou-se consistente em sua tarefa de especificar e reconhecer os gestos livres das mãos para interação natural, cumprindo seu principal objetivo. Na avaliação experimental, a linguagem foi capaz de especificar os gestos selecionados e abstrair toda a complexidade que envolve a realização de reconhecimento.

Deste modo, se pode concluir que o presente trabalho foi capaz de alcançar seus objetivos, oferecendo uma solução de fácil utilização com objetivo de facilitar a especificação e o reconhecimento de gestos por meio de uma notação textual que possa ser rapidamente aprendida e compreendida por desenvolvedores e não desenvolvedores.

6.1 Trabalhos futuros

Embora tenha se alcançado bons resultados através do presente trabalho, acreditamos que há tópicos a serem explorados e melhorados para otimizar a construção de reconhecedores de gestos por meio da solução proposta. Aqui, estão listados alguns dos trabalhos que pretende-se dar continuidade e outros que poderão ser executados futuramente.

- A construção desta mesma DSL nas diversas plataformas/linguagens no qual já existe uma API do dispositivo *Leap Motion*: *Python*, C#, *Unreal* e outras disponíveis.

- Melhorar o módulo *LeapManager* responsável por detectar as configurações assumidas pelas mãos. Leap Gesture obteve 73% de acerto durante a avaliação, muito disto devido as limitações do sensor (imprecisão nos dados obtidos, interferência por iluminação e etc.), porém torna-se claro a necessidade e possibilidade de melhorias no módulo de reconhecimento, pois isto tem um impacto imediato na experiência do usuário final.

- Nem todos elementos da notação proposta por Choi, Kim e Chung (2014) foram implementadas nesta versão de Leap Gesture. O sensor não se mostrou confiável em suas informações quando diante de poses em que os dedos se encontram oclusos ou muito próximos, além disto o sensor não apresentou resultados satisfatórios quando as mãos encontram-se em contato. Uma possível melhoria na API oferecida pelo dispositivo possibilitaria a inclusão das cláusulas oferecidas por Choi, Kim e Chung (2014) que não estão presentes em Leap Gesture.

- Extensão de Leap Gesture para uma DSL externa. As DSLs externas reduzem os ruídos sintáticos presentes em uma DSL interna (chaves, vírgulas, pontos e etc.). Uma DSL externa possibilita ainda mais uma notação legível ao usuário.

- Compatibilidade com mais dipositivos. Existe a possibilidade de Leap Gesture poder trabalhar com outros sensores, ainda que nem todos possam disponibilizar informações suficientes para que o rastreamento seja feito. Para contornar isto a DSL poderia informar se uma determinada cláusula é disponível ou não para o sensor em questão.

REFERÊNCIAS

AGGARWAL, J. K.; CAI, Q. Human motion analysis: A review. In: IEEE. *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*. [S.l.], 1997. p. 90–102.

AMARAL, W. M. d. *Sistema de transcrição da língua brasileira de sinais voltado à produção de conteúdo sinalizado por avatares 3d. 2012. 241 f.* Tese (Doutorado) — Tese (Doutorado em Engenharia Elétrica). Universidade Estadual de Campinas, Campinas-SP, 2012.

AMON, C.; FUHRMANN, F.; GRAF, F. Evaluation of the spatial resolution accuracy of the face tracking system for kinect for windows v1 and v2. In: *Proceedings of the 6th Congress of the Alps Adria Acoustics Association*. [S.l.: s.n.], 2014. p. 16–17.

ASUS. *Xtion PRO LIVE*. [S.l.]: Asus, 2017. Https://www.asus.com/us/3D-Sensor/Xtion$_{P}RO_{L}IVE$/.

BASSILY, D.; GEORGOULAS, C.; GUETTLER, J.; LINNER, T.; BOCK, T. Intuitive and adaptive robotic arm manipulation using the leap motion controller. In: VDE. *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*. [S.l.], 2014. p. 1–7.

BATTISON, R. Lexical borrowing in american sign language. ERIC, 1978.

BOLT, R. A. *"Put-that-there": Voice and gesture at the graphics interface*. [S.l.]: ACM, 1980. v. 14.

BRUCE, D. What makes a good domain-specific language? apostle, and its approach to parallel discrete event simulation. *Kamin [43]*, p. 17–35, 1997.

BUXTON, B. et al. Multi-touch systems that i have known and loved. *Microsoft Research*, v. 56, p. 1–11, 2007.

CHAMBERLIN, D. D.; BOYCE, R. F. Sequel: A structured english query language. In: *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. New York, NY, USA: ACM, 1974. (SIGFIDET '74), p. 249–264. Disponível em: <http://doi.acm.org/10.1145/800296.811515>.

CHOI, E.; KIM, H.; CHUNG, M. K. A taxonomy and notation method for three-dimensional hand gestures. *International Journal of Industrial Ergonomics*, Elsevier, v. 44, n. 1, p. 171–188, 2014.

COLGAN, A. *How Does the Leap Motion Controller Work?* [S.l.]: Leap Motion Blog, 2014. <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work>. Acesso em: 17 abr. 2017.

COSTA, P. M. F. *Aplicação para intervenção em terapia ocupacional com o Leap Motion*. Tese (Doutorado), 2014.

DAM, A. van. Beyond wimp. *IEEE Computer Graphics and Applications*, IEEE, v. 20, n. 1, p. 50–51, 2000.

DEURSEN, A. V.; KLINT, P. et al. Little languages: Little maintenance? *Journal of software maintenance*, v. 10, n. 2, p. 75–92, 1998.

DEURSEN, A. V.; KLINT, P.; VISSER, J. et al. Domain-specific languages: An annotated bibliography. *Sigplan Notices*, v. 35, n. 6, p. 26–36, 2000.

DO, T. T. N. *Development of a virtual pet game using oculus rift and leap motion technologies*. Tese (Doutorado) — Bournemouth University, 2016.

DUCHOWSKI, A. *Eye tracking methodology: Theory and practice*. [S.l.]: Springer Science & Business Media, 2007. v. 373.

FIGUEIREDO, L. S.; LIVSHITS, B.; MOLNAR, D.; VEANES, M. Prepose: Security and privacy for gesture-based programming. In: *IEEE Symposium on Security and Privacy*. [S.l.: s.n.], 2016.

FOWLER, M. *Domain Specific Languages*. 1st. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321712943, 9780321712943.

FURTADO, A. W.; SANTOS, A. L. Using domain-specific modeling towards computer games development industrialization. In: CITESEER. *The 6th OOPSLA workshop on domain-specific modeling (DSM06)*. [S.l.], 2006.

GOZA, S.; AMBROSE, R. O.; DIFTLER, M. A.; SPAIN, I. M. Telepresence control of the nasa/darpa robonaut on a mobility platform. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 2004. p. 623–629.

GRAFSGAARD, J.; WIGGINS, J. B.; BOYER, K. E.; WIEBE, E. N.; LESTER, J. Automatically recognizing facial expression: Predicting engagement and frustration. In: *Educational Data Mining 2013*. [S.l.: s.n.], 2013.

GUNA, J.; JAKUS, G.; POGAČNIK, M.; TOMAŽIČ, S.; SODNIK, J. An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 2, p. 3702–3720, 2014.

HENZE, N.; LÖCKEN, A.; BOLL, S.; HESSELMANN, T.; PIELOT, M. Free-hand gestures for music playback: deriving gestures with a user-centred process. In: ACM. *Proceedings of the 9th international conference on Mobile and Ubiquitous Multimedia*. [S.l.], 2010. p. 16.

HOLMQVIST, K.; NYSTRÖM, M.; ANDERSSON, R.; DEWHURST, R.; JARODZKA, H.; WEIJER, J. Van de. *Eye tracking: A comprehensive guide to methods and measures*. [S.l.]: OUP Oxford, 2011.

HÖYSNIEMI, J.; HÄMÄLÄINEN, P.; TURKKI, L.; ROUVI, T. Children's intuitive gestures in vision-based action games. *Communications of the ACM*, ACM, v. 48, n. 1, p. 44–50, 2005.

HQ, T. V. R. *Essential Reality P5 Gaming Glove*. [S.l.]: The Virtual Reality HQ, 2016. <https://www.thevirtualrealityhq.com/essential-reality-p5-gaming-glove>. Acesso em: 17 abr. 2017.

JOHNSON, T. E. Sketchpad iii: a computer program for drawing in three dimensions. In: ACM. *Proceedings of the May 21-23, 1963, spring joint computer conference*. [S.l.], 1963. p. 347–353.

KAÂNICHE, M. *Gesture recognition from video sequences*. Tese (Doutorado) — Université Nice Sophia Antipolis, 2009.

KAK, A. C. Purdue rvl-slll asl database for automatic recognition of american sign language. In: IEEE COMPUTER SOCIETY. *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*. [S.l.], 2002. p. 167.

KANG, B.; CHOI, E.; KWON, S.; CHUNG, M. K. Ufo-zoom: A new coupled map navigation technique using hand trajectories in the air. *International Journal of Industrial Ergonomics*, Elsevier, v. 43, n. 1, p. 62–69, 2013.

KHADEMI, M.; DODAKIAN, L.; HONDORI, H.; LOPES, C.; MCKENZIE, A.; CRAMER, S. Free-hand interaction with leap motion controller for stroke rehabilitation. *One of a CHInd*, Toronto, ON, Canada, p. 1663–1668, 2014.

KIEBURTZ, R. B.; MCKINNEY, L.; BELL, J. M.; HOOK, J.; KOTOV, A.; LEWIS, J.; OLIVA, D. P.; SHEARD, T.; SMITH, I.; WALTON, L. A software engineering experiment in software component generation. In: IEEE COMPUTER SOCIETY. *Proceedings of the 18th international conference on Software engineering*. [S.l.], 1996. p. 542–552.

KIM, D.; HILLIGES, O.; IZADI, S.; BUTLER, A. D.; CHEN, J.; OIKONOMIDIS, I.; OLIVIER, P. Digits: freehand 3d interactions anywhere using a wrist-worn gloveless sensor. *Proceedings of the 25th annual ACM symposium on User interface software and technology*, New York, NY, USA, p. 167–176, 2012.

KIM, H.; ALBUQUERQUE, G.; HAVEMANN, S.; FELLNER, D. W. Tangible 3d: hand gesture interaction for immersive 3d modeling. *Proceedings of the 11th Eurographics conference on Virtual Environments*, Aire-la-Ville, Switzerland, Switzerland, p. 191–199, 2005.

KIM, H.-J.; JEONG, K.-H.; KIM, S.-K.; HAN, T.-D. Ambient wall: Smart wall display interface which can be controlled by simple gesture for smart home. In: ACM. *SIGGRAPH Asia 2011 Sketches*. [S.l.], 2011. p. 1.

LEAPMOTION. *Leap Motion Gallery*. [S.l.]: Leap Motion, 2017. <https://gallery.leapmotion.com/>. Acesso em: 17 abr. 2017.

LEAPMOTION. *Reach into virtual reality with your bare hands* . [S.l.]: Leap Motion, 2017. <https://www.leapmotion.com/>. Acesso em: 17 abr. 2017.

LEE, J. C. Hacking the nintendo wii remote. *IEEE pervasive computing*, IEEE, v. 7, n. 3, 2008.

LEE, P.-W.; WANG, H.-Y.; TUNG, Y.-C.; LIN, J.-W.; VALSTAR, A. Transection: hand-based interaction for playing a game within a virtual reality game. In: ACM. *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. [S.l.], 2015. p. 73–76.

LI, Q.; STANKOVIC, J. A.; HANSON, M. A.; BARTH, A. T.; LACH, J.; ZHOU, G. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In: IEEE. *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*. [S.l.], 2009. p. 138–143.

LI, Y.; WANG, S.; ZHAO, Y.; JI, Q. Simultaneous facial feature tracking and facial expression recognition. *IEEE Transactions on Image Processing*, IEEE, v. 22, n. 7, p. 2559–2573, 2013.

LIMA, J. P.; SIMÕES, F.; FIGUEIREDO, L.; KELNER, J. Model based markerless 3d tracking applied to augmented reality. *Journal on 3D Interactive Systems*, v. 1, 2010.

LIPP, M. K.; MOSSMANN, J. B.; BEZ, M. R. Desenvolvimento de jogos usando a interface nui leap motion. 2015.

MEHTA, N. A flexible machine interface. *MA Sc. Thesis, Department of Electrical Engineering, University of Toronto*, 1982.

MERIANS, A. S.; JACK, D.; BOIAN, R.; TREMAINE, M.; BURDEA, G. C.; ADAMOVICH, S. V.; RECCE, M.; POIZNER, H. Virtual reality–augmented rehabilitation for patients following stroke. *Physical therapy*, v. 82, n. 9, 2002.

MICROSOFT. *Hardware Kinect*. [S.l.]: Microsoft, 2017.

MITRA, S.; ACHARYA, T. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 37, n. 3, p. 311–324, 2007.

MO, Z. *Gesture interface engine*. [S.l.]: University of Southern California, 2007.

MURTHY, G.; JADON, R. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, v. 2, n. 2, p. 405–410, 2009.

MYERS, B. A. A brief history of human-computer interaction technology. *interactions*, ACM, v. 5, n. 2, p. 44–54, 1998.

NINTENDO. *Wii Remote*. [S.l.]: Nintendo, 2017. <http://www.nintendo.com/wiiu/accessories>. Acesso em: 17 abr. 2017.

NORMAN, D. A. Natural user interfaces are not natural. *interactions*, ACM, v. 17, n. 3, p. 6–10, 2010.

QUEK, F.; MCNEILL, D.; BRYLL, R.; DUNCAN, S.; MA, X.-F.; KIRBAS, C.; MCCULLOUGH, K. E.; ANSARI, R. Multimodal human discourse: gesture and speech. *ACM Transactions on Computer-Human Interaction (TOCHI)*, ACM, v. 9, n. 3, p. 171–193, 2002.

RASKAR, R.; NII, H.; DEDECKER, B.; HASHIMOTO, Y.; SUMMET, J.; MOORE, D.; ZHAO, Y.; WESTHUES, J.; DIETZ, P.; BARNWELL, J. et al. Prakash: lighting aware motion capture using photosensing markers and multiplexed illuminators. In: ACM. *ACM Transactions on Graphics (TOG)*. [S.l.], 2007. v. 26, n. 3, p. 36.

RAUTARAY, S. S.; AGRAWAL, A. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, Springer, v. 43, n. 1, p. 1–54, 2015.

ROSENBERG, A. *Writing signed languages: In support of adopting an ASL writing system*. Tese (Doutorado) — University of Kansas, Linguistics, 1999.

SHARMA, R.; HUANG, T. S.; PAVLOVIC, V. I.; ZHAO, Y.; LO, Z.; CHU, S.; SCHUL, K. Speech/gesture interface to a visual computing environment for molecular biologists. In: IEEE. *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. [S.l.], 1996. v. 3, p. 964–968.

SMITH, G. M. et al. The radial scroll tool: scrolling support for stylus-or touch-based document navigation. In: ACM. *Proceedings of the 17th annual ACM symposium on User interface software and technology*. [S.l.], 2004. p. 53–56.

SONG, S.; LICHTENBERG, S. P.; XIAO, J. Sun rgb-d: A rgb-d scene understanding benchmark suite. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 567–576.

SONY. *PlayStation Move motion controller*. [S.l.]: Sony, 2017. <https://www.playstation.com/en-gb/explore/accessories/playstation-move-motion-controller/>. Acesso em: 17 abr. 2017.

SOUSA, L. R. N. de; SILVEIRA, I. F. Desafios das interfaces gestuais para a aprendizagem de pessoas com transtorno do desenvolvimento da coordenação. *Revista de Informática Aplicada*, v. 11, n. 2, 2016.

STARNER, T.; AUXIER, J.; ASHBROOK, D.; GANDY, M. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. In: IEEE. *Wearable computers, the fourth international symposium on*. [S.l.], 2000. p. 87–94.

STARNER, T.; WEAVER, J.; PENTLAND, A. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 20, n. 12, p. 1371–1375, 1998.

STOKOE, W. C. Sign language structure: An outline of the visual communication systems of the american deaf. *Journal of deaf studies and deaf education*, Oxford Univ Press, v. 10, n. 1, p. 3–37, 2005.

STOTTS, D.; SMITH, J. M.; GYLLSTROM, K. Facespace: endo-and exo-spatial hypermedia in the transparent video facetop. In: ACM. *Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*. [S.l.], 2004. p. 48–57.

STUART, M. *A Grammar of SignWriting*. Tese (Doutorado) — Thesis in Linguistics, University of North Dakota, 2011.

SUTTON, V. *Lessons in Sign Writing: Textbook*. [S.l.]: SignWriting, 1995.

SYSTEMS, C. *CyberGlove III*. [S.l.]: CyberGlove Systems, 2017. <https://www.cyberglovesystems.com/cyberglove-iii>. Acesso em: 19 abr. 2017.

THALMICLABS. *M.Y.O.* [S.l.]: ThalmicLabs, 2016. <https://www.myo.com/>. Acesso em: 17 abr. 2017.

VATAVU, R. D.; PENTIUC, S. G. Multi-level representation of gesture as command for human computer interaction. *Computing and Informatics*, v. 27, n. 6, p. 837–851, 2012.

VOGLER, C.; METAXAS, D. A framework for recognizing the simultaneous aspects of american sign language. *Computer Vision and Image Understanding*, Elsevier, v. 81, n. 3, p. 358–384, 2001.

WACHS, J. P.; STERN, H. I.; EDAN, Y.; GILLAM, M.; HANDLER, J.; FEIED, C.; SMITH, M. A hand gesture sterile tool for browsing mri images in the or. *Journal of the American Medical Informatics Association*, Am Med Inform Assoc, p. M2410v1, 2008.

WANG, R. Y.; POPOVIĆ, J. Real-time hand-tracking with a color glove. In: ACM. *ACM transactions on graphics (TOG)*. [S.l.], 2009. v. 28, n. 3, p. 63.