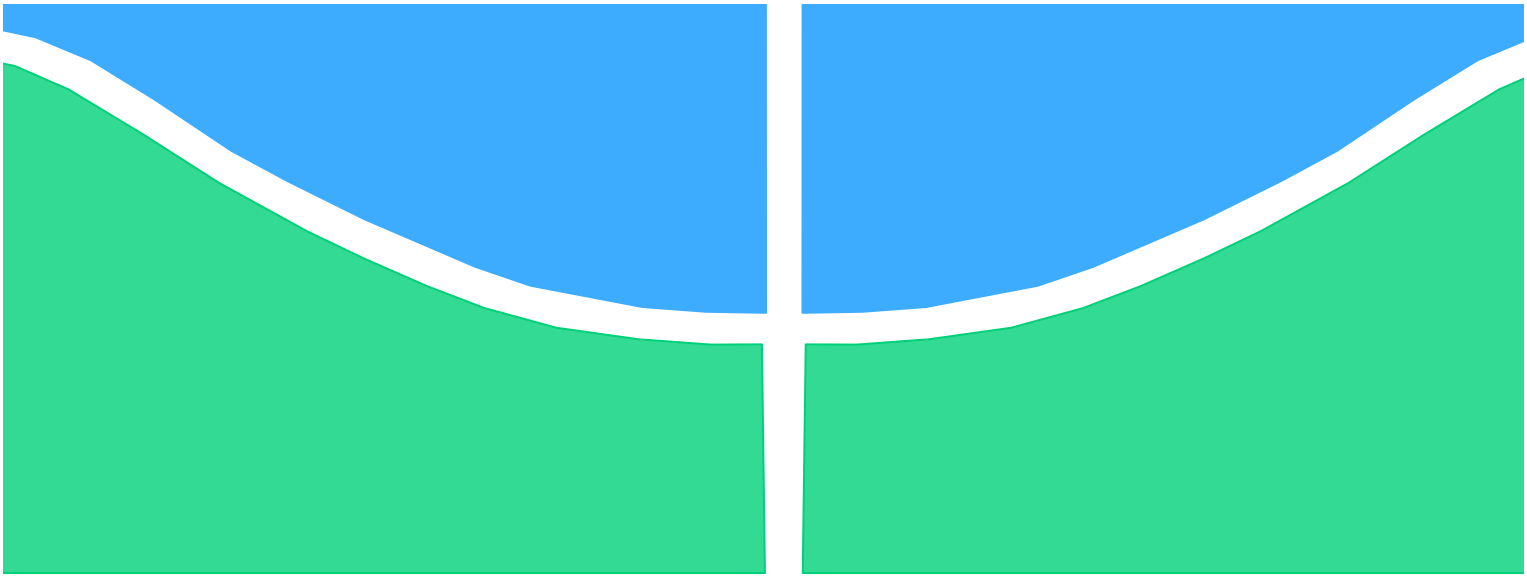


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Curso de Engenharia de Software

SISTEMA INTERATIVO BASEADO EM GESTOS PARA UTILIZAÇÃO DE COMANDOS NO COMPUTADOR

Autor: Filipe Barbosa de Almeida
Orientador: Dr. André Barros de Sales

Brasília, DF
2013

Filipe Barbosa de Almeida

SISTEMA INTERATIVO BASEADO EM GESTOS PARA UTILIZAÇÃO DE COMANDOS NO COMPUTADOR

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obter o Título de Bacharel em Engenharia de Software.

Orientador: Dr. André Barros de Sales

Brasília, DF
2013

CIP – Catalogação Internacional da Publicação

B. de Almeida, Filipe.
Sistema Interativo Baseado em Gestos para Utilização de Comandos no Computador/ Filipe Barbosa de Almeida. Brasília: UnB, 2013. 84 p.: il.; 29,5 cm.

Monografia (Graduação) – Universidade de Brasília Faculdade do Gama, Brasília, 2013. Orientação: Dr. André Barros de Sales.

1. Interativo. 2. Gestos. 3. Interface Natural. 4. Comandos no Computador I. Barros de Sales, André. II. Doutor.

CDU Classificação

REGULAMENTO E NORMA PARA REDAÇÃO DE RELATÓRIOS DE PROJETOS DE GRADUAÇÃO FACULDADE DO GAMA – FGA

Filipe Barbosa de Almeida

Monografia submetida como requisito parcial para obter o Título de Bacharel em Engenharia de Software da Faculdade UnB Gama - FGA, da Universidade de Brasília, em 22/07/2013 apresentada e aprovada pela banca examinadora abaixo assinada:

Prof. Dr. André Barros de Sales, UnB/ FGA
Orientador

Prof.ª Dra. Carla Rocha Aguiar, UnB/ FGA
Membro Convidado

Prof. Me. Hilmer Rodrigues Neri, UnB/ FGA
Membro Convidado

Brasília, DF

2013

Computação Perceptiva é um campo novo que combina a cognição visual, visão de máquina e visualização. É uma simulação computacional do discernimento humano, solução de problemas e na aprendizagem. Percepções humanas podem capturar complexos padrões, relacionamentos e exceções em um conjunto de dados. Computação perceptiva é uma forma de resumir os dados aparentemente separados em partes significativas e passar o resumo das informações a entidades de decisão.

Yang Cai

AGRADECIMENTOS

Agradeço, a Deus, que tem preparado meu caminho e guiado os meus passos e, a meus pais, que sempre me apoiaram, incentivaram e acreditaram em mim.

Agradeço também a todos aqueles que conheci durante a minha vida acadêmica, professores e alunos, que estiverem ao meu lado quando precisei. Em especial aos professores que me orientaram no desenvolvimento desse trabalho (Orientador e Banca), pelos ensinamentos, paciência e amizade.

Um agradecimento especial a minha namorada Luiza, que esteve sempre do meu lado e disposta a me ajudar em todos os momentos.

RESUMO

Nos últimos anos houve grandes evoluções na maneira de interagir com o computador. Telas sensíveis ao toque e tecnologias que reconhecem movimentos e voz ensejaram questões quanto a sua eficiência, naturalidade, intuição e usabilidade. Portanto, este trabalho objetiva desenvolver um sistema interativo baseado na detecção de gestos realizados com o corpo para executar comandos no computador, como: selecionar, clicar, arrastar, aumentar e diminuir zoom, entre outros. Buscou-se aqui, inicialmente estabelecer um embasamento teórico referente aos assuntos tratados no sistema e investigar algumas tecnologias perceptivas capazes de reconhecer movimentos. O desenvolvimento fundamentou-se em princípios de usabilidade da Interação Humano-Computador e processos da Engenharia de Software, como gerência de projeto, gerenciamento de configuração, verificação e validação e técnicas de programação. Os resultados obtidos comprovam a viabilidade da utilização de uma aplicação baseado em gestos no controle do sistema operacional, além de motivar futuros estudos na área de Interação Humano Computador, especialmente no que se refere a Computação Perceptiva.

Palavras-chave: Interativo; Gestos; Interface Natural; Comandos no Computador;

ABSTRACT

In recent years, there have been major improvements in the way you interact with the computer. With the advent of touch screens and technologies that recognize movement and voice, questions arise regarding the efficiency, naturalness, intuition and usability afforded by them. Therefore, this paper sets out to develop an interactive system based on the detection of gestures performed with the body to execute commands on the computer, such as, select, click, drag, zoom in and zoom out, among others. Initially the work sought to establish a theoretical framework on the matters dealt with in the system and investigate if some perceptual technologies can recognize movements. The development was based on principles of usability of human-computer interaction and software engineering processes, such as project management, configuration management, verification and validation and programming techniques. The results of this study demonstrate the feasibility of using a system based on gestures to control computers and motivate future studies in Human Computer Interaction, especially as regards the Perceptual Computing.

Keywords: Interaction, Gestures, Natural Interface, Computer commands.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

3D – Tridimensional

API – Application Programming Interface (Interface de Programação de Aplicativos)

CA – Critério de Aceitação

EAP – Estrutura Analítica do Projeto

GUI – Graphical User Interface (Interface Gráfica do Usuário)

IDE – Integrated Development Environment (Ambiente de Desenvolvimento Integrado)

IHC – Interação Humano-Computador

IN – Interface Natural

IR – Infrared (Infravermelho)

SDK – Software Development Kit (Kit de Desenvolvimento de Software)

SIBGCC – Sistema Interativo Baseado em Gestos para Utilização de Comandos no Computador

SO – Sistema Operacional

SR – Sistema de Rastreamento

RUP – Rational Unified Process (Processo Unificado)

TFS – Team Foundation Service

UML - Unified Modeling Language (Linguagem de Modelagem Unificada)

US – User Story (História do Usuário)

WPF – Windows Presentation Foundation

XP – Extreme Programming

SUMÁRIO

INTRODUÇÃO

Neste capítulo se explanam as principais características do trabalho, incluindo informações referentes à seu contexto e sua estrutura. Descrevem-se, ainda, sua finalidade e as questões envolvidas na sua produção.

O capítulo está divido em cinco partes: o Contexto do Trabalho, que explica onde o estudo se enquadra; a Problematização; os Objetivos; a Metodologia e o Cronograma, esclarecendo as atividades da pesquisa; e finalmente, a Estrutura do Trabalho, na qual se orienta o leitor sobre os posteriores capítulos e seus conteúdos.

1.1. CONTEXTO

Atualmente telas sensíveis a toques múltiplos e tecnologias que reconhecem voz e gesto vêm-se popularizando por permitirem ao usuário interagir com o computador de forma cada vez mais natural e intuitiva. Por esse motivo, pesquisadores no mundo todo têm-se dedicado ao assunto, buscando progressos na interação humano-computador.

A evolução dessa interação foi um dos principais fatores que permitiram a aceitação do computador como item fundamental na vida cotidiana do ser humano.

Segundo Ghirotti (GHIROTTI, 2009), diversos periféricos com interfaces bidimensionais foram desenvolvidos a fim de facilitar a interação entre humano e computador, como os altamente difundidos mouse e teclado, que apresentam limitações em naturalidade e intuitividade, apesar de seu considerável nível de acessibilidade.

No século XXI, quando sistemas informatizados estão imersos nos mais variados contextos, é imperativo desenvolver meios de interfaces eficientes que facilitem a usabilidade e a acessibilidade para pessoas com diferentes necessidades, preferências e habilidades. Considerando os diferentes perfis existentes, deve-se levar em consideração: diferentes culturas, idades,

escolaridades e históricos profissionais, experiência com computadores e necessidades especiais.

Com base nesse contexto, pretende-se desenvolver um sistema interativo baseado em gestos tridimensionais para utilizar comandos no computador.

A contribuição deste trabalho caracteriza-se, na pesquisa e avaliação das tecnologias perceptivas, como base para próximos trabalhos, na área de IHC, como o desenvolvimento de uma interface baseada em gestos que combine gestos naturais e simbólicos, e, na área de Engenharia de Software, fundamentalmente, como a apresentação prática de alguns de seus elementos no desenvolvimento de um sistema.

1.2. PROBLEMATIZAÇÃO

1.2.1. Formulação do problema

Nos últimos anos houve uma grande evolução na maneira de interagir com os dispositivos eletrônicos, mas ainda predomina a interação tátil (mecânica), com o mouse e o teclado como os principais periféricos utilizados. Embora esteja cada vez mais presente nos dispositivos modernos, a interação por toque ainda utiliza uma interface física e bidimensional.

Segundo Normam (NORMAM, 1986), essas interfaces convencionais, que utilizam botões e dispositivos de controle artificiais, trazem para o usuário limitações que exigem longos períodos de aprendizagem e adaptação à tecnologia, além de maior esforço cognitivo em atividades de interpretação e expressão das informações que o sistema processa.

Sendo assim, surgem questões quanto à viabilidade de utilizar uma interface tridimensional para manipular tais dispositivos eletrônicos, mais especificamente os computadores pessoais, além da eficiência propiciada pelas tecnologias perceptivas utilizadas por eles.

1.2.2. Solução do problema

A solução das questões levantadas é proposta através do desenvolvimento de um sistema interativo baseado em gestos para utilizar comandos no computador. Por tratar-se de um trabalho de natureza predominantemente técnica, buscou-se focar na tecnologia perceptiva e no desenvolvimento do software, desconsiderando

elementos secundários referentes à determinação dos gestos e validação da interface utilizada.

1.3. OBJETIVOS

1.3.1. Objetivo geral

Este trabalho tem como objetivo principal apresentar o desenvolvimento de um sistema interativo que utilize uma tecnologia perceptiva de detecção de movimento para reconhecer gestos pré-definidos do usuário e realizar comandos no sistema operacional.

1.3.2. Objetivos específicos

- Pesquisar tecnologias perceptivas que utilizam interface tridimensional para captar movimentos;
- Desenvolver a aplicação para um sistema operacional específico;
- Abordar alguns elementos fundamentais da Engenharia de Software no desenvolvimento do sistema.

1.4. METODOLOGIA

O trabalho iniciou com uma pesquisa bibliográfica para aprofundar o conhecimento na área de Interação Humano-Computador e Computação Perceptiva. Nesse estudo foram definidos o tema, o contexto, a problematização, os objetivos gerais e específicos e a estrutura do trabalho. Também foi utilizada a pesquisa exploratória para buscar tecnologias perceptivas com interface tridimensional que se enquadrassem no contexto do sistema a ser desenvolvido.

Essas tecnologias foram avaliadas e comparadas em suas principais características, verificando suas funcionalidades em diferentes contextos de uso e as técnicas disponíveis para reconhecer movimentos.

O desenvolvimento do sistema foi iniciado após a escolha da tecnologia considerada mais adequada e da configuração do seu ambiente de desenvolvimento. Nele, foram abordados alguns elementos fundamentais da

Engenharia de Software[1], como análise de requisitos, arquitetura de software, gerência de configuração, verificação e validação e programação.

O modelo de processo de desenvolvimento de software escolhido foi o interativo e incremental abordando o conceito de desenvolvimento ágil; utilizando alguns princípios e práticas do método conhecido como extreme programming (XP).

1.4.1. Cronograma

O trabalho foi divido em dois pacotes de trabalho e separado em três macro atividades na primeira e na segunda parte, de acordo com o exigido para desenvolver o trabalho. Na Figura 1 é apresentada a estrutura analítica do projeto, criada no início do trabalho para identificar seus elementos terminais, e serviu como base para seu planejamento.

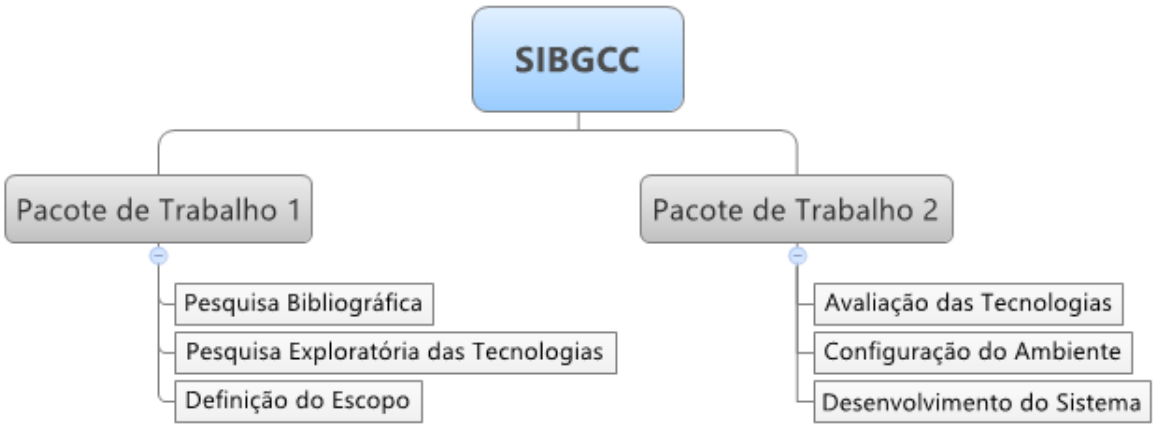


Figura 1. Estrutura analítica do trabalho

As três primeiras macro atividades foram: a Pesquisa Bibliográfica, que representa o estudo de trabalhos com temas correlatos ao deste; Pesquisa Exploratória das Tecnologias Perceptivas, que representa o estudo das principais tecnologias perceptivas e técnicas de interação existentes; e a Definição do Escopo, que define o tema do trabalho e seu contexto, problematiza a situação, define um objetivo e seu resultado esperado.

---

1 A Sociedade de Computação da IEEE define Engenharia de Software como: A aplicação de uma abordagem sistemática, disciplinada, quantificável para o desenvolvimento, operação, e manutenção de software; Ou seja, a aplicação de engenharia à software. (SWEBOOK, 2013).

O segundo pacote de trabalho foi composto pelas macro atividades de Avaliação das Tecnologias escolhidas, Configuração do Ambiente de Desenvolvimento e Desenvolvimento do Sistema. A etapa de Avaliação das Tecnologias foi de curta duração e teve o objetivo de definir a tecnologia perceptiva utilizada no sistema com base em alguns critérios inicialmente levantados; as etapas de Configuração de Ambiente e Desenvolvimento foram realizadas em sequência e se constituem como as atividades com o maior foco do trabalho.

Abaixo, a Figura 2 mostra as macro atividades definidas para cada pacote de trabalho através de um cronograma de marcos, baseado na EAP, que apresenta as tarefas, o tempo de duração e o período de execução delas, os marcos e o gráfico de Gantt.

Trabalho de Conclusão de Curso	**205 days**	**Mon 08/10/12**	**Mon 22/07/13**
Pacote de Trabalho 1	**108 days**	**Mon 08/10/12**	**Thu 07/03/13**
Pesquisa Bibliográfica	86 days	Mon 08/10/12	Mon 04/02/13
Pesquisa Exploratória das Tecnologias Perceptivas	68 days	Thu 01/11/12	Mon 04/02/13
Definição do Escopo	40 days	Thu 03/01/13	Wed 27/02/13
Apresentação do Trabalho - Primeira Parte	0 days	Thu 07/03/13	Thu 07/03/13
Entreda do Documento do TCC1	0 days	Thu 07/03/13	Thu 07/03/13
Pacote de Trabalho 2	**80 days**	**Mon 01/04/13**	**Mon 22/07/13**
Avaliação das Tecnologias Perceptivas	5 days	Mon 01/04/13	Fri 05/04/13
Configuração do Ambiente de Desenvolvimento	38 days	Mon 08/04/13	Wed 29/05/13
Desenvolvimento do Sistema	29 days	Fri 31/05/13	Wed 10/07/13
Apresentação da Segunda Parte	0 days	Mon 22/07/13	Mon 22/07/13
Entrega do Documento do TCC2	0 days	Mon 22/07/13	Mon 22/07/13

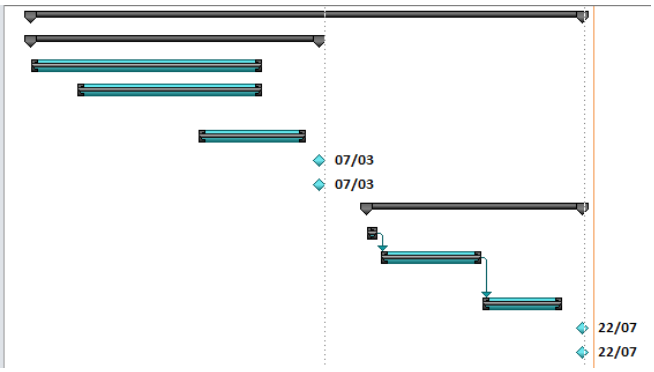


Figura 2. Cronograma de marcos definido para executar o trabalho

1.5. ESTRUTURA DO TRABALHO

Este trabalho está estruturado em cinco capítulos: Introdução, Trabalhos Correlatos, Fundamentos Teóricos e Tecnológicos, Desenvolvimento e Considerações Finais.

O capítulo Introdução destina-se a contextualizar o trabalho sucintamente, descrevendo o contexto e a problematização, além de expor os objetivos geral e específicos e a metodologia utilizada.

O capítulo “Trabalhos Correlatos” resume alguns trabalhos com propósitos semelhantes ao deste estudo, destacando os pontos relevantes de cada um em uma comparação.

O capítulo “Fundamentos Teóricos e Tecnológicos” apresenta os fundamentos teóricos necessários e relevantes para o entendimento do processo de criação do sistema, tanto na parte do software como do hardware.

O capítulo “Desenvolvimento” apresenta a ideia por trás do sistema, as funcionalidades levantadas, descreve o processo de configuração e desenvolvimento do sistema e discute os resultados alcançados.

Por fim, o capítulo “Considerações Finais” conclui o trabalho relatando as contribuições da pesquisa e do desenvolvimento do sistema, além de propor alguns estudos futuros para continuação do trabalho.

2. TRABALHOS CORRELATOS

Os trabalhos correlatos são apresentados nesse capítulo para propiciar uma visão geral na área de reconhecimento de gestos e interação natural e intuitiva com o computador. Basicamente buscaram-se na literatura experiências semelhantes que denotam a atualidade e relevância do tema tratado aqui.

As tecnologias perceptivas, o reconhecimento de gestos e a Interação Humano-Computador foram os principais temas que nortearam a pesquisa bibliográfica. A seguir, são apresentados alguns trabalhos nessa linha de pesquisa.

2.1. TÉCNICA DE NAVEGAÇÃO EM DOCUMENTOS UTILIZANDO MICROSOFT KINECT (SILVEIRA, 2011)

Desenvolvido na Universidade Federal do Rio Grande do Sul, este trabalho de graduação propõe a implementação de um método de interação entre usuário e computador empregando o Microsoft Kinect.

O objetivo da pesquisa foi investigar o poder de detecção de movimentos do dispositivo e avaliar a experiência do usuário através de uma técnica simples de interação.

O método de interação proposto permite realizar algumas atividades no computador, feitas livremente sem a proximidade com o computador ou a utilização de outros periféricos, como mouse ou teclado.

A técnica de detecção de movimento utiliza os *streams* de vídeo e profundidade fornecidos pelo SDK do Kinect para estimar as posições dos segmentos ou *joints* do esqueleto do usuário, e, a partir disso, calcular o ângulo formado pelos antebraços e executar ações pré-definidas. As ações definidas para testar a técnica proposta foram: o comando *Page Down*, com o braço direito, e o *Page Up*, com o braço esquerdo.

O autor propôs avaliar a usabilidade do sistema numa sessão de testes com pessoas de diversas áreas, faixas etárias e experiência computacional. O teste consistiu em uma apresentação pelos usuários de um conjunto de slides no Power

Point. Após o experimento, cada usuário respondeu perguntas em um questionário online.

No final o autor concluiu que, com uma técnica de detecção de gestos suficientemente sensível e precisa, é possível melhorar uma atividade como a apresentação de slides, tornando-a mais interativa, interessante e fácil. Ele ainda enfatizou que seria fácil adicionar novos comandos e gestos à técnica, ampliando seu uso para outras aplicações.

2.2. INTERAÇÃO NATURAL BASEADA EM GESTOS COMO INTERFACE CONTROLE PARA MODELOS TRIDIMENSIONAIS (MEDEIROS, 2012)

Desenvolvida na Universidade Federal da Paraíba, esta monografia investiga o uso da Interação Natural como alternativa a interfaces de usuários para manipular objetos 3D. Ela propõe desenvolver e avaliar um protótipo no cenário da saúde, que utiliza o Microsoft Kinect como tecnologia de reconhecimento de movimentos, para proporcionar ao usuário um controle de objetos tridimensionais utilizando linguagem gestual.

A técnica utilizada para reconhecer movimentos do usuário foi baseada no framework da OpenNI[2] que possibilitou trabalhar com multilinguagem e rodar em diferentes plataformas. Foram rastreados os movimentos das mãos para manipular os objetos na tela. O algoritmo[3] utilizado considera um ponto de referência através das coordenadas x, y, e z para calcular o movimento da mão e realizar comandos no sistema.

Para validar a arquitetura da aplicação e seu possível uso como auxílio na manipulação de objetos 3D, foram realizados três testes de usabilidade com usuários com roteiros pré-definidos que verificaram se o sistema cumpria os objetivos definidos.

Com o resultado da pesquisa, o autor conclui que o sistema desenvolvido pode ser utilizado satisfatoriamente para auxiliar na manipulação de interfaces já existentes. Ele comprovou que é possível utilizar tecnologias de interação natural

---

2 *Open Natural Interaction* é uma organização criada em Novembro de 2010 com o objetivo de melhorar a compatibilidade e interoperabilidade de dispositivos de interface natural, aplicações e middleware. Como um primeiro produto, a organização disponibilizou um framework de código que provê uma API para escrever aplicações utilizando interação natural (TOGORES, 2011).

3 É uma sequência finita e ordenada de passos (regras), com um esquema de processamento que permite a realização de uma tarefa (resolução de problemas, cálculos, etc).

para melhorar a manipulação de interfaces em geral, uso que facilita o trabalho dos profissionais da área de saúde por não necessitar manipular nenhum objeto físico, evitando contaminações.

2.3. INTERFACE NATURAL E IMERSIVA PARA MANIPULAÇÃO DE OBJETOS 3D (MATSUMURA et *al*., 2011)

O trabalho “Interface natural e imersiva para manipulação de objetos 3D” é um projeto do Laboratório de Tecnologias Interativas da Escola Politécnica da Universidade de São Paulo que propôs desenvolver um atlas anatômico virtual tridimensional, voltado para treinamento a distância de estudantes de medicina e áreas correlatas, usando tecnologias de realidade aumentada.

O projeto aplica as novas tecnologias de detecção e visão para criar uma interface natural multimodal baseada em reconhecimento de esqueleto e comandos de voz para manipulação direta de modelos 3D de órgãos do corpo humano. A interface proposta permite selecionar, mover, girar e ampliar os modelos, além de navegar no tempo e separar os modelos em seus componentes.

Tal sistema utiliza um método de captura de imagens 3D utilizando o Microsoft Kinect e uma série de algoritmos de reconhecimento desenvolvidos em C#, no framework .NET e com apresentação no Windows Presentation Foundation (WPF) que permitem definir comandos e manipulações intuitivas. Esses algoritmos foram criados levando em consideração os dados do esqueleto do usuário fornecidos pela API do Kinect.

A autora propôs avaliar a usabilidade do sistema com experimentos cujas métricas principais foram: eficiência do uso, facilidade no aprendizado e satisfação no uso.

Os resultados obtidos comprovaram a viabilidade do uso do sistema como opção para inúmeras aplicações quanto ao aspecto técnico, uma vez que o nível de atraso da transmissão de dados e a estrutura física da montagem não eliminaram a sensação de imersão que inviabilizaria o método de aplicação do conceito.

2.4. UM SISTEMA DE INTERAÇÃO BASEADO EM GESTOS MANUAIS TRIDIMENSIONAIS PARA AMBIENTES VIRTUAIS (GHIROTTI, 2009)

Desenvolvido no Departamento de Ciência da Computação da Universidade São Paulo, este artigo apresenta o desenvolvimento de um sistema de visão

computacional estéreo para reconhecer gestos tridimensionais de baixo custo. Ele aborda a área de IHC, na construção de uma interface baseada em gestos híbridos, que combina gestos naturais e simbólicos para a navegação e manipulação de objetos em ambiente virtual.

O sistema utiliza um algoritmo de detecção no qual as mãos e a cabeça são previamente separados usando um algoritmo *Codebook*[4] de segmentação de fundo e também um classificador de cor de pele para reduzir o espaço de procura. Nesta parte o autor descreve o sistema de visão e o seu tempo de preparação. Observando a necessidade de calibrar as câmeras e o usuário como uma limitação do projeto.

O autor ainda avaliou a utilização da interface de gestos no cenário de um jogo através de 3 experimentos pilotos. O primeiro foi uma avaliação da utilização da interface de gestos como ferramenta de navegação; o segundo como ferramenta de interação; e o terceiro como interface híbrida, formada pela integração das interfaces de gestos naturais com a de gestos simbólicos.

O trabalho apresentou um estudo piloto cujos resultados, embora preliminares, indicam que a interface híbrida combina de forma eficaz a liberdade de movimentos dos gestos naturais com a praticidade dos gestos simbólicos, sem longo período de treinamento nem adaptação do usuário.

2.5. REDES NEURAIS ARTIFICIAIS APLICADAS NO RECONHECIMENTO DE GESTOS USANDO O KINECT (ALVARENGA et *al*., 2012)

O artigo “Redes Neurais Artificiais aplicadas no Reconhecimento de Gestos usando o Kinect” é um trabalho da Universidade de São Paulo que descreve o projeto, desenvolvimento e os resultados obtidos de um sistema para aprendizado e classificação de gestos através do uso do sensor Kinect.

O objetivo foi desenvolver um sistema capaz de reconhecer gestos, através da descrição de uma sequência de movimentos das posições do braço e da mão em um espaço tridimensional. O sistema foi desenvolvido utilizando redes neurais artificiais para realizar o acompanhamento da mão direita do usuário, e assim fornecer comandos ao computador, através de uma interface natural (IN).

---

4 É um algoritmo de tempo real para segmentação de fundo desenvolvido por Kim (KIM *et. al*, 2005).

Foi utilizada a OpenNI para obter as coordenadas (x, y, z) da mão do esqueleto, analisar seu deslocamento e assim criar a rede neural com as entradas e saídas adequadas para classificar os movimentos.

Segundo os autores, o sistema demonstrou a viabilidade de realizar o aprendizado e reconhecimento de gestos em tempo real através de testes com diferentes configurações de Redes Neurais. Os resultados se mostraram promissores, porém ainda existindo a possibilidade de aprimorar o sistema a fim de obter melhores respostas no reconhecimento dos gestos.

2.6. ANÁLISE COMPARATIVA

A Tabela 1 apresenta a análise comparativa dos trabalhos correlatos supracitados. A comparação foi realizada utilizando dez critérios definidos de acordo com sua relevância para a fundamentação teórica e tecnológica do trabalho. Esses critérios são: tecnologia perceptiva utilizada, sistema operacional implantado, kit de desenvolvimento de software escolhido, técnica de reconhecimento de gesto abordada, definição de gesto, comandos no computador definidos, área de aplicação, interface gráfica para o usuário, existência de uma avaliação de usabilidade e limitações presentes no trabalho.

Tabela 1. Comparação entre os trabalhos pesquisados

	Trabalho 1	**Trabalho 2**	**Trabalho 3**	**Trabalho 4**	**Trabalho 5**
Tecnologia perceptiva	Kinect	Kinect	Kinect	Câmera convencional	Kinect
Sistema operacional	Windows	Linux	Windows	Windows	Linux
SDK	Oficial do Kinect	OpenNI	Oficial do Kinect	-	OpenNI
Técnica de reconhecimento	Ângulo formado pelos antebraços	Coordenadas das mãos	Coordenadas das mãos	Coordenadas das mãos	Coordenadas da mão direita
Definição de gesto	Movimento vertical dos braços direito e esquerdo	Movimento das mãos na horizontal e Frente/Trás	Movimento junto ou separado das mãos na horizontal, vertical e diagonal	Movimento das mãos na horizontal, vertical e Frente/Trás	Movimento da mão na horizontal, circular e Frente/Trás
Comandos no computador	Rolagem	Navegação do mouse, Rotação e Zoom	Navegação do mouse, Translação, Rotação e Zoom	Navegação do mouse, Rotação	-
Área de aplicação	Comandos no Sistema Operacional que podem ser utilizados para manipular Slides	Manipulação de objetos 3D	Manipulação de objetos 3D	Manipulação de objetos 3D	Jogos, robôs e etc.
Interface Gráfica do Usuário	Não	Sim	Sim	Não	Não
Avaliação da usabilidade	Sim	Sim	Sim	Sim	Não
Limitações	Precisão do sensor	Precisão do sensor, Calibragem do *skeleton*	Precisão do sensor	Calibração das Câmeras	Precisão do sensor, Calibragem do *skeleton*

Foi observado entre os trabalhos avaliados uma tendência nos valores dos critérios: tecnologia perceptiva utilizada, técnica de reconhecimento de gesto abordada, definição de gesto, existência de uma avaliação de usabilidade e limitações presentes no trabalho.

Pode-se dizer que em relação a tecnologia perceptiva e a técnica de reconhecimento essa tendência está vinculada ao fato da maior disponibilidade e documentação da tecnologia e da facilidade de uso da técnica. A definição dos gestos segue uma tendência devido a naturalidade do movimento realizado com os membros superiores ou suas partes quando comparada com o comando executado. As limitações estão vinculadas diretamente ao sensor escolhido.

A partir dessa pesquisa, e considerando os fatores mais convencionais no desenvolvimento de trabalhos desta natureza, foram definidas as bases conceituais e teóricas do trabalho que serão detalhadas no próximo capitulo e aplicadas no capítulo seguinte.

3. FUNDAMENTOS TEÓRICOS E TECNOLOGICOS

Neste capítulo é apresentada a fundamentação teórica e tecnológica do trabalho. São discutidos os conceitos de IHC e de percepção tecnológica como sua grande área de concentração. Também são descritas as tecnologias perceptivas avaliadas, os comandos do computador e o processo de desenvolvimento de software utilizado.

3.1. INTERAÇÃO HUMANO-COMPUTADOR

Segundo a Sociedade Brasileira de Computação (SBC, 2013), a área de Interação Humano-Computador (IHC) se dedica a estudar os fenômenos de comunicação entre pessoas e sistemas computacionais que estão na interseção das ciências da computação e informação e ciências sociais e comportamentais, envolvendo todos os aspectos relacionados à interação dos usuários com os sistemas.

A interação usuário-sistema no âmbito computacional é a combinação entre o software e o hardware necessária para viabilizar o processo de comunicação entre o usuário e a aplicação.

Segundo Normam (NORMAM, 1986), o termo interface é comumente utilizado para denominar aquilo que interliga dois sistemas. As antigas interfaces entre usuários e sistemas computacionais são criticadas por exigirem do usuário esforço cognitivo maior em atividades de interpretação e expressão das informações que o sistema processa.

Recentemente houve crescente interesse em introduzir outros meios de interação humano-computador para o campo da interface. Esses novos meios incluem uma classe de dispositivos baseados em movimentos, ou, mais especificamente, gestos.

3.1.1. Interação natural

A evolução tecnológica traz a possibilidade de criar interfaces cuja implementação se utilize progressivamente da naturalidade de agir do ser humano, resultando em interações mais intuitivas.

Segundo Garbin (GARBIN, 2010), a expressão *interação natural* está sendo utilizada por várias pesquisas de modo geral, ou como uma perceptiva à parte, em que eles a descrevem como algo que de algum modo é diferente das interfaces comuns, vistas anteriormente. Uma comparação entre as interfaces é apresentada na Figura 3. A utilização desse conceito no desenvolvimento de interfaces auxilia na boa aceitação do usuário, desde que sua influência leve a associar o que estão lidando com algo abstrato, mídia digital, mas com objetos físicos. A Interação Natural - IN ativa a cognição e a dinâmica cibernética das pessoas, o que normalmente acontece quando elas vivenciam algo na vida real (MEDEIROS, 2012).

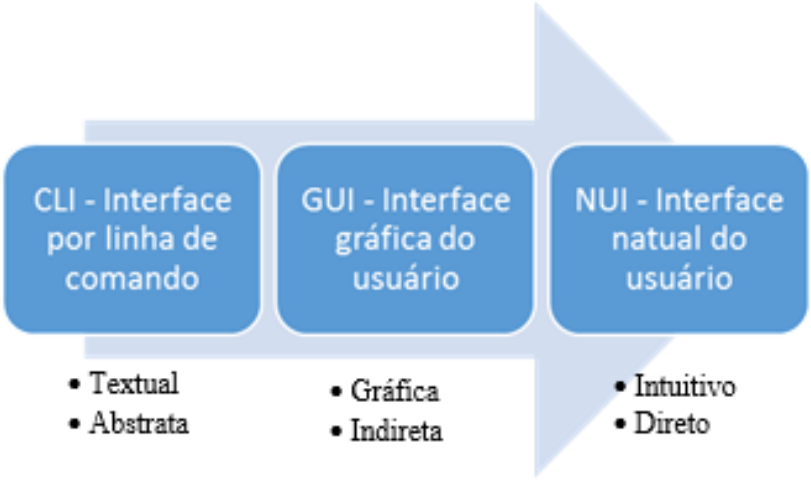


Figura 3. Evolução das interfaces. Fonte: (GABIN, 2010)

Considerada como novo paradigma de interação, a Interface Natural de Usuário é uma linguagem comum usada por designers e desenvolvedores de interface de computador para se referir a uma interface de usuário que aplica conceitos de interação natural em sua construção.

Em teoria, a interação com uma NUI deve depender somente da capacidade do usuário de interagir com o ambiente. Embora exija aprendizagem, a interface é projetada para passar a sensação de que o usuário e a máquina estão interligados.

3.1.1.1. Interação por gestos

Apesar de seus diversos significados, neste trabalho a palavra gesto será adotada com a definição proposta por Mitra e Acharya:

> "Gestos são movimentos corporais expressivos e significativos realizados através da movimentação das mãos, braços, cabeça, ou corpo com o objetivo de: 1) transmitir uma

> informação ou 2) interagir com o ambiente". (S. MITRA E T. ACHARYA, 2007)

Gestos são mecanismos complexos para mapeamento, pois podem ter significados diversos, variando de acordo com aspectos contextuais e culturais, podendo variar também de acordo com a interação com o ambiente. Um gesto, como acenar ou aplaudir, não envolve nenhum tipo de elemento externo, diferentemente de mover, empurrar, pegar, que envolvem outros objetos.

Por este trabalho estar particularmente interessado em como os gestos podem ser utilizados na comunicação com sistemas computacionais, maior enfoque será dado aos gestos de natureza semiótica propostos por Cardoz (CARDOZ, 1994), ou seja, gestos utilizados para comunicar uma informação significativa que não tem envolvimento direto com objetos externos.

A IHC mediada por gestos significa justamente comunicar-se com o computador através de ações naturalmente humanas, ou o mais próximo disso possível, de forma que não exijam longos períodos de aprendizado e adaptação à tecnologia.

Segundo Ghirotti (GHIROTTI, 2009), as novas tecnologias perceptivas são um passo para tornar interfaces naturais uma realidade no dia a dia do uso computacional; e, com base nesses conceitos estudados, são apresentadas três tecnologias escolhidas para serem aqui pesquisadas e avaliadas.

3.2. TECNOLOGIAS PERCEPTIVAS

As tecnologias perceptivas, que lidam com a interação natural, têm como objetivo principal permitir aos computadores uma detecção sensorial detalhada dos humanos. Tais tecnologias possuem um conjunto de hardware e software que trabalham principalmente com as seguintes percepções: reconhecimento de gestos, reconhecimento facial, reconhecimento de voz e reconhecimento de toque.

Um dos objetivos deste trabalho está na avaliação das tecnologias perceptivas, responsáveis pelo reconhecimento de gestos tridimensionais. Serão apresentadas três ferramentas que abordam esse tipo de reconhecimento para realizar o estudo e implementar o sistema.

Os dispositivos escolhidos foram o Microsoft Kinect™ para Windows, pela vasta documentação disponível e suas inúmeras possiblidades de implementação;

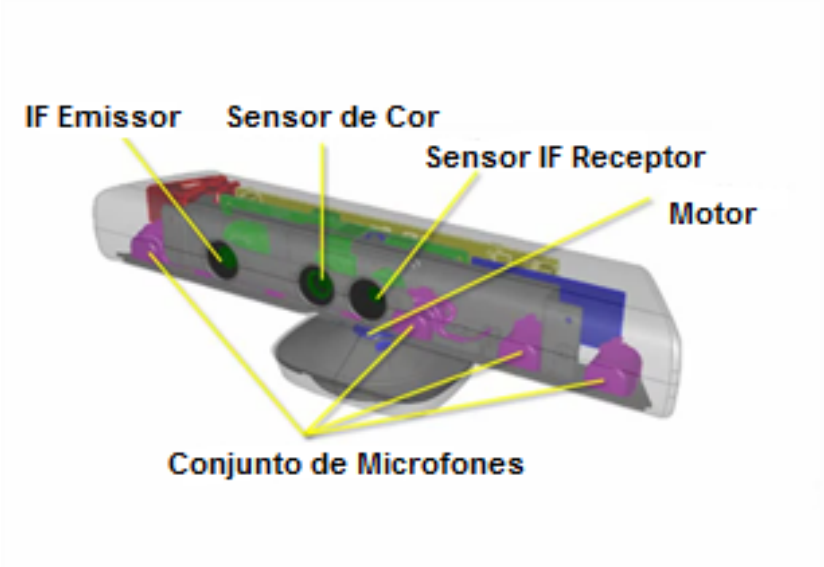
Asus Xtion, pelas características e potencial semelhante ao Kinect; e o Leap Motion Controller, pela inovação proposta pelo conceito e a oportunidade de trabalhar com uma abordagem nova em tecnologia perceptiva.

3.2.1. Microsoft Kinect para Windows

O Kinect™ é um sensor de movimento originalmente desenvolvido pela PrimeSense© e adquirido pela empresa Microsoft para ser utilizado como um dispositivo periférico do console de jogos da mesma empresa, chamado Xbox 360™. Seu objetivo principal é permitir aos usuários controlar e interagir com o console através de uma interface natural usando gestos e comandos de voz, sem utilizar controle manual.

Com o sucesso de vendas e o aumento de sua popularidade, a Microsoft decidiu investir em outras aplicações para ele, a fim de explorar ao máximo seu potencial (WIKI KINECT, 2013). Então, em fevereiro de 2012, lançou o Microsoft Kinect para Windows que consiste no sensor Kinect™, no kit de desenvolvimento e uma licença comercial necessária para desenvolver aplicações no sistema operacional Windows. O SDK do Kinect™ suporta aplicações criadas em C++, C# ou Visual Basic[5] usando a IDE da Microsoft chamada Visual Studio (MICROSOFT, 2013).

O sensor Kinect™ possui 3 câmeras: uma RGB convencional e duas infravermelhas usadas para detectar a distância do usuário, e um conjunto de



5 Linguagens de programação utilizadas pelo kit de desenvolvimento do Kinect – SDK.

microfones utilizado para captar a fala como representado na Figura 4.

Figura 4. Estrutura do Kinect. Fonte: (KINECT, 2013)

Suas características e recursos são (MICROSOFT, 2013):

- Cerca de 23 cm de comprimento horizontal, 5 cm de comprimento vertical e 5 cm de altura.
- Ângulo de visão vertical de 43,5º e um motor para alterar o ângulo de visão em + 27º e -27º, verticalmente. A Figura **5** está representado o ângulo de visão do Kinect.

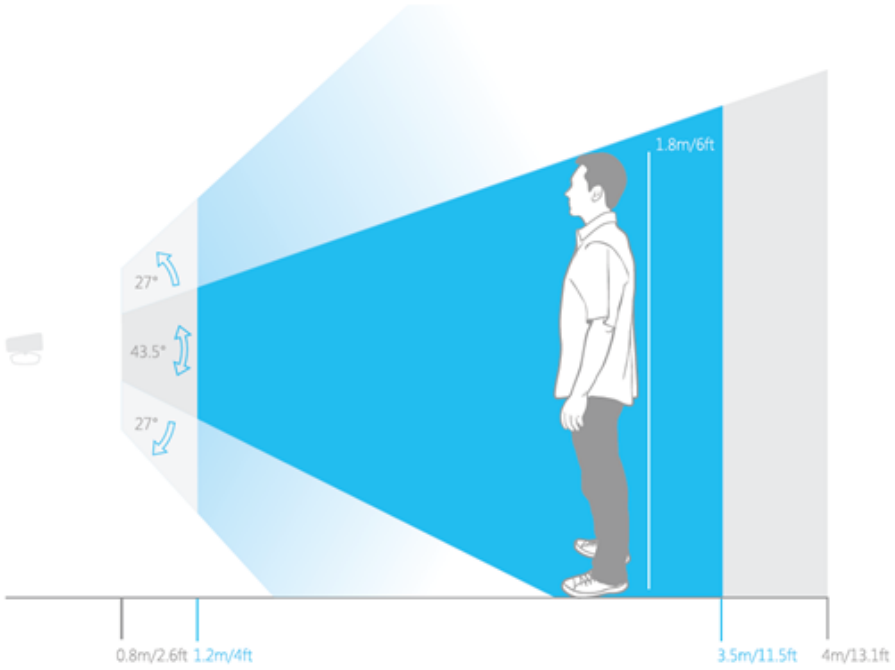


Figura 5. Campo de visão do Kinect. Fonte: (KINECT, 2013)

- Possui uma câmera RGB (Red, Green, Blue) que tira 30 frames por segundo na resolução de 640x480 e 15 frames por segundo na resolução de 1280x1024.
- Um emissor de infravermelho e um sensor de profundidade IR. O emissor emite feixes de luz infravermelha e o sensor de profundidade lê os feixes infravermelhos refletidos de volta para o sensor. Os feixes refletidos são convertidos em informação de profundidade, gerando distância entre o objeto e o sensor.

- Possui um conjunto de microfones embutidos, que além de captar as vozes mais próximas, consegue diferenciar os ruídos externos.
- Um acelerômetro de 3 eixos configurado para uma gama de 2G, onde g é a aceleração devido à gravidade.
- Possui processador embutido.
- Utiliza a interface USB 2.0 para comunicação com o computador.
- Detecta 20 pontos de articulação do esqueleto humano, conhecido como *Joints*.

Diferentemente do Kinect™ para o console de jogos, o Kinect para Windows possui um ângulo de visão mais próximo do sensor, possibilitando-lhe detectar até dez *joints* na parte superior do corpo humano e também perceber quando o usuário está sentado, uma grande vantagem para o trabalho quando se pretende obter uma visualização do usuário próximo ao computador.

Na Figura 6, a diferença do sensor do console que só possui o *Default Mode,* e o sensor para o Windows, que, além do modo anterior, possui o *Near Mode.*

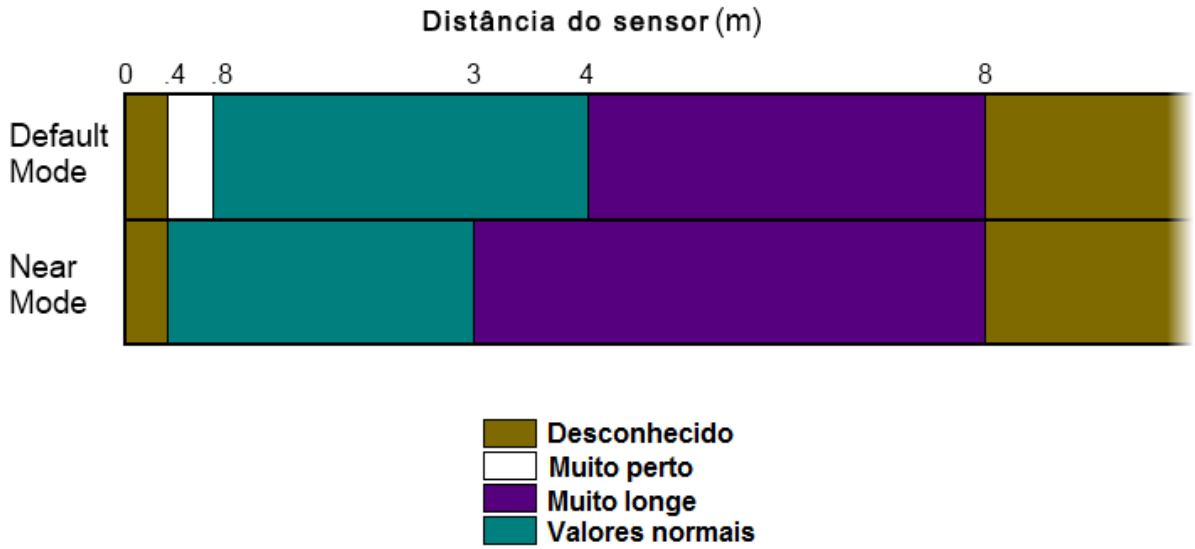


Figura 6. Comparação entre os modos de distância do Kinect. Fonte: (KINECT MODOS, 2013)

O Kinect possui um sistema de rastreamento (SR) de esqueleto que fornece as posições dos *joints* rastreados do usuário. Essas posições são fornecidas como coordenadas em um eixo tridimensional e utilizadas para muitos propósitos, como detecção de gestos, navegação, manipulação de objetos tridimensionais, entre outros.

Segundo Azimi (AZIMI, 2013), na utilização prática do sensor, um ruído aparece nas coordenadas dos *joints* retornadas por esse sistema. Muitos parâmetros afetam as características e o nível desse ruído, como iluminação da sala, tamanho do corpo do usuário, distância entre usuário e sensor, posição do usuário, localização do sensor e efeitos de arredondamento introduzidos por cálculos.

Azimi diz que as posições dos joints retornadas pelo sistema possuem acurácia, o que significa pouca diferença nas coordenadas recebidas para as posições efetivas no mundo real. No entanto, os dados das posições dos *joints* não são necessariamente precisos, variando dentro da casa dos centímetros.

Ele ainda cita que existem casos em que o sistema de rastreamento não tem informação suficiente para determinar a posição especifica de um *joint*, como oclusão por móveis ou por outras pessoas, auto-oclusão de um conjunto de *joints* e mover uma parte do corpo para fora do campo de visão do sensor, entre outras. Mas na maioria dos casos o SR é capaz de inferir a posição do *joint*.

Portanto, dois tipos de ruído estão presentes no rastreamento dos *joints*: um deles é relativamente pequeno e sempre está presente em todas as medições, causado pela imprecisão do sensor; o outro são variações temporárias causadas pela imprecisão, quando um *joint* tem um estado de rastreamento indeterminado.

3.2.1.1. Microsoft Kinect SDK

A implementação do sistema e a comunicação com o sensor Kinect™ podem ser realizados utilizando diferentes kits de desenvolvimento (SDK). A Microsoft possui seu próprio kit estável que está na versão 1.7, possui vasta documentação e integra todo processo de instalação e configuração dos drivers do sensor.

O SDK fornece acesso às interfaces de programação de aplicativos (API) do Kinect™, permitindo as seguintes funcionalidades (MICROSOFT, 2013):

- controle direto do sensor;

- acesso a imagens diretas do sensor de profundidade, e RGB;
- acesso ao microfone com capacidade de processamento de áudio, incluindo supressão de ruídos e cancelamento de eco;
- acesso aos dados do acelerômetro;
- detecção do esqueleto de usuários no campo de visão do Kinect para aplicações orientadas a gestos;
- integração com a API de reconhecimento de fala do Windows;
- suporte à última versão do sistema operacional do Windows (8.1);
- suporte à última versão da IDE da Microsoft (2012); e
- construção de aplicações em C++, C#, Visual Basic.

3.2.2. Asus Xtion

Primeiro sensor de movimento comercial exclusivo para o PC (ASUS, 2013), o Xtion é um dispositivo eletrônico distribuído pela ASUS, empresa de tecnologia de Taiwan que usa sensores do tipo infravermelho para detectar profundidade, RGB para detectar cores, e um conjunto de microfones para capturar a voz, rastreando em tempo real os movimentos realizados pelo usuário.

O dispositivo, apresentado na Figura 7, vem com um conjunto de ferramentas de desenvolvimento para facilitar que programadores criem seus próprios aplicativos baseados em gestos, sem a necessidade de escrever algoritmos de programação complexos.

Figura 7 Estrutura do Xtion. Fonte: (ASUS, 2013)

Segundo a Asus, ele é capaz de rastrear todo o corpo e seus movimentos, sendo ideal para utilizar em jogos como controle. O Xtion suporta o reconhecimento de multiusuários e diferentes sistemas operacionais.

3.2.3. Leap Motion Controller

Sensor de movimento com precisão extremamente alta, o Leap Motion Controller está sendo desenvolvido por uma pequena empresa dos Estados Unidos chamada LEAP MOTION, fundada em 2010. Considerando a data de publicação desse trabalho, ainda não está disponível no mercado.

Tal dispositivo foi escolhido para ser avaliado neste trabalho por sua grande evolução no quesito precisão em obter movimentos provenientes do usuário. Através de uma nova abordagem de reconhecimento, o Leap Motion Controller, representado na Figura 8, pretende alcançar um nível nunca experimentado de imersão, interação e navegação com um computador.

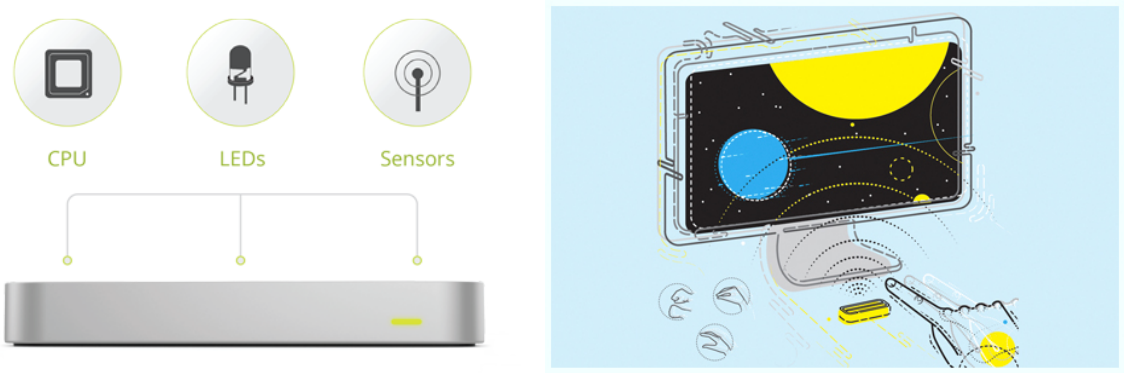


Figura 8. Funcionamento e arquitetura do Leap Motion Controller. Fonte: (LEAP MOTION, 2013)

Ainda sem muitas especificações, o dispositivo é assim definido pela própria empresa desenvolvedora da tecnologia:

> *Leap Motion Controller representa uma forma totalmente nova de interagir com os computadores. É mais preciso do que um mouse, tão confiável quanto um teclado e mais sensível do que uma tela sensível ao toque. Pela primeira vez, o usuário poderá controlar precisamente um computador em três dimensões com movimentos naturais das mãos e dos dedos.*
>
> *Este não é um sistema de jogo que aproximadamente mapeia seus movimentos da mão. A tecnologia do Leap Motion*

> *Controller é 200 vezes mais precisa do que qualquer outra coisa no mercado. Apenas com o tamanho de uma pendrive, o sensor pode distinguir os dedos e rastrear seus movimentos para baixo a um 1/100 de milímetro.* (LEAP MOTION, 2013).

Ainda sem documentação disponível a seu respeito, o sensor está previsto para chegar ao mercado no segundo semestre de 2013.

3.2.4. Comparação entre as tecnologias

Com base no estudo inicial das tecnologias, chegou-se, no segundo pacote de trabalho, a uma avaliação mais específica de cada uma delas para que se pudesse escolher a mais adequada ao sistema. Como o software a ser desenvolvido já estava com seu escopo definido, foram escolhidos alguns critérios fundados nas suas necessidades para escolher a tecnologia. Na Tabela 2, é apresentada a comparação das tecnologias utilizando esses critérios definidos.

Tabela 2. Comparação entre as tecnologias

Tecnologia Perceptiva	**KINECT**	**Xtion**	**LEAP MOTION**
Disponibilidade	SIM	SIM	NÃO
Preço	R$ 999,00	R$600,00	$218.50
Linguagem de Programação	C++, C#, VB	C++, C#, Java	C++, C#, Java
SDK	Proprietário e Aberto	Aberto	Proprietário
Ângulo de visão	43º ajustáveis em +27º e - 27º (Vertical)	45º ajustáveis manualmente (Vertical)	130º (Lado a lado) e100º (*Frente/Trás*)
Profundidade da Visão	0,4m a 4m	0,8m a 3,5m	0,02m a 0,5m
Precisão	Centímetros	Centímetros	1/100 de milímetro
Sensor Utilizado	Infravermelho e RGB	Infravermelho e RGB	Infravermelho
Posição no computador em relação ao monitor	Parte de cima ou de baixo	Parte de cima ou de baixo	Na frente e embaixo
Dimensões	28 x 7 x 7cm	18 x 3,5 x 5 cm	8 x 3 x 1,2 cm
Recursos Extras	Sim	Sim	Não
Documentação	Grande	Média	Pequena

A tabela demostra que, ao cotejar os atributos mais importantes, como disponibilidade, linguagem de programação, SDK, ângulo e profundidade de visão, o dispositivo mais adequado ao sistema é o Microsoft Kinect, que apesar de apesentar limitações de precisão, apresentadas por Azimi na descrição da tecnologia, mostra por sua superioridade em quase todos os quesitos, além de ser o mais utilizado pelos trabalhos correlatos e ter a maior documentação disponível.

3.3. COMANDOS DO COMPUTADOR

Atualmente os computadores são manipulados, em sua maioria, por meio de dispositivos de controle artificiais, como mouse ou Touchpad[6], telas sensíveis ao toque e teclado. A proposta do trabalho é explorar uma nova abordagem, possibilitando que comandos do computador sejam interpretados a partir de gestos.

Com isso, dentre as possibilidades de realizar movimentos com o corpo, serão considerados alguns comandos do sistema operacional Microsoft Windows, com base na versão 8 (oito), na qual se utiliza, principalmente, a manipulação de arquivos e janelas, navegação entre aplicações abertas e páginas e funcionalidades da área de trabalho.

Dentre os comandos que podem ser utilizados, estão: clicar, selecionar, arrastar, rotacionar, rolar, exibir menu suspenso, exibir área de trabalho, navegar entre janelas, aumentar e diminuir zoom, e fechar, maximizar e minimizar janela. A relação das funcionalidades com os comandos do sistema operacional são apresentados na Tabela 3.

---

6 É um dispositivo sensível ao toque, utilizado em computadores portáteis, para substituir o mouse.

Tabela 3**.** Relação das funcionalidades com os comandos do computador. Fonte: (MSDN, 2013)

Funcionalidade	**Comandos do Sistema Operacional**
Aumentar zoom	^{+}
Diminuir zoom	^{-}
Rotacionar	^(.) e ^(,)
Rolar	{PGDN} e {PGUP}
Arrastar	0x0008
Selecionar	0x0008
Clicar	0x0008
Navegar entre Janelas	%{TAB}
Maximizar Janela	%{BS}X
Minimizar Janela	% {BS}N
Fechar Janela	%{F14}
Exibir Menu Suspenso	0x0002
Exibir Área de Trabalho	{LWIN}

Essa especificação inicial facilitou o processo de tradução dos gestos para comandos no computador na implementação do sistema.

3.4. DESENVOLVIMENTO DE SOFTWARE

Segundo Neto (NETO, 2004), o desenvolvimento de software precisa ser reconhecido como um processo complicado; porém, mais importante é reconhecê-lo como um processo empírico, que aceita a imprevisibilidade e tem mecanismos de ação corretiva.

Esse autor (NETO, 2004) lembra que os processos de desenvolvimento de software baseados em especificações completas de requisitos, projeto, construção e teste não estão voltados a um desenvolvimento rápido. Quando os requisitos mudam ou quando os problemas com requisitos são descobertos, o sistema precisa ser retrabalhado e reavaliado. Como consequência, um processo em cascata convencional é geralmente prolongado, e o software final é entregue muito depois de ter sido originalmente especificado.

Em ambiente como o proposto neste trabalho, sujeito a mudanças que poderão causar alterações no software e atraso no prazo de entrega, deve-se utilizar um processo de desenvolvimento rápido.

Segundo Sommerville (SOMMERVILLE, 2007), processos de desenvolvimento rápido de software são projetados para criar software funcional rapidamente. Geralmente, eles são processos iterativos nos quais a especificação, o projeto e o desenvolvimento e o teste são intercalados. O software não é desenvolvido e disponibilizado integralmente, mas numa série de incrementos, e cada incremento inclui uma nova funcionalidade do sistema.

Embora existam muitas abordagens para o desenvolvimento rápido de software, serão consideradas as características mais fundamentais para este trabalho:

- Os processos de especificação, projeto e implementação são concorrentes, não existindo uma especificação detalhada do sistema. Os requisitos do sistema são definidos superficialmente.
- O sistema é desenvolvido em uma serie de incrementos. Os *Stakeholders*[7] participam da especificação e da avaliação de cada incremento. Eles podem propor alterações e novos requisitos implementáveis num incremento posterior do sistema.
- As interfaces com o usuário do sistema são geralmente desenvolvidas usando-se um sistema de desenvolvimento iterativo que permite que o projeto de interface seja criado rapidamente por desenho e inserção de funcionalidades.

Portanto, o desenvolvimento incremental envolve a produção e entrega de software em incrementos, em vez de um pacote único. Cada iteração produz um novo incremento de software.

Algumas das vantagens de adotar essa abordagem neste trabalho são: a entrega rápida e constante de funcionalidades, em que os incrementos iniciais do sistema podem fornecer uma funcionalidade de alta prioridade; e engajamento com

---

7 São os envolvidos no projeto (usuários, desenvolvedores, gerentes, engenheiros, etc.).

as funcionalidades do sistema para obter feedback constante sobre os incrementos entregues.

3.4.1. Métodos ágeis

Nas décadas de 1980 e 1990, havia uma visão geral de que a melhor maneira de obter o melhor software era por meio de um cuidadoso planejamento de projeto controlado por um rigoroso processo de desenvolvimento de software. Essa abordagem envolve um *overhead* significativo de planejamento, projeto e documentação do sistema (SOMMERVILLE, 2007).

Segundo o Manifesto Ágil (AGIL, 2001), a insatisfação com essas abordagens orientadas para a documentação levou um grupo de desenvolvedores de software a propor novos métodos de desenvolvimento. Esses permitiram que a equipe de desenvolvimento se concentrasse mais no código, em vez de no detalhamento do projeto e documentação. Geralmente, os métodos ágeis contam com uma abordagem iterativa para especificação, desenvolvimento e entrega de software, e em vez de gerenciar as atividades e esperar até que o desenvolvimento do software acabe, eles gerenciam os requisitos e os entregam a cada nova iteração (PROCESS, 2009). Podendo ser, então, propostos novos requisitos e alterações a serem incluídos nas iterações posteriores do sistema.

Embora os métodos ágeis sejam baseados na noção de desenvolvimento e entrega incrementais, eles propõem processos diferentes (SOMMERVILLE, 2007). Contudo, compartilham um conjunto de princípios comuns que serão abordados neste trabalho e são apresentados na Tabela **4**.

Tabela 4. Alguns princípios dos métodos ágeis baseados no Manifesto Ágil (AGIL, 2001).

Princípios	**Descrição**
Tolerância a Mudanças	Mudanças nos requisitos são bem-vindas, mesmo tardiamente, no desenvolvimento.
Entrega Incremental	Entregar frequentemente software funcionando, de poucas semanas a poucos meses, de preferência com a menor escala de tempo.
Técnica	Contínua atenção à excelência técnica e o bom design aumenta a agilidade.

Simplicidade	Concentrar-se na simplicidade do software e do processo em desenvolvimento.

3.4.1.1. Extreme Programming

Provavelmente o método ágil mais conhecido e amplamente utilizado é a *eXtreme Programming* (SOMMERVILLE, 2007). A XP surgiu no início de 1996, quando Kent Beck e Ward Cunningham juntaram-se para desenvolver uma nova abordagem aos projetos de software, de maneira mais simples, direta e eficiente (NETO, 2004).

Segundo a XP (XP, 2009), todos os requisitos são expressos como histórias, que são implementados diretamente como uma série de tarefas. É posposta a melhora do projeto de software através de alguns aspectos essenciais: simplicidade, feedback, respeito e coragem, representados, respectivamente, como: a manutenção de um design simples e limpo; o feedback através do teste constante; a entrega de funcionalidades do sistema o mais cedo possível, a implementação das mudanças à medida que surgem; e cada pequeno sucesso aprofundar o respeito para as contribuições únicas do desenvolvedor.

Para aplicar os valores e princípios durante o desenvolvimento de software, a XP propõe uma série de práticas. Entre elas serão citadas as utilizadas no trabalho:

- Jogo de Planejamento: O desenvolvimento é feito em iterações semanais. Ao final de cada semana, são apresentadas novas funcionalidades, testadas e prontas para serem postas em produção.
- Fases pequenas: A liberação de pequenas versões funcionais do projeto auxilia no processo de aceitação e teste.
- Design Simples: Simplicidade é um princípio da XP. Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código simples e código fácil. Nem sempre o código mais fácil de ser desenvolvido levará a solução mais simples por parte de projeto. Código fácil deve ser identificado e substituído por código simples.

- Desenvolvedor Coeso: O desenvolvedor é uma pessoa engajada e multidisciplinar (no sentido de incluir uma pessoa com cada uma das habilidades necessárias para o projeto).
- Padronização do Código: O desenvolvedor precisa estabelecer regras para programar.
- Teste de Unidade: Os testes são essenciais e vistos como uma forma de investimento que garante produtividade no médio e longo prazo. Eles ajudam a reduzir a ocorrência de erros e defeitos no sistema.
- Refatoração: É um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código fonte.
- Integração Contínua: Sempre que produzir uma nova funcionalidade, não esperar uma semana para integrar à versão atual do sistema. Isto só aumenta a possibilidade de conflitos e erros no código fonte. Integrar de forma contínua permite saber o status real da programação.

A *extreme programming* envolve um conjunto de partes pequenas que individualmente não fazem sentido, mas quando combinadas, juntas completam o seu processo (XP, 2009). A Figura 9 ilustra o processo XP simplificado que será utilizado neste trabalho.

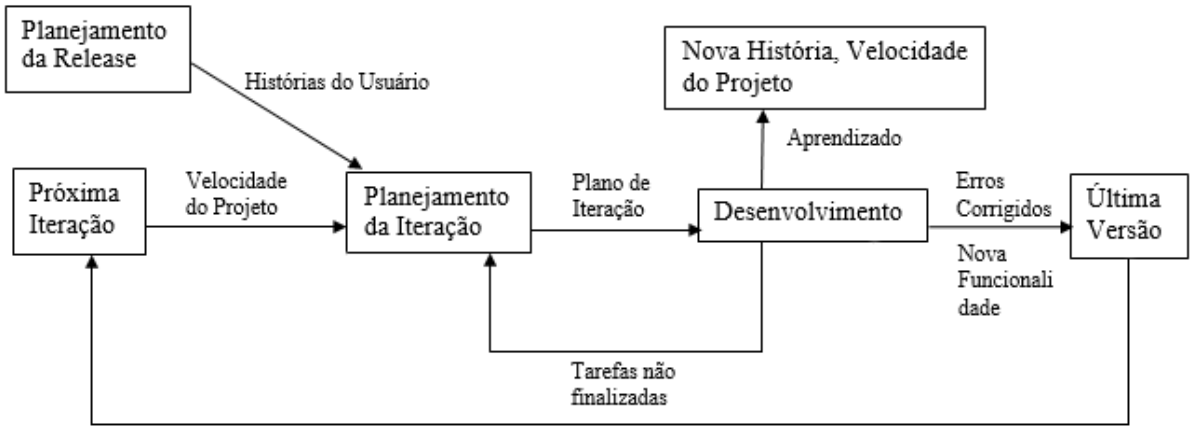


Figura 9. Ciclo de uma iteração no trabalho baseado na XP (XP, 2009).

No processo, para cada iteração é realizado um planejamento, no qual se obtém um conjunto de histórias do usuário para serem desenvolvidas. A cada história implantada são geradas novas versões do sistema, além de agregar conhecimentos tanto na área de desenvolvimento quanto de projeto. Histórias não implantadas ou não finalizadas são passadas para as próximas iterações.

Através dos fundamentos teóricos e tecnológicos apresentados neste capítulo, definiu-se a base teórica para o desenvolvimento do sistema proposto neste trabalho. A metodologia de desenvolvimento descrita anteriormente e as tecnologias perceptivas são alguns desses fundamentos essenciais para o bom andamento do projeto. No próximo capítulo será descrito o desenvolvimento do sistema.

4. DESENVOLVIMENTO DO SISTEMA

Neste capítulo são apresentados o processo e o resultado do desenvolvimento do sistema interativo baseado em gestos. A proposta foi desenvolver uma aplicação que capture os dados fornecidos pela tecnologia perceptiva, execute um tratamento para identificar gestos realizados pelos membros superiores e traduza esses gestos em comandos no computador.

O capítulo apresenta os conceitos essenciais que tratam do desenvolvimento de um sistema computacional, tais como o levantamento de requisitos, definição de um modelo arquitetural, definição dos protótipos de interface e de gestos e o próprio processo de desenvolvimento.

4.1. REQUISITOS

No início de toda a atividade de desenvolvimento de software é necessário levantar os requisitos do sistema a ser desenvolvido. Sommerville (SOMMERVILLE, 2011) propõe um processo genérico de levantamento de requisitos que contém as seguintes atividades: compreensão do domínio, coleta dos requisitos, classificação dos requisitos e definição das prioridades. A compreensão do domínio e o levantamento dos requisitos foram realizados no decorrer do levantamento bibliográfico. Para poder representar inicialmente as funcionalidades propostas, foi utilizado um diagrama que facilita o entendimento das fronteiras do sistema, conhecido como diagrama de caso de uso.

Segundo o *Rational Unified Process* (RUP, 2013), o Diagrama de Caso de Uso representa, do ponto de vista do usuário do sistema, um cenário simplificado das funcionalidades do sistema. Ele é composto de casos de uso e atores. Os atores representam uma entidade externa ao sistema, podendo ser pessoas ou outros sistemas, e os casos de uso representam as funções ou funcionalidades disponibilizadas pelo sistema.

A classificação dos requisitos foi realizada para organizá-los em grupos coerentes. Essa organização foi baseada na possível representação da funcionalidade por meio de um tipo de gesto. A partir da classificação buscou-se

priorizar os requisitos levantados segundo sua relevância perante o uso cotidiano do computador pelo usuário.

Os requisitos foram separados em dois níveis de prioridade e foram considerados para desenvolvimento, inicialmente, apenas as funcionalidades com prioridade 1 (um), sendo os de prioridade 2 (dois) considerados como proposta para trabalhos futuros. Essas funcionalidades estão representadas na Figura 10.

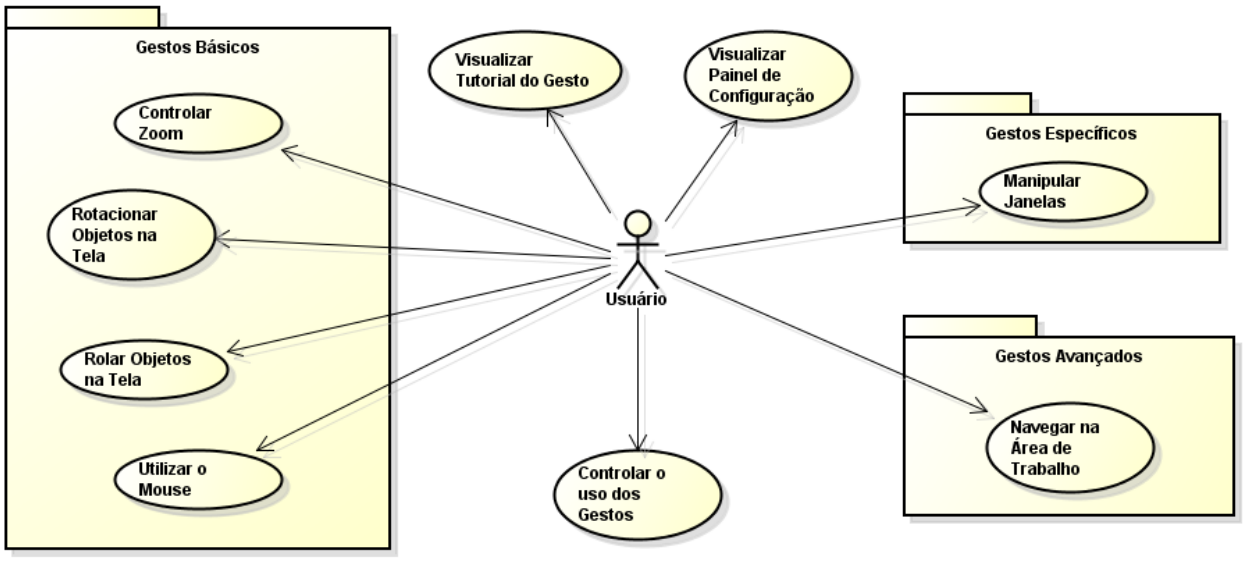


Figura 10. Diagrama de caso de uso do sistema

O diagrama foi dividido em três pacotes, de acordo com a classificação feita anteriormente. Cada pacote contém os casos de uso referentes aos gestos definidos por ele. Alguns casos de uso foram definidos fora dos pacotes por não estarem relacionados a um tipo de gesto.

A partir do entendimento inicial do sistema buscou-se representar os requisitos levantados segundo a metodologia utilizada, através de histórias de usuário. Essa representação está descrita na seção a seguir.

4.1.1. Histórias do usuário

Segundo o XP (XP, 2009), histórias do usuário ou *User Stories(US)* servem ao mesmo propósito dos Casos de Uso, porém não são a mesma coisa. As US descrevem um nível de abstração diferente e são utilizadas para criar estimativas de tempo para o desenvolvimento, além de evitar larga documentação dos requisitos.

Usualmente, cada história é um conjunto de frases curtas, em média 3 sentenças, escritas para definir o que o sistema deve fazer utilizando linguagem não técnica. Uma história de usuário deve prover detalhes suficientes para fazer uma estimativa razoável de tempo necessário para implantá-la. E deve ainda instruir a criação de testes para verificar se as funcionalidades estão sendo implantadas segundo sua especificação.

Na Tabela 5, as histórias do usuário definidas para o sistema. As histórias e seus respectivos critérios de aceitação (CA) foram baseados na especificação inicial dos requisitos. A descrição das histórias teve em vista os critérios de qualidade de histórias de usuário convencionados, conhecidos pelo acrônimo INVEST, no qual se recomenda que as histórias sejam, sempre que possível: independentes, negociáveis, valoráveis, estimáveis, pequenas e testáveis.

Tabela 5. Histórias do usuário.

Prior.	Id.	História de Usuário	Critérios de Aceitação
1	US 1	Como usuário, desejo visualizar informações sobre os comandos disponíveis no sistema para me situar adequadamente em relação à aplicação.	Deve existir uma janela sobre o sistema.
			O sistema deve possuir, em sua janela, abas dedicadas a cada tipo de gesto: Gestos Básicos, Gestos Específicos e Gestos Avançados.
			Cada aba deve apresentar as opções de “habilitar” ou “desabilitar” cada gesto, acompanhado de um vídeo tutorial.

1	US 2	Como usuário desejo rolar páginas para que eu possa visualizar todo o conteúdo na tela.	O sistema deve permitir que o usuário realize a rolagem vertical de conteúdo.
1	US3	Como usuário eu desejo rotacionar conteúdos na tela para que eu possa visualizá-los melhor.	O usuário deve poder, através do sistema, rotacionar um conteúdo (imagem) que esteja em foco.
1	US4	Como usuário desejo aumentar o zoom de conteúdos na tela para que eu possa visualizá-los melhor.	O sistema deve possibilitar que o usuário aumente a aproximação de conteúdos na tela através de gestos.
1	US5	Como usuário desejo diminuir o zoom de conteúdos na tela para que eu possa visualizá-los melhor.	O sistema deve possibilitar que o usuário reduza a aproximação de conteúdos na tela através de gestos.
1	US6	Como usuário desejo ter a opção de desabilitar um gesto para que o sistema não o reconheça.	Deve existir uma opção na janela do sistema para desabilitar determinado gesto
			O sistema deve permitir que com apenas um clique o usuário desabilite o reconhecimento de determinado gesto que esteja ativado.
1	US7	Como usuário desejo ter a opção de habilitar um gesto para que eu possa realizá-lo.	Deve existir uma opção na janela do sistema para habilitar determinado gesto
			O sistema deve permitir que com apenas um clique o usuário habilite o reconhecimento de determinado gesto que esteja

			desativado.
1	US8	Como usuário desejo clicar em informações da tela para que eu possa realizar alguma ação.	Quando o gesto de “clique” for habilitado, arquivos ou informações clicáveis da tela em foco poderão ser clicadas.
1	US9	Como usuário desejo selecionar algum conteúdo da tela para que eu possa realizar alguma ação.	Quando o gesto de “seleção” estiver habilitado, arquivos ou informações selecionáveis da tela em foco poderão ser selecionados.
1	US10	Como usuário desejo arrastar algum conteúdo da tela para que eu possa realizar alguma ação.	Quando o gesto de “arraste” estiver habilitado, arquivos ou informações arrastáveis da tela em foco poderão ser arrastados.
2	US11	Como usuário desejo navegar entre as janelas abertas na tela para que eu possa utilizar alguma.	Quando o gesto de “navegação” estiver habilitado, o usuário deve conseguir, através de gesto, alterar a tela em foco por outra que estiver aberta em segundo plano, podendo alternar entre as telas.
2	US12	Como usuário desejo minimizar uma janela aberta para que eu possa aumentar meu campo de visão na tela.	O sistema deve permitir que o usuário minimize a janela em foco, quando o gesto de “minimizar” estiver ativado.
2	US13	Como usuário desejo maximizar uma janela aberta	O sistema deve permitir que o usuário maximize a janela em

		para que eu possa melhor visualizar o conteúdo dela.	foco, quando o gesto de “maximizar” estiver ativado.
2	US14	Como usuário desejo fechar uma janela aberta para que eu possa não vê-la mais.	O sistema deve permitir que o usuário feche a janela em foco, quando o gesto de “fechar” estiver ativado.
2	US15	Como usuário desejo exibir o menu suspenso de um conteúdo na tela para que eu possa realizar alguma ação.	Quando o gesto de “exibir menu suspenso” estiver habilitado, o usuário deve conseguir, através de gesto, exibir o menu suspenso de arquivos ou informações que possuam essa opção na tela em foco.
2	US16	Como usuário desejo exibir a área de trabalho para que eu possa visualizar seu conteúdo.	O usuário deve ter a opção de ser redirecionado diretamente para a área de trabalho com apenas um gesto, independente da tela em que estiver o foco no momento.
1	US17	Como usuário desejo visualizar as configurações do sistema, para ter informações sobre o dispositivo eletrônico e o estado atual da aplicação.	Deve estar disponível na janela do sistema uma aba adicional contendo informações sobre o status do sistema: Conexão do sensor, Rastreio de esqueleto, Modo de orientação da mão, Modo de distância.

		O sistema deve conter informações dinâmicas sobre todas as ações que ocorrerem no sistema, como: gesto habilitado ou desabilitado, alteração de modo de distância, conexão com o sensor, e todas as ações possíveis do usuário.

4.1.1.1. Planejamento das releases

Após a escrita das histórias, o XP (XP, 2009) recomenda a criação de um plano de release. Esse plano especifica quais histórias serão implantadas em cada entrega do sistema e qual a duração de cada entrega. Isto possibilita a escolha das funcionalidades que serão desenvolvidas a cada iteração. As histórias selecionadas são traduzidas em tarefas individuais de programação que são implantadas durante a iteração para completar a história.

O desenvolvimento do sistema foi dividido em três *releases* para se adequar ao cronograma, seguir as recomendações da XP e distribuir melhor a entrega das funcionalidades. Elas ficaram com duas iterações cada, com todas tendo duração semanal. Essa estrutura está representada na Tabela 6, que mostra o período estimado para desenvolver cada US, sua classificação por prioridade e a quantidade de histórias por iteração.

Tabela 6. Planejamento das releases do sistema.

Período	Releases	Iterações	Histórias
Junho 2013	Release 1	Iteração 1	US 1: Visualizar Janela do Sistema
		Iteração 2	US 2: Realizar Rolagem de Página
			US 3: Realizar Rotação de Página
			US 17: Visualizar Janela de Configuração do Sistema
	Release 2	Iteração 3	US 4 Aumentar Zoom
			US 5: Diminuir Zoom
		Iteração 4	US 6: Desabilitar Gesto
			US 7: Habilitar Gesto
Julho 2013	Release 3	Iteração 5	US 8: Clicar
			US 9: Selecionar
			US 10: Arrastar
		Iteração 6	US 11: Visualizar Tutorial do Gesto

Segundo a XP, a essência do planejamento das releases é estimar as histórias em termos de semanas ideias de desenvolvimento[8]. Esse planejamento pode ser por tempo ou por escopo, onde, o primeiro, é medido através da quantidade de histórias que podem ser implementadas antes de um determinado tempo e, o segundo, quanto tempo uma história demora para ser finalizada.

Nesse trabalho foi escolhida a estimativa por tempo buscando alocar as histórias de forma que elas fossem finalizadas antes do final de cada iteração. As histórias foram classificadas por prioridade e divididas entre as iterações para evitar excesso de trabalho e atraso no prazo de entrega. A primeira e a última iteração foram subestimadas para possibilitar uma adaptação no prazo caso ocorresse algum imprevisto.

8 Quanto tempo se gasta para desenvolver uma história sem distrações (XP, 2009).

4.2. PROTÓTIPO DE INTERFACE

Segundo Gabin (GABIN, 2010), a forma como os usuários interagem com as máquinas e sistemas de informação, de um modo geral, tem sido alvo de estudos há muitos anos. Esse estudo é fundamental para que os usuários conheçam bem o modo como utilizar o sistema. Não adiantando um software realizar todas as funcionalidades que o cliente deseja se o próprio cliente não sabe como utilizá-las. Nesse relacionamento do tipo humano-computador, os elementos envolvidos devem comunicar-se de alguma forma e, para isso, utilizam um componente chamado de interface gráfica, que possibilita essa interação entre os dois meios.

Dessa forma, Gabin diz que o desenvolvimento de um sistema eficiente compreende não apenas a execução correta de tarefas, mas também requer uma interface simples e fácil de usar para o usuário. Logo, o desenvolvimento de software exige também um bom design de interação.

Segundo Preece (PREECE *et al*., 2005), design de interação significa criar experiências que melhorem e estendam a maneira como as pessoas trabalham, comunicam-se e interagem. O design de interação possui dois grupos de metas: metas de usabilidade e metas decorrentes da experiência do usuário. O primeiro compreende:

- **Eficácia**: se o sistema faz o que se espera dele.
- **Eficiência**: como o sistema auxilia os usuários na realização de tarefas.
- **Segurança**: evitar que o usuário seja exposto a condições perigosas ou indesejáveis.
- **Utilidade**: refere-se à medida na qual o sistema viabiliza as funcionalidades que o usuário deseja.
- **Facilidade de aprendizado**: refere-se à facilidade com que o usuário aprende a usar o sistema.
- **Facilidade de lembrança**: refere-se à facilidade de lembrar como utilizar o sistema.

O segundo está relacionado principalmente à melhoria da eficiência e da produtividade no trabalho. Assim, de acordo com Preece *et al*. (PREECE *et al.*, 2005), esse grupo avalia sistemas que sejam satisfatórios, agradáveis,

interessantes, úteis, motivadores, esteticamente apreciáveis, incentivadores de criatividade, compensadores e emocionalmente adequados.

Os protótipos de interface gráfica podem ser entendidos como modelos funcionais construídos com base em especificações preliminares para simular a aparência e a funcionalidade de um software a ser desenvolvido, ainda que de forma incompleta. Por meio de um protótipo, os futuros usuários do software e aqueles que irão desenvolvê-lo poderão interagir, avaliar, alterar e aprovar as características mais marcantes da interface e da funcionalidade da aplicação.

Dessa forma, a proposta de protótipo de interface do sistema interativo que foi desenvolvido é apresentado na Figura 11, onde estão ilustradas os quatro telas disponíveis para o usuário.

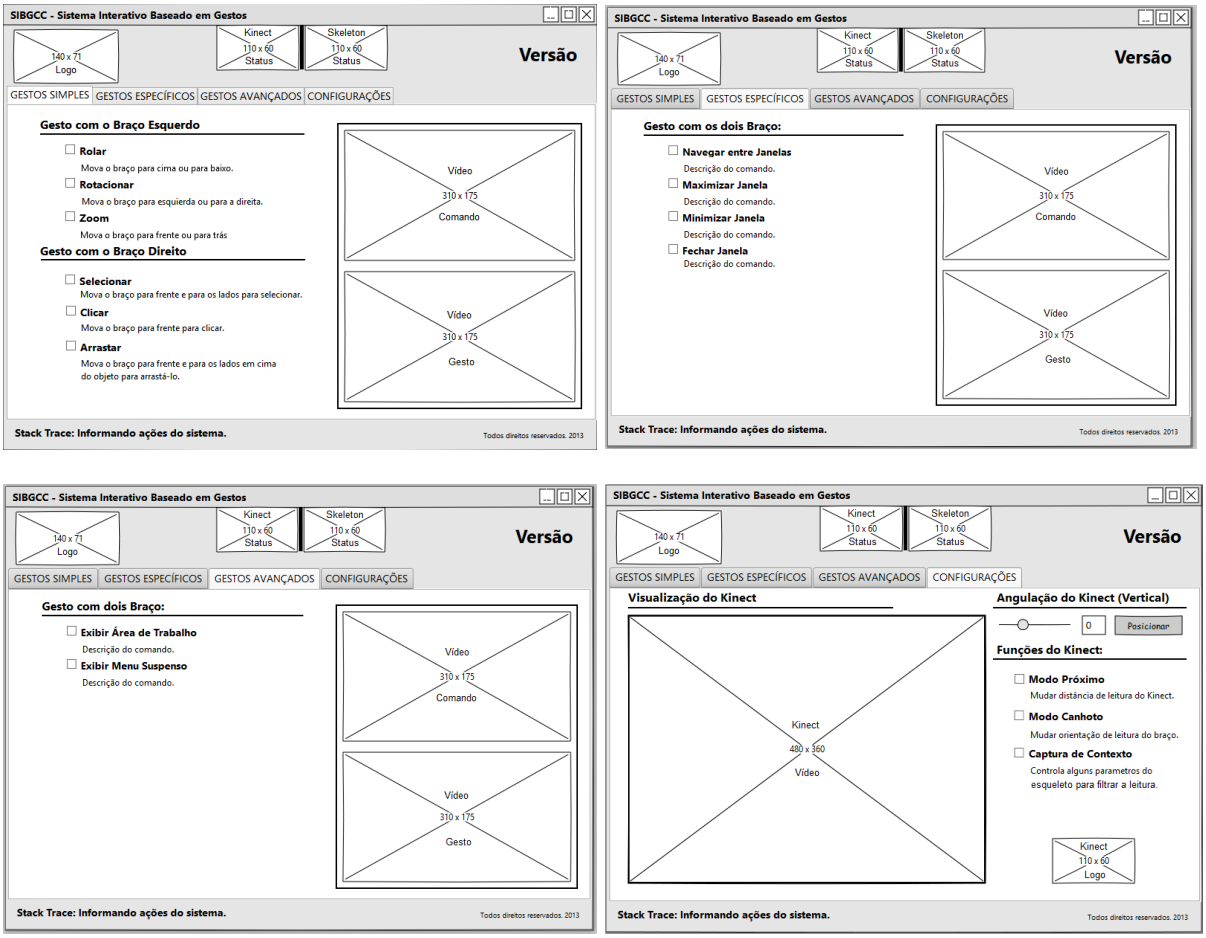


Figura 11. Protótipos de interface do sistema (Apêndice I)

Foram definidas inicialmente três telas de acordo com a divisão dos gestos (geral, específico e avançado) e uma tela durante o desenvolvimento para configurar o dispositivo eletrônico. Tal divisão foi realizada devido à complexidade de implantação e execução dos movimentos e ao entendimento do comando.

Como a ideia foi desenvolver um sistema resposivo à mudanças, os protótipos das telas sofreram alterações no decorrer do desenvolvimento. Eles estão representados na versão final no Apêndice I desse documento

4.3. PROTÓTIPOS DOS GESTOS

Assim como foi realizada a prototipação da interface gráfica, foi necessário prototipar os gestos que seriam realizados pelo usuário para que se pudesse observar as limitações do corpo humando na movimentação dos membros superiores e consequentemente as fronteiras do algoritmo de reconhecimento dos gestos.

Com base em tal cenário, foram procuradas na literatura referências que pudessem fundamentar ou até justificar a definição dos gestos para cada comando. Os trabalhos correlatos e um guia de interface do usuário para a última versão do SDK do Kinect (KINECT UI. 2013) foram utilizados para tal tarefa. O primeiro serviu como a base central na definição dos gestos, onde se especificou quais partes dos membros do corpo seriam utilizadas e quais movimentos essas partes poderiam fazer (horizontal, vertical Frente/Traz ou diagonal) e o segundo serviu como um guia para orientar quanto às considerações de design dos gestos.

A partir desse levantamento foram definidos os protótipos dos gestos de arrastar e selecionar, clicar, rolar, rotacionar e aplicar zoom apresentados na Figura 12, respectivamente.

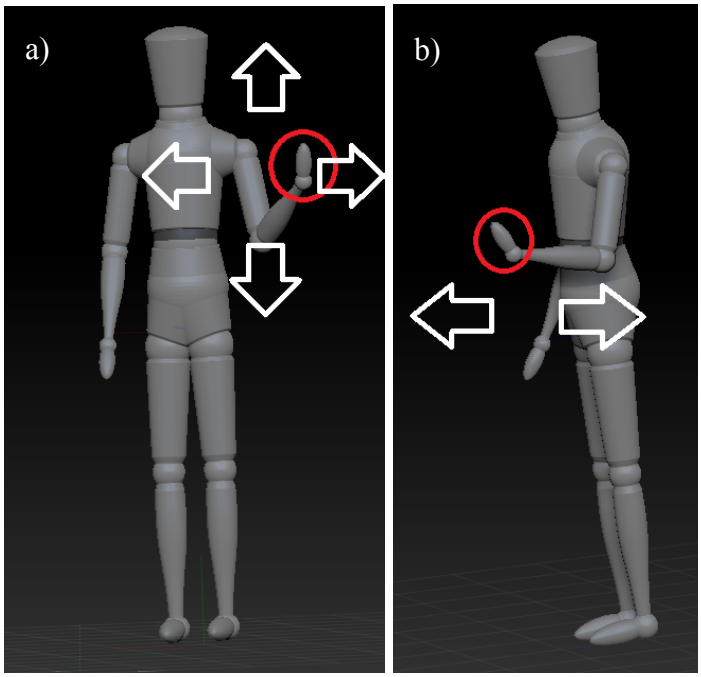


Figura 12 Protótipos dos gestos: a) Arrastar e Selecionar b) Clicar

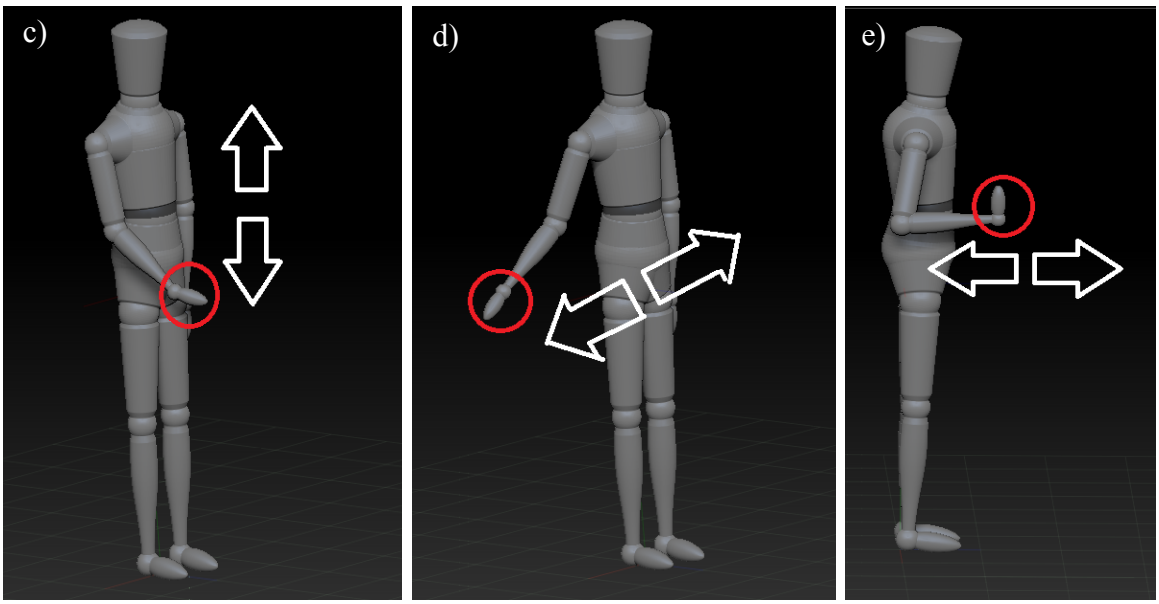


Figura 12. Protótipos dos gestos: c) Rolar, d) Rotacionar, e) Aplicar Zoom

Os protótipos definidos podem ser entendidos como modelos simples construídos a partir de recomendações e considerações de design preliminares para

simular a aparência do gesto a ser implantado, ainda que de forma visual. Eles foram divididos em duas orientações: os realizados pela mão direta e os realizados pela mão esquerda, onde houve uma oposição na mão entre os comandos clicar, selecionar e arrastar com os de rolar, rotacionar e aplicar zoom, seguindo a lógica do mouse e do teclado, respectivamente.

A representação do gesto de navegação do mouse foi baseada na sua manipulução bidimensional (movimento da mão na horizontal e vertical) através do protótipo a da Figura 12 e os comandos do mouse (movimento para Frente/Traz) através da combinação com o gesto do protótipo b.

Os gestos de rolar, rotacionar e aplicar zoom foram baseados nos protótipos c, d e e da Figura 12, respectivamente. O gesto de rolar através de um movimento da mão na vertical, o de rotacinar na horizontal e o de aplicar zoom para frente e para traz.

Através da prototipação dos gestos foi possível avaliar, alterar e testar alguns paramentos utilizados nos algoritmos de reconhecimento baseado em Catuhe (CATUHE, 2012) que serão descritos mais à frente, como tempo de execução, duração, altura, largura, profundidade e sentido. A Tabela 7 apresenta uma especificação inicial dos valores dos paramentros dos gestos rolar, rotacionar e aplicar zoom.

Tabela 7**.** Paramentos do algoritmo de reconhecimento dos gestos: rolar, rotacionar e aplicar zoom

Gesto	Altura	Largura	Profundidade	Sentido	Período	Tamanho
Rolar	0.4	0.4	0.3	X+ X-	950ms	0.35
Rotacionar	0.4	0.4	0.3	Y+ Y-	950ms	0.35
Aplicar Zoom	0.4	0.4	0.3	Z+ Z-	950ms	0.35

Inicialmente foram utilizados os valores dos paramentros defnidos na referência. No decorrer do desenvolvimento esse paramentros foram testados e

atualizados de acordo com o contexto de utilização. Os gestos de manuseio do mouse utilizam um algoritmo simplificado que será explicado mais a frente.

4.4. ARQUITETURA

Segundo o RUP (RUP, 2013), a arquitetura de um sistema consiste na definição dos componentes do software, suas propriedades externas e seus relacionamentos com outros softwares e hardwares. O termo também se refere a um modelo conceitual que facilita a transição de requisitos para a implantação, registra as decisões iniciais acerca do projeto de alto nível, permite compreender melhor o software e facilita o reuso do projeto, dos seus componentes e padrões.

O modelo de visão arquitetural de software “4+1” desenvolvida por Philippe Kruchten (KRUCHTEN, 1995) tem o objetivo de descrever o funcionamento de sistemas de software baseando-se no uso de múltiplas visões concorrentes.

As visões são usadas para mostrar o sistema sob várias perspectivas, como usuário final, desenvolvedores e gerentes de projetos. As quatro visões do modelo são: visão lógica, visão de desenvolvimento, visão de processo e visão física. A visão de caso de uso é usada para ilustrar, também, a arquitetura e representa o +1.

No contexto desse trabalho será apresentada a visão lógica para representar a arquitetura do software que será desenvolvido. Ela se concentra nas funcionalidades disponibilizas para o usuário final, descrevendo como o sistema é estruturado na forma de unidades de implementação: pacotes, classes ou interfaces (KRUCHTEN, 1995). Essa visão utiliza entre outros o diagrama de comunicação da UML[9] para apresentar, de maneira semelhante ao diagrama de sequência, a colaboração dinâmica entre os objetos.

A definição da estrutura do sistema foi baseada na arquitetura apresentada por Matsumura (MATSUMURA et *al*., 2011), mostrada na Figura 13, possibilitando o desenvolvimento paralelo de cada pacote, ao encapsular níveis de abstração distintos e permitir a fácil reutilização deles.

---

9 *Unified Modeling Language* ou Linguagem de Modelagem Unificada.

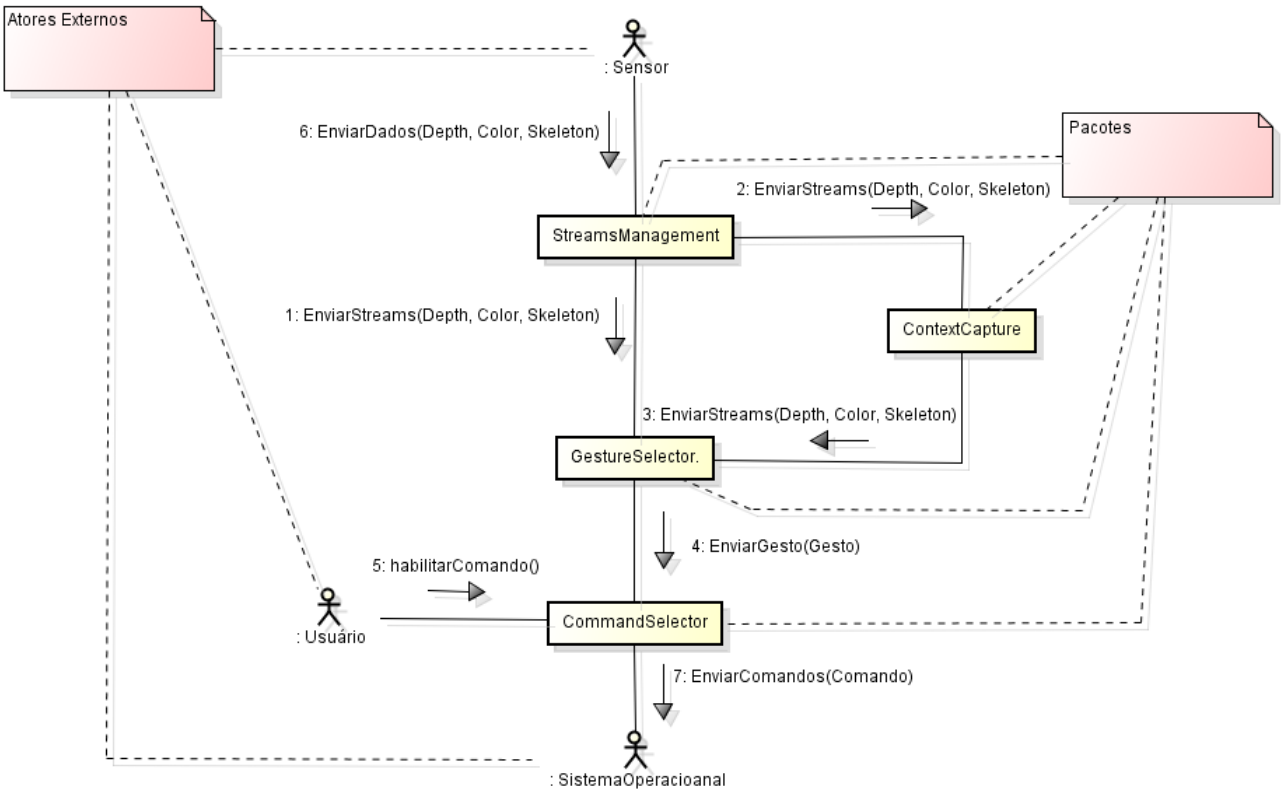


Figura 13. Diagrama de comunicação do sistema

A representação do diagrama apresentado acima é feita utilizando os pacotes e atores do sistema, descritos abaixo:

- **Gerenciador de Streams**: possui comunicação direta com a tecnologia perceptiva e é responsável pelo rastreamento e gerenciamento das informações do usuário providas pelo sensor.
- **Captura de Contexto**: responsável por filtrar dos dados recebidos do Gerenciador de Streams opcionalmente e retorná-los ao Seletor de Gestos.
- **Seletor dos Gestos**: responsável pela seleção dos gestos baseada no rastreamento das informações do usuário.
- **Seletor de Comandos**: responsável pela execução dos comandos habilitados pelo usuário, que modificam o estado da cena (sistema operacional) ao reconhecê-los através do gesto.

A partir dessa definição inicial da arquitetura buscou-se entender melhor o escopo de desenvolvimento do sistema. Ele compreendeu à obtenção dos dados do sensor, captura do contexto, interpretação dos gestos executados pelo usuário e sua tradução em comandos para o sistema operacional, apresentada através de uma interface simplificada.

4.5. CONFIGURAÇÃO DO AMBIENTE

A configuração do ambiente foi considerada uma etapa importante do projeto devido à necessidade de abordar alguns elementos da Engenharia de Software. Inicialmente, foi realizada uma pesquisa para obter algumas ferramentas de gerência de projeto, desenvolvimento de sistemas, gerência de configuração e controle de *builds*[10] e testes.

Considerou-se utilizar uma abordagem com ferramentas de código aberto a exemplo do Eclipse, como IDE, IceScrum, como ferramenta de gerência de projeto, Maven, como ferramenta de controle de *build*, Jenkins, como ferramenta para integração continua, e Git, como ferramenta para gerência de configuração. Mas devido à utilização de um hardware proprietário e do padrão de configuração observado nas referências pesquisadas, optou-se pela escolha de ferramentas integradas à tecnologia perceptiva escolhida. O pacote de ferramentas utilizado para desenvolver o sistema é apresentado a seguir.

4.5.1. Ferramenta de gerência do projeto

A ferramenta de gerência de projeto escolhida é integrada nativamente ao ambiente de desenvolvimento integrado (IDE) recomendado para o Kinect. Conhecida como Team Foundation Service (TFS), ela possui um acoplamento direto com a IDE Visual Studio e disponibiliza alguns serviços que auxiliam na entrega de software (TFS, 2013).

Os serviços disponibilizados por essa ferramenta e utilizados no trabalho são: planejamento ágil, controle de versão, integração contínua e execução de testes. A Figura 14 apresenta uma simplificação da estrutura do TFS.

---

10 Versão "compilada" de um software ou parte dele que contém um conjunto de recursos que poderão integrar o produto final (BUILDWIKI, 2013).

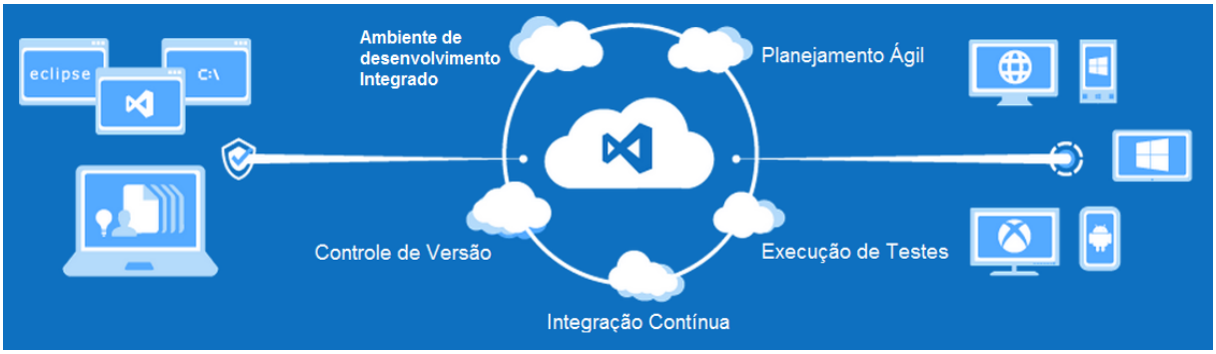


Figura 14. Estrutura do TFS. Fonte: (TFS, 2013)

Para desenvolver o sistema foi criado um projeto de nome SIBGCC na ferramenta que pode ser acessado pelo link "https://filipebarbosa.visualstudio.com/DefaultCollection/SIBGCC". Nele foi possível obter um histórico de todo o processo de desenvolvimento do sistema. Onde pode-se observar a estrutura das iterações descritas anteriormente no Planejamento das Releases (Figura 15). Para cada iteração foram definidas suas respectivas funcionalidades e histórias.

Iterations	Start Date	End Date
SIBGCC	31/05/2013	12/07/2013
Release 1	31/05/2013	28/06/2013
Iteração 1	31/05/2013	07/06/2013
Iteração 2	07/06/2013	14/06/2013
Release 2	14/06/2013	28/06/2013
Iteração 3	14/06/2013	21/06/2013
Iteração 4	21/06/2013	28/06/2013
Release 3	28/06/2013	12/07/2013
Iteração 5	28/06/2013	05/07/2013
Iteração 6	05/07/2013	10/07/2013

Figura 15. Cronograma do SIBGCC no TFS. Fonte: (SIGBCC, 2013)

Através das histórias foram definidas tarefas para implantar sua funcionalidade (Fig. 16). Essas tarefas foram estimadas seguindo a lógica de duração explicada anteriormente, e caso ficassem maiores que o definido, elas seriam divididas em tarefas menores. A maioria das tarefas foi vinculada a um *commit*[11] para representar sua execução.

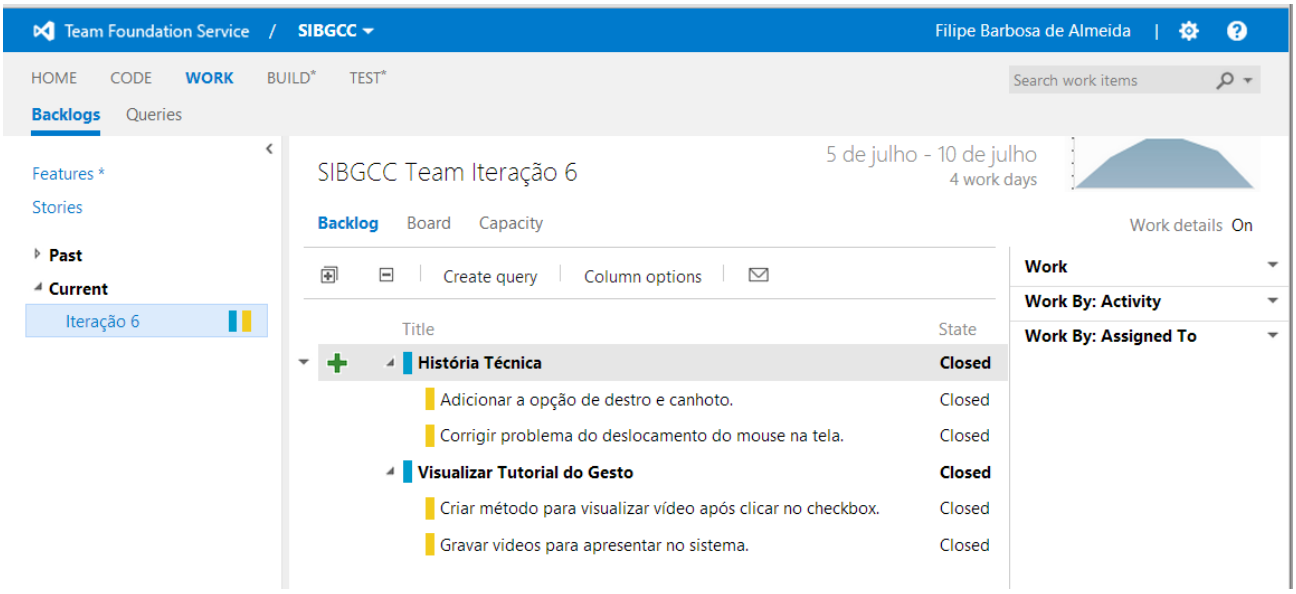


Figura 16. Visualização das histórias e tarefas no TFS. Fonte: (SIGBCC, 2013)

O TFS propiciou um serviço de integração contínua, onde a cada *commit* realizado para o repositório remoto era criada uma *build* do sistema e eram executados os testes implantados. Quando a criação da *build* e/ou a execução dos testes encontrava algum erro, o próprio TFS informava por e-mail o ocorrido e criava uma tarefa com a descrição do problema.

4.5.2. Gerência de configuração

O gerenciamento de configuração do sistema foi realizado de forma integrada com a IDE e o TFS. Definiu-se no início do desenvolvimento que para cada iteração seria criada uma *branch*, ou seja, uma ramificação do código principal do sistema (*master*), para facilitar o controle das versões do software produzidas a cada ciclo de desenvolvimento. Poderia ter-se utilizado uma única linha de desenvolvimento por

11 Serviço de envio de código disponibilizado pela integração do Visual Studio com o TFS.

ser mais simples, mas por opção definiu-se essa estratégia para facilitar a visualização das versões do código. No final da iteração, a *branch* criada era integrada novamente a *master* do código.

Também foi definida uma estrutura de versionamento baseada nas recomendações da IDE, onde a numeração da versão foi definida usando a estrutura “0.0.0”, sendo o primeiro digito a versão principal, baseada na release corrente; o segundo, a versão secundária, baseada no número de *branchs* finalizadas na release e o terceiro na quantidade de *builds* já geradas na *release*.

4.6. SISTEMA - SIBGCC

O desenvolvimento do sistema interativo baseado em gestos para utilização de comandos no computador foi realizado utilizando o paradigma orientado à objetos, algumas técnicas de escrita de código segundo o guia de programação C# da Microsoft (MSDN C#, 2013), tratamento de exceções, e uma abordagem *Top-Down*, onde iniciou-se pela interface para implantar uma funcionalidade até chegar ao código.

O Visual Studio é uma IDE que propicia esse tipo de abordagem através da separação da interface em uma classe *XAML* (Fig. 17) e do código em uma classe *C#.*

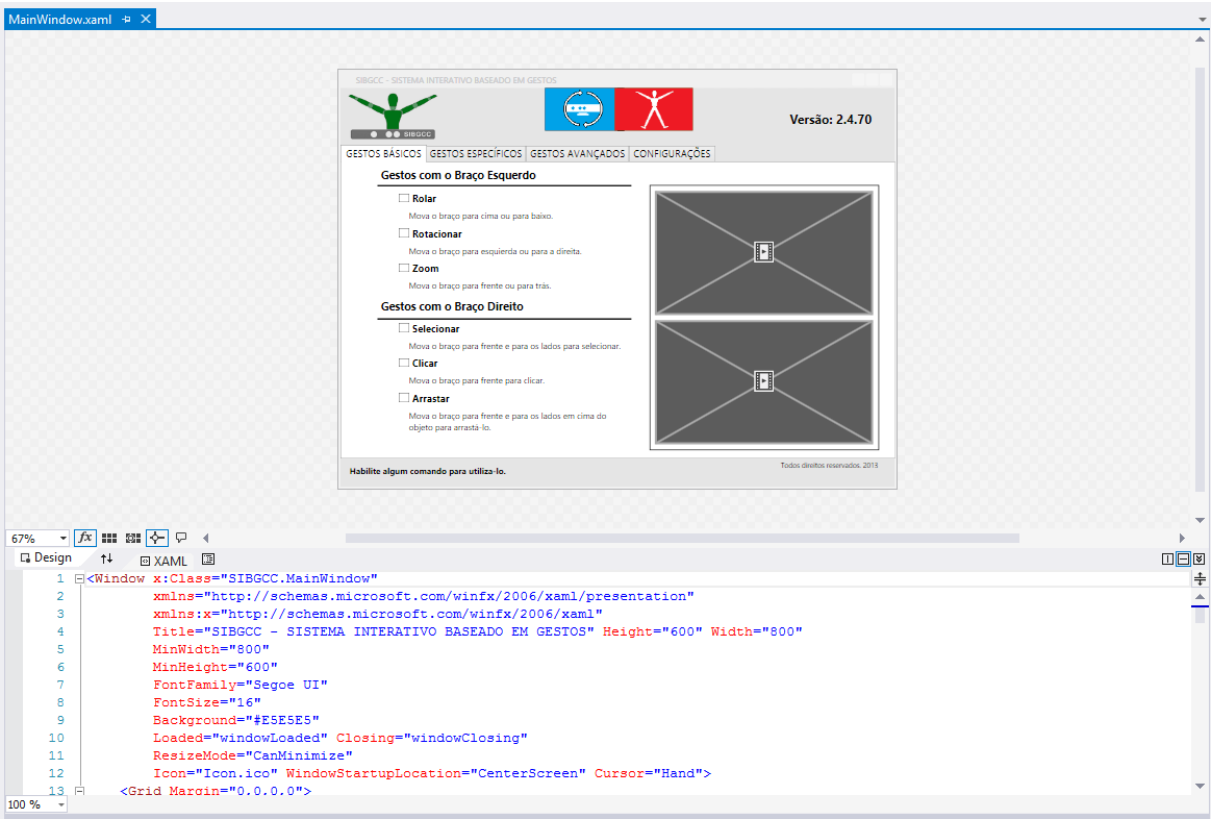


Figura 17. Construindo a interface do sistema.

Na classe XAML é possível editar a interface gráfica do sistema através de um código simples e da manipulação de objetos na tela. Como exemplo pode-se citar a adição de um botão, cuja instância é criada inicialmente na interface e seu método é criado no código em C#.

O problema inicial a ser abordado no desenvolvimento de aplicações para o Kinect é escolher o kit de desenvolvimento e suas APIs. Com a grande popularização do dispositivo, a Microsoft passou a disponibilizar novas versões do seu SDK proprietário, com lançamentos de versões mais frequentemente e mais estáveis que a concorrência (KEDAVRA, 2011). A última versão lançada e escolhida para desenvolver o sistema foi a versão 1.7, que trouxe algumas melhorias na parte de reconhecimento das articulações das mãos e na interface com o usuário.

Com a escolha do SDK, o início do desenvolvimento passa a ser facilitado com a utilização de classes e métodos pré-estabelecidos disponíveis na API. Onde uma das principais vantagens do SDK proprietário, segundo Kedavra, é sua capacidade de encontrar as *Joints* do usuário usando um sistema de

reconhecimento rápido e preciso que não exige pré-configuração, porque uma máquina de aprendizagem já foi instruída para reconhecer o esqueleto.

Convém salientar que instanciar um objeto do tipo *Kinectsensor* e executar os métodos de captura do *Stream* de cor, profundidade e esqueleto são implantações relativamente simples, como é apresentado no trecho de código demostrado na Figura 18.

```
private KinectSensor sensor;

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
sensor.ColorFrameReady += sensorColorFrameReady;

sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
sensor.DepthStream.Range = DepthRange.Near;
sensor.DepthFrameReady += sensorDepthFrameReady;

sensor.SkeletonStream.EnableTrackingInNearRange = true;

sensor.SkeletonStream.TrackingMode = SkeletonTrackingMode.Seated;
```

Figura 18. Trecho de código para inicialização do Kinect.

Pode-se observar no código acima que a habilitação dos *Streams* possibilita acesso a uma imagem RGB de resolução 640 por 480 a 30 *frames* por segundo, a uma imagem de profundidade de resolução 640 por 480 a 30 *frames* por segundo e à leitura das partes do corpo do usuário em uma classe do tipo *Skeleton* no modo de rastreamento próximo e sentado.

A classe *Skeleton* possui uma coleção de 20 juntas ou *Joints,* que são representações de partes do corpo, como pode ser visto na Figura 19. Quando um esqueleto é detectado, ele chama o método *SkeletonFrameReadyArgs* enviando um *SkeletonFrame* como parâmetro, e dentro dele temos uma coleção de seis esqueletos (*SkeletonData*), onde apenas dois deles são mapeados.

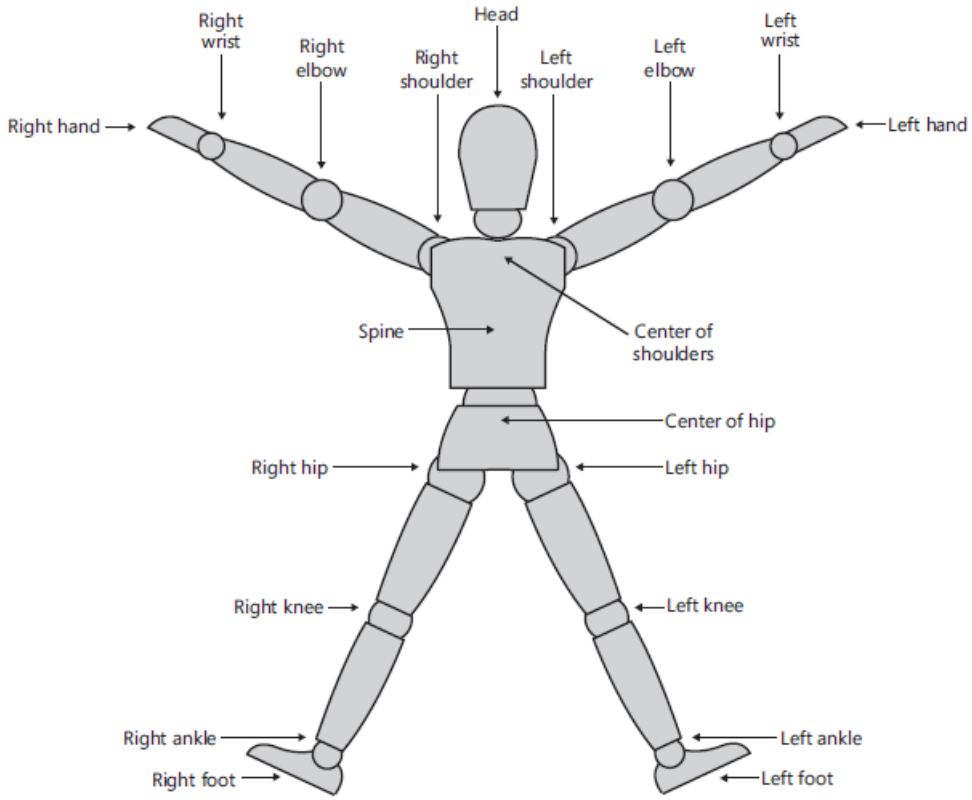


Figura 19. Os 20 *Joints* do *Skeleton*. Fonte: (CATUHE, 2012)

Cada *Joint* é definido por uma posição (x, y, z) expressa no espaço *skeleton.* Esse espaço é definido em volta do sensor, que é localizado no ponto (0,0,0) – ponto onde os eixos x, y e z se encontram na Figura 20. As coordenadas são expressas em valores decimais, onde os eixos x e y variam de -1 a 1 e o eixo z de acordo com a distância em metros do Kinect.

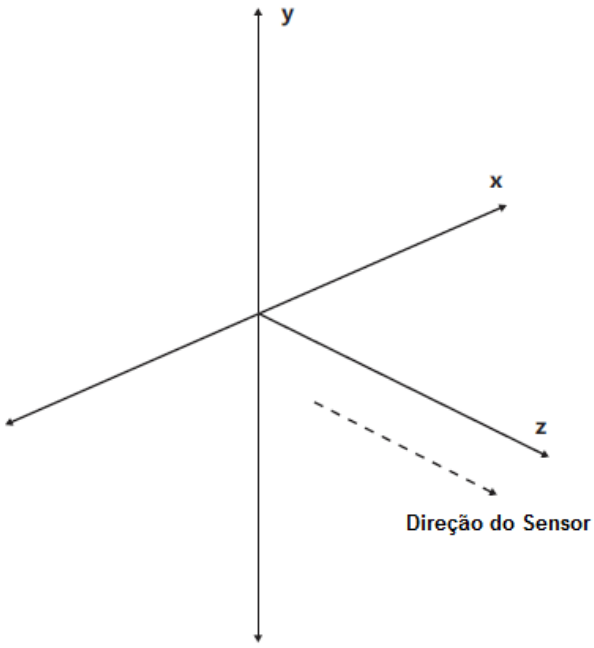


Figura 20. Eixo de reconhecimento do Kinect. Fonte: (CATUHE, 2012)

Como se observa na Figura 20, o eixo x estende-se para a direita (do ponto de vista do utilizador), o eixo y se estende para cima, e o eixo z é orientado partindo do sensor para o usuário.

A partir do SDK 1.5, foi adicionada ao Kinect a capacidade de rastrear os usuários sentados. Nesse caso, apenas as 10 articulações da parte superior do corpo (cabeça, ombros, cotovelos, braços e pulsos) são rastreadas.

4.6.1. Captura de Contexto

Antes de começar a reconhecer gestos Catuhe (CATUHE, 2012) recomenda compreender o “contexto em discussão”. Esse contexto define as características que ajudam a entender o que o usuário quer expressar.

Os gestos são categorizados por ele em: desejados e não desejados. Os desejados são aqueles que o usuário produz voluntariamente e que são os que o sistema quer detectar. Os não desejados são aqueles que o sistema detecta erroneamente.

A capturar o contexto auxilia o algoritmo de reconhecimento a filtrar os gestos falsos dos gestos verdadeiros, que são os definidos. Para o sistema, não é uma

questão de entender o que o usuário está fazendo e sim usar o contexto para obter algumas dicas sobre a vontade do usuário.

A referência cita quatro ferramentas para implementar a captura de contexto: verificar a estabilidade do *skeleton,* verificar a velocidade de deslocamento do *skeleton*, determinar a orientação do *skeleton* e detectar a posição dos olhos do *skeleton*. Dentre as opções, optou-se por implementar no sistema a segunda e a terceira ferramenta.

A segunda verifica a velocidade de deslocamento do *skeleton* considerando que o corpo humano naturalmente não fica totalmente parado. Sendo necessário para realização do cálculo as características de posição e tempo do centro de gravidade do *skeleton*.

A terceira determina a direção do usuário, onde a orientação do *skeleton* é importante porque o gesto pode ser só detectado quando o usuário estiver de frente, centrado e olhando para o sensor. Para detectar a orientação, verifica se a distância dos ombros do *skeleton* em relação ao sensor é a mesma, dado um valor limite.

A incorporação destas ferramentas no sistema é feita através do trecho de código apresentado abaixo (Fig. 21).

```
...
    if (skeletons.Length != 0){
        foreach (var skeleton in skeletons){
            if (skeleton.TrackingState == SkeletonTrackingState.Tracked){
                skeletonManager.Draw(skeleton);
                contextTracker.Add(skeleton.Position.ToVector3(), skeleton.TrackingId);
                if (!contextTracker.IsStableRelativeToCurrentSpeed(skeleton.TrackingId) ||
                contextTracker.IsShouldersTowardsSensor(skeleton) == false)
                    continue;
...
```

Figura 21. Trecho de código sobre captura de contexto

Nele é verificado inicialmente a existência de algum *skeleton* de usuário. Então para cada skeleton é validado sua condição de rastreamento e caso seja positiva ele é adicionado pelo método *Add* da classe *contextTacket* para sua

verificação. No final, são analisadas uma ou duas condições do *skeleton*: a velocidade relativa em relação a velocidade atual pelo método *IsStableRelativeToCurrentSpeed* (ANEXO I) e a orientação dos ombros pelo *IsShouldersTowardsSensor* (ANEXO II). Após a captura de contexto é implementado o algoritmo de reconhecimento dos gestos.

4.6.2. Algoritmo de reconhecimento

Segundo o Catuhe (CATUHE, 2012), o Kinect pode ser considerado uma excelente ferramenta para se comunicar com o computador. E uma das maneiras mais óbvias de se realizar essa comunicação é através de gestos. Para ele, um gesto é o movimento de uma parte do corpo ao longo do tempo, como quando se move a mão da direita para a esquerda para simular uma direção.

Como o gesto é nada mais que um movimento, tentar detectá-lo pode ser entendido como um processo de detecção de um movimento pré-definido, solução que pode ser adotada para detectar movimentos lineares, como o deslizar da mão em algum sentido (Fig. 22).

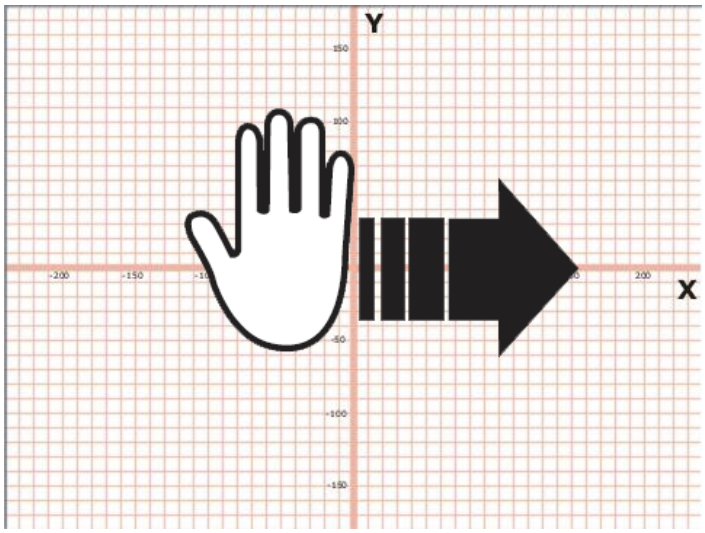


Figura 22. Caminho do Gesto

O princípio por trás da captura do gesto para usá-lo como comando não é complicado. A ideia é capturar as últimas n posições de determinado *Joint* e aplicar um algoritmo de reconhecimento sobre elas para detectar um gesto. Essas posições

são disponibilizadas no formato do eixo de reconhecimento do Kinect e podem ser interpretadas seguindo a lógica da figura apresentada acima.

Para detectar um gesto no sistema foram definidas as seguintes etapas baseadas no Catuhe (CATUHE, 2012):

- Capturar um ou mais *Joints;*
- Obter seus valores x, y e z;
- Verificar se os pontos estão em progressão para algum sentido.
- Verificar se os pontos não estão muito longe do início em relação ao eixo de orientação.
- Verificar se o primeiro e último pontos estão a uma distância específica um do outro.
- Verificar se o primeiro e último pontos foram criados dentro de determinado período de tempo.

Esse método de detecção é uma maneira genérica de verificar as junções do usuário. Ele busca todas as entradas e verifica para ter certeza que elas respeitam os parâmetros de tempo de execução, duração, altura, largura, profundidade e sentido, definidos independentemente para cada gesto de acordo com seu protótipo.

A implantação da estrutura do algoritmo no código é apresentada nas figuras abaixo referentes ao método, em específico, de detecção do gesto de rotação para direta (Fig. 23) e do método *ScanPositions* padrão (Fig. 24). O primeiro verifica a altura, a largura e a profundidade do movimento do *Joint.*

```
…
if (ScanPositions((p1, p2) => Math.Abs(p2.Y - p1.Y) < SwipeMaximalHeight,(p1, p2) => p2.X -
p1.X > -0.01f, (p1, p2) => Math.Abs(p2.X - p1.X) > SwipeMinimalLength,
SwipeMininalDuration, SwipeMaximalDuration)){
            RaiseGestureDetected("DeslizarParaDireita");
   return;
}

 …
```

Figura 23. Trecho de código sobre método para detectar o deslizar da mão para direita

A verificação é realizada através de uma condição *if* que considera como valor o resultado do método *ScanPositions*. O segundo verifica o sentido, o período e o tamanho.

```
…
protected bool ScanPositions(Func<Vector3, Vector3, bool> heightFunction, Func<Vector3, Vector3,
bool> directionFunction, Func<Vector3, Vector3, bool> lengthFunction, int minTime, int maxTime){

        int start = 0;

        for (int index = 1; index < Entries.Count - 1; index++){

         if (!heightFunction(Entries[0].Position, Entries[index].Position) ||
         !directionFunction(Entries[index].Position, Entries[index + 1].Position)){
              start = index;
           }

           if (lengthFunction(Entries[index].Position, Entries[start].Position)){
              double totalMilliseconds = (Entries[index].Time - Entries[start].Time).TotalMilliseconds;

              if (totalMilliseconds >= minTime && totalMilliseconds <= maxTime){
                 return true;
              }
           }
        }
        return false;
     }

…
```

Figura 24. Trecho de código sobre verificação das junções.

Após identificar um potencial gesto a condição chama o método *RaiseGestureDetected* (Fig. 25) para comparar se o tempo entre o gesto reconhecido no momento e o anterior é maior que um valor pré-definido. Caso a condição seja positiva, é chamado o método *OnGestureDetected*.

```
...
        protected void RaiseGestureDetected(string gesture){

            if (DateTime.Now.Subtract(lastGestureDate).TotalMilliseconds >
            MinimalPeriodBetweenGestures){

                if (OnGestureDetected != null)
                    OnGestureDetected(gesture);

                lastGestureDate = DateTime.Now;
            }
        }

...
```

Figura 25. Trecho do código sobre a comparação do tempo entre a execução dos gestos.

Esse método recebe como parâmetro uma *string* com o valor passado na condição do *ScanPositions* e a partir dele um controle de fluxo seleciona qual comando habilitado deve ser executado pelo sistema (Fig. 26).

```
...
        private void onGestureDetected(string gesture){

            switch(gesture){
                case "DeslizarParaDireita":
                    if (rolarCheck.IsChecked == true){
                        SendKeys.SendWait("^(.)");
                        statusBarText.Text = gesture;
                    }
                break;
                case "DeslizarParaEsquerda":
                    if(rotacionarCheck.IsChecked == true){
                        SendKeys.SendWait("^(,)");
                        statusBarText.Text = gesture;
                    }
                break;
...
```

Figura 26. Trecho do código sobre a execução do comando.

No caso do manuseio do mouse utilizou-se uma abordagem um pouco diferente. Como um *joint* já estava sendo utilizado para detectar um conjunto suficiente de gestos, determinou-se outro para poder executar a navegação do mouse e os comandos de clique, seleção e arraste. O processo de detecção do movimento é o mesmo, do qual são obtidos os valores x, y, e z do *joint* específico, mas diferentemente do processo anterior, se convertem os valores x e y obtidos, variando de -1 a 1, para o valor do tamanho da tela do computador. No trecho de código a seguir é apresentado o método de escalonamento implantado no sistema (Fig. 27). Esse método foi obtido pela biblioteca *Coding4Fun.Kinect.Wpf* (CODEPLEX, 2013).

```
…
if (clicarCheck.IsChecked == true){

    scaledJoint = joint.ScaleTo((int)SystemParameters.PrimaryScreenWidth,
    (int)SystemParameters.PrimaryScreenHeight, SkeletonMaxX, Skele-
    tonMaxY);

    if ((headJoint - joint.Position.Z) >= 0.3f && leftClickCount == 0){
        leftClick = true;
    if (selecionarCheck.IsChecked == false && arrastarCheck.IsChecked ==
    false)
        leftClickCount = 1;
    }else{
        leftClick = false;
        if (leftClickCount <= 10)
            leftClickCount++;
        else
            leftClickCount = 0;
     }

    MouseCommnads.MouseControl((int)scaledJoint.Position.X, (int)scale-
    dJoint.Position.Y, (int)SystemParameters.PrimaryScreenWidth, (int)Sys-
    temParameters.PrimaryScreenHeight, leftClick);
}

…
```

Figura 27. Trecho de código sobre método para manusear o mouse.

Através do trecho do código exposto acima, é possível observar o escalonamento realizado pelo método *ScaleTo* utilizando os parâmentos de largura e altura da tela e um limitador de movimento, além da conversão dos valores para o método *SendMouseInput*. Esse método da classe *NativeMethods* executa o resto do processo de comunicação com o sistema operacional. Os comandos de clique, selecionar e arrastar são validados através da coordenada z do *Joint*.

4.7. RESULTADOS

Dentre os objetivos do trabalho, cita-se inicialmente como importante resultado o conhecimento agregado no decorrer do levantamento bibliográfico e da avaliação das tecnologias perceptivas. Com um foco prático, o aprendizado sobre computação perceptiva e sobre conceitos relacionados à área de Engenharia de Software trouxe grande benefício para o desenvolvimento do sistema. Considerada a atividade mais relevante do trabalho, ela demandou uma atenção maior por sua relação direta com o resultado almejado.

A configuração do ambiente trouxe benefícios para o desenvolvimento do software, como o controle das tarefas, o controle da versão do código e a automação dos *builds* e testes. Alguns problemas de integração da IDE com TFS,

causados pela dificuldade de conexão com o repositório remoto, acarretaram diminuição no tempo de desenvolvimento do sistema.

As histórias foram implantadas seguindo o planejamento descrito anteriormente, e a cada tarefa de história finalizada era realizado um *commit* vinculado a ela para controle.

O processo de codificação foi realizado utilizando boas práticas de programação (KERNIGHAN, 1999), como simplicidade, clareza, generalização e internacionalização, e de forma gradativa devido às dificuldades técnicas e conceituais encontradas tanto na linguagem de programação C# quanto em definir os parâmetros dos gestos e traduzi-lo para um algoritmo de reconhecimento. Abordou-se, também, alguns padrões de projeto e técnicas de refatoração para melhorar a manutenção e reutilização do código. Entre os padrões utilizados pode-se citar a fábrica abstrata e *singleton* descritos por Gamma (GAMMA, 2000) e entre as técnicas de refatoração renomeação, extração de método, mover método (ANEXO III) e extração de classe descritas por Folwer (FOLWER *et. al*, 2004).

Ocorreram atrasos em algumas iterações devido ao mal dimensionamento das histórias, mas como isso já havia sido previsto, houve uma compensação nas iterações seguintes. Na Figura 28 é apresentado o fluxo acumulado de histórias durante o projeto.

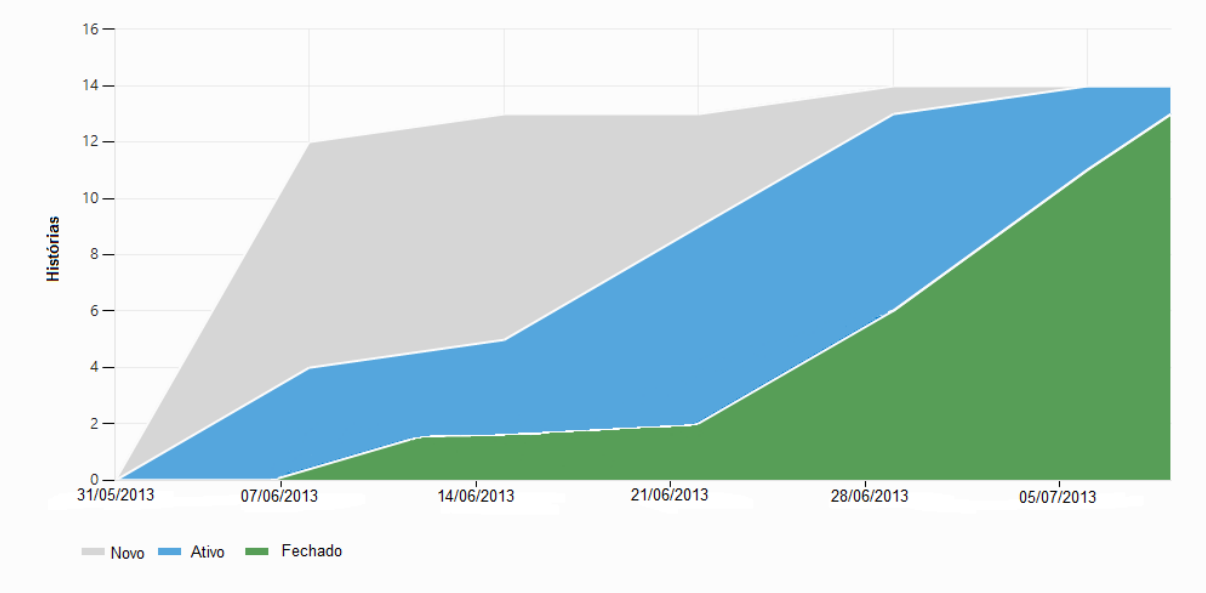


Figura 28. Fluxo acumulado das histórias

Nesse gráfico se observa a evolução na implantação das histórias através da sua mudança de estado de nova (“Acabou de ser criada”), para ativa (“Entrou em desenvolvimento”) e depois para fechada (“Funcionalidade implantada”). Segundo o XP (XP, 2009), uma história de usuário é considerada pronta para entrega quando está funcional e testada; então, para cada funcionalidade desenvolvida buscou-se criar testes, seguindo os critérios de aceitação, de forma a contemplar uma cobertura de código acima de 80% (ANEXO IV).

O sistema foi desenvolvido no período de 6 (seis) semanas como havia sido planejado e as funcionalidades com prioridade 1 (um) listadas no planejamento das releases foram implantadas como haviam sido especificadas. Na Figura 29, abaixo, o software final em funcionamento.

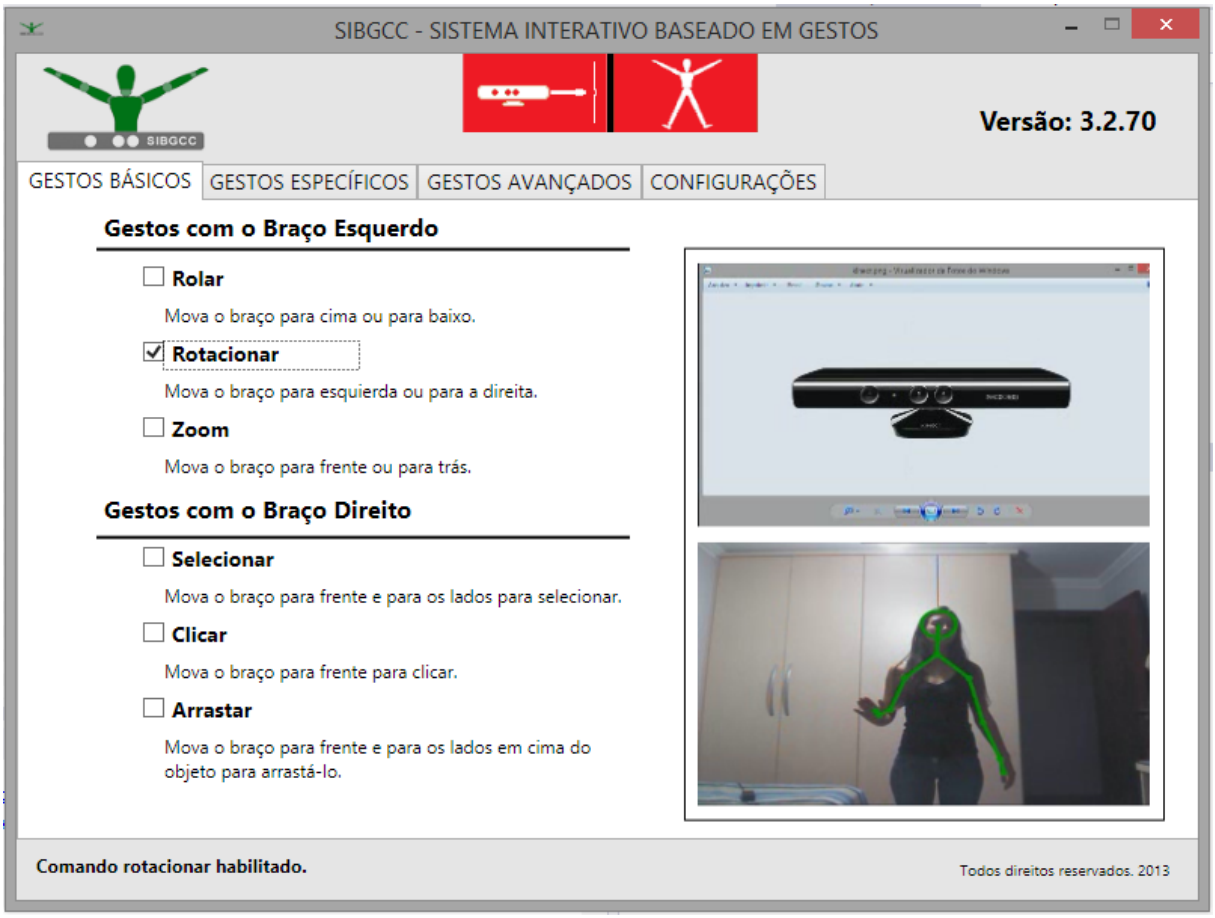


Figura 29. Sistema em funcionamento

Ao executar a aplicação, a tela inicial apresenta todas as informações relevantes para o usuário. Na parte superior da tela é apresentado o logotipo do sistema e duas imagens, contendo: uma, o status do Kinect (Conectado ou não) e a outra, o status do rastreamento do esqueleto do usuário (Rastreado ou não), além da informação da versão do sistema. Na parte central da janela são apresentados as abas com os comandos executados pelo sistema através de textos e caixas de marcar, além de dois campos de vídeo com o primeiro mostrando como o comando é executado no computador e o segundo como realizar o gesto. Na parte inferior é apresentado um campo de texto que informa todas as ações que são realizadas no sistema.

Dentre as quatro abas disponíveis, as três primeiras seguem a mesma estrutura, mas a quarta apresenta um conteúdo central diferente, para informar algumas características e propriedades do Kinect (Fig. 30).

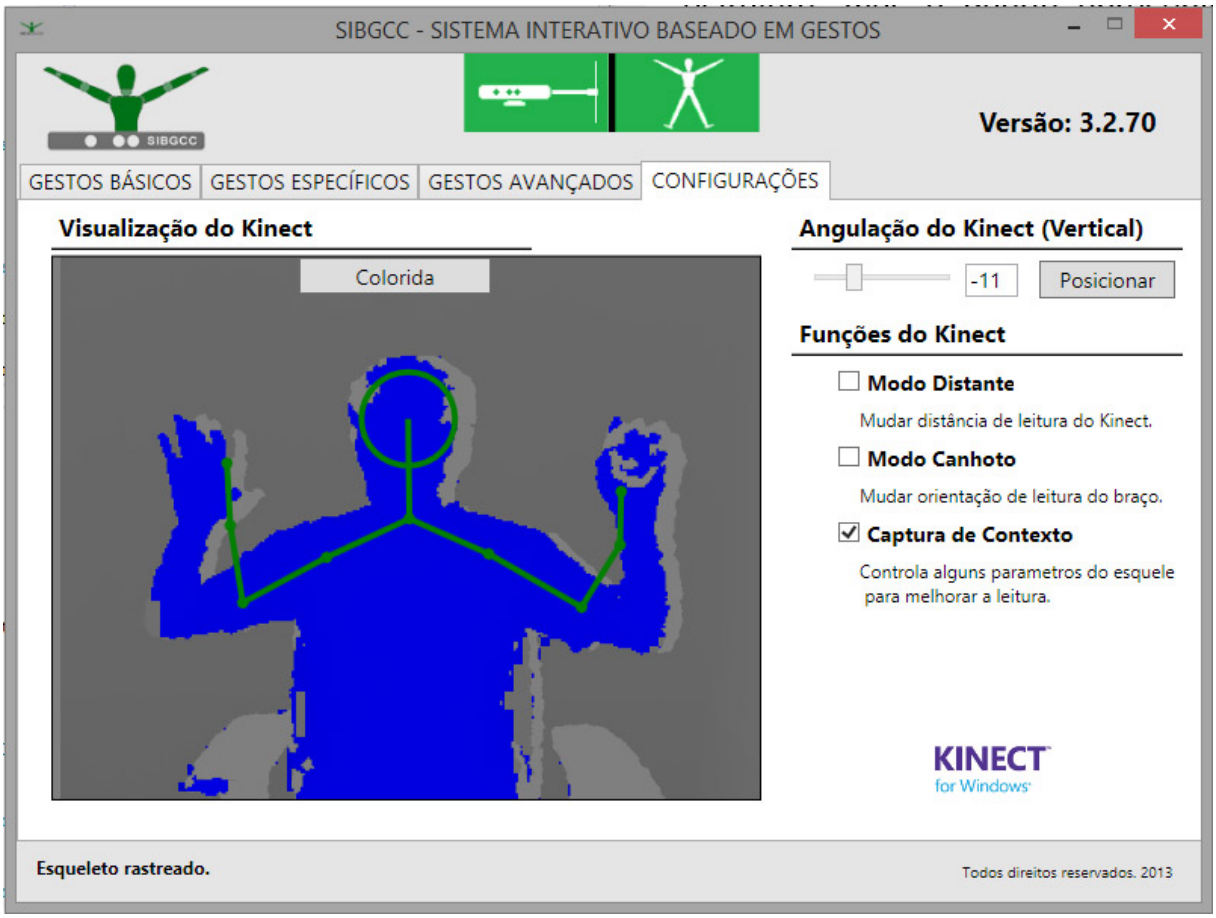


Figura 30. Tela de configuração do Kinect

Nesta aba é apresentada uma janela com a visão do Kinect, sendo possível alterar entre a visão em cores e de distância do sensor, e um painel de configuração à direita contendo algumas propriedades, como a inclinação do sensor, o modo de distância de leitura e o modo de orientação de reconhecimento do braço e a opção de controlar a captura de contexto.

Todas as funcionalidades apresentadas acima estão implantadas e testadas seguindo as especificações iniciais do sistema. Observou-se que através da utilização do sistema é possível facilitar a realização de alguns comandos no computador, como rolar e navegar com mouse. Percebeu-se alguma dificuldade na realização dos gestos em sequência e, principalmente, a dificuldade da execução de gestos pelos dois braços ao mesmo tempo. Para melhorar utilização do sistema é necessário calibrar os parâmetros de duração, tamanho e intervalo do algoritmo de reconhecimento de acordo com o ambiente externo (distância e posição do usuário) que o sistema é utilizado.

Quanto aos aspectos de usabilidade do sistema, espera-se que o nível de atraso no processamento de dados e as dificuldades citadas acima não sejam fatores que eliminem a sensação de imersão do sistema, expectativa que somente poderá ser confirmada através da avaliação da sua usabilidade. No capítulo seguinte, entre outras considerações, é apresentada uma proposta de processo de avaliação de usabilidade que poderá ser aplicada para validar o sistema futuramente.

5. CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as considerações a serem feitas em relação ao trabalho através das contribuições obtidas com a pesquisa e o desenvolvimento do sistema e uma proposta de avaliação do sistema.

Os estudos realizados mostraram que a computação perceptiva é uma área promissora para futuras interações humano-computador, uma vez que a viabilidade do desenvolvimento de interfaces baseadas em gestos é plausível, pelo avanço crescente das tecnologias disponíveis.

5.1. CONCLUSÃO

Através do resultado obtido no trabalho, é possível afirmar que o sistema desenvolvido pode ser utilizado para auxiliar na manipulação do computador pessoal. Utilizando uma interface simplificada e abordando alguns dos comandos mais utilizados no computador, o sistema se mostrou, mesmo que ainda não avaliado formalmente, usual e prático.

Desde o início do desenvolvimento buscou-se implantar um sistema modular e extensível, utilizando processos, ferramentas e técnicas da Engenharia de Software. O objetivo principal do trabalho foi desenvolver uma aplicação para um sistema operacional específico em um contexto de uso específico, o que não invalida a possibilidade de adaptá-lo para utilização em outros sistemas operacionais e outros contextos, como: auxilio na educação, manuseio de objetos tridimensionais, auxilio na reabilitação de pacientes com problemas locomotores, controle de jogos e acoplado em outros sistemas menores.

A aplicação desenvolvida também se mostra como um diferencial na forma como aborda esse tipo de tecnologia, pois comparado com outros sistemas, como os desenvolvidos nos trabalhos correlatos, apresentou teoricamente maior eficiência, facilidade de uso e aprendizado, além de maior flexibilidade e aplicação devido a estar relacionado ao sistema operacional como um todo e não a uma aplicação específica.

Na Tabela 8 é apresentada uma relação entre os critérios definidos para a avaliação dos trabalhos correlatos e as características do sistema desenvolvido no trabalho.

Tabela 8. Análise dos critérios de avaliação dos trabalhos em relação ao SIBGCC

	SIGBCC
Tecnologia perceptiva	Kinect
Sistema operacional	Windows
SDK	Oficial do Kinect
Técnica de reconhecimento	Coordenadas das mãos
Definição de gesto	Movimento separado das mãos na horizontal, vertical e Frente/trás.
Comandos no computador	Manipulação do mouse, Rolagem, Rotação e Zoom
Área de aplicação	Comandos no Sistema Operacional que podem ser utilizados em qualquer aplicação
Interface gráfica do usuário	Sim
Avaliação da usabilidade	Não
Limitações	Precisão do sensor e algoritmo de reconhecimento do gesto

Observou-se no final que um conjunto de propriedades comuns nos trabalhos correlatos se repetiram no SIGBCC. Os critérios levantados desses trabalhos serviram como base para a fundamentação teórica e tecnológica, além do próprio desenvolvimento do sistema. Desde a tecnologia perceptiva utilizada até as limitações do projeto passando pela técnica de reconhecimento, comandos no computador e área de aplicação. O sistema desenvolvido nesse trabalho procurou escolher os atributos que melhor se encaixavam no escopo do projeto.

5.2. TRABALHOS FUTUROS

Uma proposta inicial de trabalho futuro seria finalizar a implantação das histórias levantadas e avaliar a usabilidade do sistema. A implementação dessas funcionalidades seguiria o mesmo processo já utilizado, necessitando apenas de uma definição dos protótipos dos novos gestos, e a avaliação da proposta quanto ao novo paradigma de interação com sistemas computacionais seria considerada a partir do ponto de vista do usuário.

Assim, a proposta de avaliar a usabilidade do sistema poderia ser baseada nas métricas: ciência do uso, facilidade no aprendizado e satisfação no uso.

A Usabilidade, segundo a norma ISO/IEC 9126 (ISO 9126, 2004), é a "facilidade com que um usuário pode aprender a operar, preparar entradas para e interpretar as saídas de um sistema ou componente". A usabilidade de um sistema é um conceito que se refere à qualidade da interação do sistema com os usuários e depende de vários aspectos. Alguns desses fatores são:

- Facilidade de aprendizado do sistema: tempo e esforço necessários para que os usuários atinjam determinado nível de desempenho;
- Facilidade de uso: avalia o esforço físico e cognitivo do usuário durante o processo de interação, medindo a velocidade do número de erros cometidos durante a execução de determinada tarefa;
- Satisfação do usuário: avalia se o usuário gosta e sente prazer em trabalhar com o sistema;
- Flexibilidade: avalia a possibilidade de o usuário acrescentar e modificar as funções e o ambiente iniciais do sistema. Assim, esse fator mede também a capacidade do usuário de utilizar o sistema de maneira inteligente e criativa, realizando novas tarefas que não estavam previstas pelos desenvolvedores;
- Produtividade: se o uso do sistema permite ao usuário ser mais produtivo do que seria se não o utilizasse.

A especificação de requisitos de usabilidade serve como referência para avaliar se a interface apresenta um nível satisfatório de qualidade de uso em termos de usabilidade.

A importância da realização de medidas pode ser apreciada na citação dos autores Good, M. e outros:

> “Engenharia de Usabilidade é um processo através do qual as características de usabilidade são especificadas, antecipadamente e de forma quantitativa no processo de desenvolvimento, e medidas durante todo o processo. “(GOOD, M. et al, 1986)

Sem especificações mensuráveis, não é possível determinar metas de usabilidade e dizer se o sistema final alcança essas metas. Então, são sugeridas no tópico a seguir algumas medidas para auxiliar na determinação dessas metas.

5.2.1. Medidas da qualidade em uso

A norma da ISO 9126-4 (ISO 9126-4, 2004) apresenta métricas baseadas em normas de qualidade, engenharia de software e requisito de usabilidade para medir a qualidade de um produto de software de qualquer aplicação.

Essas métricas são colocadas através de um conjunto de questões que devem ser observadas durante a avaliação de usabilidade. Esse conjunto é amplo e genérico o bastante para possibilitar que seja adaptado a qualquer aplicação, inclusive aplicações de interação.

O foco deste trabalho foi desenvolver um sistema com qualidade em uso. Então, é interessante estabelecer o contexto de uso, ou seja, definir o perfil dos usuários, identificar as tarefas realizadas, os equipamentos envolvidos, ter uma visão do ambiente físico e social e, também, estabelecer as métricas de qualidade adequadas para a avaliação.

Pode-se extrair desta norma um modelo de qualidade para diferenciar qualidade interna, externa e de uso, como ser visto na Figura 31.

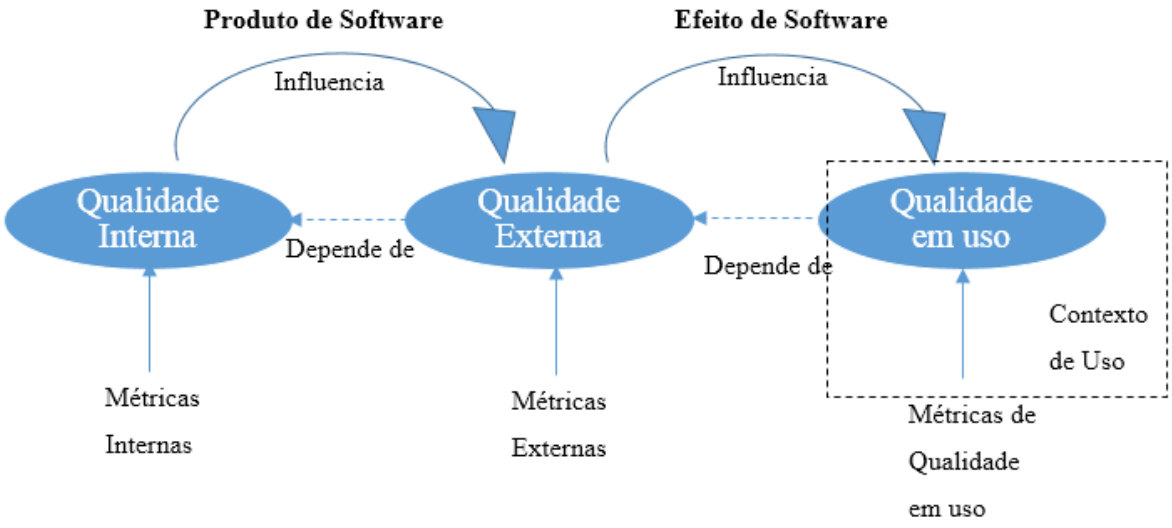


Figura 31. Relação entre métricas – Baseada na ISO 9126-4

As métricas de qualidade de uso podem ser utilizadas no futuro para verificar o atendimento das necessidades dos usuários para atingir objetivos específicos, como eficiência, produtividade, segurança e satisfação em um contexto de uso específico.

REFERÊNCIAS BIBLIOGRÁFICAS

AGIL, Manifesto. 2013. Manifesto Ágil [Online] 2013 http://agilemanifesto.org/.

ALVARENGA, Matheus L., Correa, Diogo S. e Osório, Fernando S. 2012. Redes Neurais Artificiais aplicadas no Reconhecimento de Gestos usando o Kinect. Universidade de São Paulo. 2012. p. 10.

ASUS. 2013 Xtion PRO developer solution to make motion-sensing applications and games [Online] 2013. http://www.asus.com/Multimedia/Xtion_PRO_LIVE/.

AZIMI, Mehran. 2013. Skeletal Joint Smoothing White Paper [Online] 2013 http://msdn.microsoft.com/en-us/library/jj131429.aspx

BUILDWIKI. 2013. Conceito de Build [Online] 2013 https://pt.wikipedia.org/wiki/Build.

CADOZ, Claude. 1994. Gesto um Canal de Comunicação Homem Máquina: A comunicação Instrumental e TSI. Tecnologia em Ciência da Informação.

CATUHE, David. 2012. Programming with the Kinect for Windows Sofware Develpmento Kit. Redmond, Washington. Microsoft Press, 2012. p. 210. ISBN: 978-0-7356-6681-8.

CODEPLEX. 2013. Coding4Fun Kinect Toolkit. Project Hosting for Open Source Software [Online] 2013 http://c4fkinect.codeplex.com/

FOLWER, Martin. 2004 Refatoração – Aperfeiçoando o Projeto de Código Existente. Bookman 2004. p.774. ISBN 0-201-48567-2.

Gamma, Erich. 2000. Padrões de Projeto: Soluções reutilizáveis de software orientado à objetos. Porto Alegre: Bookman, 2000. p. 370. ISBN 978-85-7307-610-3.

GARBIN, Sander M. 2010. Estudo da evolução das interfaces homem-computador. Universidade de São Paulo, São Carlos.

GHIROTTI, Silvia E. e MORIMOTO, Carlos H. 2009. Um sistema de interação baseado em gestos manuais tridimensionais para ambientes virtuais. Departamento de Ciência da Computação – IME/USP.

GILB, Tom. 1981. Design por Objetivos, Manuscritos não publicados.

GOOD, M., ET. Al.1986. User-derived impact analysis as a tool for usability engineering. Proceedings of Human Factors in Computing Systems.

ISO 9126-4. 2004. International Oraganization for Standardization. Software engineering - Product quality - Part 4: Quality in use metrics. Genebra.

KEDAVRA, Avada. 2011. Official Kinect SDK vs. Open-source alternatives [Online] 2011. http://stackoverflow.com/questions/7706448/official-kinect-sdk-vs-open-source-alternatives.

KIM, Kyungnam, Chalidabhongse, Thanarat, Harwood, David, Davis, Larry. 2005. Real-time foreground–background segmentation using codebook model. University of Maryland, Mongkut's Institute of Technology. 2005. p. 14.

KINECT, 2013. Arquitetura. do Kinect [Online] 2013. http://msdn.microsoft.com/en-us/library/jj131033.aspx.

KINECT UI. 2013. Human Interface Guidelines. Microsoft Corporation. 2013. 135p.

KINECT MODOS. 2013. Kinect Modos de Operação. [Online] 2013. http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx.

KRUCHTEN, Philippe. 1995. Architectural Blueprints—The "4+1" View Model of Software Architecture. Rational Software Corp. p. 15.

LEAP MOTION. 2013. Site oficial do dispositivo. [Online] 2013. https://www.leapmotion.com/product.

MATSUMURA, Keila K., SONNINO, Roberto. 2011. FUSION4d - Interface Natural e Imersiva para Manipulação de Objetos 3D. Universidade de São Paulo. 2011. 109p.

MEDEIROS, Anna Carolina S. 2012. Interação Natural baseada em Gestos como Interface de Controle para Modelos Tridimensionais. Universidade Federal da Paraíba, 2012. 63 p.

MEDEIROS, Marco A. 1999. ISO 9241: Uma Proposta de Utilização da Norma para Avaliação do Grau de Satisfação de Usuários de Software. Universidade Federal de Santa Catarina. 1999. 159p.

MICROSOFT, Kinect for Windows. 2013, Site oficial do dispositivo. [Online] 2013. http://www.microsoft.com/en-us/kinectforwindows/Develop/New.aspx.

MSDN. 2013 SendKeys Method [Online] 2013. http://msdn.microsoft.com/en-us/library/office/aa202943(v=office.10).aspx.

MSDN C#, 2013. Guia de Programação C# [Online] 2013. http://msdn.microsoft.com/pt-BR/library/67ef8sbd(v=VS.90).aspx

NETO, Oscar N. de Souza, 2004. Análise Comparativa das Metodologias de Desenvolvimento de Softwares Tradicionais e Ágeis. Universidade da Amazônia, 2004. 74p.

NORMAM, Donald. 1986. A. User Centered System Design: New Perspectives on Human-Computer Interaction.

PREECE, Jennifer, ROGERS, Yvonne, SHARP, Helen. Design de Interação. Além da interação home-computador. Editora Bookman. Porto Alegre – RS, 2005. 348 p.

PROCESS, AGIL. 2013. Agile Software Development: A gentle introduction [Online] 2013. http://www.agile-process.org/.

RIBEIRO, H. L. 2006. Reconhecimento de gestos usando segmentação de imagens dinâmicas de mãos baseada no modelo de mistura de Gaussianas e cor de pele. 2006. 144p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, SP.

RUP. 2013. Conceitos: Arquitetura de Software. [Online] 2013. http://www.wthreex.com/rup/process/workflow/ana_desi/co_swarch.htm.

S. MITRA E T. ACHARYA. 2007. Reconhecimento de Gestos: Uma Pesquisa. Sistemas, Homem, e Cibernético, Transcrição IEEE no, 37(3):311-324.

SBC. 2013. Sociedade Brasileira de Computação. Interação Humano Computador. [Online] 2013. http://www.sbc.org.br/index.php?option=com_content&view=category&layout=blog&id=45&Itemid=66.

SWEBOOK. 2013. Chapter 1 Introduction To The Guide [Online] 2013 http://www.computer.org/portal/web/swebok/html/ch1.

SILVA, Bárbara Dariano. 2008. Avaliação de usabilidade situada para aperfeiçoamento de equipamentos médicos / B.D. Silva. -- São Paulo, 2008. p. 89.

SILVEIRA, Marcus Almeida. Técnica de Navegação em Documentos Utilizando Microsoft Kinect. Universidade Federal do Rio Grande do Sul. 2011. p. 36.

SOMMERVILLE, Ivan. 2011. Engenharia de Software, 8ª edição; tradução: Selma Shin Shimizu, Reginaldo Arakaki, Edílson de Andrade Barbosa; Pearson Addison-Wesley p.

TFS. 2013. Official site Team Foundation Service. [Online] 2013. http://tfs.visualstudio.com/.

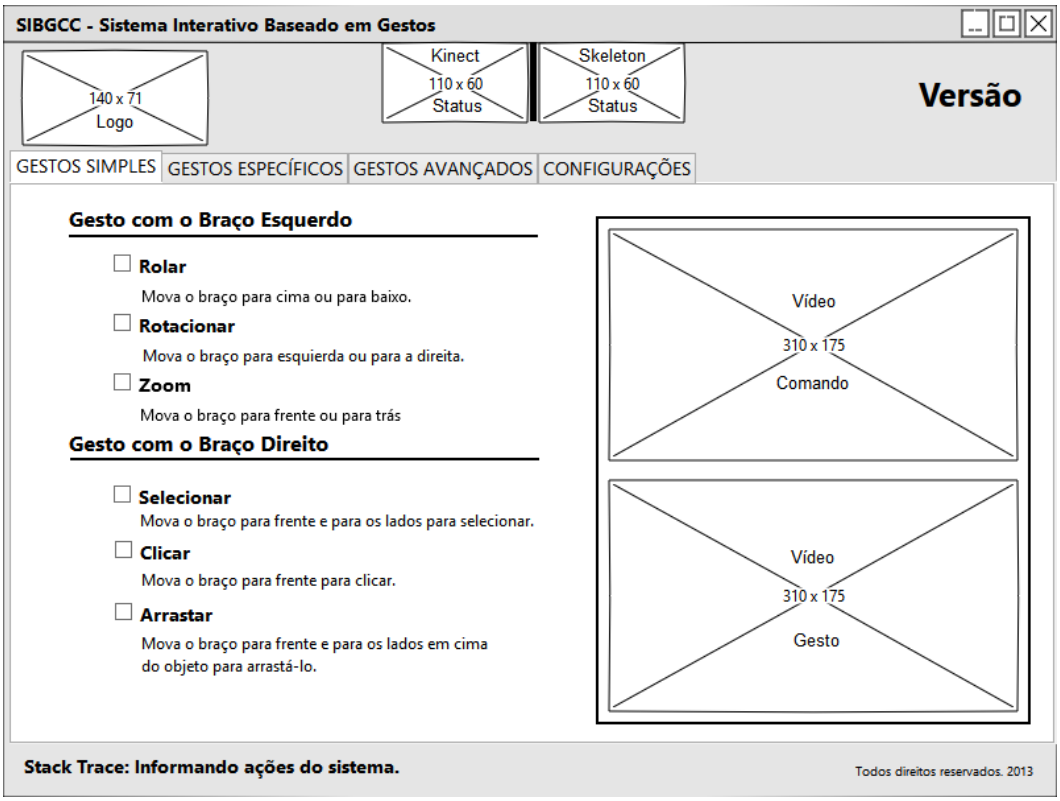
TOGORES, Thiago Andrade. 2011. Vitruvius - Um Reconhecedor de Gestos para o Kinect. Universidade de São Paulo. 2011. p. 42.

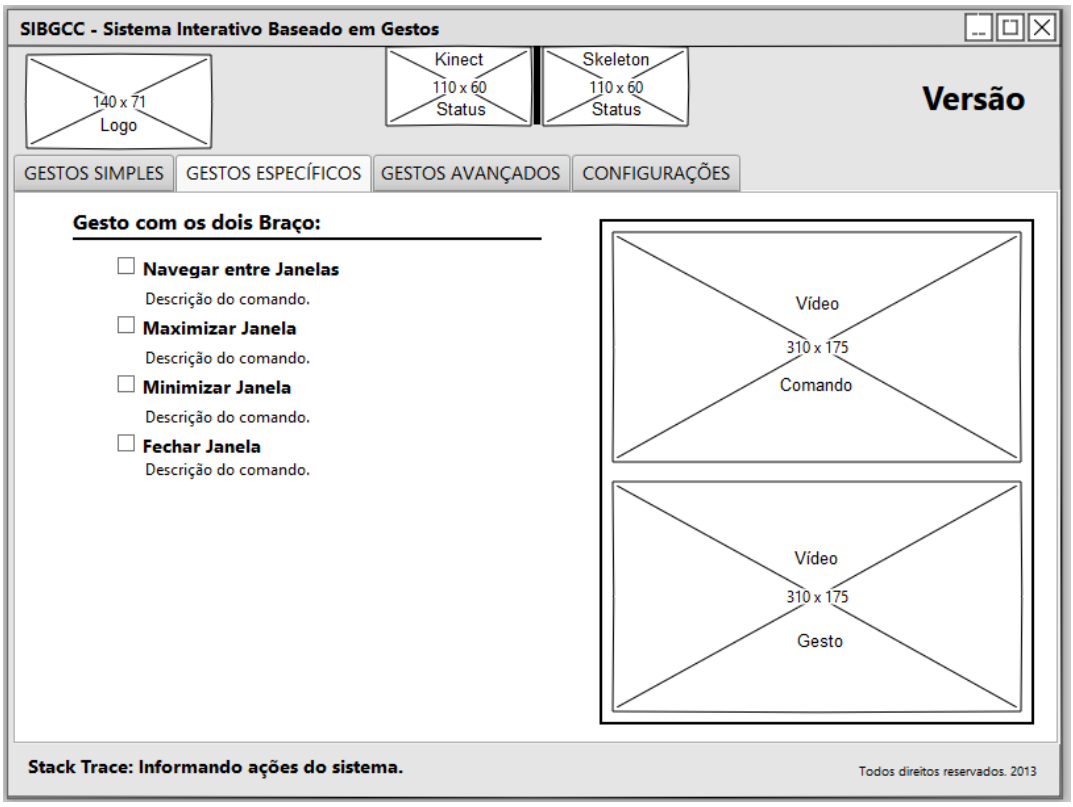
WIKI KINECT. 2013. Dispositivo sensorial Kinect. [Online] 2013. http://en.wikipedia.org/wiki/Kinect.

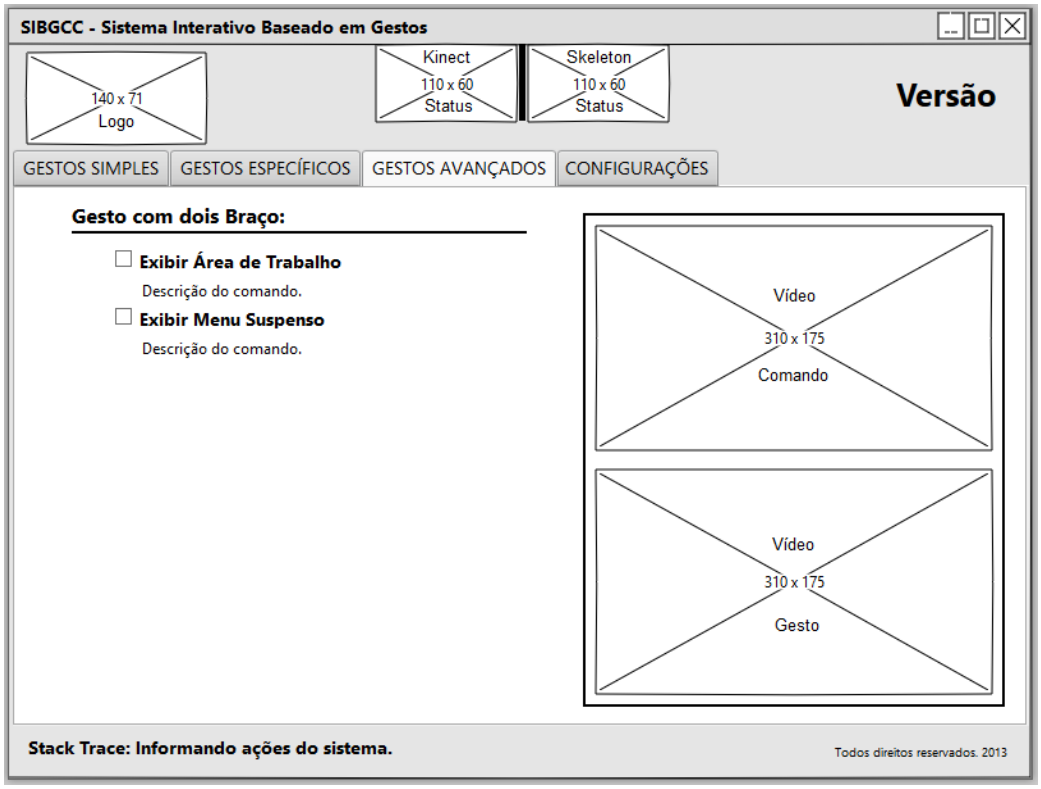
XP. 2009. Official site Extreme Programming: A gentle introduction. [Online] 2009. http://www.extremeprogramming.org.

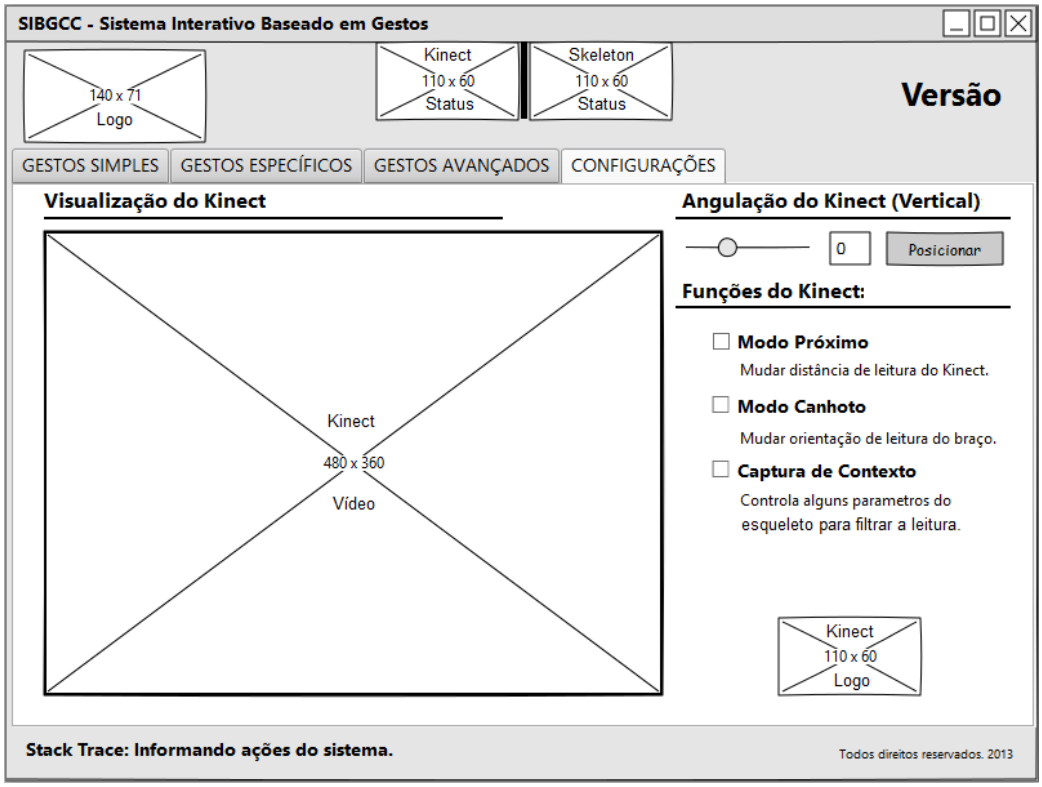
APÊNDICE

APÊNDICE I – Protótipos de Interface






SIBGCC - Sistema Interativo Baseado em Gestos
140 x 71
Logo
Kinect
110 x 60
Status
Skeleton
110 x 60
Status
Versão
GESTOS SIMPLES
GESTOS ESPECÍFICOS
GESTOS AVANÇADOS
CONFIGURAÇÕES
Gesto com dois Braço:
Exibir Área de Trabalho
Descrição do comando.
Exibir Menu Suspenso
Descrição do comando.
Vídeo
310 x 175
Comando
Vídeo
310 x 175
Gesto
Stack Trace: Informando ações do sistema.
Todos direitos reservados. 2013


SIBGCC - Sistema Interativo Baseado em Gestos
140 x 71
Logo
Kinect
110 x 60
Status
Skeleton
110 x 60
Status
Versão
GESTOS SIMPLES
GESTOS ESPECÍFICOS
GESTOS AVANÇADOS
CONFIGURAÇÕES
Visualização do Kinect
Kinect
480 x 360
Vídeo
Angulação do Kinect (Vertical)
0
Posicionar
Funções do Kinect:
Modo Próximo
Mudar distância de leitura do Kinect.
Modo Canhoto
Mudar orientação de leitura do braço.
Captura de Contexto
Controla alguns parametros do esqueleto para filtrar a leitura.
Kinect
110 x 60
Logo
Stack Trace: Informando ações do sistema.
Todos direitos reservados. 2013

ANEXOS

ANEXO I – Método para verificar a velocidade relativa do *skeleton* em relação a velocidade atual

O método verifica se a velocidade do *skeleton* (mudança de posição dos pontos em relação ao tempo) é maior que um valor pré-definido.

```
…
    public bool IsStableRelativeToCurrentSpeed(int trackingID){
        List<ContextPoint> currentPoints = points[trackingID];

        if (currentPoints.Count < 2)
            return false;

        Vector3 previousPosition = currentPoints[currentPoints.Count - 2].Position;
        Vector3 currentPosition = currentPoints[currentPoints.Count - 1].Position;
        DateTime previousTime = currentPoints[currentPoints.Count - 2].Time;
        DateTime currentTime = currentPoints[currentPoints.Count - 1].Time;

       var currentSpeed = (currentPosition - previousPosition).Length / ((currentTime - previous-
       Time).TotalMilliseconds);

        if (currentSpeed > Threshold)
            return false;

        return true;
    }

…
```

ANEXO II – Método para verificar a orientação dos ombros

O método verifica através dos valores dos *Joints* ombro direito e ombro esquerdo se o modulo da subtração desses dois valores é maior que um valor limite pré-determinado.

…

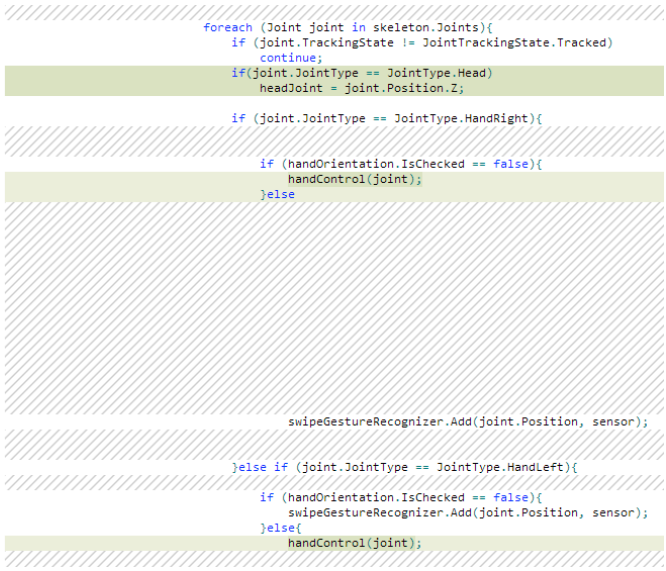
```
public bool IsShouldersTowardsSensor(Skeleton skeleton){

    var leftShoulderPosition = Vector3.ToVector3(skeleton.Joints.Where(j => j.JointType ==
    JointType.ShoulderLeft).First().Position);
    var rightShoulderPosition = Vector3.ToVector3(skeleton.Joints.Where(j => j.JointType ==
    JointType.ShoulderRight).First().Position);

   var leftDistance = leftShoulderPosition.Z;
   var rightDistance = rightShoulderPosition.Z;

   if (Math.Abs(leftDistance - rightDistance) > Threshold)
      return false;

   return true;
}
```

…

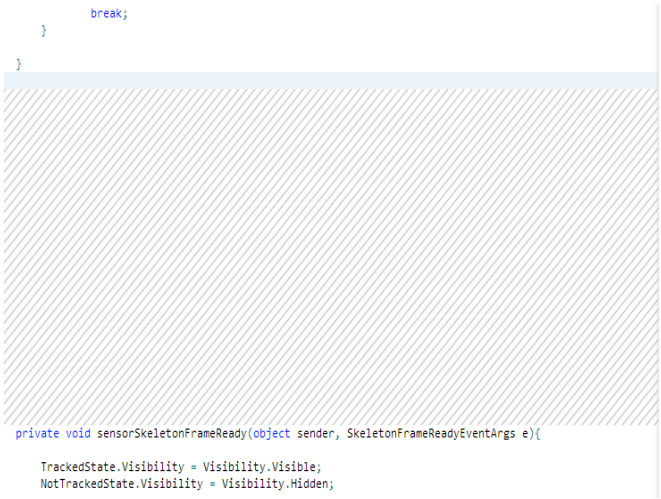
ANEXO III – Exemplo de aplicação da técnica de refatoração extração de método

As imagens foram retiradas do TFS e representam do lado esquerdo, o código antes da refatoração, e do direito, o código depois da refatoração.

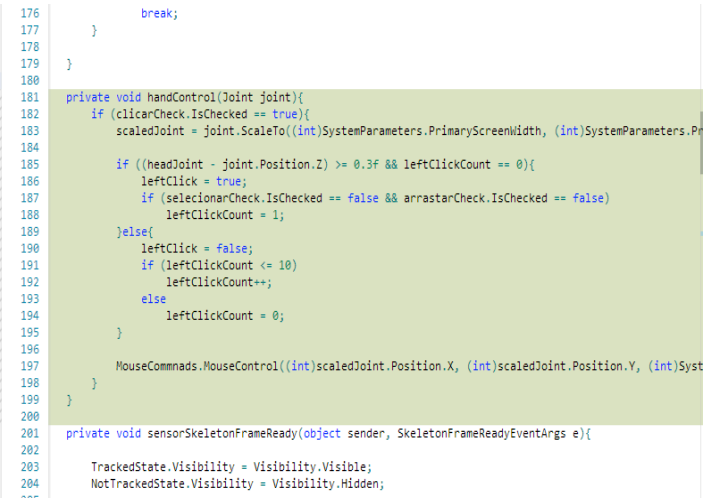
```
foreach (Joint joint in skeleton.Joints){
    if (joint.TrackingState != JointTrackingState.Tracked)
        continue;


    if (joint.JointType == JointType.HandRight){
        circleGestureRecognizer.Add(joint.Position, sensor);

        if (handOrientation.IsChecked == false){

            if (clicarCheck.IsChecked == true){
                scaledJoint = joint.ScaleTo((int)SystemParameters.PrimaryScreenWid

                cursorX = (int)scaledJoint.Position.X;
                cursorY = (int)scaledJoint.Position.Y;

                if (Math.Abs(joint.Position.Z - previousDepth) > 0.05f)
                    leftClick = true;
                else
                    leftClick = false;

                NativeMethods.SendMouseInput(cursorX, cursorY, (int)SystemParamete
                previousDepth = joint.Position.Z;
            }
        }else{
            swipeGestureRecognizer.Add(joint.Position, sensor);
        }

    }else if (joint.JointType == JointType.HandLeft){

        if (handOrientation.IsChecked == false){
            swipeGestureRecognizer.Add(joint.Position, sensor);
        }else{
            if (clicarCheck.IsChecked == true){
                scaledJoint = joint.ScaleTo((int)SystemParameters.PrimaryScreenWid
```



```
253                     foreach (Joint joint in skeleton.Joints){
254                         if (joint.TrackingState != JointTrackingState.Tracked)
255                             continue;
256                         if(joint.JointType == JointType.Head)
257                             headJoint = joint.Position.Z;
258
259                         if (joint.JointType == JointType.HandRight){

260                             if (handOrientation.IsChecked == false){
261                                 handControl(joint);
262                             }else
263                                 swipeGestureRecognizer.Add(joint.Position, sensor);

264                         }else if (joint.JointType == JointType.HandLeft){

265                             if (handOrientation.IsChecked == false){
266                                 swipeGestureRecognizer.Add(joint.Position, sensor);
267                             }else{
268                                 handControl(joint);
```



```
            break;
        }

    }


private void sensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e){

    TrackedState.Visibility = Visibility.Visible;
    NotTrackedState.Visibility = Visibility.Hidden;
```



```
176             break;
177         }
178
179     }
180
181     private void handControl(Joint joint){
182         if (clicarCheck.IsChecked == true){
183             scaledJoint = joint.ScaleTo((int)SystemParameters.PrimaryScreenWidth, (int)SystemParameters.Pr
184
185             if ((headJoint - joint.Position.Z) >= 0.3f && leftClickCount == 0){
186                 leftClick = true;
187                 if (selecionarCheck.IsChecked == false && arrastarCheck.IsChecked == false)
188                     leftClickCount = 1;
189             }else{
190                 leftClick = false;
191                 if (leftClickCount <= 10)
192                     leftClickCount++;
193                 else
194                     leftClickCount = 0;
195             }
196
197             MouseCommnads.MouseControl((int)scaledJoint.Position.X, (int)scaledJoint.Position.Y, (int)Syst
198         }
199     }
200
201     private void sensorSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e){
202
203         TrackedState.Visibility = Visibility.Visible;
204         NotTrackedState.Visibility = Visibility.Hidden;
```

ANEXO VI – Cobertura de código segundo a IDE

A imagem apresenta a cobertura de código após a execução de todos os testes.