UNIVERSIDADE SAGRADO CORAÇÃO

MICHEL HIDEO GOTO HIRATA

DESENVOLVIMENTO DE UM PROTÓTIPO QUE UTILIZA O EQUIPAMENTO KINECT

BAURU
2015

MICHEL HIDEO GOTO HIRATA

DESENVOLVIMENTO DE UM PROTÓTIPO QUE UTILIZA O EQUIPAMENTO KINECT

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas como parte de requisitos para obtenção do título bacharel em Ciência da Computação sob orientação da Prof.º Esp. André Luiz Ferraz Castro

BAURU
2015

Hirata, Michel Hideo Goto

H6683d

Desenvolvimento de um protótipo que utiliza o equipamento Kinect / Michel Hideo Goto Hirata. -- 2015.

86f. : il.

Orientador: Prof. Esp. André Luiz Ferraz Castro.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Exergame. 2. Kinect. 3. Movimento. 4. Software. I. Castro, André Luiz Ferraz. II. Título.

MICHEL HIDEO GOTO HIRATA

DESENVOLVIMENTO DE UM PROTÓTIPO QUE UTILIZA O EQUIPAMENTO KINECT

Trabalho de Conclusão de Curso apresentado ao centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Esp. André Luiz Ferraz Castro

BANCA EXAMINADORA:

_____

Prof. Esp. André Luiz Ferraz Castro
Universidade Sagrado Coração

_____

Prof. Dr. Elvio Gilberto da Silva
Universidade Sagrado Coração

_____

Prof. Me. Patrick Pedreira Silva
Universidade Sagrado Coração

Bauru, 10 de dezembro de 2015

Dedico este trabalho para meu ídolo, meu objetivo, meu tudo ao meus pais que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida. Amo muito vocês.

AGRADECIMENTOS

Agradeço aos meus pais pelo carinho, paciência, conselho e apoio durante a faculdade, pois sem eles nada disso seria possível. Vocês sempre serão meu objetivo de vida. Obrigado por me dar essa oportunidade, de não deixar faltar nada, sei que não foi fácil enfrentando todas dificuldades. Agora é a minha vez de retribuir. Amo muito vocês.

Agradeço aos meus irmãos Naomi, Lucas e sobrinhos, que sempre me apoiou com muito amor mesmo distante.

As (aos) minhas (meus) tias (tios), meus primos, minhas (meus) avós (avôs), que me ajudaram e continua ajudando desde que cheguei do Japão incentivando a importância do estudo e sempre cuidando de mim.

Agradeço aos meus amigos Gabriel Takachi, Rafael Nunes, Alexandre Suriano, Thiago Klein, companheiros de trabalhos e na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida.

A mãe do meu filho Daniela Kumagai, que me apoiou, esteve ao meu lado nos momentos difíceis sempre cuidando de mim, motivando o término da faculdade e ao meu filho Arthur que veio para iluminar a minha vida com um amor imenso.

Agradeço o meu orientador, Prof. Esp. André Luiz Ferraz Castro pelo empenho dedicado, apoio, várias ideias e por ser compreensivo com minhas dificuldades.

Agradeço os professores que me ensinaram e auxiliaram durante a faculdade construindo a base para me tornar um profissional na área de T.I. Me. Patrick Pedreira Silva e Dr. Elvio Gilberto da Silva.

Meus sinceros agradecimentos a todos aqueles que de alguma forma doaram um pouco de si para que a conclusão deste trabalho se tornasse possível.

"O sucesso é ir de fracasso em fracasso sem perder entusiasmo." (Winston Churchill)

RESUMO

O mundo do game revolucionou com o surgimento do Exergame, uma tecnologia com baixo custo que utiliza os movimentos corporais como dados de entrada para que o usuário possa interagir com o jogo eletrônico. Mas essa tecnologia não ficou só no game, como também para diversas áreas no mundo inteiro; educação física, fisioterapia, medicina, segurança, educação e outros.

Neste trabalho foi feito um protótipo de um software que consiga verificar se o movimento do usuário está correto utilizando o console Kinect da Microsoft. Entre vários dispositivos que conseguem captar o movimento, o Kinect foi escolhido devido à qualidade de captura dos dados e a sua grande comunidade de utilizadores onde facilmente se pode encontrar suporte para eventuais problemas.

Para verificar se o Kinect consegue reconhecer o movimento do usuário, foi comparado o movimento feito pelo usuário com o movimento previamente cadastrado via código de programação. Através dos testes feitos com o sistema desenvolvido, o console Kinect mostrou que tem capacidade para reconhecimento de movimento para ajudar os profissionais em diversas área. Para o desenvolvimento da ferramenta, foi utilizado Visual Studio 2015 com a linguagem C#, SDK do Kinect, OpenCV.

Palavras-chaves: Exergame; Kinect; Movimento; Software

ABSTRACT

The gaming World revolutionized with the emerge of "Exergame", a low cost technology using body movements as data input, so the user can interact with the game. But the technology didn't just stop at games, it spread to various areas worldwide; physical education, physical therapy, medicine, security, education and many others.

Based on this paper, will be developed a software using Microsoft's Kinect console to verify if a user's movement is correct. In a wide range of devices that can capture movement. Kinect was chosen due to it's quality to capture data, and its large user community, where you can easily find support for eventual problems.

To verify if the Kinect can recognize the user´s movement, the move made by the user is compared to a movement previously registered by programming code. Through the tests made with the developed system, the Kinect console showed that is able to recognize movement to help professionals in varius area. For the development of the tool was used Visual Studio 2015 with c# language, SDK Kinect, Open CV.

Keywords: Exergame; Kinect; Movement; Software

LISTA DE ILUSTRAÇÕES

Figura 1- Rede de conexão sináptica 18
Figura 2 - Interação Homem Máquina 20
Figura 3 - Tipos de Interfaces 21
Figura 4 - Reabilitação com Wii 26
Figura 5 - Console Kinect 28
Figura 6 - Raios de visão do Kinect 30
Figura 7- Fluxograma de funcionamento 31
Figura 8 - Tabela de configuração da câmera RGB 32
Figura 9- Imagem YUV 32
Figura 10- Comparação entre formatos da câmera RGB 33
Figura 11- Luz infravermelho do Kinect 33
Figura 12- Configuração do sensor de profundidade 34
Figura 13- Esqueleto Kinect 36
Figura 14 - Ciclo de vida de um software 39
Figura 15- Reabilitação com Kinect 42
Figura 16 - Cabo de conexão do Kinect for Xbox 360 53
Figura 17 - Tela inicial do sistema 54
Figura 18 - Sensor de profundidade 57
Figura 19- Hierarquia das articulações 62
Figura 20- Esqueleto completo 63
Figura 21- Movimentos 64
Figura 22- Fórmula do produto escalar 66
Figura 23 - Produto dos vetores 67
Figura 24- Produto do modulo dos vetores 67
Figura 25 - Quadro chave do movimento 69
Figura 26- Rastreamento e identificação de movimento 70

Figura 27– Diagrama de casos de uso 71

LISTA DE ABREVIATURA E SIGLAS

.NET:	Framework de Desenvolvimento Microsoft
AVC:	Acidente Vascular Cerebral
C:	Linguagem de Programação
C++:	Linhagem de Programação
CLI:	Command Line Interface
EXG:	Exergame
GUI:	Graphic User Interfaces
IHC:	Interface Human Computer
JAVA:	Linguagem de Programação
NUI:	Natural User Interfaces
RGB:	Red Green Blue
SDK:	Software Development Kit
USB:	Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO 13
1.2 OBJETIVO 16
1.2.1 Objetivo Geral 16
1.2.2 Objetivos Específicos 16
2 REABILITAÇÃO FISIOTERAPÊUTICA 17
2.1 EXERCÍCIOS TERAPÊUTICOS 17
2.2 REABILITAÇÃO NEUROLÓGICA 17
3 INTERAÇÃO HOMEM – MÁQUINA 20
3.1 INTERFACE DE LINHA DE COMANDO (CLI - COMMAND-LINE INTERFACE) 21
3.2 INTERFACE GRÁFICA DO USUÁRIO *(*GUI – GRAFIC USER INTERFACE) 21
3.3 INTERFACE NATURAL DO USUÁRIO *(NUI – NATURAL USER INTERFACE)* 22
4 SIMULAÇÃO DE AMBIENTE REAL 23
4.1 REALIDADE VIRTUAL 23
4.2 REALIDADE AUMENTADA 23
5 HISTÓRIA DO EXERGAME 24
5.1 ATARI PUFFER 24
5.2 DANCE DANCE REVOLUTION 24
5.3 NINTENDO WII 25
5.4 MICROSOFT KINECT 26
6 FUNCIONALIDADE DO KINECT 28
6.1 ACELERÔMETRO 28
6.2 Eixo Motorizado 29
6.3 Fluxo do Kinect 30
6.3.1 Fluxo da Cor 31
6.3.2 Fluxo de Profundidade 34
6.3.3 Fluxo de Esqueleto 35
6.4 Microfone Multi-Matriz 36
6.5 SOFTWARE 37
7 ENGENHARIA DE SOFTWARE 38
7.1 PROCESSO DE SOFTWARE 38
7.1.1 Planejamento 39
7.1.2 Análise e Especificação de Requisitos 40

7.1.3 Projeto **40**
7.1.4 Implementação **40**
7.1.5 Testes **40**
7.1.6 Entrega a Implementação **41**
7.1.7 Operação **41**
7.1.8 Manutenção **41**
8 TRATAMENTO DE REABILITAÇÃO COM KINECT **42**
8.1 CLÍNICA DE FISIOTERAPIA DA UNIVERSIDADE DE SÃO PAULO (UNICID) 43
8.2 ABP (ASSOCIAÇÃO BRASIL PARKINSON) – SÃO PAULO 43
8.2.1 Jintronix-Jrs **43**
8.2.2 Virtualware **44**
8.2.3 Virtualrehab **44**
8.3 CENTRO DE REABILITAÇÃO NO EXTERIOR 44
8.3.1 The banner good samaritan reabilitation center (Phoenix -EUA) **44**
8.3.2 Northem Arizona University (Phoenix -EUA) **45**
9 KINECT HACKING - KINECT NA SOCIEDADE **46**
9.1 EDUCAÇÃO 46
9.2 ARQUEOLOGIA 46
9.3 REABILITAÇÃO 47
9.4 VITRINE VIRTUAL 47
9.5 TRADUTOR DE SINAIS DE SURDOS 47
9.6 CÃO-GUIA ROBÔ 48
9.7 MEDICINA 48
9.8 CARRINHO DE MERCADO 48
9.9 JOGO ADULTO 48
10 METODOLOGIA 50
10.1 FERRAMENTAS UTILIZADOS 51
10.1.1 VISUAL STUDIO **51**
10.1.2 LINGUAGEM DE PROGRAMAÇÃO C# **51**
10.1.3 .NET FRAMEWORK **51**
10.1.4 UML (UNFIED MODELING LANGUAGE) **52**
11 DESENVOLVIMENTO **53**
11.1 Tela Inicial 53

11.2 Iniciando o sistema 55
11.3 Escala Cinza 57
11.4 CAPTURANDO ESQUELETO DO USUÁRIO 58
11.5 Cadastrando movimento 63
11.5.1 Movimento T 64
11.5.2 Movimento Mão Esquerda 45 graus 65
11.5.3 Movimento Aceno 68
11.6 Funcionamento do Sistema 71
12 CONCLUSÃO 73
13 REFERÊNCIAS 74
14 APENDICE A - CÓDIGOS FONTE 81

1 INTRODUÇÃO

Com o incremento dos componentes gráficos nos anos 80 surgiram as primeiras Graphic User Interfaces (GUI), que são interfaces gráficas que remontavam metáforas do cotidiano do usuário, criando uma interação mais amigável. A partir daí as interfaces gráficas se desenvolveram e tornaram-se a principal ponte de interação entre humano e computador. (MULING, 2012).

Nessa época os videogames eram feitos com um hardware simples, dispondo de processamento de apenas 8*bits*, alguns kilobytes de memória, além de interfaces e joysticks limitados. Com o avanço da tecnologia, o setor de videogame está cada vez mais inovando tanto no gráfico quanto de hardware, ultrapassando as fronteiras estabelecidas do setor de games beneficiando diferentes áreas por meio de atividades de pesquisa e desenvolvimento. (MACHADO, 2013).

Uma possibilidade atual para propiciar jogos à população é a utilização de jogos virtuais que podem ser definidos como uma atividade lúdica com as mesmas características dos jogos convencionais, mas praticados em ambiente virtual. Um jogo que interage e se mistura com exercícios físicos tem chamado a atenção de profissionais de diversas áreas são denominados de Serious Games ou Exergame. (HUIZINGA, 2003; MACHADO, 2011; PARIZOCOVA, CHIN, 2003).

Os jogos que se misturam ao exercício são denominados Exergame (EXG) sendo uma nova tecnologia com baixo custo que utiliza os movimentos corporais como dados de entrada para que o usuário possa interagir com o jogo eletrônico que vem sendo usada por Profissionais de Educação Física, Fisioterapeutas e até Médicos como auxílio na prática da atividade física e reabilitação, e cria um ambiente de aprendizagem que proporciona gasto calórico e entretenimento, o que não acontece com os games tradicionais que são associados à obesidade e sedentarismo. (PARIZKOVA; CHIN, 2003).

Em novembro de 2010, a tecnologia deu um grande avanço com o lançamento do novo console da Microsoft, o Kinect do jogo Xbox360. Primeiro game que pode jogar sem o controle, ou seja, o próprio corpo do jogador é o controle. Devido ao seu enorme sucesso entre pesquisadores e admiradores, foi reconhecida a importância do dispositivo além do mundo dos jogos. Logo, a *Microsoft* anunciou o desenvolvimento de aplicações para a área de assistência à saúde, reabilitação e

educação. (MICROSOFT, 2015; XBOX360, 2015*;* PIRES, 2012*).*

Muitas pessoas precisam passar por algum tipo de reabilitação como consequência de diversas doenças ou de algum acidente. Essa reabilitação consiste de diversas sessões de fisioterapia com movimentos específicos, repetitivos para a terapia surte algum efeito e isto pode se tornar um fator de desmotivação ao paciente. O uso do Kinect junto a esses tratamentos aumenta a motivação do paciente e o índice de acerto dos movimentos feitos por ele. (CHAG; CHEN; HUANG; 2011).

A Jornalista Júlia Brasil relata: Hélio Martins de Paula, gerente da casa de apoio do Hospital Mário Penna diz que o videogame ajuda os pacientes a passar pela fase e tratamento de forma menos dolorosa tanto os adultos quanto as crianças que estão temporariamente hospedadas na casa para tratamento de câncer aderiram ao brinquedo doado pela Microsoft. (BRASIL, 2012).

O uso do conceito de Reabilitação Virtual através de tecnologias como Kinect em tratamentos fisioterapêuticos introduz a possibilidade de motivar os pacientes, pois permite que seja estabelecido um contexto para a realização das atividades que extrapolam o tratamento. Bruckheimer diz que na fisioterapia tradicional o paciente deve realizar uma sequencia de atividades repetitivas e que, através de recursos computacionais, é possível utilizar um jogo, por exemplo, onde o paciente conseguiria realizar estes mesmo movimentos. (BRUCKHEIMER, 2011).

Espiridião Elias Aquim, Chefe de setor de fisioterapia do Hospital VITA Curitiba, explica que o uso de videogames é um grande parceiro da fisioterapia, pois dá força muscular, melhora o equilíbrio, ajuda no condicionamento físico e ainda melhora nas habilidades motoras do paciente. A Instituição começou a utilizar console de videogames em 2009 com o primeiro videogame com console que consegue captar os movimentos do corpo. O médico afirma que, já foi indicado para quase dois mil pacientes do hospital e o resultado são surpreendentes e que com 25 anos de carreira, ele nunca conheceu um tratamento que despertasse tanto interesse e bem-estar aos pacientes. (BRASIL, 2012).

O grande desafio dos hospitais é entregar o paciente para a sociedade com maior funcionalidade e independência possível, para que ele possa voltar à sua vida normal. Os pacientes submetidos ao tratamento com videogames ficam mais motivados e interessados nas sessões, o que melhora percepção deles em relação

à recuperação e evolução do tratamento. (BRASIL, 2012).

Uma grande vantagem de jogos voltados à reabilitação é a utilização da realidade virtual e da realidade aumentada como formas de interação do jogo com o paciente. O autor define a RV como uma maneira de definir os mundos virtuais desenvolvidos com o uso de alta tecnologia para convencer o usuário que ele se encontra em outra realidade. Pode-se dizer que é uma interface humano-computador, onde ocorre a simulação de um ambiente real, utilizando imagens virtuais para o mesmo. (MONTERO; ZANCHET, 2013).

Segundo Fisioterapeuta Eduardo Arca, coordenador na área de ciência e saúde na Universidade do Sagrado Coração, afirma que o Exergame é apenas uma ferramenta e não podem substituir os exercícios de tratamento padrões para reabilitação. Só é eficiente quando o paciente está na fase final ou é utilizado no final das sessões dos exercícios. Portanto, durante os exercícios padrões, o Kinect pode auxiliar o fisioterapeuta durante na avaliação. A proposta deste projeto é desenvolver um protótipo de um software para verificar e analisar os movimentos corporais do usuário utilizando o Kinect da Microsoft. Os usuários visualizarão na TV como devem realizar e praticar os movimentos inclusive com a contagem de vezes ou tempo. Isso evitará que o paciente tente ludibriar a sessão, isto é, na prática normal, alguns pacientes fingem fazer os exercícios ou até mesmo se desmotivam por sentir dificuldades, assim, com um sistema será possível direcionar melhor cada paciente e estimulá-los de forma mais divertido, a realizarem a sessão de exercícios repetidos. Também servirá para distrair e estimular os pacientes minimizando o “tédio” que pode causar durante uma série de exercícios biomecânicos repetitivos.

1.2 OBJETIVO

1.2.1 Objetivo Geral

Desenvolver um protótipo de um software com o console da Microsoft Kinect para observar os movimentos corporais, tornando-se assim, futuramente, uma ferramenta para os profissionais de fisioterapia.

1.2.2 Objetivos Específicos

- Pesquisar a história do console da Microsoft Kinect e suas funcionalidades.
- Realizar um levantamento dos exercícios ortopédicos que são feitos na fisioterapia
- Desenvolver o software que identificará os movimentos do usuário no monitor com o Kinect na linguagem C# no programa Visual Studio 2015 para plataforma desktop.
- Testar o software verificando se o console realmente consegue o movimento pré cadastrado.

2 REABILITAÇÃO FISIOTERAPÊUTICA

O Conselho Federal de Fisioterapia e Terapia Ocupacional (CONFEITO, 2012) define a fisioterapia como “É uma ciência da Saúde que estuda, previne e trata os cinéticos funcionais intercorrentes em órgãos e sistemas do corpo humano, gerados por alterações genéticas, por traumas e por doenças”. Fundamenta suas ações em mecanismos terapêuticos próprios, sistematizados pelos estudos da Biologia, das ciências morfológicas, das ciências fisiológicas, das patologias, da bioquímica, da biofísica, da cinesia, da sinergia funciona, e da cinesia patologia de órgãos e sistemas do corpo humano e as disciplinas comportamentais e sociais. (CONFEITO, 2012).

2.1 EXERCÍCIOS TERAPÊUTICOS

Há muito tempo sabe-se que exercícios físicos trazem inúmeros benefícios para a saúde e podem ser usados de forma terapêutica para tratar problemas de saúde bem específicos como problemas respiratórios, cardiovasculares, neurológicos e é claro comprometimento das funções motoras. Segundo os autores, o exercício terapêutico é considerado um elemento central na maioria dos planos de assistência da fisioterapia, complementado por outras intervenções, com a finalidade de aprimorar a função e reduzir uma incapacidade. (KISNER; COLBY, 1998).

O treinamento sistemático e planejado de movimentos corporais, posturais ou atividade físicas com a intenção de proporcionar ao paciente meios de tratar ou prevenir comprometimentos, melhorar, restaurar ou aumentar a função física, prevenir ou reduzir fatores de risco relacionados à saúde, otimizar o estado geral de saúde ou a sensação de bem-estar. (KISNER; COLBY, 1998).

2.2 REABILITAÇÃO NEUROLÓGICA

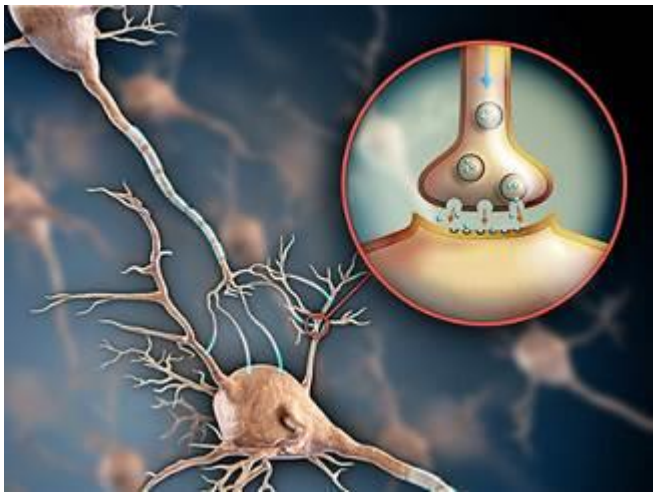
Compreendendo que as patologias neurológicas podem provocar deficiência, incapacidades e desvantagens, tem-se que a reabilitação deve buscar impedir, reverter/ corrigir, minimizar ou, pelo menos, proporcionar um mínimo de qualidade de vida durante o período de acometimento destes problemas. Os autores afirmam que

a reabilitar significa adquirir novamente uma habilidade tenha ela sido perdida ou reduzida. (ROCHA; DEFAVARI; BRANDÂO, 2012).

> O sistema nervoso central possui uma rede neural com células que determinam a sensibilidades e as ações motoras, traduzindo-as em comportamento. Quando existem lesões, surge um desfranjo nesta rede e o processo nervoso reinicia seus processos de reorganização e regeneração. (RIBEIRO, 2005, p. 2).

A aprendizagem motora é assumida em todos os níveis do sistema nervoso central, sendo que a necessidade ou desejo de interagir com o ambiente de forma adequada é um fator importante nas alterações que ocorrem a esse nível. Os movimentos repetitivos, o feedback e a motivação do paciente também são princípios básicos na reaprendizagem motora. (FERNANDES, SANTOS, 2012).

Figura 1- Rede de conexão sináptica



Fonte: CANNE (2015)

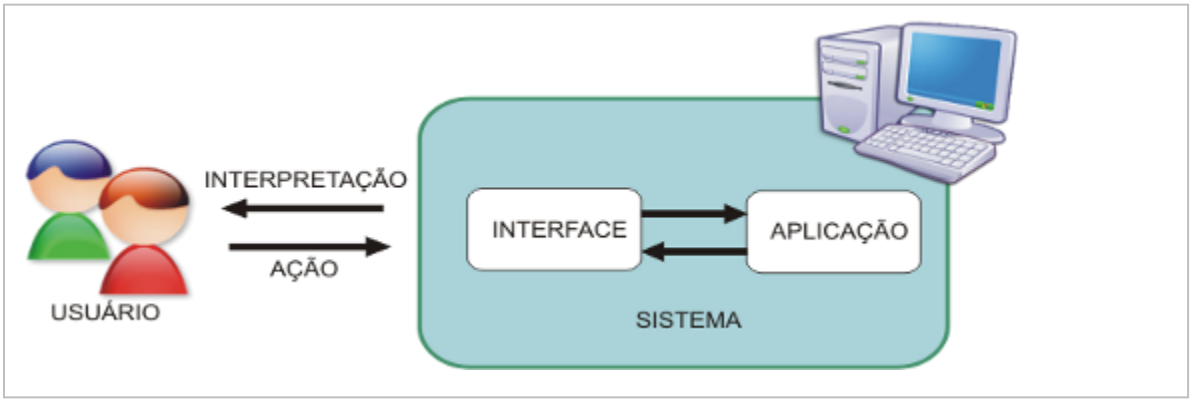
Os movimentos repetitivos têm grandes benefícios para o resultado funcional, pois elas descrevem que certos efeitos de estimulação cerebral só ocorrem se a área em questão for estimulada repetitivamente. Conforme mostra na Figura 1, quanto mais intensa e variada for a estimulação, mais sinapses (zonas ativas de contato entre uma terminação nervosa e outros neurônios) serão formadas. Estas repetições devem ser reguladas a fim de assegurar que fatores como cansaço ou

dor sejam minimizados. (FERNANDES; SANTOS, 2012; CEANNE, 2015).

3 INTERAÇÃO HOMEM – MÁQUINA

Também chamada de Interação Humano – Computador (IHC), interação entre humanos e máquinas acontece através da interface do utilizador, formado por software e hardware, relaciona com o design de sistemas computacionais que apoiem as pessoas de forma que elas possam conduzir suas atividades de forma produtiva e com segurança. O papel da IHC está presente em todos os tipos de sistemas, incluindo controle de tráfego aéreo e plantas nucleares, onde a segurança é imprescindível, planilhas eletrônicas e processadores de texto onde produtividade é fundamental, e até mesmo jogos eletrônicos, onde a satisfação e excitação são fundamentais. (PREECE, 1994)

Figura 2 - Interação Homem Máquina



Fonte: PAULA (2012)

Como mostra na Figura 2, na Computação, a interface atua como um tradutor, transpondo a linguagem artificial, de zero e uns, para a linguagem natural, humana. Em outras palavras, a relação governada pela interface é uma relação semântica, caracterizada por significado e expressão, não por força física. (JOHNSON, 2001).

Durante a história da computação, surgiram diversas maneiras de interação entre humanos e máquina. Dentre elas, os três principais tipos são: CLI (Interface de Linha de Comando), GUI (Interface Gráfica do Usuário) e NUI (Interface Natural do Usuário) como mostrado na Figura 3. (SÁ, 2011).

Figura 3 - Tipos de Interfaces

Fonte: Wormann (2014)

3.1 INTERFACE DE LINHA DE COMANDO (CLI - COMMAND-LINE INTERFACE)

Consiste de um mecanismo para interação com um computador através da digitação de comandos de texto para a realização de tarefas específicas. (STEPHENSON, 1999).

Surgiu por volta de 1950 com a utilização dos teletipos, máquinas de escrever eletromecânicas usadas para transmitir mensagens de um ponto a outro. Apesar das interfaces de comunicação continuam avançando, as características da CLI de baixo consumo de recursos da máquina, maior velocidade de navegação e maior controle sobre o sistema operacional fizeram com que o CLI evoluísse paralelamente com as outras interfaces. (SÁ, 2011; GARBIN, 2010).

3.2 INTERFACE GRÁFICA DO USUÁRIO *(*GUI – GRAFIC USER INTERFACE)

Popularizada pelo Macintoch da Apple, criado pela empresa Xerox Palo Alto Reserch Center em 1973, a interface GUI permite a interação do usuário através de manipulação direta movendo até o objeto como ícone e outros indicadores visuais e selecionando-os. Existem quatro elementos importantes dentro do GUI: Windows, Icon, Menu e Pointing device (Janela, Ícone, Menu e Cursor). (SÁ, 2011; GARBIN, 2010).

3.3 INTERFACE NATURAL DO USUÁRIO *(NUI – NATURAL USER INTERFACE)*

Interface de usuário que utiliza comportamentos naturais humanos para interagir diretamente com o conteúdo digital. O software em aplicação computacional faz a mediação das interações do usuário com o computador de uma forma intuitiva e natural, através do corpo, gesto, voz e toque. A palavra “natural” é usada porque a maioria das interfaces de computador utilizam dispositivos de controle artificial. (WIGDOR, 2011; NUIGROUP, 2012).

4 SIMULAÇÃO DE AMBIENTE REAL

4.1 REALIDADE VIRTUAL

Jaron Lanier introduziu o termo realidade virtual (RV) no final da década e 1980. Artista e cientista da computação, Lanier conseguiu convergir dois conceitos antagônicos em um novo e vibrante conceito, capaz de captar a essência dessa tecnologia: a busca pela fusão do real com virtual. Apesar que Lanier tenha definido o RV, as primeiras propostas e pesquisas que fundamentaram o estudo da realidade virtual foi o cineasta Morton Heiling na década de 1950 com o seu Sensorama, um equipamento no qual o usuário é submetido a diferentes estímulos como, movimentos, odores e ventos. (KIRNER, 2006).

De acordo com o autor, RV consiste em três conceitos fundamentais: interação, imersão e envolvimento. A interação se baseia em uma forma de permitir com que o jogador possa sentir-se parte do ambiente. Para isso, podem ser utilizados aparelhos de captura de movimentos para uma interação natural. A imersão consiste em fazer com que o usuário se sinta presente no ambiente tridimensional. O envolvimento consiste em fazer com que o paciente tenha motivação para completar determinada atividade estabelecida. (MACHADO, 2011).

4.2 REALIDADE AUMENTADA

Trabalha com o uso de objetos virtuais envolvidos no meio real, ambiente virtual e real, uma mistura de ambiente para o usuário se sentir mais confortável e identificado. A interação com este ambiente ocorre por meio de diversos tipos de dispositivos de entrada intuitivos, seja por toque, gestos ou fala, como se fosse o mundo real do usuário. (SÁ, 2011; GARBIN, 2010).

5 HISTÓRIA DO EXERGAME

A seguir serão apresentadas as principais invenções e uma breve história de onde começou a surgir essa combinação com o exercício e videogame.

5.1 ATARI PUFFER

O pioneiro nesta área foi à empresa Autodesk, que desenvolveu dois sistemas, o HighCycle e Raquetebol Virtual, no início dos anos 80. O HighCycle foi uma bicicleta ergométrica em que um usuário poderia pedalar por uma paisagem virtual. Se o usuário pedalar rápido o suficiente, a moto virtual decolaria e voar sobre a paisagem. Raquetebol Virtual Rastreados a posição e a orientação de uma raquete real que foi usado para bater uma bola virtual em um ambiente virtual. Mas a verdadeira tentativa foi a Atari, que lançou o vídeo game Puffer. Como no HighCycle, o jogador controla a velocidade pedalando e direcionando-o com o Gamepag. (BBC, 2015).

O usuário pedala visualizando na tela com uma paisagem virtual a sua velocidade, tempo, grau de inclinação e outros. De acordo com o RACERMATE, site de treinamento a bicicleta, foi a melhor criação feita para treinamento interno. Hoje, ele é utilizado em vários lugares integrado com o Sistema Windows da Microsoft. (RACERMATE, 2015).

5.2 DANCE DANCE REVOLUTION

Próxima invenção da Nintendo foi a Power Pad, jogo que necessita o movimento físico. O acessório é colocado para fora na frente do monitor de vídeo para vários jogos, geralmente ligados à segunda porta de controlador NES (Nintendo Entertainment System), os jogadores pisam nos botões para controlar o jogo. (BBC, 2014).

De acordo com a Ana Maria Nascimento da UFPE, o que marcou o “início” do Exergame foram o Dance Dance Revolution da Konami em 1998. (NASCIMENTO, 2012).

Projetado somente para entretenimento, os jogos do Dance Dance Revolution

foi um grande sucesso mundial. Trata-se de um aparelho de fliperama onde o jogador deve acompanhar o ritmo de uma música pré-determinada com pessoa em uma superfície de interatividade. A música pode ser desde uma suave balada até um frenético Techno, exigindo além de coordenação motora uma boa resistência cardiorrespiratória. (VIEIRA, 2015).

A tecnologia permite que o usuário tenha uma experiência real do movimento de diversos esportes e atividades físicas que proporcionam o desenvolvimento de habilidades sensoriais e motoras, devido à possibilidade de praticar exercícios por meio de mecanismos de realidade virtual e tecnologias de rastreio e atuação. (NASCIMENTO, 2015).

5.3 NINTENDO WII

De 2005 até o final de 2006, sob o nome Revolution, o Nintendo Wii trouxe uma mudança de hábitos fundamental para entender games na atualidade. Não vieram gráficos inovadores, mas sim, novamente, os games característicos dos 'nintendistas' e o Wii Mote, que difundiu o controle interativo nos jogos e despontou nas vendas. Um controle que na maioria dos jogos será necessário movimentar para realizar diversas ações importantes, como rebater uma bola de tênis ou dar socos nos adversários. Toda movimentação é captada por um sensor infravermelho colocado próximo à sua televisão, além da conexão Bluetooth existente no sistema. Esse desempenho acabou atraindo até pessoas que antes não jogam nada ou jogavam muito pouco de vídeo game. Com jogos simples, de jogabilidade intuitiva e extremamente divertidos, o Wii conquistou uma legião de fãs. (ZAMBARDA, 2010; MARCO, 2015; FONSECA, 2009).

Os games começaram a passar além de entretenimento, para a área de saúde e doenças no mundo inteiro. De acordo com Vieira (2015), no mundo, todos profissionais de Fisioterapia e educação Física começaram a adaptar os jogos eletrônicos Wii Fit (jogo do Wii com referencia a fitness, academia) para o uso com seus clientes. Com isso, nasceu o conceito de “Wii Reabilitação” que se tornou tão forte e aceito pelos profissionais de saúde que em 2008 o governo do Reino Unido oficializou o console Wii como instrumento da rede pública de Saúde. A autora acrescenta que, os benefícios da utilização do Nintendo Wii na fisioterapia, como

ferramenta terapêutica na literatura, incluem as correções da postura e do equilíbrio, o aumento da capacidade de locomoção, da amplitude e movimento dos membros superiores e inferiores, além da motivação do paciente, como mostra na Figura 4. (MERIANS, 2002)

Figura 4 - Reabilitação com Wii

Fonte: PINTO (2015)

5.4 MICROSOFT KINECT

Mas o hit do ano de 2010 foi o Kinect da Microsoft. Desta vez, o usuário não precisa segurar nada na mão e nem em cima de um console para controlar o peso. O Kinect leva o conceito (e o movimento) do Wii até sua culminação natural, livra-se completamente do controle. O Wii recuperou os videogames de seu nicho perpétuo ao lançar um console que era simples de usar. O Kinect está levando o videogame para o futuro ao literalmente ver e ouvir você. O Kinect é mais fácil de usar que o Wii, e traz para todo o seu corpo uma experiência de entretenimento eletrônico de uma forma que nunca existiu antes. Há um pequeno número de jogos Kinect em meio a um mar de jogos tradicionais não-Kinect. Mas o que acontecerá quando os desenvolvedores começarem a incorporar as características do Kinect? (THE NEW YORK TIMES, 2010).

Levar o Kinect além dos jogos para área de saúde, educação, segurança e outros. A Microsoft apelidou isso de Efeito Kinect. (XBOX, 2015).

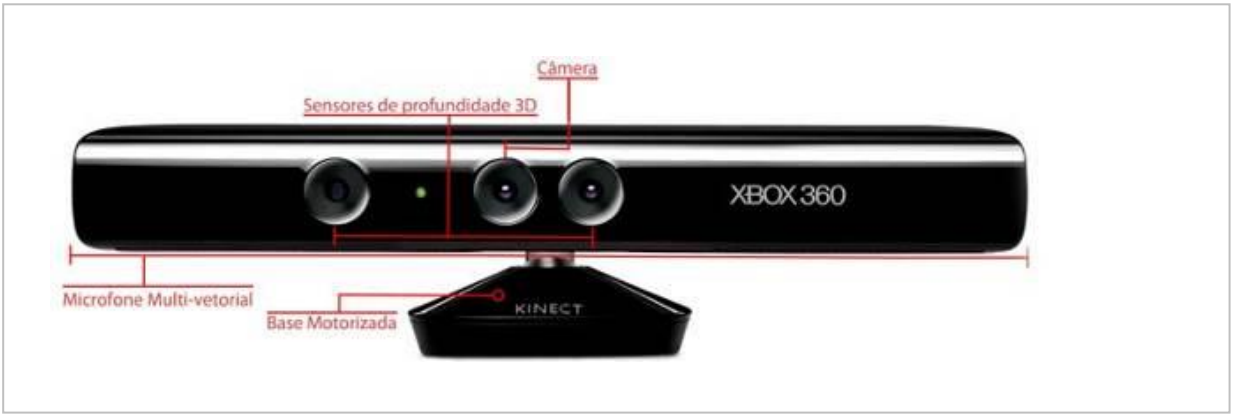
Depois desse movimento, a Microsoft lançou o kit de desenvolvimento de software para o Kinect, gratuitamente, bem como desenvolveu o Kinect para o Windows PC. (MICROSOFT; JINTRONIX, 2015).

6 FUNCIONALIDADE DO KINECT

O Kinect é capaz de detectar os movimentos dos jogadores com duas câmeras, um para captar as imagens e outro funciona como um sensor de profundidade no ambiente. Possui também um microfone e tudo isso funciona como um software próprio e cria um esqueleto digital em três dimensões, em dezenas articulações, com uma infinidade de movimentos que podem ser executados com o corpo inteiro. (XBOX, 2015).

No Kinect, existe um trio de inovações de hardware trabalhando junto com o sensor, conforme mostra a Figura 5:

Figura 5 - Console Kinect



Fonte: CANALTECH (2015)

6.1 ACELERÔMETRO

Um acelerômetro nada mais é que um sensor para medir a aceleração sobre um objeto. Ele é capaz de fazer medidas nas três dimensões: X, Y, Z. Além disso ele está configurado para suportar um intervalo de 2G, onde G é a força que a gravidade implica sobre o sensor, permitindo a sua orientação em relação a gravidade. Este sensor possui uma sensibilidade para temperaturas, o valor do seu eixo pode variar em até três graus positivamente ou negativamente e este desvio é necessário fazer uma comparação entre o eixo Y do acelerômetro e os dados de profundidade. O acelerômetro é extremamente importante, pois diversos projetos em que o Kinect é utilizado em um dispositivo que se move, ou seja, ele se estende

por todos os casos em que orientação do sensor é alterada. (SCHADE, 1998, p. 21).

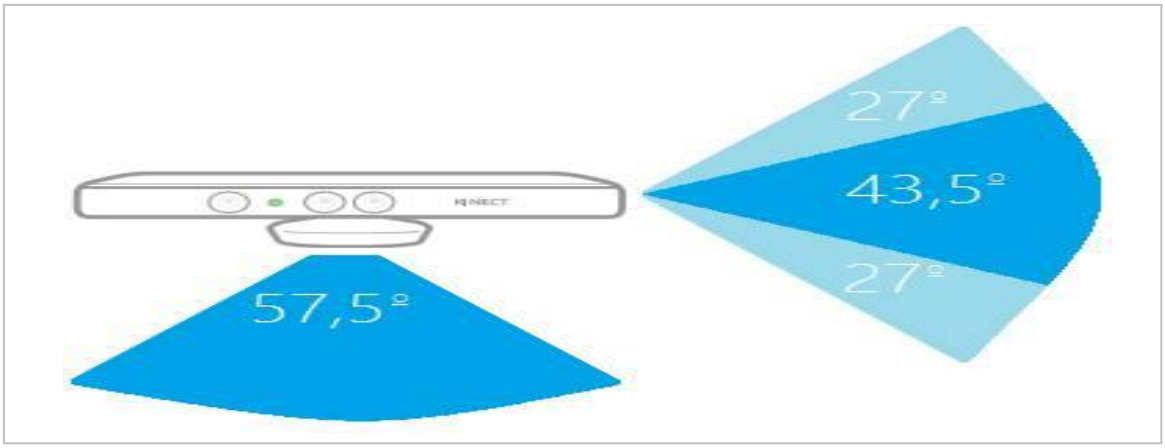
As Informações do acelerômetro no SDK são disponibilizadas através de um método conhecido como *kinectSensor.AccelerometerGetCurrentReading(),* retornando as informações sobre o acelerômetro em um vetor de 4 posição (X, Y, Z e W) onde o W sempre terá o valor zero. O motivo de ter a posição W em um sistema de coordenadas de três dimensões, é porque usa o conceito de coordenadas homogêneas, é uma ideia bem simples e poderosa. É representada as coordenadas X, Y e Z em uma quádrupla ordenada*[X, Y, Z e W] onde X= X/W+1, Y = Y/W+1 e Z= Z/W +1. O sistema de coordenadas do acelerômetro do sensor está centralizado com o dispositivo e por padrão, o retorno do método, caso o Kinect esteja em uma superfície plana, será próximo a (X: 0, Y: -1.0, Z:0 e W:0). O método para adquirir os valores é o seguinte:

```
AtualizarValoresAcelerometro(){
Vector4 resultado = kinectSensor.AccelerometerGetCurrentReading();
// método para adquirir os valores
// atribuindo os valores X,Y,Z respectivamente.
labelX.Content = Math.Round(resultado.X, 3);
 labelY.Content = Math.Round(resultado.Y, 3);
labelZ.Content = Math.Round(resultado.Z, 3);
}
```

6.2 Eixo Motorizado

O eixo motorizado é uma peça importante para o Kinect, através dele é possível alterar o ângulo de elevação do componente onde ficam as câmeras do sensor, alterando assim a visão que o sensor possui do ambiente. Este eixo movimenta o sensor apenas para cima e para baixo, atualmente não há uma forma de movimentá-lo horizontalmente. (SCHADE, 1998, p. 22).

Figura 6 - Raios de visão do Kinect



Fonte: SCHADE (1998, P.23)

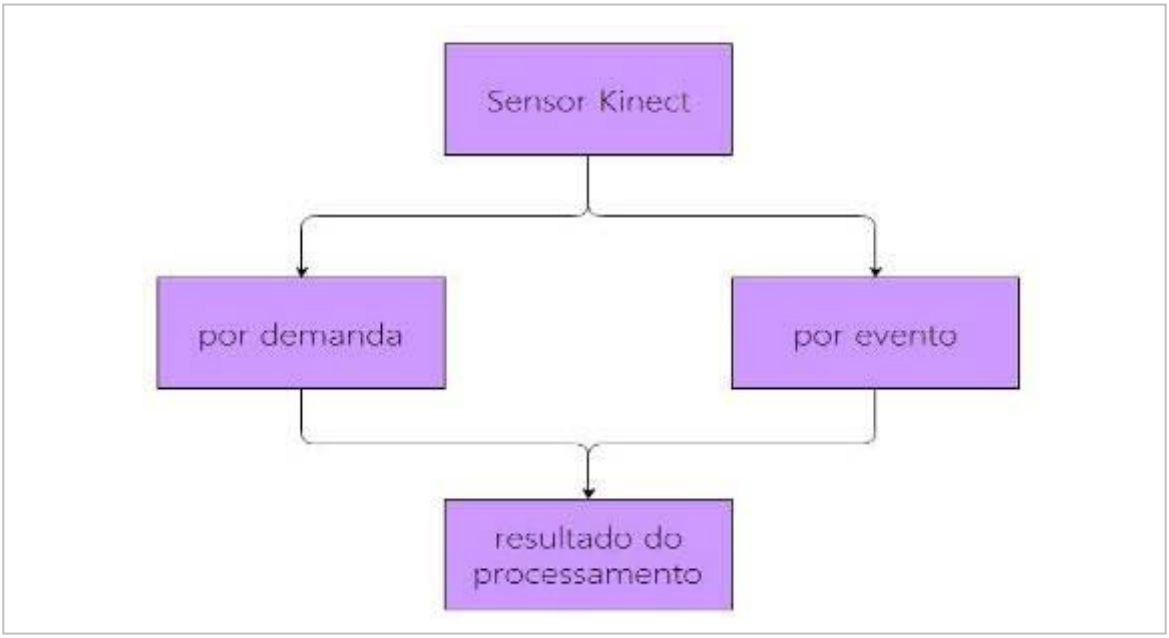
Como mostra na Figura 6, o raio de visão do Kinect é de 57,5 grau na horizontal e 43,5 na vertical, contudo o eixo motorizado pode movimentar a visão vertical para 27 graus para cima ou para baixo. Para alterar o grau de elevação do eixo motorizado é utilizado seguinte método:

```
private void AtualizarValores(){
kinectSensor.ElevationAngle = Convert.ToInt32(slider.Value); // alterando o grau
label.Content = kinect.ElevationAngle; // mostrar o ângulo do kinect
}
```

6.3 Fluxo do Kinect

Existem três tipos de processamento que geram o fluxo (streams) que fazem parte da classe *kinectSensor:* fluxo de cores (colorStream), fluxo de profundidade (DepthStream) e fluxo de esqueleto do usuário (SkeletonStream). Todos possuem algumas características em comum e estão ligados a um ou mais sensores que envolvem seu processamento. O processo do Kinect possui dois caminhos, um para obter um quadro, ou utilizando um método para solicitar o último quadro já processado, ou através de um evento que é disparado cada vez que um novo quadro deste fluxo está pronto, ou seja, já completou sua etapa de processamento. A Figura 7 mostra o fluxograma de funcionamento de todos os fluxos.

Figura 7- Fluxograma de funcionamento



Fonte: SCHADE (1998, p.26)

6.3.1 Fluxo da Cor

A câmera de vídeo ajuda no reconhecimento facial e na detecção de outras características ao detectar três cores componentes: vermelho, verde e azul. A Microsoft chama isso de “câmera RGB” se referindo aos componentes de cor que ela detecta. A Câmera de vídeo quanto à câmera com sensor de profundidade tem resolução de 640 x 480 pontos e funcionam a 30 quadros por segundo. Possibilitando a gravação perfil do jogador e fazer o seu reconhecimento. (STEPHANIE, 2010; XBOX, 2015).

A câmera de cores Kinect possui uma série de configurações, que influência diretamente na qualidade da imagem e na quantidade de quadros por segundo que o Kinect consegue processar.

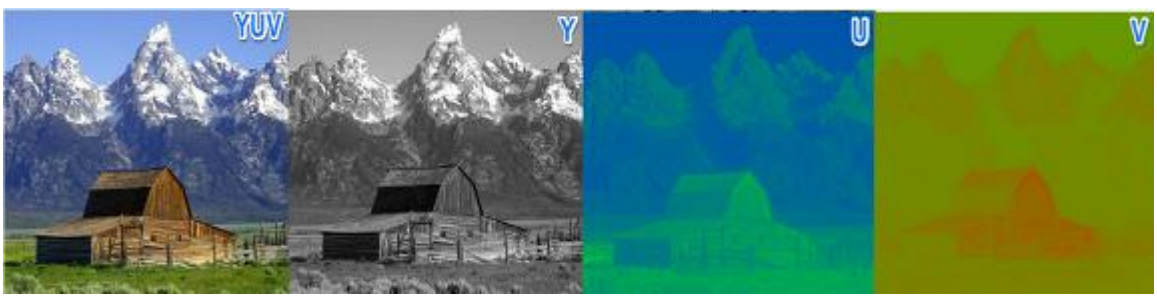
Figura 8 - Tabela de configuração da câmera RGB

Formatos	RGB		YUV	Bayer		IR
Bits por pixel	32		16	32		16
Resolução do eixo X	640	1280	640	640	1280	640
Resolução do eixo Y	480	960	480	480	960	480
Quadros por segundo	30	12	12	30	12	12

Fonte: SCHADE (1998, P.28)

Na Figura 8 mostra os 4 formatos que o Kinect consegue processar com a sua câmera: RGB, YUV, Bayer e IR. O formato RGB (Red – Green – Blue) é bastante conhecido no ramo de computação, neste formato cada pixel da imagem possui uma quantidade da cor vermelha, verde e azul. As somas destes três valores representam a cor do pixel. O formato YUV, que não é muito popular, codifica uma imagem ou vídeo levando em conta a percepção humana. Ele permite uma redução da largura de banda e de uma transmissão. Pois utiliza somente 16bits por pixel. Ele é muito útil para comprimir imagens, pois a perda é mascarada pela percepção humana, enquanto que o RGB possui o dobro de tamanho por pixel e nem faz diferença para os nossos olhos. Na sigla YUV, o Y representa a quantidade de luminosidade percebida pelo olho humano no pixel, o U e V representa dois componentes diferentes que os nossos olhos percebem as cores. Como mostra a seguir, na Figura 9. (SCHADE, 1998, p. 28).

Figura 9- Imagem YUV



Fonte: SCHADE (1998, P.29)

O formato Bayer é bastante similar ao RGB, mas ele altera a imagem proporcionalmente à nossa percepção para cores, por exemplo, verde é mais levado em consideração do que o vermelho ou azul. A diferente visual entre o formato RGB, YUV e Bayer são bem sutis como mostra na Figura 10 com uma mesma imagem nos três formatos com a resolução máxima. (SCHADE,1998, p.30).

Figura 10- Comparação entre formatos da câmera RGB

Fonte: SCHADE (1998, P.30)

Formato IR (InfraRed ou Infravermelho) é bem diferente dos formatos citados anteriormente. Quando a câmera é habilitada com este formato, podemos visualizar a forma com que o Kinect vê o ambiente. O resultado é uma imagem escura com diversos pontos luminosos que representa os pontos infravermelhos que são lançados pelo emissor de luz infravermelho no ambiente como mostra na Figura 11.

Figura 11- Luz infravermelho do Kinect

Fonte: BBZIPPO (2010)

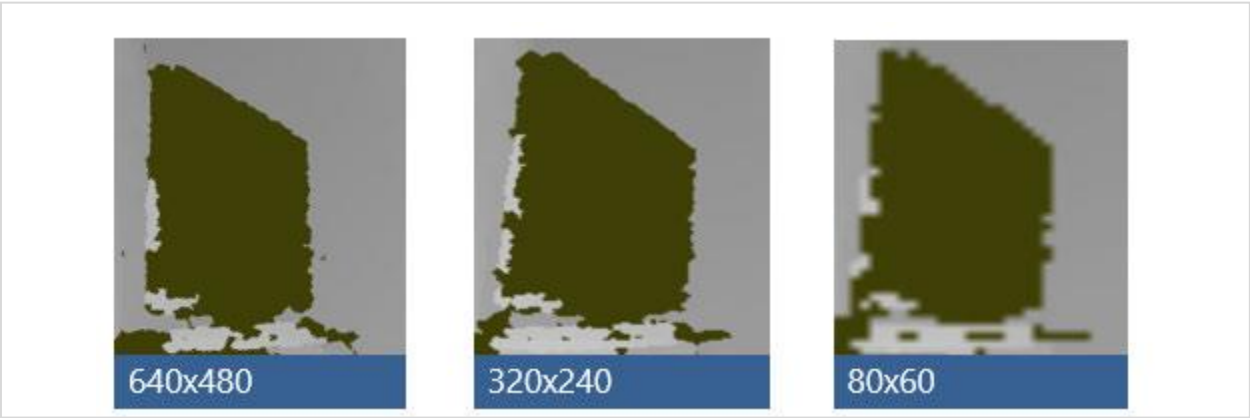
6.3.2 Fluxo de Profundidade

Um projetor infravermelho e um sensor CMOS (um tipo de tecnologia empregada na fabricação de circuitos integrados onde se incluem elementos de lógica digital. São baixíssimo consumo de energia e a possibilidade de alta densidade de integração) monocromático trabalham juntos para “ver” a sala em 3D sem levar em contas as condições de iluminação. (STEPHANIE, 2010; XBOX, 2015).

O Kinect cria um esqueleto digital de você baseado nas informações captadas. Portanto quando você se move para esquerda, direita ou pula pela sala, o sensor vai capturar e colocar as informações dentro do jogo. (XBOX, 2015).

O sensor de profundidade do Kinect também possui diferentes tipos de formato. Neste fluxo é possível alterar somente a resolução, pois em todos os formatos o FPS (Frames Per Second / quadros por segundo), permanece 30 FPS e a quantidade de bits por pixel permanece 16 FPS. A Figura 12 mostra a resolução do fluxo de profundidade. (XBOX, 2015).

Figura 12- Configuração do sensor de profundidade



Fonte: SCHADE (1998, P.48)

A imagem reconhecida pelo Kinect pode ser descrita em um formato conhecido como RGBD, ou seja, Red- Green – Blue – Depth. Todo pixel de profundidade possui 16 bits, sendo 13 para informações referentes à profundidade e 3bits que identificam se o pixel pertence à um humano. A câmera de profundidade é

capaz de reconhecer até 6 usuários em frente ao sensor e todas estas informações citadas são obtidas através da classe *DepthImagePixel.* (STEPHANIE, 2011).

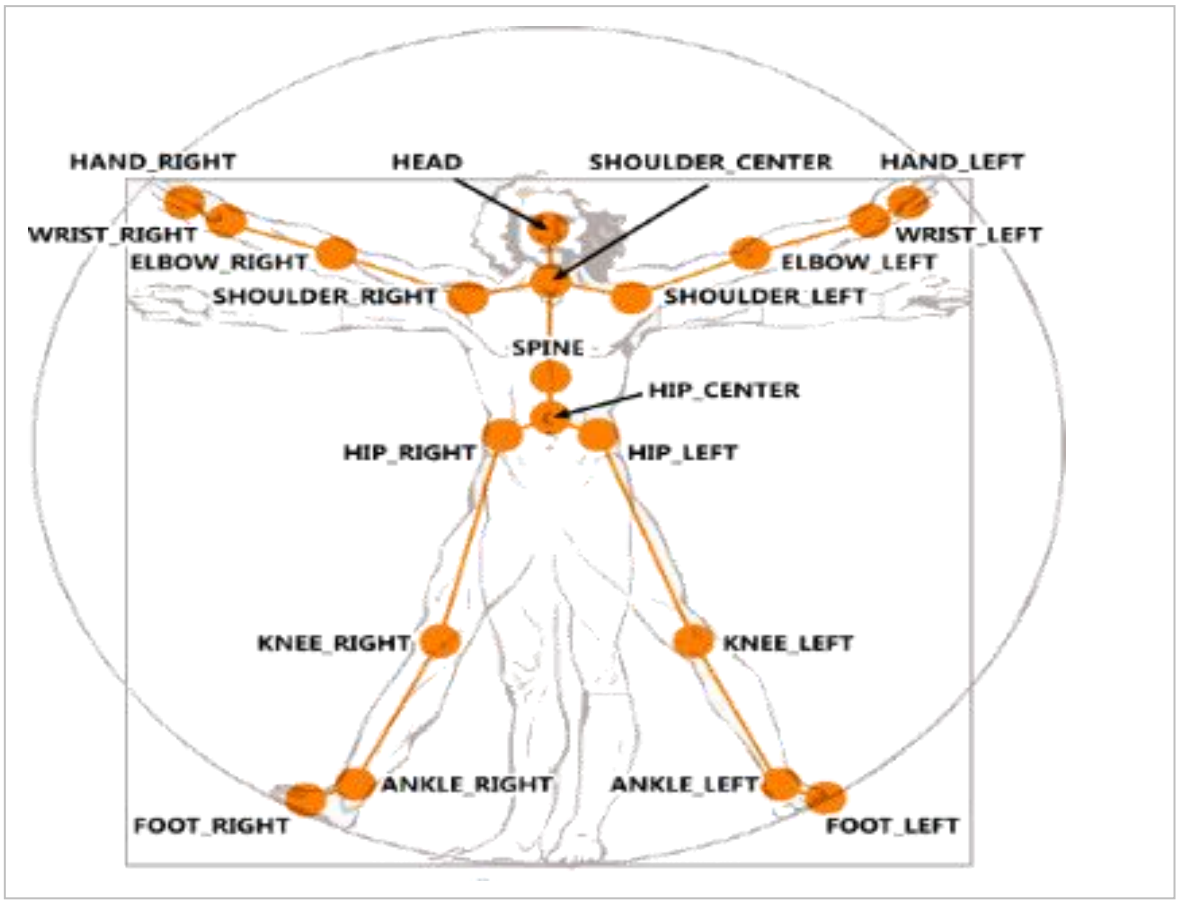
O fluxo de profundidade está intimamente ligado ao fluxo de esqueleto, pois ambos utilizam recursos uns dos outros para obterem informações. Para captar a profundidade do ambiente, o sensor utiliza o efeito parallax (mover o fundo de uma velocidade mais lenta do que as imagens de primeiros planos. Técnica que causa uma ilusão de profundidade nas interfaces) com seus dois sensores de profundidade e um algoritmo para detectar a profundidade de cada objeto na imagem. Claro que a técnica do Kinect é bem mais complexa, mas isto é uma breve explicação do sensor de profundidade. Podemos dizer que este fluxo é o mais importante do Kinect, pois praticamente toda aplicação utilizando o Kinect é feita baseada em movimentos. (SCHADE, 1998, p. 50).

6.3.3 Fluxo de Esqueleto

Este fluxo também é bastante semelhante aos demais já apresentados, entretanto não há um sensor físico que cria o esqueleto. Este fluxo é um conjunto de processamento e de sensores que tornam o Kinect capaz de identificar o usuário. Existem dois tipos de movimentos que são utilizados a partir do esqueleto do usuário: poses/ posturas e gestos. Uma pose é uma forma de manter o corpo parado por determinado tempo até que isto tenha algum significado, e gesto são os movimentos. (SCHADE, 1998, p. 51).

A Figura 13 apresenta o esqueleto fornecido pelo sensor que contém um conjunto de até vinte diferentes articulações identificadas, cada uma dessas articulações representa uma articulação correspondente no corpo humano. (SÁ, 2011).

Figura 13- Esqueleto Kinect



Fonte: ABREGO (2013)

6.4 Microfone Multi-Matriz

Uma série de quatro microfones conseguem isolar as vozes dos jogadores do barulho da sala. Ela possibilita ao jogador estar a poucos metros de distância do microfone e ainda usar controles por voz. (XBOX, 2015).

O diretor de incubação para Xbox da Microsoft, Alex Kipman explica que cada simples movimento do corpo é uma entrada que cria combinações de ações aparentemente sem fim. Sabendo disso, os desenvolvedores decidiram não programar essa combinação aparentemente sem fim em ações e reações preestabelecidas no software. Em vez disso, ela ensinaria ao sistema como reagir baseado em como os humanos aprendem classificando os gestos das pessoas no mundo real. (STEPHANIE, 2011).

6.5 SOFTWARE

Para começar o processo de ensino, os desenvolvedores do Kinect reuniram quantidades massivas de dados de captura de movimento em cenário da vida real. Em seguida, processaram esses dados usando uma máquina de aprendizado de algoritmo de Jamie Shotton, um pesquisador do Microsoft Research Combridge, na Inglaterra. (SHOTTON, 2011).

Eles conseguiram mapear os dados para modelos representando pessoas de diferentes idades, tipos de corpos, gênero e vestuário. Com os dados selecionados, os desenvolvedores conseguiram também, ensinar o sistema a classificar movimentos do esqueleto de cada modelo, enfatizando as juntas e as distancias entre essas juntas. (KUCHINSKA, 2010).

De acordo com os pesquisadores do Microsoft Research, uma vez que as partes do corpo são identificadas, o sistema calcula a provável localização das juntas para construir um esqueleto 3D. O Xbox roda o algoritmo 200 vezes por segundo, o que é cerca de 10 vezes mais rápido do que as técnicas anteriores de reconhecimento corporal. Com isto, os jogadores podem ser rastreados com rapidez suficiente para que seus movimentos sejam incorporados nos jogos. (SHOTTON, 2011)

7 ENGENHARIA DE SOFTWARE

A utilização de Computadores nas mais diversas áreas do conhecimento humano tem gerado uma crescente demanda por soluções computadorizadas e com isso, o desenvolvimento de software tornou uma extrema importância para a sociedade contemporânea. Os fundamentos da engenharia de software envolvem o uso de modelos abstratos e precisos que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades. Além disso, a engenharia de software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento de um sistema computacional. (ALMEIDA, 2005).

No processo de criação de um software é essencial se preocupar com a qualidade do produto. Mas isso varia de usuário, desenvolvedor e cliente. O usuário dirá que um bom produto é satisfazer suas necessidades, fácil de usar, eficiente e confiável. Por outro lado, para um desenvolvedor, é ser fácil de manter sendo o produto de software observado por uma perspectiva interna. Já o cliente pensa que o produto deve agregar o valor a seu negócio. Para isso, é necessário que a qualidade seja incorporada ao produto ao longo do seu processo de desenvolvimento. (ALMEIDA, 2005).

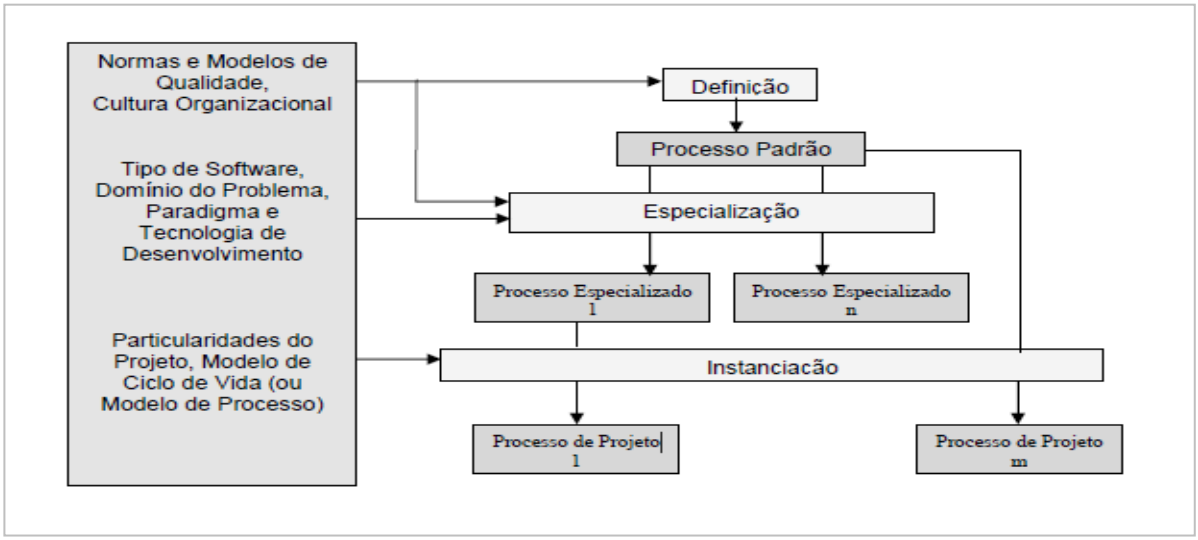
7.1 PROCESSO DE SOFTWARE

Há vários aspectos a serem considerados na definição de um processo de software, mas no centro de arquitetura estão as atividades-chaves: análise e especificação de requisito, projeto, implementação e testes, esses são a base sobre a qual o processo de desenvolvimento deve ser construído. (ALMEIDA, 2005).

A definição de um processo envolve a escolha de um modelo de ciclo de vida, o detalhamento de seu macro atividades, a escolha de métodos, técnicas e procedimentos para a sua realização e a definição de recursos e artefatos necessários e produzidos, pois um processo de software não pode ser definido de forma universal. Para construir um produto de boa qualidade, um processo deve ser adequado ao domínio da aplicação e ao projeto específico, ele deve ser definido

caso a caso, considerando-se as especificidades da aplicação, a tecnologia a ser adotada na sua construção, a organização onde o produto será desenvolvido para atingiras necessidades dos usuários finais, dentro de um cronograma de um orçamento previsíveis. A Figura 7 mostra de maneira geral, as fases do ciclo de vida de um software. Além das fases, fatores que influenciam a definição de um processo: tipo de softwares, paradigma, domínio da aplicação, tamanho e complexidade, características da equipe e outros. (ALMEIDA, 2005).

Figura 14 - Ciclo de vida de um software



Fonte: ALMEIDA (2005, P.9).

7.1.1 Planejamento

Fornecer uma estrutura que possibilite ao gerente fazer estimativas razoáveis de recursos, custo e prazos. À medida que os projetos progridem o planejamento deve ser detalhado e atualizado regularmente, pelo menos ao final de ada uma das fases do desenvolvimento. O planejamento e o acompanhamento do progresso fazem parte do processo de gerência de projeto. (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.2 Análise e Especificação de Requisitos

Nesta fase, o processo de levantamento de requisito é intensificado. O escopo deve ser refinado e os requisitos identificados. O engenheiro de software tem de compreender o domínio do problema, funcionalidade e o comportamento esperados. Uma vez identificado os requisitos do sistema a ser desenvolvido, estes devem ser modelados, avaliados e documentados. Uma parte vital desta fase é a construção de um modelo descrevendo o que o software tem de fazer. (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.3 Projeto

Responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema, modelados na fase anterior e, portanto, requer que a plataforma de implementação seja conhecida. Basicamente, envolvem duas grandes etapas. A primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. A segunda etapa é detalhar o projeto do software para cada componente identificado na etapa anterior, os componentes de software devem ser sucessivamente refinados em níveis de maior detalhamento, até que possam ser codificados e testados (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.4 Implementação

O projeto deve ser traduzido para uma forma possível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.5 Testes

Inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser

testada e os resultados documentados, e assim, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Depois disso, finalmente o sistema como um todo, pode ser testado. (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.6 Entrega a Implementação

Uma vez testado, o software deve ser colocado em produção. Para isso, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados. O propósito desta fase é estabelecer que o software satisfaz os requisitos dos usuários. Isto é feito instalando o software e conduzindo testes de aceitação. Quando o software tiver demostrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada. (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

7.1.7 Operação

O software é utilizado pelos usuários no ambiente de produção (PRESMAN, 2002).

7.1.8 Manutenção

O software sofrerá mudanças após ter sido entregue para o usuário e isso ocorre devido os erros que foram encontrados. O software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho, e isso depende do tipo e porte da manutenção necessária. Esta fase pode requerer a definição de um novo processo, onde cada uma das fases procedentes é reaplicada no contexto de um software existente ao invés de um novo (PRESMAN, 2002; SOMMERVILLE, 2003; PFLEEGER, 2004).

8 TRATAMENTO DE REABILITAÇÃO COM KINECT

Nesta seção serão apresentadas algumas clínicas que realiza o tratamento de fisioterapia com o Kinect. A abordagem contemporânea da reabilitação utilizando tarefas em ambientes virtuais poupa os fisioterapeutas do trabalho manual, oferecendo reabilitação para um maior número de pacientes, pois pode resultar no mesmo efeito em menor tempo. Ao mesmo tempo, as tarefas de realidade virtual apresentam uma motivação extra para o paciente. O autor explica que a telereabilitação permite ao paciente manter contato com os profissionais de saúde, médicos, terapeutas em domicílio, além de possibilita a continuação da reabilitação em longo prazo, com maior eficiência funcional. (Figura 15) (KRPIČ, 2013).

Figura 15- Reabilitação com Kinect

Fonte: Xbox (2015)

Ana Carolina da Silva, terapeuta ocupacional da AACD (Associação de Assistência à Criança Deficiente) comenta, “Como não há necessidade de controles existem sensores de movimento para permitir a interação com o game somente pelo deslocamento do corpo, os pacientes com dificuldades motoras têm maior facilidade para realizar as atividades”. (MICROSOFT, 2015).

8.1 CLÍNICA DE FISIOTERAPIA DA UNIVERSIDADE DE SÃO PAULO (UNICID)

De acordo com o fisioterapeuta que supervisiona o uso do game na clínica, professor Fabio Navarro Cyrillo, a ideia de usar o Kinect no tratamento em Fisioterapia surgiu após o sucesso obtido com o console da Nintendo e as possibilidades que essa nova tecnologia oferece. Evidências científicas recentes indicam que o uso da realidade virtual com videogame oferece além da motivação ganhos funcional para os pacientes, ajuda a reconquistar o equilíbrio, coordenação, resistência e força muscular, ele deixa a terapia mais dinâmica e envolvente. Além de tudo, o movimento é preciso, completo e mais semelhante aos realizados nas atividades diárias (NOGUEIRA; DIAS; RIGONATO; 2011).

8.2 ABP (ASSOCIAÇÃO BRASIL PARKINSON) – SÃO PAULO

No intuito de melhorar a qualidade no tratamento dos pacientes, foi inaugurada uma unidade de fisioterapia gamificada com a parceria da Jintronix Brasil (Plataforma virtual de reabilitação). O sistema exige apenas o Kinect para Windows, um computador e um software para captura de movimento, sem a necessidade de qualquer driver para o usuário. Cada Sessão é gravada em um banco e dados, assim o fisioterapeuta tem acesso às informações e acompanhamento de cada paciente. (ABOUTME; VIRTUALREHAB, 2014).

O programa motiva a rotina diária de exercícios dos pacientes e monitora a evolução podendo ajustar o tratamento. Além disso, os associados da ABP podem fazer o uso do sistema em casa enviando os resultados dos exercícios praticados via internet para os fisioterapeutas. (ABOUTME, 2014).

8.2.1 Jintronix-Jrs

A empresa desenvolveu um sistema de reabilitação em 2009 no Canadá que está sendo uma solução para os fisioterapeutas. Eles podem receber uma série de dados durante e após cada avaliação, e com eles em mãos, podem reforçar ou aliviar as séries e acelerar a recuperação. Danilo Castro, gerente da Jintronix-Brasil

comenta, O foco comercial da empresa não está somente em clínicas e hospitais, mas também nos profissionais e pacientes de fisioterapia. (ABOUTME, 2014).

O Jintronix foi desenvolvido principalmente para as pessoas com condições neurológicas, como depois de uma incidência de acidente vascular cerebral ou uma lesão cerebral traumática e condições ortopédicos. (JINTRONIX, 2015).

8.2.2 Virtualware

Empresa que se especializam no desenvolvimento de hardware e software de soluções baseados em imersividade e interatividade. Fundada em 2004, hoje é uma empresa global que tem mostrado um crescimento constante desde a sua criação em tecnologia de realidade virtual. (VIRTUALWARE, 2015).

8.2.3 Virtualrehab

VirtualRehab promove interação social, quer em centros de atendimento ou através da internet. É a primeira plataforma que incorpora a tecnologia na nuvem. Ele é baseado no Microsoft Azure, que permite o acesso on-line e controle, assegurando a confiabilidade do serviço e segurança, O aspecto tele reabilitação do produto, aumentam às possibilidades de tratamento, melhora a cobertura das necessidades das pessoas com deficiência, e com ela a sua independência e qualidade de vida. Ela combina técnicas modernas de captura de movimento através do Microsoft Kinect. O fisioterapeuta pode agendar sessões que permitem reabilitar funções motoras diferentes através de programas personalizados de reabilitação física. Até a data atual, foi registrada mais de 12mil sessão totalizando mais de 1200 horas de jogo terapêutico. (VIRTUALREHAB, 2014).

8.3 CENTRO DE REABILITAÇÃO NO EXTERIOR

8.3.1 The banner good samaritan reabilitation center (Phoenix -EUA)

O diretor médico do centro de reabilitação afirma que os pacientes aproveitam

dos jogos de vídeo, porque eles são uma maneira de se divertir e fugir daquelas difíceis sessões de fisioterapia. Isso dá as pessoas à ideia de que eles serão capazes de retornar as atividades divertidas no futuro e há algo para eles olharem. (BOEHM, 2013).

8.3.2 Northem Arizona University (Phoenix -EUA)

A diretora Kathleen Ganley do programa de terapia física da Universidade afirma que, os videogames tornaram um tem popular para o mundo de fisioterapia. Eles podem ser muito eficazes na terapia e fornecem uma fuga das sessões repetitivas de fisioterapia. [...]" é muito mais fácil dizer a pessoa derrubar o pino no videogame do que pedir para agachar, se levantar, movimentar o braço, etc. A terapia se torna difícil só quando o paciente é uma pessoa idosa e não tem contato com jogos". (BOEHM, 2013).

9 KINECT HACKING - KINECT NA SOCIEDADE

Depois que a Microsot liberou o kinect para os desenvolvedores, vários grupos de várias áreas viram isso como uma oportunidade. As possibilidades do sensor de movimentos podem ir além de apenas transformar o corpo em joystick. Foi o início da era Kinect Hacking, que abrirá o caminho para tecnologia futura ajudando tanto no setor público e privado da sociedade. (KINECTHACKS, 2015).

Alessandro Vieira dos Reis que desenvolve jogos eletrônicos para a saúde na FisioGame (*www.fisiogames.com.br*) e analista do comportamento afirma que, os jogos para a saúde surgiram devido à somatória de três contingências atuais. Primeira cosia é a inserção das tecnologias de informação nas instituições de saúde. Em Seguida vem o game como mídia e artefato cultural cada vez mais presente no cotidiano. Por último é a humanização da saúde, que traz a necessidade do bem-estar emocional e mesmo da ludicidade para o contexto clínico. (VIEIRA, 2015).

9.1 EDUCAÇÃO

Os Engenheiros e tecnólogos do instituto de tecnologia (FIT) desenvolveram um painel interativo para ajudar no aprendizado. Como um projeto piloto, foi criado um mapa do mundo no qual os usuários navegam pelos continentes utilizando somente as mãos ou a fala e obtém informações sobre 60 países. A um metro da TV, as pessoas podem conhecer a diferentes culturas e até responder a um jogo geográfico sobre os países, suas bandeiras e capitais. (SANTA TECNOLOGIA, 2012).

9.2 ARQUEOLOGIA

Pesquisadores de uma Universidade da Califórnia conseguiram transformar o Kinect em uma ferramenta para o mapeamento de sítios arqueológicos. Colocado em funcionamento junto com um sistema chamado de Cali2, o gedget consegue realizar uma identificação tridimensional do terreno. (KARASINSKI, 2011).

9.3 REABILITAÇÃO

Uma equipe de Pesquisadores da Universidade de Southampton do Reino Unido desenvolveu uma abordagem que utiliza o Kinet para fornecer estratégias de reabilitação realistas para pessoas que sofreram acidente vascular cerebral (AVC). A pesquisadora Chery Metcalf comenta que para recuperar de um acidente vascular cerebral pode ser um momento difícil e penoso para os pacientes e suas famílias. Através da pesquisa, muitas pessoas em reabilitação classificam seus exercícios caseiros como repetitivos e muitas vezes desmotivadores. Utilizando o Kinect, são capazes de aplicar um produto já comercialmente disponível e desenvolver uma ferramenta altamente inovadora que pretende ser tanto rentável quanto clinicamente útil. (ISAUDE, 2011)

9.4 VITRINE VIRTUAL

A empresa FaceCake Marketing criou um protótipo que a TV serve de espelho, fazendo com que o cliente experimente as peças como roupa, bolsa, joias e outros. Além de selecionar as peças, também podem mudar o fundo combinando com a peça, por exemplo, cenário de festa com um vestido, cenário de neve com roupa de esqui, cenário de praia para biquíni. Por mais que facilita os clientes, ainda tem suas barreiras como: Se realmente as roupas vão vestir bem, conforto do tecido, a qualidade da TV pode diferenciar nas cores, pode ser divertido na compra on-line, mas não vai resolver muito de seus desafios reais. (FACECAKE, 2015).

9.5 TRADUTOR DE SINAIS DE SURDOS

Os desenvolvedores da Microsoft Research criaram um protótipo que lê a linguagem de sinais dos surdos e traduzi-lo em texto falado. O tradutor reconhece mais de 370 gestos na língua chinesa e americana. Futuramente um surdo poderá fazer apresentações para multidões ou até trabalhar em quiosque de informação facilmente se comunicar com os visitantes. (KAN, 2013).

9.6 CÃO-GUIA ROBÔ

A empresa japonesa NSK está desenvolvendo um cão-guia robô baseada na tecnologia do Kinect para identificar obstáculo, diversas superfícies e escadas. As articulações possibilitam a subir e descer facilmente como um cão de verdade. Nas patas do robô foram equipadas com sensores para evitar buracos e impacto forte. A comercialização do cão-guia está prevista para 2020 e pretendem incluir comando de voz para que o robô seja comandado pelo dono e sistema de GPS para indicar o local desejado com a melhor rota. (BARROS, 2011).

9.7 MEDICINA

Em 2012, o hospital Evangélico de Londrina (PR) desenvolveram aplicativos semelhantes, que utilizam os recursos de reconhecimento de gestos do Kinect para que os profissionais de saúde possam acessar o banco de dados da instituição e visualizar arquivos digitais, imagens, exames e vídeos, assim como a ficha completa do paciente, apenas com gestos das mãos e a distância, em uma TV ligada ao acessório dentro do centro cirúrgico, evitando assim, risco de infecções para os pacientes que estão no ambiente esterilizado da sala de cirurgia. (MICROSOSFT, 2015).

9.8 CARRINHO DE MERCADO

Luis Matos do departamento de informática da Universidade de Beira Interior (Portugal) criou um carrinho de supermercado que segue pessoas com dificuldades de locomoção, como idoso, gestante, cadeirante, deficiente físico e outros. O carrinho é equipado com notebook, e como o sensor principal, kinect para evitar colisões mantendo uma distancia segura com a pessoa. (DICYT, 2015).

9.9 JOGO ADULTO

A empresa ThriXXX desenvolveu um jogo que é um simulador de sexo em 3D com mulheres seminuas. O jogador faz simulação com gestos, comando de voz para

interagir com ‘elas’. Kyle Machulis, especialista mundial em jogos e brinquedos digitais de cunho erótico comenta, ”Eu estou impressionado com rápida reviravolta no desenvolvimento desta plataforma, na obtenção de novos controles que funciona em uma engine como esta, especialmente quando utilizando recursos da comunidade open-source”. (JORNAL DO BRASIL, 2010).

10 METODOLOGIA

Neste trabalho foi desenvolvido um sistema de computador utilizando o sensor Kinect para futuramente, auxiliar profissionais da área de fisioterapia durante sua sessão. O usuário pode visualizar o seu esqueleto no monitor e verificar se seu movimento está correto através do sensor Kinect. O trabalho foi feito com base em uma pesquisa exploratória, com intuito de compor um protótipo de um sistema complementar aos trabalhos de fisioterapia utilizando a plataforma Microsoft Kinect. Suas potencialidades serão estudadas através de livros, artigos publicados, métodos de medição de movimentos utilizados por fisioterapeutas, a documentação oficial do SDK, sites oficiais mantidos pela Microsoft para uso da comunidade de desenvolvedores. Net (Framework de Desenvolvimento Microsoft) e sites não oficiais que publicam estudos e aplicativos que estão sendo desenvolvidos para serem utilizados com o sensor Kinect.

O sistema foi desenvolvido utilizando a linguagem C# com a plataforma .Net 4.5. Para a criação da camada de visualização, foi utilizada a tecnologia WPF a qual permite, além do desenho de formulários simples, o desenho de objetos 2D e 3D, no IDE (Integrated Development Enviroment) Visual Studio 2015 com o console Kinect for Xbox 360. Para isso precisa do seu adaptador para conectá-lo no computador (MICROSOFTc, 2015).

Para obter a comunicação com o Kinect foi utilizado o SDK oficial da Microsoft que atualmente está na versão 2.0, mas foi utilizado a versão 1.8 por compatibilidade do sistema operacional, que é possível adquirir através do site da Microsoft (MICROSOFTa, 2015).

Etapa 1: Foi feito uma busca referencial bibliográfico na área de fisioterapia para compreender procedimentos que viabilizem a composição da proposta do sistema.

Etapa 2: Compreender a análise e modelagem conceitual do sistema e entendimento das ferramentas que viabilizem o desenvolvimento para essa plataforma interface natural. Para isso será usada à linguagem de modelagem unificada (UML em inglês, Unified Modeling Language).

Etapa 3: Construir um sistema que identificará os movimentos do corpo

cadastrando previamente os movimentos para serem analisados, podendo assim, visualizar em um monitor/ TV como o seu movimento está sendo executado. Com o programa Visual Studio 2015 da Microsoft, Linguagem de programação C#, Kinect for Xbox 360, e o recurso do SDK distribuído pela Microsoft.

Etapa 4: Visualizar e testar se o Kinect consegui identificar e acompanhar o movimento cadastrado previamente.

10.1 FERRAMENTAS UTILIZADOS

10.1.1 VISUAL STUDIO

Ferramenta para desenvolver o software é baseada em componentes e outras tecnologias para a criação de aplicativos avançados de alto desempenho. Além disso, o Visual Studio é otimizado para projeto, desenvolvimento e implantação em equipe. (MICROSOFTa, 2015).

10.1.2 LINGUAGEM DE PROGRAMAÇÃO C#

A linguagem C Sharp, ou C#, é uma linguagem de programação desenvolvida pela Microsoft para compor a plataforma NET. Considerada uma evolução de linguagem C++, que teve origem na linguagem estruturada C, o C# é um código de sintaxe orientado a objetos, altamente escalável e fortemente tipado, assim como a linguagem de programação Java (MICROSOFT a, 2015).

10.1.3 .NET FRAMEWORK

Será necessário a plataforma .NET Framework para a aplicação da linguagem C#. É uma plataforma de desenvolvimento popular para criação de aplicativos para Windows, Windows Store, Windows Server e Microsot Azure. A plataforma .NET Framework inclui as linguagens de programção C# e Visual Basic, o Common Language Runtime e uma ampla biblioteca de classes. (MICROSOFTa, 2015).

10.1.4 UML (UNFIED MODELING LANGUAGE)

Linguagem de modelagem unificada (UML) é uma linguagem que define uma série de artefatos que nos ajuda a especificar, visualizar e documentar modelos dos sistemas, incluindo a estrutura e design atendendo todos os requisitos, muito útil principalmente em sistemas orientados a objetos. (OMG, 2015).

11 DESENVOLVIMENTO

Primeira coisa que precisamos para utilizar o Kinect for Xbox é o adaptador para poder conectar no computador. O seu cabo de conexão possui uma entrada específica, fonte, conector para o Kinect e a entrada USB como mostra na Figura 16.

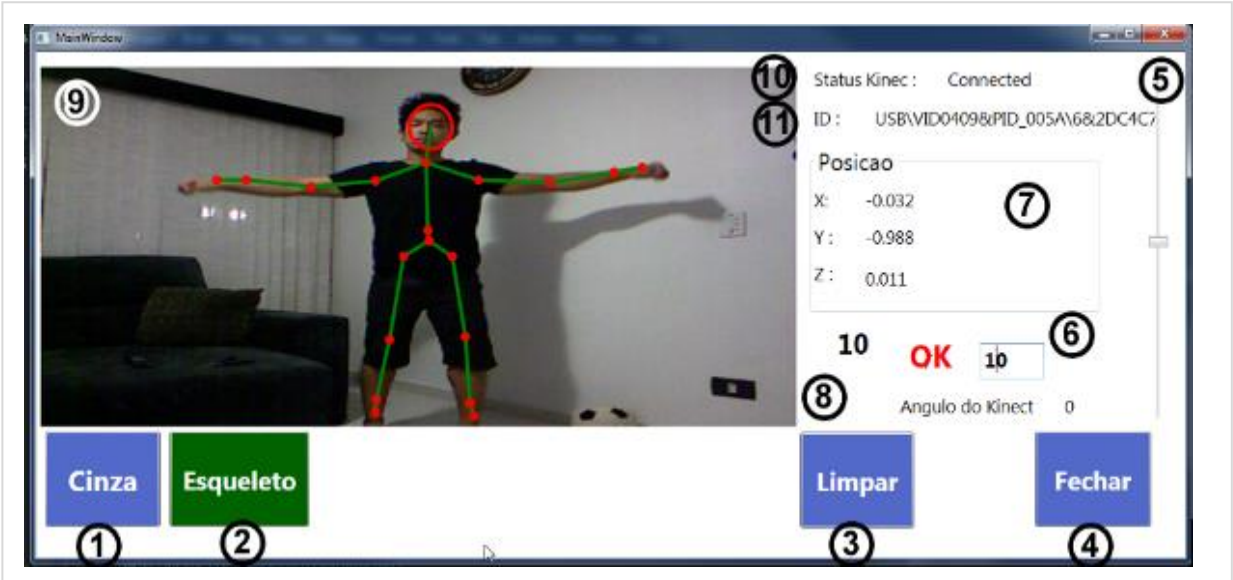
Figura 16 - Cabo de conexão do Kinect for Xbox 360

Fonte: SCHADE (1998, P.3)

11.1 Tela Inicial

A Figura 17 mostra a tela inicial do sistema de reconhecimento através do Kinect. Logo quando executa ele, com o Kinect conectado no USB, o Kinect é ligado automaticamente mostrando apenas a imagem obtido pela câmera frontal. Para a aplicação utilizei o WPF application (Windows Presentation Foundation) é um tipo de projeto proposto no Visual Studio para aplicação desktop para Windows, apesar que o Kinect não se limita a aplicação WPF, existem diversas ferramenta inclusa no SDK que nos auxiliam neste tipo de projeto. Foi feito algumas funções do Kinect utilizando o Windows Form Application e também funcionou perfeitamente. A diferença dos dois é que o WPF utiliza o XAML (eXtensivle Application Markup Language) para a construção da interface, similar ao HTML e CSS para aplicações Web.

Figura 17 - Tela inicial do sistema



Fonte: Elaborada pelo autor

1. Aplica um filtro preto e branco na imagem mostrada. O Kinect é capaz de converter e manipular cada pixel da imagem da imagem mostrada, mas desta vez vamos utilizar um algoritmo simples para converter as cores RGB.
2. Mostra o esqueleto do usuário que está em frente da câmera.
3. Limpa a contagem
4. Fecha o sistema.
5. Barra de rolagem para alterar o ângulo de elevação do sensor. Ele é possível subir e descer 27 graus.
6. Inserir quantidade que exercício que precisa fazer
7. Mostra os valores do acelerômetro do sensor X, Y, Z. (responsável para escancear o corpo do jogador ajudando o motor a ter mais precisão de movimento (KINECT, 2015).
8. Mostra quantidade feira pelo usuário
9. Tela onde é exibido o ambiente do usuário
10. Status do Sensor Kinect
11. Identificação do sensor Kinect

11.2 Iniciando o sistema

Para iniciar o sistema com todos os sensores do Kinect funcionando, temos que ativar os métodos que vamos utilizar no *MainWindow (onde inicia a tela)*.

```
private void InicializarKinect()       {
        kinectSensor kinect = kinectSensor();      //SDK do kinect
        slider.Value = kinect.ElevationAngle;       //setando o Angulo de elevação do motor do kinect
        kinect.DepthStream.Enable();                //ativando o sensor de profundidade
        kinect.SkeletonStream.Enable();           // ativando o sensor de esqueleto
        kinect.ColorStream.Enable();              //ativando o sensor de cor
        kinect.AllFramesReady += kinect_AllFramesReady;
//criando evento onde todos os quadros(profundidade,esqueleto e cor) vão trabalhar
    }
```

Assim, logo vai disparar o evento *kinect_AllFramesReady* para visualizar o usuário na tela. Dentro desse evento é onde geram ou deixa no aguardando os eventos que o usuário solicitar. Podemos ver que primeira coisa que temos que fazer é obter a imagem do ambiente, para isso pegamos o quadro (pixel que o sensor gera por segundo) e aplicamos ele em uma imagem, que é a visão do nosso ambiente.

```
 private void kinect_AllFramesReady(object sender, AllFramesReadyEventArgs e)    {
      // capturando imagemdo ambiente
         byte[] imagem = ObterImagemSensorRGB(e.OpenColorImageFrame());
      //capturando a profundidade da imagem obtida
         FuncoesProfundidade (e.OpenDepthImageFrame(), imagem, 2000);
      // verificação para executar só quando o sensor conseguir obter a imagem do ambiente
     if (imagem != null)
    // atribuindo a imagem na tela
           canvasKinect.Background = new
ImageBrush(BitmapSource.Create(kinect.ColorStream.FrameWidth, kinect.ColorStream.FrameHeight,
96, 96, PixelFormats.Bgr32, null, imagem,
                        kinect.ColorStream.FrameWidth * kinect.ColorStream.FrameBytesPerPixel));

//limpando as imagens anteriores para não sobrecarregar na memória
        canvasKinect.Children.Clear();
```

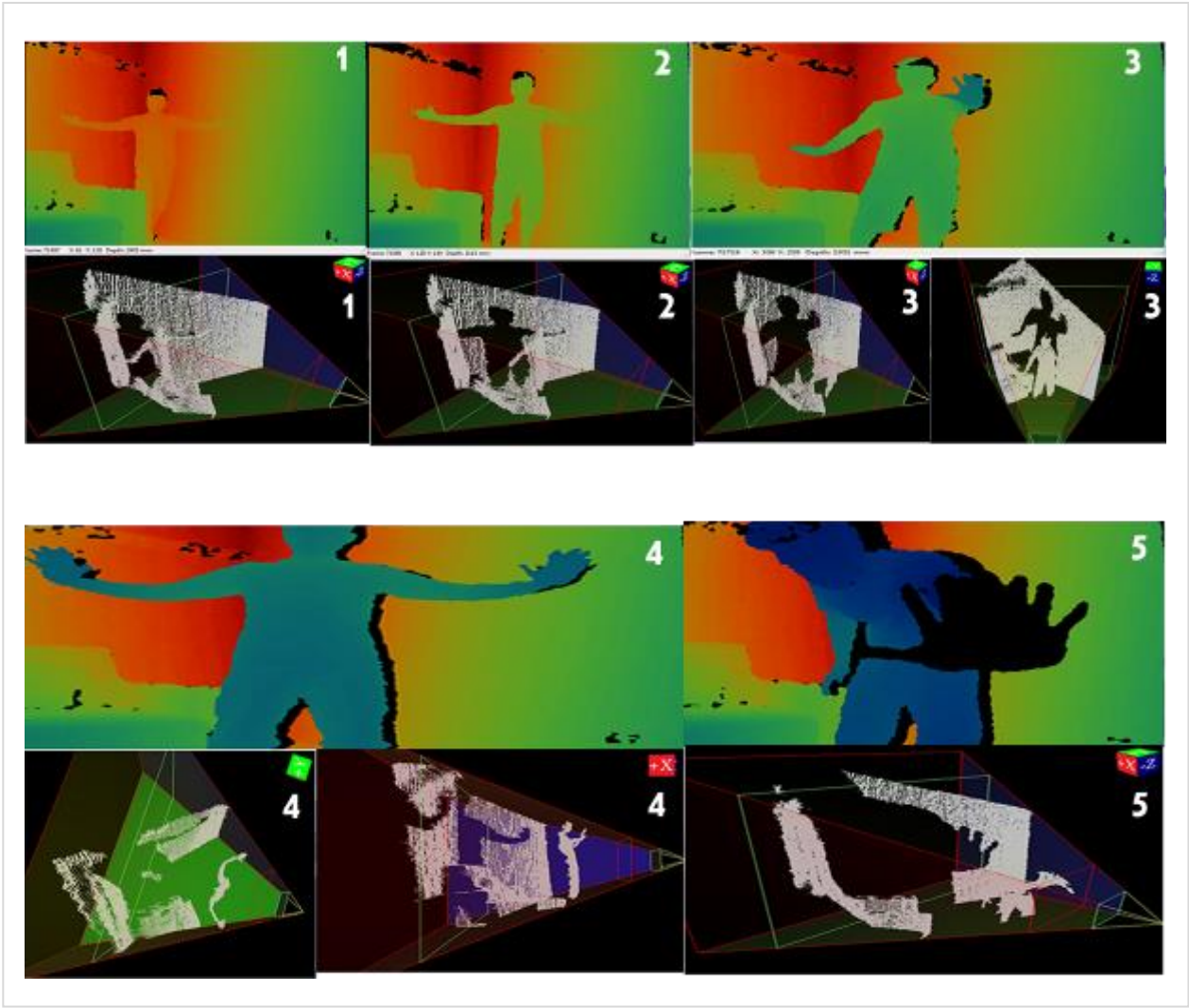
```
//capturando o esqueleto do usuário. Só será executada se o usuário solicitar no botão do esqueleto
            FuncoesEsqueletoUsuario(e.OpenSkeletonFrame());
        }

private byte[] ObterImagemSensorRGB(ColorImageFrame quadro)
        {
//verificação para quando o sensor não conseguir obter a imagem, não fazer nada.
            if (quadro == null) return null;
            using (quadro)    // utilizando a imagem
            {
//capta o pixel da imagem obtida e retorna a imagem
                byte[] bytesImagem = new byte[quadro.PixelDataLength];
                quadro.CopyPixelDataTo(bytesImagem);
                return bytesImagem;
            }
        }
```

Ao executar esses métodos, já podemos visualizar o nosso ambiente na tela. Por dentro do Kinect já está executando o fluxo RGB e da profundidade, só o esqueleto que ainda não por não ativar o botão do esqueleto. A Figura 18 mostra como o Kinect visualiza o ambiente e como diferencia a profundidade do ambiente. Ao falar do fluxo de profundidade, citamos que o utiliza o conjunto de sensores para poder capturar profundidade e esqueleto, e que consegue emitir infravermelho cheio de pontos para analisar a profundidade, pois esta imagem podemos ver nitidamente como ele está visualizando. A parte de cima que está tudo colorido, ele divide a profundidade com RGB, quando está distante a cor fica avermelhado, distancia intermediário esverdeada e por final azulada. Quando chega muito próximo do sensor ele fica preto, isso ocorre porque o Kinect for Xbox 360 não tem a função near(perto) para poder identificar a proximidade. Mas a distância de ficar preto já é o suficiente para ele analisar e montar a imagem. Na imagem número 5 da profundidade da cor parece ser um erro do sensor por mostrar mais uma mão ao aproximar. Mas isto é por ele está mostrando a sombra que gera pela aproximação. Na imagem número 5 da visão IR podemos ver a sombra do usuário.

Na Figura 18 mostra a imagem do infravermelho que o sensor é capaz de detectar do ambiente. Com a imagem capturada ele é capaz de mostrar 360º do

ambiente. A imagem número 1, 2 e 3(esquerda) mostra todos os lados X, Y, Z do ambiente, agora no número 3(direita) já mostra o ambiente de frente focando do lado Z. a imagem do número 4(esquerda) é o focado do lado Y, por cima do ambiente e o 4(direita) o lado X do ambiente.

Figura 18 - Sensor de profundidade



Fonte: Elaborada pelo autor

11.3 Escala Cinza

Para aplicar um filtro na imagem precisamos manipular cada pixel da imagem do quadro atual. Para isso basta converter as cores RGB do pixel em escala cinza. Depois de armazenar os dados da imagem para o array de pixel byte[]byteimagem,

faremos um laço de repetição para percorrer todos os pixels substituindo o valor das cores R, G e B pelo valor mais alto dos três. É importante saber que um pixel não é necessariamente um byte, dependendo do formato de cores são necessários mais de um byte para preencher um pixel. Por padrão cada pixel é representado por 4 bytes, um para o R, um para o G, um para o B e um para o alpha (transparência). O código para atribuir o filtro é o seguinte.

```
using (quadro)
 {
  DepthImagePixel[] imagemProfundidade = new epthImagePixel[quadro.PixelDataLength];
  quadro.CopyDepthImagePixelDataTo(imagemProfundidade);
  DepthImagePoint[] pontosImagemProfundidade = new DepthImagePoint[640 * 480];
          kinect.CoordinateMapper.MapColorFrameToDepthFrame(kinect.ColorStream.Format,
kinect.DepthStream.Format, imagemProfundidade, pontosImagemProfundidade);

   for (int i = 0; i < pontosImagemProfundidade.Length; i++)
    {
     var point = pontosImagemProfundidade[i];
     if (point.Depth < distanciaMaxima && KinectSensor.IsKnownPoint(point))
      {
        var pixelDataIndex = i * 4;

        byte maiorValorCor = Math.Max(bytesImagem[pixelDataIndex],
Math.Max(bytesImagem[pixelDataIndex + 1], bytesImagem[pixelDataIndex + 2]));

        bytesImagem[pixelDataIndex] = maiorValorCor;
        bytesImagem[pixelDataIndex + 1] = maiorValorCor;
        bytesImagem[pixelDataIndex + 2] = maiorValorCor;
      }
    }  }
```

11.4 CAPTURANDO ESQUELETO DO USUÁRIO

Como foi citado anteriormente, não há um método nativo que faça aparecer o esqueleto do usuário como imagem, é necessário criar um método de extensão utilizando o SDK do Kinect *SkeletonFrame* com a classe *skeleton*, que define a implementação virtual do esqueleto do usuário. E para obter as articulações do

usuário, usa-se a propriedade *Joint,* esta propriedade possui um indexador de acordo com o tipo de articulação. Antes de começar a trabalhar com as articulações, precisa pegar as coordenadas das articulações desejada. Para isso, criamos dois valores do tipo double que devem conter os valores referentes à alta e largura do componente projetado. A seguir é um código simples para obter as coordenadas.

```
private ColorImagePoint ConverterCoordenadasArticulacao(Joint articulacao, double larguraCanvas,
double alturaCanvas)       {
//irá rernar as coordenadas de uma articulação convertida para um plano de duas dimensões
          ColorImagePoint posicaoArticulacao =
kinect.CoordinateMapper.MapSkeletonPointToColorPoint(articulacao.Position,
kinect.ColorStream.Format);
          posicaoArticulacao.X = (int)(posicaoArticulacao.X * larguraCanvas) /
kinect.ColorStream.FrameWidth;
          posicaoArticulacao.Y = (int)(posicaoArticulacao.Y * alturaCanvas) /
kinect.ColorStream.FrameHeight;
          return posicaoArticulacao; }
```

Pegando a coordenada, agora temos que desenhar o esqueleto utilizando o objeto *Ellipse.*

```
 private Ellipse CriarComponenteVisualArticulacao(int diametroArticulacao, int larguraDesenho, Brush
corDesenho)
      {
//recebe o diametro do circulo,largura da borda e a cor que será desenhado.
          Ellipse objetoArticulacao = new Ellipse();
          objetoArticulacao.Height = diametroArticulacao;
          objetoArticulacao.Width = diametroArticulacao;
          objetoArticulacao.StrokeThickness = larguraDesenho;
          objetoArticulacao.Stroke = corDesenho;
          return objetoArticulacao;
              }
```

Agora que já conseguimos a coordenada e criar o componente visual, atribuímos os dois métodos para fazer o desenho da articulação em sua posição na imagem.

```
        public void DesenharArticulacao(Joint articulacao, Canvas canvasParaDesenhar)       {
      int diametroArticulacao = articulacao.JointType == JointType.Head ? 50 : 10;
      int larguraDesenho = 4;
      Brush corDesenho = Brushes.Red;
      Ellipse objetoArticulacao = CriarComponenteVisualArticulacao(diametroArticulacao,
larguraDesenho, corDesenho);

        ColorImagePoint posicaoArticulacao = ConverterCoordenadasArticulacao(articulacao,
canvasParaDesenhar.ActualWidth, canvasParaDesenhar.ActualHeight);

        double deslocamentoHorizontal = posicaoArticulacao.X - objetoArticulacao.Width / 2;
        double deslocamentoVertical = (posicaoArticulacao.Y - objetoArticulacao.Height / 2);

        if (deslocamentoVertical >= 0 && deslocamentoVertical < canvasParaDesenhar.ActualHeight
            && deslocamentoHorizontal >= 0 && deslocamentoHorizontal <
canvasParaDesenhar.ActualWidth)
        {
            Canvas.SetLeft(objetoArticulacao, deslocamentoHorizontal);
            Canvas.SetTop(objetoArticulacao, deslocamentoVertical);

            canvasParaDesenhar.Children.Add(objetoArticulacao);
        }
    }
```

Com esses métodos, já consegue desenhar os pontos de articulações na tela do usuário. O código a seguir mostrar como desenha os pontos.

```
private void DesenharEsqueletoUsuario(SkeletonFrame quadro)       {
        if (quadro == null) return;
        using (quadro)
        {
            Skeleton[] esqueletos = new Skeleton[quadro.SkeletonArrayLength];
            quadro.CopySkeletonDataTo(esqueletos);
            IEnumerable<Skeleton> esqueletosRastreados = esqueletos.Where(esqueleto =>
esqueleto.TrackingState == SkeletonTrackingState.Tracked);
            if (esqueletosRastreados.Count() > 0)
            {
                Skeleton esqueleto = esqueletosRastreados.First();
```

```
                EsqueletoUsuarioAuxiliar funcoesEsqueletos = new EsqueletoUsuarioAuxiliar(kinect);
                funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HandRight],
canvasKinect);
                funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HandLeft],
canvasKinect);            }        }
```

O código acima só criará dois pontos na mão direita e esquerda. Para acrescentar mais pontos, é preciso acrescentar o *Joint* especificando qual parte do esqueleto deseja visualizar. Na Figura 18 mostra como os pontos das articulações

Figura 18 - Criando ponto de articulação

Fonte: Elaborada pelo autor

```
using (quadro)          {
            Skeleton[] esqueletos = new Skeleton[quadro.SkeletonArrayLength];
            quadro.CopySkeletonDataTo(esqueletos);
            IEnumerable<Skeleton> esqueletosRastreados = esqueletos.Where(esqueleto =>
esqueleto.TrackingState == SkeletonTrackingState.Tracked);
            if (esqueletosRastreados.Count() > 0)
            {
                Skeleton esqueleto = esqueletosRastreados.First();
                EsqueletoUsuarioAuxiliar funcoesEsqueletos = new EsqueletoUsuarioAuxiliar(kinect);
                funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HandRight],
anvasKinect);
```

```
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HandLeft], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.Head], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HipCenter], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HipLeft], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.HipRight], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.WristLeft], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.WristRight], canvasKinect);
funcoesEsqueletos.DesenharArticulacao(esqueleto.Joints[JointType.Spine], canvasKinect);
        }       }
```

Figura 19- Hierarquia das articulações

Fonte: SCHADE (1998, P.78)

Na Figura 19, mostra a definição das relações entre as articulações e através dela, conseguimos obter informações sobre a relação das articulações, por exemplo, a rotação de uma articulação em relação a outra. Para acessar a propriedade dessas articulações, utiliza o *BoneOrientations* da *classe skeleton*. Ao acrescentar um laço de repetição, é possível ligar os pontos das articulações fazendo o desenho completo do esqueleto do usuário como mostra na Figra 20. A seguir mostra o código do laço de repetição.

```
EsqueletoUsuarioAuxiliar esqueletoUsuarioAuxiliar =new EsqueletoUsuarioAuxiliar(kinectSensor);
        foreach (BoneOrientation osso in esqueleto.BoneOrientations)
        {
        esqueletoUsuarioAuxiliar
        .DesenharOsso(esqueleto.Joints[osso.StartJoint],
        esqueleto.Joints[osso.EndJoint],
        canvasParaDesenhar);
        esqueletoUsuarioAuxiliar
        .DesenharArticulacao
        (esqueleto.Joints[osso.EndJoint], canvasParaDesenhar);
        }}
```

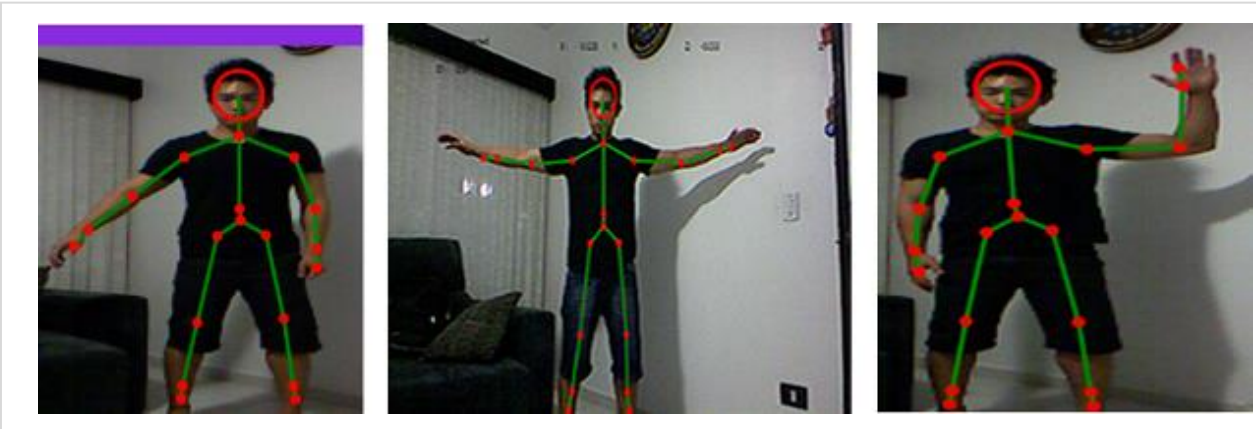
Figura 20- Esqueleto completo

Fonte: Elaborada pelo autor

11.5 Cadastrando movimento

Para que uma pose tenha validade é necessário que ela esteja sendo feita por um período de tempo, para isso, criamos duas etapas que avalia a pose ao longo do tempo chamando R*astreamento* e para quando é reconhecida, *Identificação*. Para o rastreamento, precisa usar a interface *IRastreador* do SDK do Kinect, que recebe por parâmetro o esqueleto do usuário e não precisa retornar nenhum tipo de valor. Assim, ele consegue rastrear um determinado movimento cadastrado no sistema, para isso utiliza-se o *Generics*. Generics é uma funcionalidade da linguagem C# que permite ao desenvolvedor criar um design de classes que podem aplicar funcionalidades a um determinado tipo que será definido apenas na aplicação final. A Figura 21 mostra os 3 movimentos que foram cadastrados previamente.

Figura 21- Movimentos



Fonte: Elaborado pelo Autor

11.5.1 Movimento T

Consiste em manter os dois braços esticados para os lados em largura paralela aos ombros, fazendo com que o formato do esqueleto do usuário se assemelhe com a letra “T”. Para todo movimento cadastro, tem que implementar uma margem de erro na comparação dos valores, pois é inviável exigir ao usuário que os movimentos sejam exatamente iguais.

```
public static bool CompararComMargemErro
(double margemErro, double valor1, double valor2){
return valor1 >= valor2 - margemErro &&
valor1 <= valor2 + margemErro; }
```

Para identificar se o usuário completou a posição “T”, foi feito a seguinte lógica recebendo as articulações do esqueleto como parâmetro para comparar as posições das articulações em seus eixos. Para isso, foi comparado à altura (eixo Y) e a distância (eixo X) das mãos e dos cotovelos em relação ao centro dos ombros e se o eixo X das mãos estão mais distantes de seu respectivo cotovelo. Nessa validação, foi aplicado a margem de erro 0.30. próximo código mostra o método para validar o movimento T.

```
        Joint centroOmbros = esqueletoUsuario.Joints[JointType.ShoulderCenter];
        Joint maoDireita = esqueletoUsuario.Joints[JointType.HandRight];
        Joint cotoveloDireito = esqueletoUsuario.Joints[JointType.ElbowRight];
        Joint maoEsquerda = esqueletoUsuario.Joints[JointType.HandLeft];
        Joint cotoveloEsquerdo = esqueletoUsuario.Joints[JointType.ElbowLeft];
        double margemErro = 0.30;
        bool maoDireitaAlturaCorreta = Util.CompararComMargemErro(margemErro,
maoDireita.Position.Y, centroOmbros.Position.Y);
        bool maoDireitaDistanciaCorreta = Util.CompararComMargemErro(margemErro,
maoDireita.Position.Z, centroOmbros.Position.Z);
        bool maoDireitaAposCotovelo = maoDireita.Position.X > cotoveloDireito.Position.X;
        bool cotoveloDireitoAlturaCorreta = Util.CompararComMargemErro(margemErro,
cotoveloDireito.Position.Y, centroOmbros.Position.Y);
        bool cotoveloEsquerdoAlturaCorreta = Util.CompararComMargemErro(margemErro,
cotoveloEsquerdo.Position.Y, centroOmbros.Position.Y);
        bool maoEsquerdaAlturaCorreta = Util.CompararComMargemErro(margemErro,
maoEsquerda.Position.Y, centroOmbros.Position.Y);
        bool maoEsquerdaDistanciaCorreta = Util.CompararComMargemErro(margemErro,
maoEsquerda.Position.Z, centroOmbros.Position.Z);
        bool maoEsquerdaAposCotovelo = maoEsquerda.Position.X < cotoveloEsquerdo.Position.X;

        return maoDireitaAlturaCorreta &&
            maoDireitaDistanciaCorreta &&
            maoDireitaAposCotovelo &&
            cotoveloDireitoAlturaCorreta &&
            maoEsquerdaAlturaCorreta &&
            maoEsquerdaDistanciaCorreta &&
            maoEsquerdaAposCotovelo &&
                    cotoveloEsquerdoAlturaCorreta;
```

11.5.2 Movimento Mão Esquerda 45 graus

Este movimento é um pouco mais complexo que comparações de eixos como no movimento “T”, pois além das comparações, precisa calcular o ângulo entre três articulações para ver quão aberto deve estar o braço do usuário. O movimento é levantar o braço esquerdo levemente para lateral no ângulo de 20 graus. Para a validação precisa do ângulo entre quadril esquerdo, ombro esquerdo e mão

esquerda. Além disso verificar se a mão está na mesma distância que o quadril e se ela está depois do cotovelo. O cálculo do ângulo entre três articulações utiliza o método conhecido como produto escalar como mostra na Figura 22. Utiliza-se dois vetores (V e W) para calcular um espaço de 3 dimensões.

Figura 22- Fórmula do produto escalar

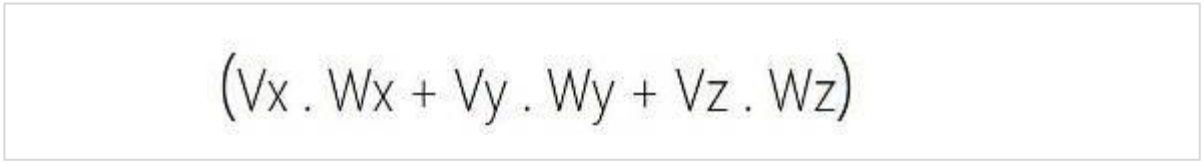
Fonte: SCHADE (1998, P.96)

```
public static double CalcularProdutoEscalar(Joint articulacao1, Joint articulacao2, Joint articulacao3)
{
        Vector4 vetorV = CriarVetorEntreDoisPontos(articulacao1, articulacao2);
        Vector4 vetorW = CriarVetorEntreDoisPontos(articulacao2, articulacao3);

        double resultadoRadianos = Math.Acos(ProdutoVetores(vetorV, vetorW) /
ProdutoModuloVetores(vetorV, vetorW));
        double resultadoGraus = resultadoRadianos * 180 / Math.PI;

        return resultadoGraus;      }
```

Depois de obter os valores dos vetores V e W, precisa aplicar em outra formula para calcular o resultado do produto como ilustra a Figura 23 e o código a seguir.

```
private static double ProdutoVetores(Vector4 vetorV, Vector4 vetorW)      {
        return vetorV.X * vetorW.X +
            vetorV.Y * vetorW.Y +
            vetorV.Z * vetorW.Z;       }
```

Figura 23 - Produto dos vetores

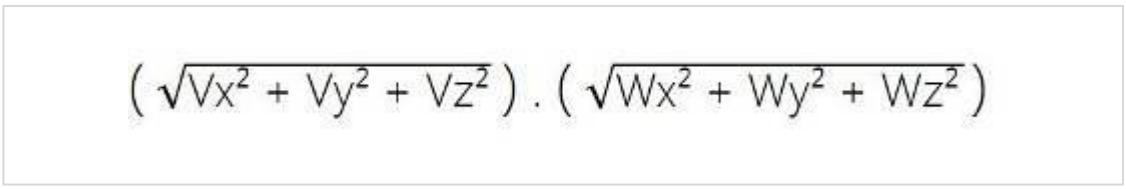


Fonte: SCHADE (1998, P.97)

Por último, precisa calcular o produto do módulo dos vetores multiplicando a raiz quadrada do resultado da soma do quadrado de cada eixo do vetor como mostra na Figura 24 e o código a seguir.

```
private static double ProdutoModuloVetores(Vector4 vetorV, Vector4 vetorW)       {
        return Math.Sqrt(Math.Pow(vetorV.X, 2) +
                    Math.Pow(vetorV.Y, 2) +
                    Math.Pow(vetorV.Z, 2))
                    *
                Math.Sqrt(Math.Pow(vetorW.X, 2) +
                    Math.Pow(vetorW.Y, 2) +
                    Math.Pow(vetorW.Z, 2));       }
```

Figura 24- Produto do modulo dos vetores



Fonte: SCHADE(1998,P.98)

O resultado do produto escalar será em radianos e precisa converter para graus multiplicando por 180 e dividir pelo número PI. Com isso consegue validar o movimento que levanta o braço esquerdo 45 grau para o lado esquerdo.

O tempo para executar e rastrear este movimento é aproximadamente um segundo e para exercer corretamente, na tela, é exibido uma barra de tempo para o usuário visualizar se está fazendo corretamente. E do mesmo jeito do movimento T,

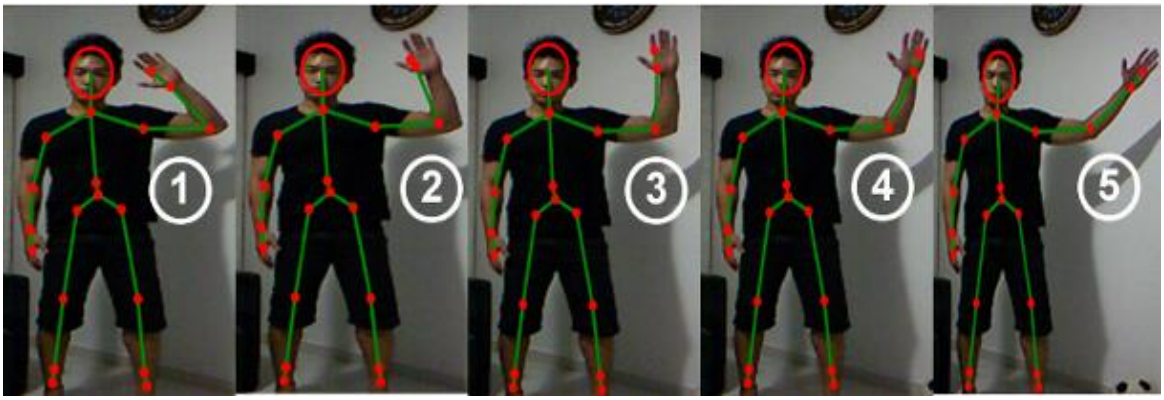
quando terminar o movimento certo, será contabilizado, podendo visualizar na tela. Abaixo mostra a sequência de código para validar o movimento.

```
const double ANGULO_ESPERADO = 25;
        double margemErroPosicao = 0.30;
        double margemErroAngulo = 10;

        Joint quadrilEsquerdo = esqueletoUsuario.Joints[JointType.HipLeft];
        Joint ombroEsquerdo = esqueletoUsuario.Joints[JointType.ShoulderLeft];
        Joint maoEsquerda = esqueletoUsuario.Joints[JointType.HandLeft];
        Joint cotoveloEsquerdo = esqueletoUsuario.Joints[JointType.ElbowLeft];

        double resultadoAngulo = Util.CalcularProdutoEscalar(quadrilEsquerdo, ombroEsquerdo,
maoEsquerda);

        bool anguloCorreto = Util.CompararComMargemErro(margemErroAngulo, resultadoAngulo,
ANGULO_ESPERADO);

        bool maoEsquerdaDistanciaCorreta = Util.CompararComMargemErro(margemErroPosicao,
maoEsquerda.Position.Z, quadrilEsquerdo.Position.Z);
        bool maoEsquerdaAposCotovelo = maoEsquerda.Position.X < cotoveloEsquerdo.Position.X;
        bool maoEsquerdaAbaixoCotovelo = maoEsquerda.Position.Y < cotoveloEsquerdo.Position.Y;

        return anguloCorreto &&
   maoEsquerdaDistanciaCorreta &&   maoEsquerdaAposCotovelo &&maoEsquerdaAbaixoCotovelo;
```

11.5.3 Movimento Aceno

Para o último movimento foi utilizado o algoritmo conhecido como keyframes, interpretar o movimento com uma série de quadro para serem reconhecida em um determinado espaço de tempo, é uma coleção de pose que são executadas uma após a outra e todas elas possuem uma margem de tempo para execução. Para cada quadro chave do gesto é necessário ter uma pose, que é a pose que o usuário deve estar e dois valores para controlar a quantidade mínima e máxima de quadros que devem ter se passado desde que o quadro chave anterior foi aceito. Para isso, precisou cadastrar três tipos de quadro chave para formar um movimento e poder reconhece-lo.

Figura 25 - Quadro chave do movimento.



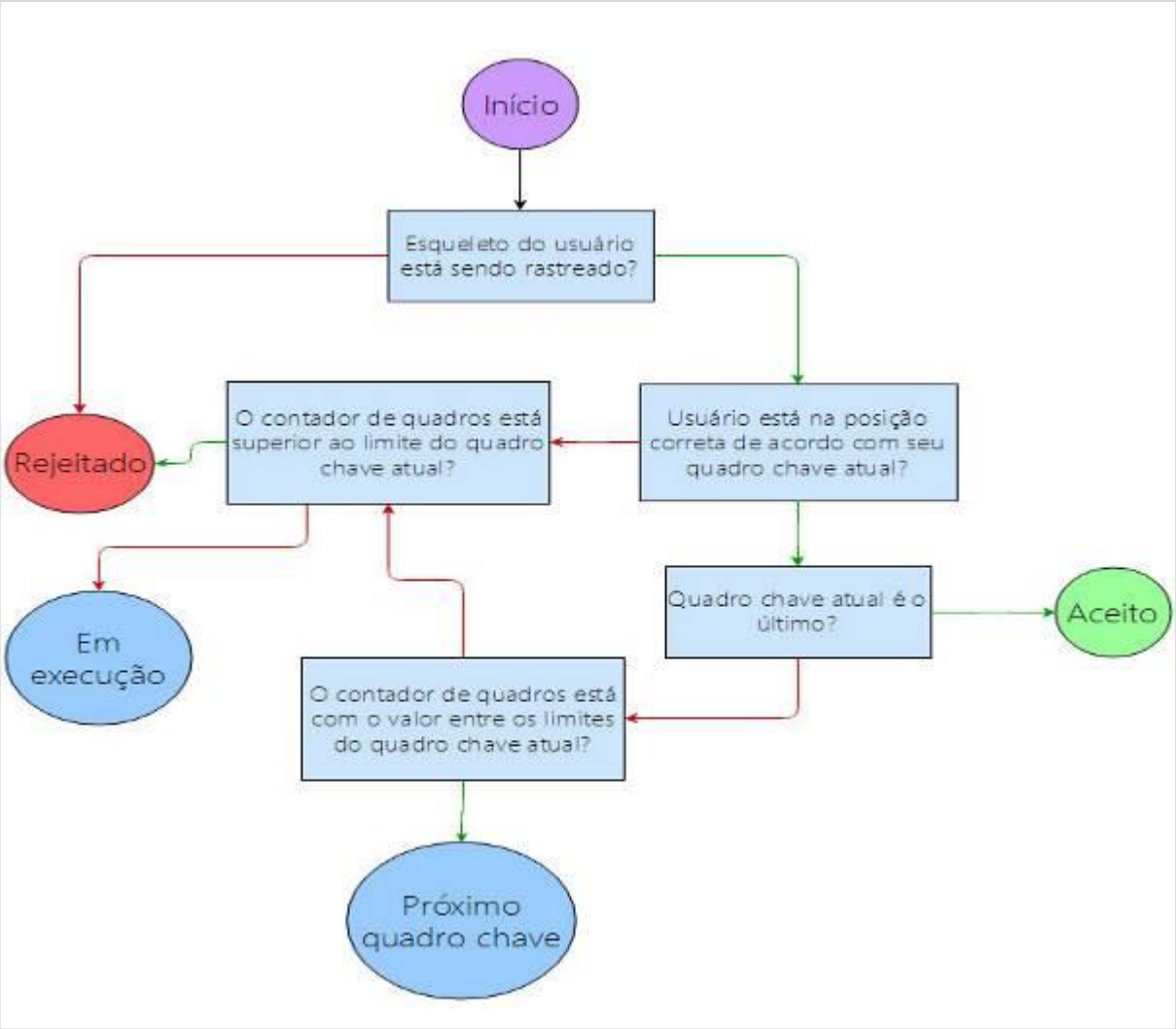
Fonte: Elaborado pelo autor

Para verificar o movimento (imagem 5) da Figura 25, é necessário que a mão do usuário esteja mais longe do corpo e mais alta que seu cotovelo, como foi feito no cadastramento do movimento T. O movimento da imagem 1 e 2, precisa verificar se a mão do usuário está mais próxima do corpo do usuário ao invés de mais longe. Por último, a imagem 3 precisa validar se a mão do usuário está mais alta e alinhada verticalmente com o seu cotovelo. Criando esses movimentos, já pode ligar todos e gerar um único movimento. Na Figura 26 mostra um fluxograma de como um quadro é executado para seguir o movimento e abaixo o método para validar o movimento.

```
protected override bool PosicaoValida(Microsoft.Kinect.Skeleton esqueletoUsuario)
    {
        const double ANGULO_ESPERADO = 25;
        double margemErroPosicao = 0.30;
        double margemErroAngulo = 10;


        Joint quadrilEsquerdo = esqueletoUsuario.Joints[JointType.HipLeft];
        Joint ombroEsquerdo = esqueletoUsuario.Joints[JointType.ShoulderLeft];
        Joint maoEsquerda = esqueletoUsuario.Joints[JointType.HandLeft];
        Joint cotoveloEsquerdo = esqueletoUsuario.Joints[JointType.ElbowLeft];
        double resultadoAngulo = Util.CalcularProdutoEscalar(quadrilEsquerdo, ombroEsquerdo,
maoEsquerda);
```

```
        bool anguloCorreto = Util.CompararComMargemErro(margemErroAngulo, resultadoAngulo,
ANGULO_ESPERADO);
        bool maoEsquerdaDistanciaCorreta = Util.CompararComMargemErro(margemErroPosicao,
maoEsquerda.Position.Z, quadrilEsquerdo.Position.Z);
        bool maoEsquerdaAposCotovelo = maoEsquerda.Position.X < cotoveloEsquerdo.Position.X;
        bool maoEsquerdaAbaixoCotovelo = maoEsquerda.Position.Y < cotoveloEsquerdo.Position.Y;
  return anguloCorreto &&  maoEsquerdaDistanciaCorreta &&  maoEsquerdaAposCotovelo &&
              maoEsquerdaAbaixoCotovelo;   }
```

Figura 26- Rastreamento e identificação de movimento

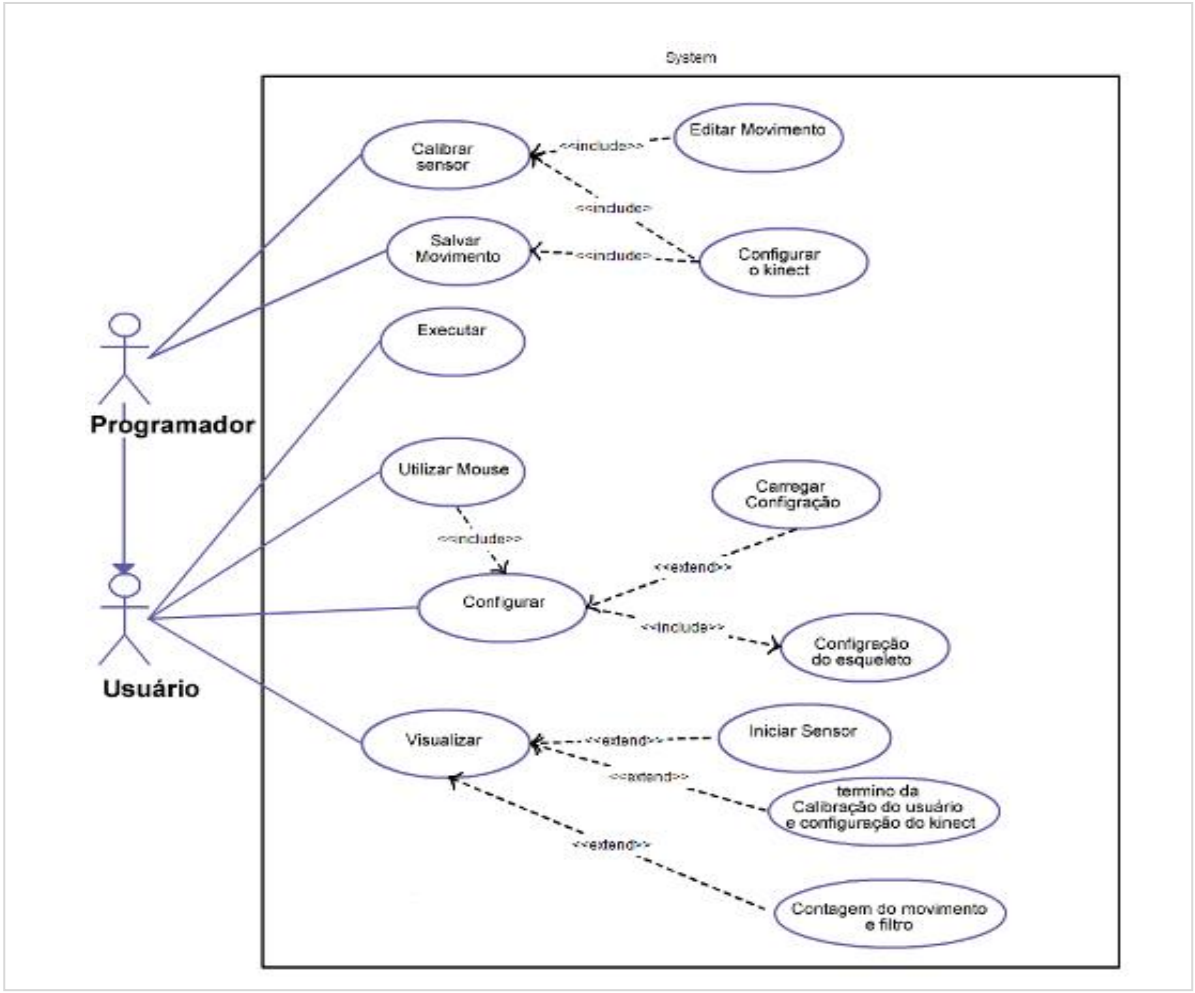


Fonte: SCHADE (1998, P.112)

11.6 Funcionamento do Sistema

Ao iniciar o sistema o usuário consegue visualizar a sua imagem através do dispositivo de saída do computador (TV, monitor), com isso ele pode optar por visualizar o seu esqueleto ou não. Os movimentos podem ser analisados mesmo sem o esqueleto do usuário ativado e será contabilizado se conseguir completar o movimento certo. Na tela principal tem a opção de calibrar a altura do Kinect para a sua melhor visualização. E também um campo para inserir quantas repetição querem fazer o movimento, visualizando sua contagem. A Figura 27 apresenta os casos de uso do sistema.

Figura 27– Diagrama de casos de uso



Fonte: Elaborado pelo autor

Primeiramente, o cadastrador (programador) vai calibrar o Kinect para salvar o movimento para o usuário poder exercer. Com o Kinect calibrado, ambiente mostrado no monitor, o usuário pode optar por configurar o angulo do Kinect, visualizar a sua imagem ou clicando com o mouse, o seu esqueleto. Ao iniciar o movimento cadastrado previamente, será contabilizado a quantidade de movimento.

12 CONCLUSÃO

A escolha deste tema foi um desafio por trabalhar com uma área desconhecida de reconhecimento de gestos, mas foi muito satisfatória por poder conhecer a tecnologia Kinect e suas funcionalidades. Além da aplicação que executei, é possível fazer inúmeras aplicações com ela. Acredito que é uma tecnologia do futuro por ser um dispositivo barato com suas incríveis funcionalidades, que ainda não estão sendo adaptado no nosso cotidiano. Por este motivo, instigou a vontade de continuar a desenvolver aplicações com o uso do sensor Kinect e aprimorar minha aplicação. Com uma boa criatividade e conhecimento básico de programação, qualquer pessoa pode criar um sistema com facilidade, pois o SDK que a Microsoft disponibilizou facilitou o uso do sensor Kinect.

Com o levantamento bibliográfico dos assuntos relacionado a fisioterapia, mostrou quanto o Kinect pode ajudar os profissionais da área e os pacientes que precisam do tratamento, na clínica ou até via internet para quem não tem condições de locomover.

Considera-se que o objetivo principal deste projeto foi atingido, de fato, o sistema proporciona as facilidades, interação com o usuário e reconhecimento dos movimentos cadastrado. Próxima etapa deste projeto é poder salvar os movimentos via sensor e não através de código, previamente cadastrado. Assim a facilita o cadastro de tratamento pelos fisioterapeutas para aplicar nos seus pacientes, podendo assim, acompanhar seus pacientes visualizando seu desempenho.

13 REFERÊNCIAS

ABOUTME, **Unidade de fisioterapia gamificada é inaugurada**, 2014. Disponível em: <http://aboutme.com.br/noticias/visualizar/341>. Acessado em: 20 de abril de 2015.

ABREGO M., **Tutorial Esqueleto Básuci Kinect,** Kinect for Windows Experienc. Disponível em :< https://malenyabrego.wordpress.com/2013/01/10/tutorial-esqueleto-basico-kinect/>. Acessado em 10 de maio de 2015.

ALMEIDA R. F., **Engenharia de Software,** 2005 Notas de Aula. Universidade Federal do Espírito Santo, Vitória.

ALMEIDA R. F., **Engenharia de Software.** Universidade Federal do Espírito Santo, 2005

BARROS T., **Empresa japonesa cria cão-guia robô com tecnologia kinect**, 2011. Disponível em: < http://www.techtudo.com.br/noticias/noticia/2011/11/empresa-japonesa-cria-cao-guia-robo-com-tecnologia-kinect.html>. Acessado em: 03 e maio e 2015

BBC**, From Atari Joyboard to wii Fit: 25 years of "exergaming".** Disponível em: <http://gadgets.boingboing.net/2008/05/15/from-atari-joyboard.html>, acessado em 11 de abril de 2015.

BBZIPPO, **Kinect in infrared**, Drawing Blanks. Disponível em: https://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/. Acessado em 5 de novembro de 2015.

Bem viver – Fisioterapia e Centro de Convivência. Disponível em: <http://nucleobemviver.com.br/>. Acessado em 10 de abril de 2013.

BOEHM J. **Medicine increasingly turning to videogames to speed recovery,** 2013. Disponível em: < http://cronkitenewsonline.com/2013/05/medicine-increasingly-turning-to-video-games-to-speed-recovery/>. Acessado em: 27 de abril de 2015.

BRASIL J., **Gameterapia ganha adeptos no Brasil; pesquisas focam aprimoramento de técnica,** 2012 Disponível em: <http://www.cienciaempauta.am.gov.br/2012/11/gameterapia-ganha-adeptos-no-brasil-pesquisas-focam-aprimoramento-de-tecnica/>. Acessado em 10 de abril de 2015.

BRUCKEIMER, Alessandro D. **Detecção corporal 3D na reabilitação virtual,** 2011. 86f. Trabalho de conclusão de curso, Curso de Ciência da Computação, UNIVERSIDADE ESTATUAL DE SANTA CATARINA, Joinville.

CABREIRA A., MULLING T. **Perspectivas para novas interfaces: Kinect e integrações gestuais sob o panorama de interfaces naturais do usuário,** 4.2012,

São Paulo.

CANALTECH**, Como Funciona o Kinect?** Disponível em:<http://canaltech.com.br/o-que-e/kinect/Como-funciona-o-Kinect/>. Acessado em 20 de abril de 2013.

CARVALHO, INGRED T. **JOGO VIRTUAL CONTROLADO PELOS SINAIS MIOELÉTRICOS NA RECUPERAÇÃO DE PACIENTES COM LESÃO MUSCULAR NOS MEMBROS SUPERIORES E/OU INFERIORES**, BRASILIA,2013. Disponível em:<http://www.repositorio.uniceub.br/bitstream/235/4917/1/20911162.pdf>. Acessado em 10 de abril de 2015.

CEANNE, NEUROPLASTICIDADE E REABILITAÇÃO. Disponível em: < http://www.ceanne.com.br/revista/neuroplasticidade-e-reabilitacao/ >. Acessado em: 20 de maio de 2015.

CHAN Y., CHEN S., HUANG J.D. **A kinect-base for physical rehabilitation: A pilot study for young adults with motor disabilities**, In Research in developmental disabilities, vol 32, N 6, 2011, 76f

CONFEITO – **Conselho Federal de Fisioterapia e Terapia Ocupacional**. 2015.Disponível em: < http://www.coffito.org.br/site/index.php/fisioterapia/definicao.html>. Acessado em: 01 de maio de 2015.

DB-ENGINES, **DB-Engines Ranking**. Disponível em <http://db-engines.com/en/ranking>. Acessado em 20 de maio de 2015.

DICYT, **Carrinho de Compra inteligente para deficiente**. Disponível em: < http://www.dicyt.com/noticia/carrinho-de-compras-inteligente-para-deficientes>. Acessado em 15 de Maio de 2015.

DIOGO R., **As melhores técnicas e exemplos do efeito parallax, LADO design,** Disponível em: <http://www.ladodesign.com.br/desenvolvimento-frontend/as-melhores-tecnicas-e-exemplos-do-efeito-parallax/>. Acessado em 10 de novembro de 2015.

FACECAKE, **Changing the way people shop.** Disponível em: <http://www.facecake.com/>. Acessado em: 01 de maio de 2015.

FERNANDES Carina I.S; SANTOS F. **Reaprendizagem motora e fisioterapia neurológica – revisão bibliográfica,** 2012, Trabalho de Conclusão de curso (Licenciatura em Fisioterapia) – Departamento de Ciência da Enfermagem e Tecnologia e Saúde, UNIVERSIDADE FERNANDO PESSOA, Lisboa.

FERRAZ, Leonardo T. D.; YAMASHIDA, Renato K. S. **Desenvolvimento de um jogo para reabilitação utilizando um sensor de com e movimento (Kinect)**. 2012. 7 f. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) – Departamento de Engenharia

Mecatrônica e de Sistemas Mecânicos, ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO, São Paulo.

FONSECA W. **Como funciona o Nintendo Wi**i, Tecmundo. Disponível em: <http://www.tecmundo.com.br/video-game-e-jogos/2504-como-funciona-o-nintendo-wii.htm>. Acessado em 15 de abril de 2015.

GARBIN S. M. **Estudo da Evolução das interfaces homem-computador**, Monografia- Curso de Engenharia Elétrica, Departamento de Engenharia de São Carlos, UNIVERSIDADE DE SÃO PAULO, São Carlos,2010

HALL, Carrie M; BRODY, Lori T. **Exercício terapêutico**: na busca da função. Rio de Janeiro: Guanabara,2007.

HUIZINGA J. **O jogo como elemento da cultura,** Perspectiva: 5 (1).2003;

Human Pose Estimation for Kinect. **Microsoft Research** Disponível em:< http://research.microsoft.com/en-us/projects/vrkinect/default.aspx>. Acessado em 10 de maio de 2013.

Indoor Training Gets Fun **BIKESPORT**. Disponível em: <http://www.bikesportmichigan.com/sales/computrainer.shtml>. Acessado em 8 de abril de 2013.

JINTRONIX. **Reforçar a reabilitação física por meio da tecnologia de captura de movimento.** Disponível em: <http://www.jintronix.com/>. Acessado em: 25 de Abril de 2015.

JOHNSON J. **From Atari Joyboard to Wii Fit: 25 years of "exergaming",** 2008. Disponível em: <http://gadgets.boingboing.net/2008/05/15/from-atari-joyboard.html>. Acessado em 12 de abril de 2015.

JOHSON S., **Cultura da inteface: como o computador transforma nossa maneira de criar e comunicar,** Rio de Janeiro, Jorge Zahar Ed, 2001

JORNAL DO BRASIL, **Primeiro jogo erótico chega ao kinect,**2010. Disponível em:< http://www.jb.com.br/ciencia-e-tecnologia/noticias/2010/12/21/primeiro-jogo-erotico-chega-ao-kinect/>. Acessado em: 04 de maio de 2015.

KAN M. **Microsoft usa kinect para interpretar a linguagem de sinais de surdos,** 2013.Disponível em :<http://www.tudosobretecnologia.com.br/2013/11/microsoft-usa-kinect-para-interpretar.html>. Acessado em: 03 de maio de 2015.

KARASINSKI L., **Como a tecnologia do kinect revolucionou o mundo,** 2011. Disponível em: <http://www.tecmundo.com.br/kinect/16237-como-a-tecnologia-do-kinect-revolucionou-o-mundo.htm>. Acessado em 01 de maio de 2015.

KARASINSKI V., RAMOS D. **Análise XBOX ONE.** 2013. Disponível em: <http://www.tecmundo.com.br/xbox-one/analise-xbox-one.htm>. Acessado em 20 de abril de 2015.

KINECT **HACKS, Utilizando Kinect**, Disponível em :<http://www.kinecthacks.com/>. Acessado em: 01 de maio de 2015.

KIRNER C., TORI R., SISCOUTTO R. A., **Fundamentos e tecnologia de realidade virtual e aumentada**. Editora SBC, 2006.

KISNER C.; COLBY L. A. **Exercícios Terapêuticos – Fundamentos e Técnicas.**3. Ed. São Paulo: Manolete, 1998.

KRPIC A, SAVANOVI A, CIKAJLO I. **Telerehabilitation: remote multimedia-supported assistance and mobile monitoring of balance training outcomes can facilitate the clinical staff's effort**. Int J Rehabil Res. p. 162–171.

KUCHINSKAS S. Binary body double: **Microsoft reveals the science behind project Natal for Xbox 30.** Disponível em: <http://www.scientificamerican.com/article/microsoft-project-natal/>. Acessado em

MACHADO L. S.; MORAES R. M., NUNES F. L., COSTA R.M.E., **Serious games baseados em realidade virtual para educação médica, Revista brasileira de educação médica**, 2011.

MACHADO Ricardo N. **PROPOSTA DE SISTEMA BASEADO NA PLATAFORMA KINECT PARA SUPORTE A REABILITAÇÃO DE PACIENTES COM PATOLOGIA LIGAMENTARES NOS JOELHOS,** 2013, Trabalho de Conclusão de curso (Bacharel em Tecnologias da Informação e Comunicação) – UNIVERSIDADE FEDERAL DE SANTA CATARINA, Araranguá.

MARCO D., **A revolução Nintendo**, AperteStart. Disponível em :<http://www.apertestart.com.br/a-re-revolucao-nintendo/>. Acessado em: 15 de abril de 2015.

MICROSOFT a**, Impacto no Brasil**, Disponível em: <http://www.microsoft.com/pt-br/about/impacto-no-brasil/v2/inovacao-e-ped/casos-de-sucesso/reabilitacao-rima-com-diversaov2.aspx>. Acessado em 10 de abril de 2015.

MICROSOFT b, **Recurso de Visual C#**. Disponível em: <https://msdn.microsoft.com/pt-br/vstudio/hh341490>. Acessado em 12 de maio de 2015.

MICROSOFT c, **Windows Presentation Foundation.** Disponível em: < https://msdn.microsoft.com/pt-br/library/ms754130.aspx>. Acessado em 01 de novembro de 2015.

MONTERO E. F., ZANCHET D. J., **Realidade virtual e a medicina**, Acta cirúrgica Brasileira, vol 18, n5,2003

NASCIMENTO A. **Vídeo game pode ajudar a combater obesidade infantil**. Disponível em:<http://www.diariodepernambuco.com.br/app/noticia/vida-urbana/2012/06/16/interna_vidaurbana,379349/videogame-pode-ajudar-a-combater-obesidade-infantil.shtml>. Acessado em 15 de abril de 2015.

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

NOGUEIRA D., DIAS S., RIGONATO A., **Realidade Virtual: Xbox Kinect é utilizado para reabilitação na Fisioterapia.** 2011.Disponível em :<http://www.unicid.edu.br/realidade-virtual-xbox-kinect-e-utilizado-para-reabilitacao-na-fisioterapia/>. Acessado em 25 de Abril de 2015.

NUIGROUP, **Natural User Interface**, Disponível em: <http://wiki.nuigroup.com/Natural_User_Interface>. Acessado em: 12 de maio de 2015.

OMG, **Introduction to OMG´s Unified Modeling Language (UML)**. Disponível em: < http://www.omg.org/gettingstarted/what_is_uml.htm>. Acessado em 03 de maio de 2015.

PACIEVITCH Y., **MYSQL**. Disponível em: <http://www.infoescola.com/informatica/mysql/>. Acessado em 20 de Maio de 2015.

PARIZKOVA, J, CHIN, M**. Obesity prevention and health promotion during early periods of growth and development:** J. Exercise Sci. Fitness, 2003; 1, 1-14.

PAULA A. S., **A Importância da Interação Humano-Computador,**TIQx. Disponível em: <http://tiqx.blogspot.nl/2012/02/compreenda-importancia-da-interacao.html>. Acessado em 25 de março de 2015.

PFLEEGER S.L., **Engenharia de Software**, São Paulo, 2004,

PINTO H., **Terapia** >. Acessado em 25 de março de 2015.

PIRES, B. **Revista Programar**. Edição 33. 2012.

PREECE J., **Human-Computer Interaction**, Addison-Wesley, 1994.

PRESMAN R. S., **Engenharia de Software**, Rio de Janeiro, 2002

RACERMATE. Disponível em :< https://www.racermateinc.com>. Acessado em 15 de abril de 2015.

RIBEIRO N., **O ambiente terapêutico como agente orimizador na neuroplaticidade em reabilitação de pacientes neurológicos,** revista diálogo possíveis, Salvador 4, n2, p.107,2005.

ROCHA, Pollyeverlin R.; DEFAVARI, Alex H.; BRANDÂO Pierre S., **Estudo da viabilidade da utilização do kinect como ferramenta no atendimento fisioterapêutico de pacientes neurológicos.**In: SIMPÓSIO BRASILEIRO DE JOGOS E ENTRETENIMENTO DIFITAL, 11, 2012, Brasília.

SÁ, JOBERT G. P., **Construindo uma DSL para reconhecimento de gestos utilizando Kinect.**2011,76f, Trabalho de conclusão de Curso, Centro de Informática, UNIVERSIDADE FEDERAL DE PERNAMBUCO, Recife.

SANTATECNOLOGIA, **Kinect é utilizado para ajudar as pessoas com deficiência.** Disponivel em: < https://adenilsontecnologia.wordpress.com/2012/09/14/kinect-e-utilizado-para-ajudar-na-educacao-de-pessoas-com-deficiencias/>. Acessado em: 15 de Maio de 2015.

SCHADE G. C., **Microsoft Kinect Criando:** Aplicações interativas com o Microsoft Kinect**,** São Paulo – SP – Brasil, Casa do Código, 1998

SHOTTON J. ,FITZBBON A. ,COOK M., SHARP T., FINOSHIO M.,MOORE R., KIPMAN A., BLAKE A., **Real-Time Human Pose Recognition in Parts from a Single Depth,** 2011**.** Disponível em: <http://research.microsoft.com/apps/pubs/default.aspx?id=145347>. Acessado em: 20 de abril de 2015.

SOMMERVILLE I., **Engenharia de Software**, São Paulo, 2003

SOUZA F. H. O uso do **Nintendo Wii como instrumento de reabilitação na Fisioterapia: revisão bibliográfica**, p.2. Disponível em :<https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&ved=0CDcQFjAE&url=http%3A%2F%2Fwww.fisioterapia.com%2Fpublic%2Ffiles%2Fsalvar_como.php%3Ftxt_path%3Dartigo%2Fartigo03_1.pdf&ei=WGNLVfehIIWngwSR64H4Bw&usg=AFQjCNFHsk0OBLQZAD19Qvedfdm9eFPpfQ&bvm=bv.92765956,d.eXY&cad=rja> . Acessado em: 15 de abril de 2015.

STEPHANIE, C. **Como Funciona o Microsoft Kinect**. **HowStuffWorks Brasil.** Disponível em :<http://eletronicos.hsw.uol.com.br/microsoft-kinect3.htm>. Acessado em 10 de maio de 2013.

STEPHANIE, C. **How Microsoft Kinect Works**. **HowStuffWorks**. Disponível em: <http://www.howstuffworks.com/microsoft-kinect.htm>. Acessado em 10 de Maio de 2013.

STEPHENSON N.**In the begining was the command line**, Avon Books, 1999.

THE NEW YORK TIMES, **Kinect e Move avançam com força no território do Wii**,2010. Disponível em :<http://veja.abril.com.br/noticia/vida-digital/kinect-e-move-avancam-com-forca-no-territorio-do-wii/>. Acessado em 17 de abril de 2015.

VIEIRA, A.R. **Vídeo games para a saúde.** Disponível em: <http://psiquecienciaevida.uol.com.br/ESPS/Edicoes/49/imprime163682.asp>. Acessado em: 15 de abril de 2015

VIRTUALREHAB, **Reabilitação virtual**. Disponível em: < http://www.virtualrehab.info/es>. Acessado em: 25 de Abril de 2015.

VIRTUALWARE GROUP, **Reabilitação virtual**. Disponível em: <http://virtualwaregroup.com/en>. Acessado em: 25 de Abril de 2015.

WIGDOR D, WIXON D. B., **and NUI World: designing natural user interfaces for touch and gesture**, 2011

WIKIPEDIA, **EXERGAMING.** Disponível em:<http://en.wikipedia.org/wiki/Exergaming>. Acessado em 15 de abril de 2015.

Wormann M., **Speak, wave, touch: How to do it right. User research insights about Natural User Interfaces.** SlideShare, slide 5. Disponível em: < http://www.slideshare.net/MichaelWrmann/speak-wave-touch-how-to-do-it-right-user-research-insights-about-natural-user-interfaces>. Acessado em 25 de março de 2015.

XBOX, **Configuração do Kinect no Xbox 360**. Disponível em :<http://support.xbox.com/pt-BR/xbox-360/kinect/kinect-sensor-setup>. Acessado em: 20 de abril de 2015.

XBOX360, **O Efeito Kinect. Xbox**. Disponível em:<http://www.xbox.com/pt-BR/Kinect/Kinect-Effect>. Acessado em 10 de abril de 2015

ZAMBARDA P. **A revoluçã nas mãos da Nintendo,** NINTENDO BLAST. Disponível em: <http://www.nintendoblast.com.br/2010/03/revolucao-nas-maos-da-nintendo.html>. Acessado em 15 de abril de 2015

14 APENDICE A - CÓDIGOS FONTE

MainWindow.xaml.cs

```
using AuxiliarKinect.FuncoesBasicas;
using Microsoft.Kinect;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect.Toolkit.Controls;
using AuxiliarKinect.Movimentos.Poses;
using AuxiliarKinect.Movimentos;
using AuxiliarKinect.Movimentos.Gestos;
using AuxiliarKinect.Movimentos.Gestos.Aceno;
using Microsoft.Kinect.Toolkit.Interaction;
using System.IO;
using System.Windows.Threading;
using TCCProjetoKinect.Auxiliar;

namespace TCCProjetoKinect
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        KinectSensor kinect;
        InicializadorKinect inicializador;
        List<IRastreador> rastreadores;
        InteractionStream fluxoInteracao;

        private IInteractionClient canvasDesenho;

        public MainWindow()
        {
            //onde inicializa tudo
            InitializeComponent();
            InicializarSeletor();
            InicializarRastreadores();
            InicializarTimer();
        }

        private void InicializarTimer()
        {
            DispatcherTimer timer = new DispatcherTimer();
            //timer para sempre estar mostrando as coordenadas x, y, z
            timer.Interval = TimeSpan.FromMilliseconds(100);
```

```
        timer.Tick += timer_Tick;
        timer.Start();
    }

    private void timer_Tick(object sender, EventArgs e)
    {
        //executa o método a cada 100 miliseconds
        AtualizarValoresAcelerometro();
    }

    private void AtualizarValoresAcelerometro()
    {
        //mostrando os valores X , Y , Z
        Vector4 resultado = kinect.AccelerometerGetCurrentReading();
        lblX.Content = Math.Round(resultado.X, 3);
        lblY.Content = Math.Round(resultado.Y, 3);
        lblZ.Content = Math.Round(resultado.Z, 3);
    }
    private void InicializarSeletor()
    {

        inicializador = new InicializadorKinect();
        inicializador.MetodoInicializadorKinect = InicializarKinect;
    }

    private void InicializarKinect(KinectSensor kinectSensor)
    {
        // instanciando SDK do kinect
        kinect = kinectSensor;
        slider.Value = kinect.ElevationAngle;
        lblAngulo.Content = "0";
        lblStatus.Content = "Connected";

        lblID.Content = kinect.DeviceConnectionId;
        kinect.DepthStream.Enable();
        kinect.SkeletonStream.Enable();
        kinect.ColorStream.Enable();
        kinect.AllFramesReady += kinect_AllFramesReady;
}

    private void InicializarRastreadores()
    {
        //inicializando o rastreamento
        rastreadores = new List<IRastreador>();

        Rastreador<PoseT> rastreadorPoseT = new Rastreador<PoseT>();
        rastreadorPoseT.MovimentoIdentificado += PoseTIdentificada;

        Rastreador<PosePause> rastreadorPosePause = new Rastreador<PosePause>();
        rastreadorPosePause.MovimentoIdentificado += PosePauseIdentificada;
        rastreadorPosePause.MovimentoEmProgresso += PosePauseEmProgresso;

        Rastreador<Aceno> rastreadorAceno = new Rastreador<Aceno>();
        rastreadorAceno.MovimentoIdentificado += AcenoIndentificado;

        rastreadores.Add(rastreadorPoseT);
        rastreadores.Add(rastreadorPosePause);
        rastreadores.Add(rastreadorAceno);
    }
```

```
    private void AcenoIndentificado(object sender, EventArgs e)
    {
        intCountT++;
        // ação quando o movimento de braço é identificado
        lblContagem.Content = intCountT.ToString();
        try {
            if (intCountT == Convert.ToInt32(txtQtd.Text))
                lblOK.Visibility = Visibility.Visible;
            else
                lblOK.Visibility = Visibility.Hidden;
        }
        catch { }
        }

    private void PosePauseEmProgresso(object sender, EventArgs e)
    {
        PosePause pose = sender as PosePause;
        //ação quando o movimento com o braço esquerdo está em progresso
        Rectangle retangulo = new Rectangle();
        retangulo.Width = canvasKinect.ActualWidth;
        retangulo.Height = 20;
        retangulo.Fill = Brushes.Black;

        Rectangle poseRetangulo = new Rectangle();
poseRetangulo.Width = canvasKinect.ActualWidth * pose.PercentualProgresso / 100;
        poseRetangulo.Height = 20;
        poseRetangulo.Fill = Brushes.BlueViolet;

        canvasKinect.Children.Add(retangulo);
        canvasKinect.Children.Add(poseRetangulo);
    }

    private void PosePauseIdentificada(object sender, EventArgs e)
    {
        intCountT++;
        //quando o movimento com o braço esquerdo é identificado
        lblContagem.Content = intCountT.ToString();
        try {
            if (intCountT == Convert.ToInt32(txtQtd.Text))
                lblOK.Visibility = Visibility.Visible;
            else
                lblOK.Visibility = Visibility.Hidden;
        }
        catch { }
    }

    private int intCountT = 0;
    private void PoseTIdentificada(object sender, EventArgs e)
    {
        intCountT++;
        // quando o movimento T é identificado
        lblContagem.Content = intCountT.ToString();
        try
        {
            if (intCountT == Convert.ToInt32(txtQtd.Text))
                lblOK.Visibility = Visibility.Visible;
            else
                lblOK.Visibility = Visibility.Hidden;
        }
```

```
            catch { }
        }

        private void kinect_AllFramesReady(object sender, AllFramesReadyEventArgs e)
        {
            byte[] imagem = ObterImagemSensorRGB(e.OpenColorImageFrame());
            //onde os sensores fica executando
            FuncoesProfundidade(e.OpenDepthImageFrame(), imagem, 2000);

            if (imagem != null)
            {
                canvasKinect.Background = new
ImageBrush(BitmapSource.Create(kinect.ColorStream.FrameWidth,
kinect.ColorStream.FrameHeight,
                                    96, 96, PixelFormats.Bgr32, null, imagem,
                                    kinect.ColorStream.FrameWidth *
kinect.ColorStream.FrameBytesPerPixel));

            }

            canvasKinect.Children.Clear();
            FuncoesEsqueletoUsuario(e.OpenSkeletonFrame());

        }

        private void FuncoesEsqueletoUsuario(SkeletonFrame quadro)
        {
            if (quadro == null) return;

            using (quadro)
            {
                //rastreando o esqueleto
                Skeleton esqueletoUsuario = quadro.ObterEsqueletoUsuario();

                    foreach (IRastreador rastreador in rastreadores)
                        rastreador.Rastrear(esqueletoUsuario);

                    if (btnEsqueleto.Background==Brushes.DarkGreen)
                        quadro.DesenharEsqueletoUsuario(kinect, canvasKinect);

            }
        }

        private byte[] ObterImagemSensorRGB(ColorImageFrame quadro)
        {
            if (quadro == null) return null;
            //obter imagem do ambiente
            using (quadro)
            {
                byte[] bytesImagem = new byte[quadro.PixelDataLength];
                quadro.CopyPixelDataTo(bytesImagem);

                return bytesImagem;
            }
        }

        private void FuncoesProfundidade(DepthImageFrame quadro, byte[] bytesImagem,
int distanciaMaxima)
        {
            if (quadro == null || bytesImagem == null) return;
```

```
            //analisando a profundidade
            using (quadro)
            {
                DepthImagePixel[] imagemProfundidade = new
DepthImagePixel[quadro.PixelDataLength];
                quadro.CopyDepthImagePixelDataTo(imagemProfundidade);

               // if (btnDesenhar.IsChecked)
                 ///   fluxoInteracao.ProcessDepth(imagemProfundidade,
quadro.Timestamp);
                 if (btnCinza.Background==Brushes.DarkBlue)
                    ReconhecerProfundidade(bytesImagem, distanciaMaxima,
imagemProfundidade);
            }
        }

        private void ReconhecerProfundidade(byte[] bytesImagem, int distanciaMaxima,
DepthImagePixel[] imagemProfundidade)
        {
        DepthImagePoint[] pontosImagemProfundidade = new DepthImagePoint[640 * 480];

kinect.CoordinateMapper.MapColorFrameToDepthFrame(kinect.ColorStream.Format,
kinect.DepthStream.Format, imagemProfundidade, pontosImagemProfundidade);
            //reconhecendo a profundidade
            for (int i = 0; i < pontosImagemProfundidade.Length; i++)
            {
                var point = pontosImagemProfundidade[i];
                if (point.Depth < distanciaMaxima && KinectSensor.IsKnownPoint(point))
                {
                    var pixelDataIndex = i * 4;

                    byte maiorValorCor = Math.Max(bytesImagem[pixelDataIndex],
Math.Max(bytesImagem[pixelDataIndex + 1], bytesImagem[pixelDataIndex + 2]));

                    bytesImagem[pixelDataIndex] = maiorValorCor;
                    bytesImagem[pixelDataIndex + 1] = maiorValorCor;
                    bytesImagem[pixelDataIndex + 2] = maiorValorCor;
                }
            }
        }

        private void slider_DragCompleted(object sender,
System.Windows.Controls.Primitives.DragCompletedEventArgs e)
        {
            //metodo para mudar o angulo do kinect
            kinect.ElevationAngle = Convert.ToInt32(slider.Value);
            lblAngulo.Content = kinect.ElevationAngle;
        }

        private void btnFecharClick(object sender, RoutedEventArgs e)
        {
            //fecha o sistema
            Application.Current.Shutdown();
            this.Close();
        }
`
        private void btnEsqueleto_Click(object sender, RoutedEventArgs e)
        {
            //executando o botao esquerdo
            if(btnEsqueleto.Background != Brushes.DarkGreen)
```

```
            btnEsqueleto.Background = Brushes.DarkGreen;
            else
                btnEsqueleto.Background = Brushes.DarkBlue;
        }

        private void button_Copy1_Click(object sender, RoutedEventArgs e)
        {
            //limpando a area de pontos
            lblContagem.Content = "0";

            intCountT = 0;
            lblOK.Visibility = Visibility.Hidden;
        }

        private void btnCinza_Click(object sender, RoutedEventArgs e)
        {
            //filtro de cinza
            if (btnCinza.Background != Brushes.DarkGreen)
                btnCinza.Background = Brushes.DarkGreen;
            else
                btnCinza.Background = Brushes.DarkBlue;
        }
    }
}
```