UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA - CT
DEPARTAMENTO DE ELETRÔNICA E COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO - CCC

DESENVOLVIMENTO DE UM APLICATIVO COM INTERFACES HUMANO-COMPUTADOR PARA UM SISTEMA DE REABILITAÇÃO MOTORA UTILIZANDO MICROSOFT KINECT

TRABALHO FINAL DE GRADUAÇÃO

Diego João Cargnin

Santa Maria, RS, Brasil
2012

DESENVOLVIMENTO DE UM APLICATIVO COM INTERFACES HUMANO-COMPUTADOR PARA UM SISTEMA DE REABILITAÇÃO MOTORA UTILIZANDO MICROSOFT KINECT

Por

Diego João Cargnin

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Eletrônica e Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação.**

Orientador: Prof. Dr. Marcos Cordeiro d'Ornellas (UFSM)

Trabalho de Graduação N.338

Santa Maria, RS, Brasil

2012

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UM APLICATIVO COM INTERFACES HUMANO-COMPUTADOR PARA UM SISTEMA DE REABILITAÇÃO MOTORA UTILIZANDO MICROSOFT KINECT

elaborado por
Diego João Cargnin

Como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

_____
Prof. Dr. Marcos Cordeiro D'Ornellas (UFSM)
(Presidente/Orientador)

_____
Profª. Drª. Giliane Bernardi (UFSM)

_____
Profª. Drª. Ana Lucia Cervi Prado (UFSM)

Santa Maria, 06 de julho de 2012

AGRADECIMENTOS

Dedico meus agradecimentos primeiramente a minha família que me apoiou e aconselhou neste período importante da minha vida. Ao meu pai Valmir, a minha mãe Jane e ao meu irmão Marcos. Aos meus amigos da graduação pelas risadas e momentos de alegria e jogos disputados no cc-craques, mas também pelos momentos de stress, estudos e trabalhos difíceis. Aos meus amigos do Wx3 parceiros de filmes, festas e jogatinas e aos grandes amigos do Ensino Médio, Bonito, Mosquito e São Nunca. Aos colegas da Animati com os quais aprendi muito no período em que estive lá. A todos os colaboradores do projeto de reabilitação motora da fisioterapia e computação em especial Dudu e Tuquinha.

“Não há nada mais trabalhoso que viver sem trabalhar!”
- RAMÓN VALDÉZ

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM APLICATIVO COM INTERFACES HUMANO-COMPUTADOR PARA UM SISTEMA DE REABILITAÇÃO MOTORA UTILIZANDO MICROSOFT KINECT

AUTOR: DIEGO JOÃO CARGNIN
ORIENTADOR: PROF. DR. MARCOS CORDEIRO D'ORNELLAS
Local e data da defesa final: Santa Maria, 06 de julho de 2012

Com o avanço da tecnologia de detecção de movimento, a utilização desta começa a ser explorada em mais áreas. Visando a utilização desta tecnologia para melhorar o tratamento fisioterapêutico de pacientes com dificuldades de movimentos, o LaCA em colaboração com o curso de Fisioterapia da UFSM está desenvolvendo um sistema capaz de apresentar jogos que estimulem e motivem estes pacientes a realizar os exercícios propostos. Estes jogos são desenvolvidos com base nos requisitos obtidos a partir da observação dos pacientes e das necessidades dos fisioterapeutas para que estes possam auxiliar na medição e avaliação dos tratamentos realizados.

O objetivo deste trabalho de conclusão é utilizar técnicas de engenharia de software empregando análise de requisitos e a modelagem de software para a concepção de um sistema de reabilitação motora utilizando interfaces humano-computador, projetando a interface de um menu principal que terá a função de encapsular os jogos criados para fisioterapia e a criação um jogo que será baseado em um teste de mesa. O sistema projetado visará a ludicidade e dinamismo além de auxiliar o trabalho dos fisioterapeutas.

Palavras-chave: Microsoft Kinect, Reabilitação Fisioterapêutica, Unity3D, interfaces, humano-computador

ABSTRACT

Trabalho de Graduação
Undergraduate Program in Computer Science
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM APLICATIVO COM INTERFACES HUMANO-COMPUTADOR PARA UM SISTEMA DE REABILITAÇÃO MOTORA UTILIZANDO MICROSOFT KINECT

AUTHOR: DIEGO JOÃO CARGNIN
ADVISOR: PROF. DR. MARCOS CORDEIRO D'ORNELLAS
Local e data da defesa final: Santa Maria, 06 de julho de 2012

With the advancement of technology in motion detection, its use begins to be explored in more areas. Aiming to use this technology to improve the physical therapy of patients with movement difficulties, the LaCA in collaboration with the course of physiotherapy of UFSM is developing a system capable of displaying games that stimulate and motivate these patients to perform the exercises. These games are developed based on the requirements derived from the observation of the needs of patients and therapists so that they can assist in measuring and evaluating treatments.

The objective of this conclusion work is using techniques of software engineering and requirements analysis using modeling software to design a system for motor rehabilitation using human-computer interfaces, designing the interface of a main menu that will serve to encapsulate the games designed for physiotherapy and creating a game that is based on a test table. The designed system will aim at playfulness and dynamism as well as assist the work of the physiotherapists.

Palavras-chave: Microsoft Kinect, Rehabilitation, Phisical Therapy, Unity3D, interfaces, Human-Interface

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

3D – Três Dimensões

9-HTP – Nine Hole Peg Test

ADM – Amplitude de Movimento Articular

API – Application Programming Interface

AVE – Acidente Vascular Encefálico

FPS – First Person Shooter

HUSM – Hospital Universitário de Santa Maria

LaCA – Laboratório de Computação Aplicada

RGB – Red, Green and Blue

SDK – Software Development Kit

SNC – Sistema Nervoso Central

VB – Visual Basic

VGA – Video Graphics Array

WPF – Windows Presentation Foundation

XAML – eXtended Aplication Markup Language

XML – Extensible Markup Language

ZDK – Zigfu Development Kit

SUMÁRIO

1 INTRODUÇÃO

A integração entre a computação e a área da saúde tem se tornado recorrente. O uso de computadores para realização e análise de exames tem se consolidado no auxílio ao profissional da saúde no diagnostico de doenças (Oak, 2011). Pacientes de fisioterapia, vítimas de acidentes, derrames e lesões são submetidos por vários meses a sessões repetitivas de exercícios para reabilitação, dependendo quase que totalmente do auxílio e acompanhamento do fisioterapeuta. Os exercícios devem ser supervisionados e avaliados regularmente, com o intuito de medir o progresso do paciente no tratamento. A ascensão de tecnologias de interação com o paciente através de sensores de detecção movimentos de baixo custo possibilita a criação de softwares de apoio ao tratamento fisioterapêutico.

Nesse sentido O LaCA em cooperação com o curso de fisioterapia da UFSM está desenvolvendo um projeto para a criação de um sistema de reabilitação utilizando o aparelho de reconhecimento de movimentos Microsoft Kinect que tem como um dos seus objetivo o desenvolvimento de um sistema de jogos de baixo custo financeiro e computacional que combine ludicidade aos pacientes e feedback ao fisioterapeuta.

Este trabalho propõe o desenvolvimento de um sistema que auxilie o fisioterapeuta na análise da qualidade e eficiência da sessão de exercícios e que estimule os pacientes, através do uso de um jogo simulador que utiliza medidas quantitativas para esta analise.

1.1 Objetivos

1.1.1 Objetivo Geral

Tem como objetivo o estudo de métodos para a concepção de uma interface que visa a facilidade e integração com o usuário, criando assim um programa capaz de ajudar o profissional da fisioterapia na avaliação da qualidade da execução dos exercícios pertinentes ao tratamento utilizando sistemas de detecção de movimento e interfaces naturais.

1.1.2 Objetivos específicos

- Utilização dos padrões de programação utilizados pelo Microsoft Kinect, Unity3d, Windows Presentation Foundation e suas aplicações.

- Criação de um menu principal que integrará o sistema.
- Estudo dos requisitos necessários para obtenção de dados confiáveis e relevantes, relativos aos exercícios fisioterapêuticos e a construção de uma interface adequada.
- Aplicação de interfaces naturais e gráficas no desenvolvimento de aplicações com Microsoft Kinect.
- Implementação de um módulo funcional para medição de dados como angulação, qualidade e amplitude de movimento dos exercícios propostos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capitulo são descritos os conceitos teóricos e os detalhes das ferramentas que fundamentam o desenvolvimento da aplicação. São apresentadas as definições do Microsoft Kinect, os detalhes da API para programas utilizados Unity3d bem como as características das linguagens utilizadas e dos padrões de modelagem.

2.1 Windows Presentation Foundation

O Windows Presentation Foundation ou simplesmente WPF é a tecnologia de apresentação da Microsoft que será utilizado no programa para a criação de uma interface que encapsulará os módulos de jogos a serem criados posteriormente e integrados ao programa. Foi escolhido, pois permite a separação da criação da interface e do código para que assim módulos sejam adicionados e o programa de banco de dados desenvolvido de forma independente da interface.

O WPF utiliza a linguagem de marcação XAML que nada mais é do que um arquivo XML com características especiais e um código para a plataforma .NET, que é um framework de software que pode usar algumas linguagens como VB.net, C#, entre outras.(MICROSOFT, 2012)

O arquivo XAML é gerado através de uma ferramenta de desenho visual (Figura 2.1), no caso o Visual Studio 2010. O arquivo gerado é relacionado diretamente com a classe do arquivo de código da linguagem escolhida podendo assim através dele realizar-se o que se deseja. O arquivo XAML contém as diretrizes da interface e as características que o ligam à classe respectiva no código.

Para construir no ambiente o desenvolvedor deve arrastar o componente desejado escolhido na lista para a tela onde se encontra o modelo de visualização da interface. Uma vez colocado o componente este pode então ser selecionado e adicionada uma função à uma ação que pode ser escolhida em um menu de ações. A ação escolhida é gerada automaticamente no código da classe do objeto. As propriedades do componente podem ser alteradas no modelo de visualização ou no próprio código XAML gerado.

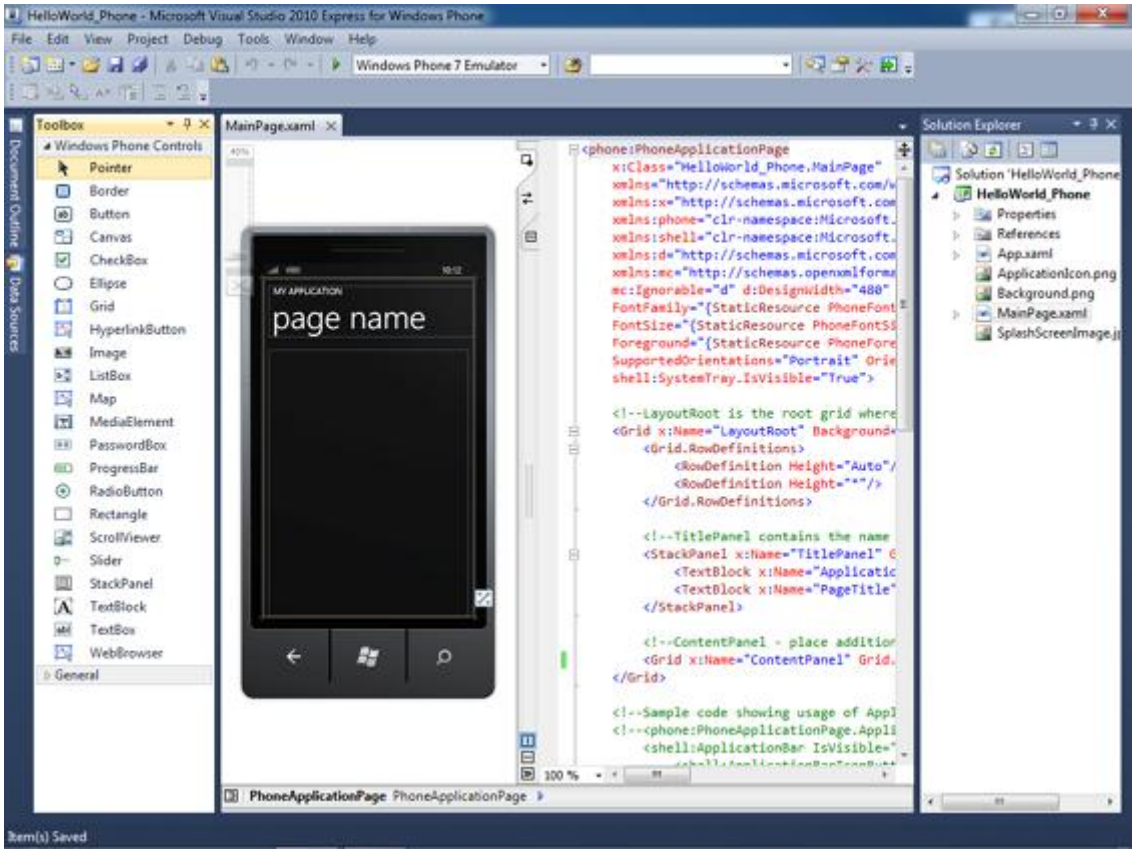

Figura 2.1: Exemplo do menu de criação de interfaces do WPF.

2.2 Microsoft Kinect

O Microsoft Kinect é um dispositivo de reconhecimento de movimentos criado pela Microsoft inicialmente para o videogame XBOX 360 e que teve o uso posteriormente liberado para computadores com o Microsoft Windows através do Kinect SDK. O objetivo do dispositivo é a eliminação de controles físicos, permitindo ao usuário a interação com a máquina por meio de gestos e fala.

2.2.1 Especificações técnicas do hardware

O Kinect conta com a tecnologia desenvolvida pela empresa israelense Primesense, especializada em visão computacional e câmeras 3d. O dispositivo funciona com sensores de cor e profundidade além de um projetor de infravermelho. Conta também com microfones que são utilizados na detecção de vozes e motor de inclinação para mover verticalmente o ângulo de visão do sensor (Figura 2.2). Os dados técnicos do aparelho são os seguintes (PRIMESENSE, 2010):

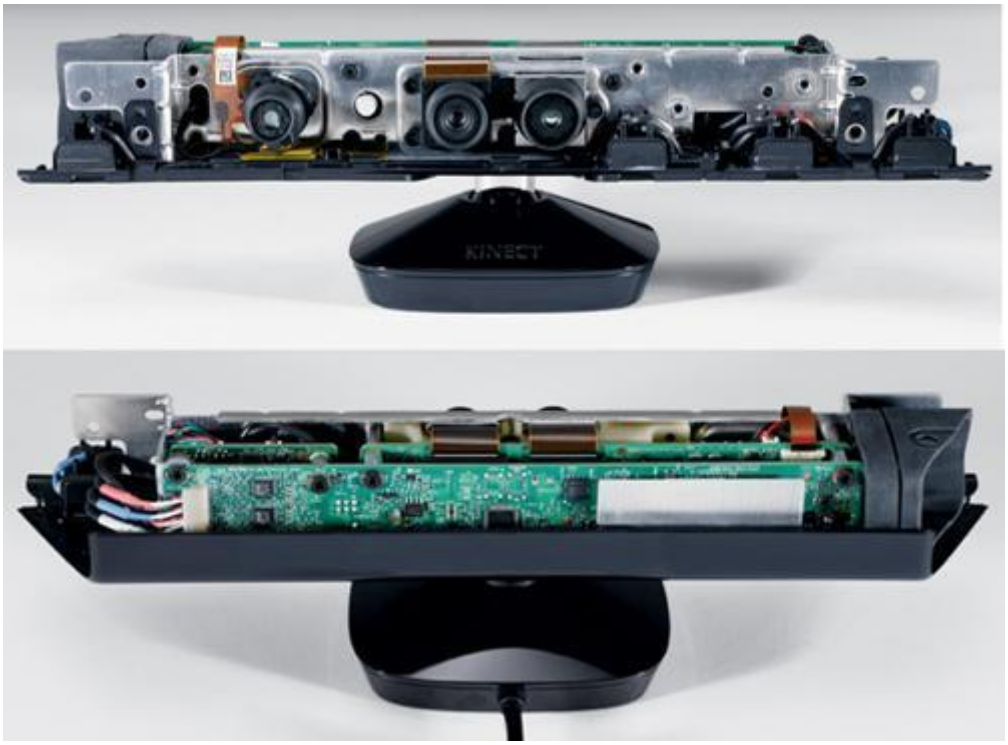

Figura 1.2: Microsoft Kinect aberto mostrando o hardware.

- Câmera de vídeo VGA - Esta câmara de vídeo auxilia no reconhecimento facial e outras características de detecção através da detecção das três cores básicas vermelho, verde e azul, assim sendo chamada de câmera RGB. Possui resolução de 640x480 pixels com uma velocidade de 30 frames/sec.
- Sensor de profundidade - Um projetor de raios infravermelhos e um sensor monocromático trabalham juntos para visualizar o ambiente independentemente das condições de iluminação. Possui uma resolução de 320x240 pixels e uma velocidade de 30 frames/sec.
- Microfones - Contém um conjunto de quatro microfones que podem isolar as vozes dos jogadores do barulho na sala. Isso permite que o jogador esteja a poucos metros de distância do microfone e ainda poder usar os controles de voz.
- Campo de visão – O Kinect tem um ângulo de visão horizontal de 57 graus e 43 graus vertical podendo o segundo inclinar 27 graus. Funciona também entre uma profundidade de 1,8 e 3,5 metros (Figura 2.3).

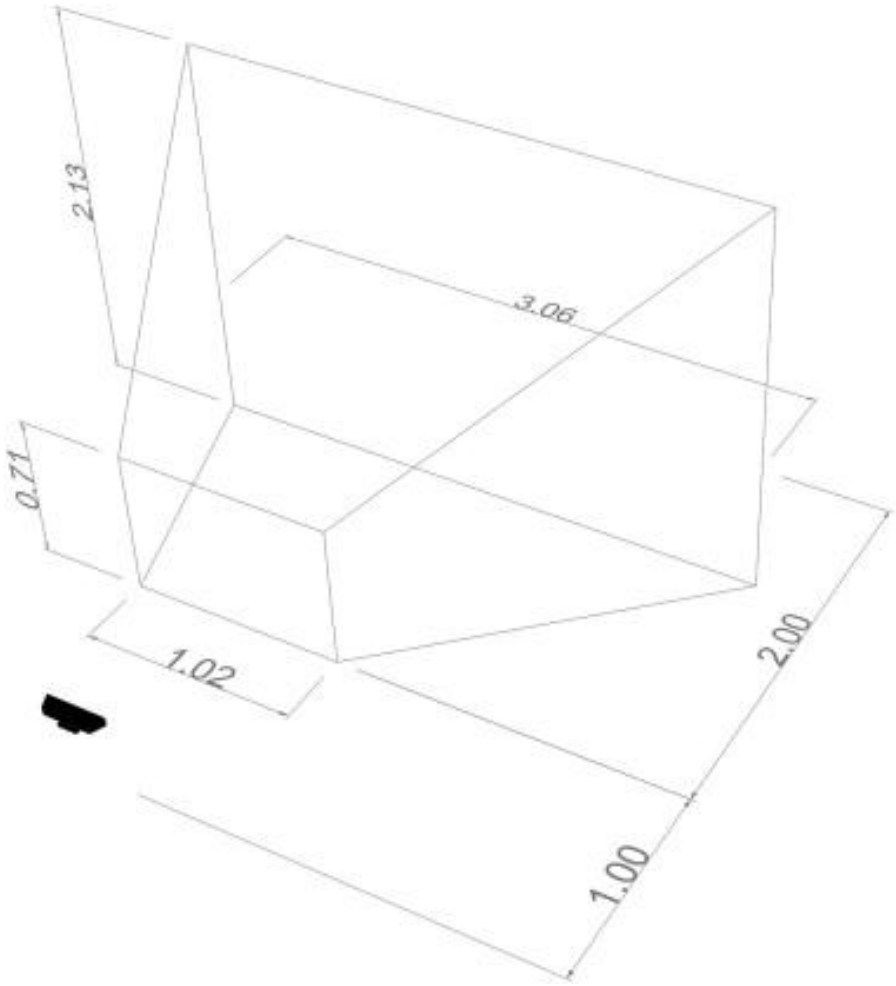

Figura 2.3: – campo de visão do Microsoft Kinect

2.3 Unity3d

Game engines ou *engines* gráficas(também conhecidas como motor de jogo) são o principal componente de software de um jogo ou outras aplicações 3D. Elas são responsáveis pela criação de objetos, por facilitar renderizações e a criação de efeitos especiais tais como fogo e fumaça, além de tratar das interações entre os objetos dos jogos. Algumas *engines* são consideradas *middlewares*, devido ao grande número de elementos inclusos no software que auxiliam e facilitam no desenvolvimento para jogos como físicas aprimoradas, sons e gráficos de ponta. Algumas *game engines* tem propósito voltado para um tipo de finalidade, tais como jogos FPS (atirador em primeira pessoa) que são especializadas em simular ambientes 3D para o uso neste tipo de jogos por exemplo que conta com *engines* como Unreal Engine e CryEngine 2 (WARD 2008). Para este projeto foi usada a *engine* Unity3d de propósito mais generalizado quando o assunto é a criação de jogos.

A *engine* gráfica Unity3d é uma ferramenta para o desenvolvimento de conteúdo 3D, criada pela empresa Unity Tecnologies. É usada para a criação de jogos, animações 3D entre outros, para múltiplas plataformas tais como Microsoft Windows, Mac, Xbox 360, Wii e de celulares. A Unity3d constitui-se em um editor de desenvolvimento e design e uma *engine* de jogos responsável pela integração dos objetos gerados e dos comandos do produto final.

A Unity3d aceita o uso de vários tipos de objetos 3d gerados através de outras ferramentas que serão usados no projeto, tais como modelos de software modelados no Blender, 3dMax e Maya. Além disso, a Unity3d também aceita a utilização de varias linguagens para a programação de scripts o que facilita a programação para os objetos gerados e permite seu uso para muitas plataformas. A interface visual da Unity3d permite que os objetos gerados possam ser manipulados através da interface (imagem x) de modo simples utilizando botões e boxes contendo as propriedades dos objetos.

A Unity3d é modelada através da orientação a objetos. Assim permite que cada objeto na cena tenha suas propriedades e ações mesmo sendo oriundo de um mesmo modelo, mostrando assim sua propriedade de heranças, bem como o encapsulamento de cada objeto que é visto quando há o uso de um mesmo script para vários objetos de um mesmo tipo porém sem influenciarem-se entre si. Cada objeto na cena pode ter varias propriedades que são definidas pelo programador, mas neste trabalho estão definidas apenas as utilizadas na criação deste sistema. (UNITY3D, 2012).

2.3.1 Componentes da Unity3d

A Unity3d utiliza o conceito de *scenes*(cenas) para a criação dos ambientes visualizados no jogo. Cada *scene* contém os obstáculos e objetos que formam o jogo. Podemos pensar numa *scene* como um nível de um jogo onde todos os inimigos e objetivos estão fixados e com suas funções preparadas. Na *scene* são posicionados todos os *GameObjects* com a especificação de suas propriedades, e toda a física e movimentação dos mesmos são preparadas pelos scripts adicionados.(UNITY3D, 2012).

GameObjects são todos os objetos que podem ser colocados na *scene*, como iluminação, componentes do jogo como cubos, chão e objetos de modelos prontos como armários e cadeiras por exemplo. A cada *GameObject* na scene são atribuídos vários componentes que nada mais são que propriedades especificas de cada *GameObject*.(Figura 2.4).

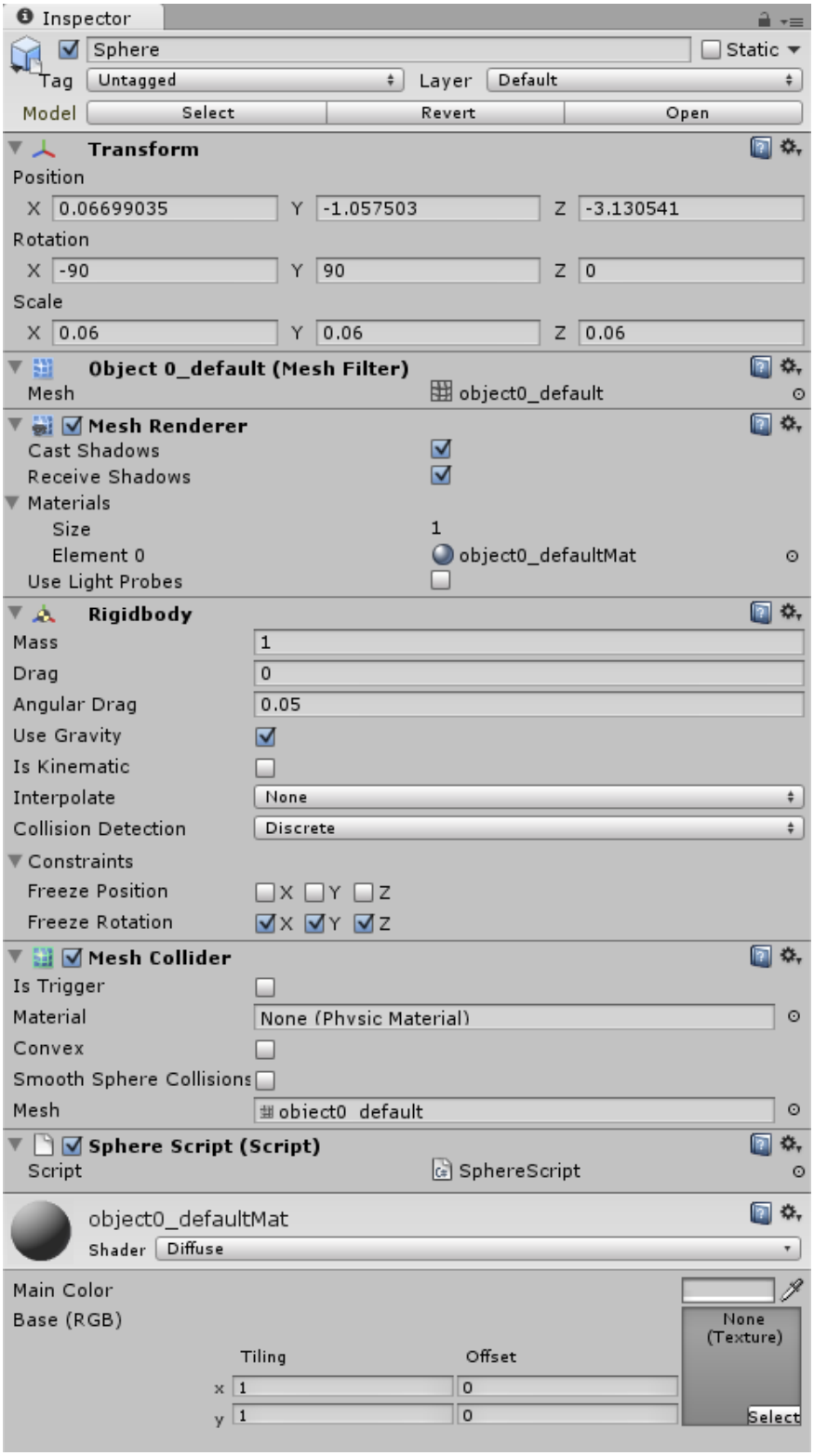

Figura 2.4: Componentes de um GameObject na Unity3d.

O componente *Transform* define o posicionamento dos objetos na tela. A partir dele é possível posicionar objetos na cena e movimentá-los através de botões ou em tempo real através dos scripts.(UNITY3D, 2012).

Mesh Renderer é o componente responsável pela representação do objeto em sua forma 3d, contendo todos os vértices e informações das texturas do objeto. A Unity3d não possui as ferramentas necessárias para a criação de modelos em 3 dimensões, porem ela possui integração com diversos programas de modelagem de objetos 3d tais como Blender, Maya, 3DS MAX e Modo. A Unity3d utiliza modelos do tipo .FBX, .3DS e .obj que devem ser obtidos através da conversão dos objetos dos programas de modelagem citados anteriormente.(UNITY3D, 2012).

O componente *Rigidbody* trata-se de um corpo fixo que é atribuído ao modelo 3d do *GameObject* que está aplicado a cena. Este corpo possui informações relevantes ao peso, movimentação e tamanho do objeto, proporcionando a este uma movimentação realística quando elementos como vento e gravidade são inseridos no ambiente. O *Rigidybody* contem o mesmo formato do *GameObject*. (UNITY3D, 2012).

O componente *Collider* trabalha em coordenação com o *Rigidbody* para aplicar colisões realistas entre os objetos da cena. Porém ele pode ser definido como qualquer tipo de tamanho ou forma ficando a cargo do programador escolher como ele pode interagir com os outros objetos. (UNITY3D, 2012).

2.3.2 Scripting

As iterações entre os objetos das cenas dos jogos acontecem através do motor de jogo presente na Unity3d que integra todos os elementos necessários para a execução do mesmo. O motor de jogo calcula automaticamente a física presente em cada objeto em particular, suas posições e efeitos, porém o comportamento destes objetos em relação a outros pode ser controlado e modificado através da funcionalidade de scripts que podem ser anexados a cada tipo de objeto como um tipo de componente assim como *Colliders* e *Rigidybodys*.

Cada script de objeto da Unity3d contém automaticamente as funções sobre o que deve acontecer ao objeto na sua inicialização ou a cada *update* de frame. Também podem ser adicionadas funções pré definidas pela Unity3d como do tipo *OnCollisionEnter()* que é ativada caso haja colisão do *Rigidbody* do objeto que contem este script com outro *Rigidybody* qualquer da cena. Os scripts podem ser usados para criar ou apagar objetos, além de mudar as propriedades dos mesmos de acordo com o comportamento definido.

Os scripts são muito importantes para este projeto, pois possibilitam a integração do Microsoft Kinect com a Unity3d, através do plugin Zigfu que contém todas as informações do esqueleto do usuário Kinect e possibilita a utilização do mesmo na forma de scripts editáveis que podem ser modificados de acordo com a necessidade do programador.

A unity3d permite que seus scripts sejam programados nas linguagens c#, Boo e javascript, possuindo uma comunidade ativa e crescente que suporta o estudo da linguagem e gera um repositório vasto de conhecimento a partir dos fóruns da Unity3d.

2.4 ZDK - Plugin Zigfu

Para utilizar o Kinect com a Unity 3D foi escolhido o ZDK, um plugin da empresa Zigfu, que é conhecido por integrar diversas plataformas de desenvolvimento ao Microsoft Kinect. A utilização deste Development Kit é justificada pela sua praticidade, pouco custo computacional e sua compatibilidade com a biblioteca OpenNI e com o SDK Microsoft Kinect. Para este trabalho, somente as funcionalidades referentes à SDK foram utilizadas.

O SDK é um kit de desenvolvimento da Microsoft para o aparelho Microsoft Kinect, que contém drivers, API, interface de dispositivos e documentação referentes a linguagens de programação nativas ao dispositivo. O SDK descreve com exemplos como utilizar os conceitos de cor, profundidade, esqueleto e reconhecimento de voz do kinect. Além disso, o SDK habilita o programador a acessar os sensores de áudio e de vídeo do aparelho para que novos programas possam ser criados. Contudo as aplicações criadas pelo SDK são feitas a partir de um aplicativo de programação como o Microsoft Visual Studio, tornando a criação de um jogo 3D muito complicada. Para isso a Unity3d foi utilizada em conjunto com o plugin ZDK.

O ZDK abstrai os comandos de inicialização do Kinect advindos do SDK para inicializar o dispositivo. Através de scripts ligados a um objeto da Unity 3D, o plugin gera o mapeamento do jogador no ambiente 3D (Figura 2.5). Com o posicionamento do usuário e as coordenadas 3D das suas articulações geradas pelo Kinect é possível realizar a atualização das posições de objetos no mundo 3D gerado pela Unity (Figura 2.6).

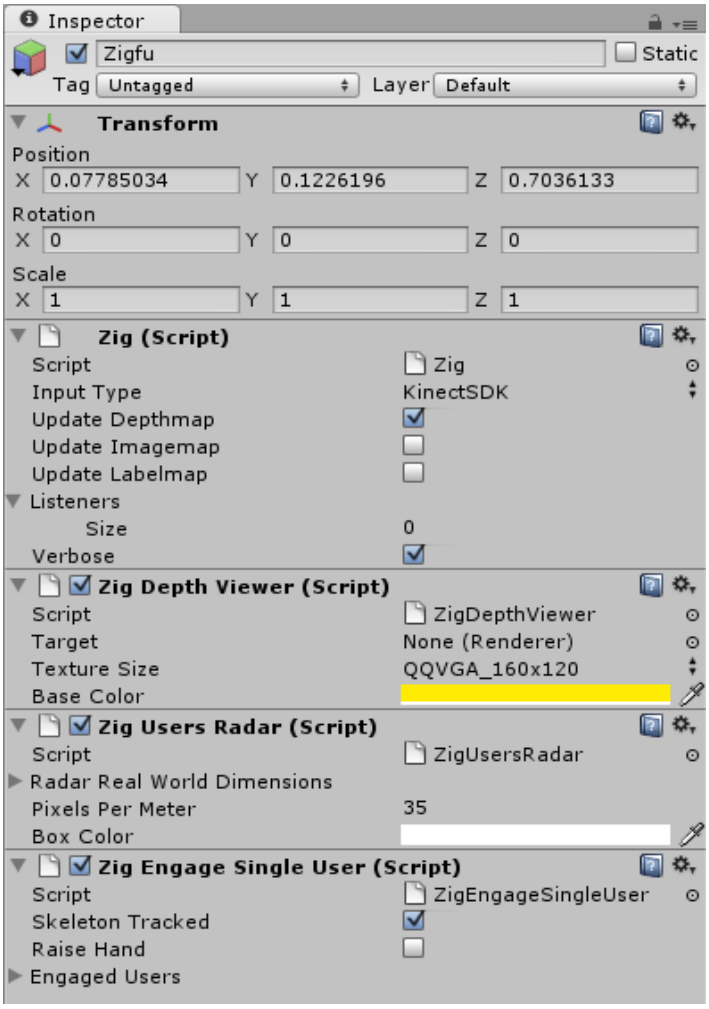

Figura 2.5: Script Zig ligado à um objeto Unity 3D.

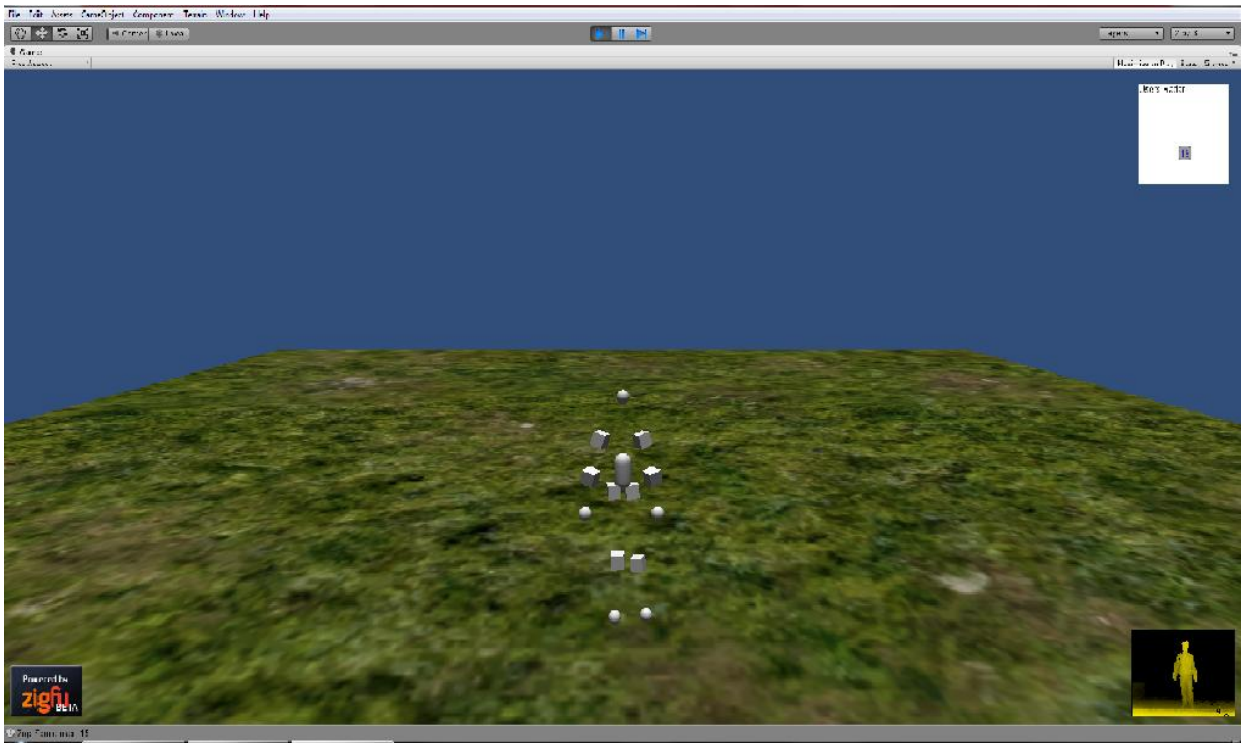

Figura 2.6: Disposição do esqueleto através das coordenadas 3D geradas pelo Kinect.

Relacionando às Figuras 1 e 2, é possível entender o funcionamento do ZDK. A Figura 1 mostra quatro scripts (*Zig, ZigDepthViewer, ZigUsersRadar e ZigEngageSingleUser*) ligados ao objeto, cada um com uma funcionalidade demonstrada em funcionamento pela Figura 2. As funções chamadas pelos scripts são script, visão da câmera de profundidade, criação de um radar de usuários (canto superior direito da tela) e modo monousuário. Cada script é chamado uma vez a cada atualização do Kinect (logo, 30 vezes por segundo) Percebe-se, então, a possibilidade do programador desenvolver ou customizar estes scripts, de acordo com os requisitos do aplicativo desejado. No sistema de reabilitação fisioterapêutica em questão, serão principalmente necessários os módulos de detecção do esqueleto para medição da interação entre as articulações. Através destas, será possível traçar vetores entre as articulações e com estes, calcular seu ângulo de abertura, informação que será posteriormente será enviada a um banco de dados e armazenada.

A grande vantagem da utilização da ZDK neste trabalho de graduação será a praticidade de sua instalação e manutenção, além da programação orientada ao ambiente Unity3D ser altamente simplificada, e várias funções de calibração e detecção (esqueleto, e quantidade de usuários) já estarem disponíveis para uso.

2.5 Linguagem C#

A C# é a linguagem de programação utilizada para na criação dos scripts da Unity3d e nas funções dos elementos criados no WPF. Foi concebida pela Microsoft como parte do projeto .NET Framework da Microsoft para ser a principal linguagem deste. Foi distribuída pela primeira vez em Julho de 2000. (ECMA, 2006).

A linguagem foi criada para ser simples, moderna, orientada a objetos e de propósito geral. Além disso, foi projetada para a criação tanto de programas pequenos como de grandes programas.

Mesmo sendo projetada para ser econômica em termos de memória e processamento, C# não compete diretamente com Assembly e C. A linguagem é parecida em termos de codificação com C++, porem diferencia-se desta por ter coletor de lixo (garbage collector), permitir uso de ponteiros em modo *unsafe* e como o Java, não permite herança múltipla, mas uma classe pode implementar várias interfaces abstratas. (ECMA, 2006).

Além disso, C# diferencia-se de outras linguagens por não incluir nenhum conjunto de classes de funções. Ao invés disso, C# utiliza as classe e funções vinculadas ao framework

.NET que são acessadas através de um conjunto de *namespaces* que agrupam as classes com funções similares, como *System.Collections* para estruturas de dados e *System.Windows.Forms* para o sistema Windows Form que foi utilizado neste projeto para a criação da interface inicial. (ECMA, 2006).

2.6 Acidente Vascular Encefálico

O acidente vascular encefálico (AVE) caracteriza-se por um distúrbio neurológico focal, que afeta partes específicas do corpo como o lado esquerdo do rosto ou o braço direito, ou às vezes global durando mais que 24 horas, com desenvolvimento rápido dos sintomas. Podem ser classificados como isquêmicos ou hemorrágicos. (LEITE, 2009). Resulta de alteração na irrigação sanguínea, no encéfalo, promovendo alterações das funções motoras, sensitivas, mentais, perceptivas e de linguagem. (MINUTOLI, 2007; GOMES, 2006).

2.6.1 Hemiplegia

A deficiência na mobilidade motora gerada muitas vezes pelo AVE, estabelece uma seqüela denominada hemiplegia que consiste em um estado físico caracterizado por uma paresia ou uma paralisia da metade de um corpo (hemicorpo), determinando um padrão de membro superior em flexão e abdução e membro inferior em extensão e abdução. Este padrão leva à incapacidade ou dificuldade em realizar diversas tarefas da vida diária que podem interromper atividades de extrema importância na realização pessoal. A hemiplegia torna os membros severamente fracos em um lado do corpo, e pode trazer consigo inúmeros problemas como dificuldade em se manter em pé ou andar, segurar ou pegar objetos, espasmos musculares, depressão, irritabilidade, entre outros. (GOMES, 2006). A Figura 2.7 mostra os tipos de plegias e as áreas afetadas.

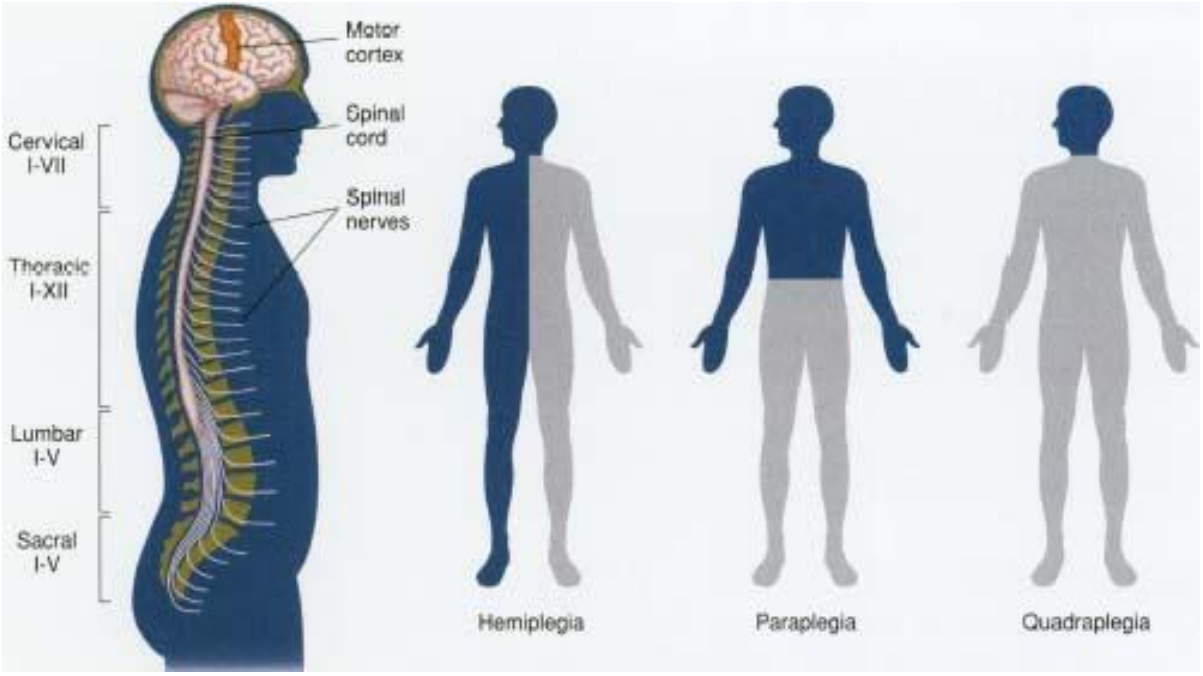

Figura 2.7: Diferentes formas de plegias.

2.6.2 Amplitude de Movimento Articular (ADM)

Segundo Batista (Batista, 2006), A medida da amplitude de movimento articular é um componente importante na avaliação física, pois identifica as limitações articulares, bem como permite aos profissionais acompanharem de modo quantitativo a eficácia das intervenções terapêuticas durante a reabilitação.

O instrumento mais utilizado para medir a ADM é o Goniômetro Universal (Figura 2.8). Consiste em duas barras que representam a parte fixa e móvel, além de um transferidor com uma escala de ângulos para a medição da ADM (Batista, 2006).

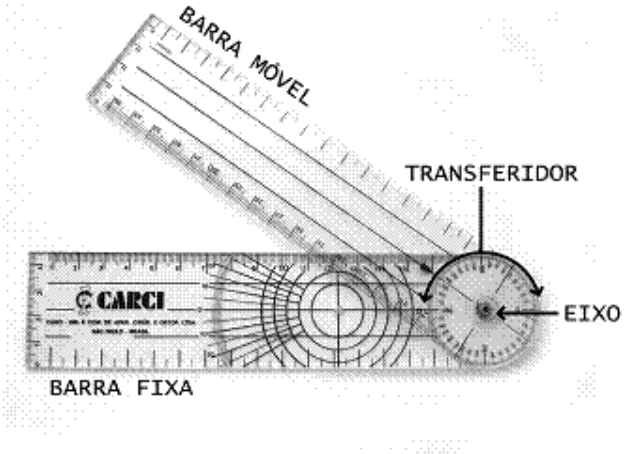

Figura 2.8: Goniômetro Universal

Exemplo de exercícios realizados na fisioterapia que inibem o padrão hemiplégico.

- Abdução do ombro:

O movimento ocorre no plano frontal. A abdução da articulação gleno-umeral é acompanhada por elevação clavicular, seguida por rotação lateral do úmero (Figura 2.9).

Amplitude articular: 0°- 170°/180°(JOÃO, 2011).

A Figura xxx mostra o movimento de abdução do braço esquerdo medido pelo instrumento chamado goniômetro.

Figura 2.9: Abdução gleno-umeral

- Flexão e extensão do cotovelo:

A articulação do cotovelo é uma articulação em dobradiça uniaxial. O movimento teste ocorre no plano sagital. O movimento de extensão é considerado o retorno da flexão (Figura 2.10).

Amplitude Articular: 0- 140°/145°(JOÃO, 2011).

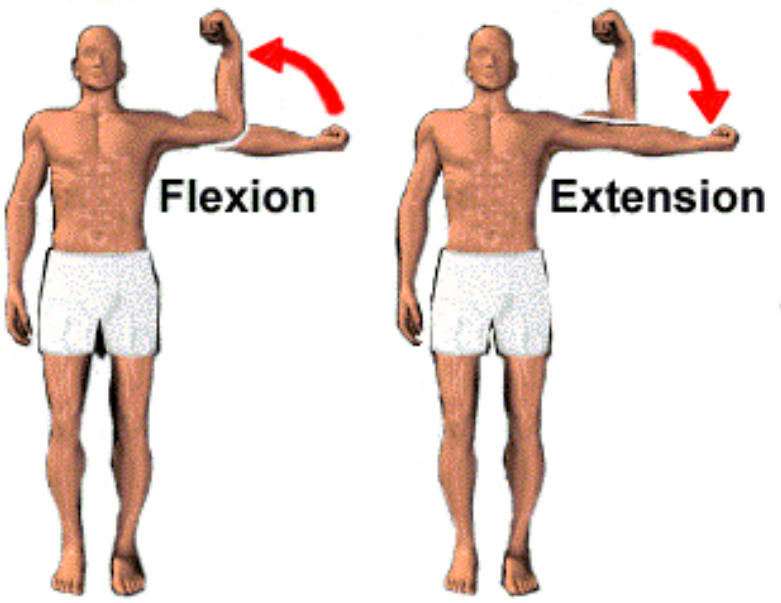

Figura 2.10: Movimento de Flexão do cotovelo

2.7 Nine Hole Peg Test (9-HTP)

O instrumento que serviu de molde para o jogo criado neste trabalho chama-se *Nine Hole Peg Test*. Segundo Lima (LIMA 2008) o jogo “nove pinos e nove buracos” é um instrumento de medição quantitativa do funcionamento dos membros superiores. O instrumento constitui-se por um tabuleiro contendo duas partes. A primeira parte do tabuleiro contem um recipiente que contém os nove pinos a serem utilizados no teste. A segunda parte à esquerda da primeira deve conter os nove buracos onde os pinos encaixam-se. O propósito é colocar os 9 pinos do recipiente, 1 por 1, em cada 1 dos buracos ao lado (Figura 2.11).

O jogo ocorre com quatro tentativas de completar o tabuleiro. Na primeira tentativa, o jogo teste ser feito com o braço dominante na forma de prática. Na segunda tentativa, ainda com o braço dominante, o teste requer que haja a medição do tempo em segundos. A medição ocorre a partir do momento em que o paciente toca o primeiro pino até o momento em que ele posiciona o último pino no ultimo buraco. A terceira e quarta tentativas são iguais a primeira e segunda tentativas respectivamente, apenas alterando-se o braço a ser testado. (MATHIOWETZ 1985).

Figura 2.11: Nine Hole Peg Test

3 MODELAGEM

Este capítulo descreve os passos para o desenvolvimento das interfaces implementadas, como os requisitos e a modelagem de dados, bem como a implementação da interface do jogo 3D e da interface do menu principal.

3.1 Levantamento de Requisitos

Com o fim de obter os dados necessários para a elaboração das interfaces desejada, a coleta dos dados deu-se por diferentes meios. Em visitas ao centro de reabilitação do HUSM da UFSM as informações foram obtidas a partir da visualização dos pacientes e também por entrevistas com professores a respeito do tratamento nas diversas especialidades da fisioterapia.

Para o primeiro módulo de jogos, foi proposta a criação de um jogo que estimulasse o movimento de abdução do braço de pacientes hemiplégicos, permitindo medir o grau de amplitude do movimento e também a velocidade de realização da tarefa. A análise quantitativa das medidas registradas permitirá ao fisioterapeuta avalias a evolução do quadro. O jogo, que neste trabalho será denominado “Jogo das estacas” consiste em mover objetos de um ponto inicial específico para um ponto final predeterminado para o objeto em uma caixa como descrito pelo modelo do instrumento de teste *Nine Hole Peg Test*.

Segundo Preece (2011), varias técnicas de coletas de dados podem ser utilizadas na tarefa de identificar os requisitos. Elas dependem do tipo de informação necessária e dos recursos disponíveis para obtê-las. Também se leva em consideração o envolvimento dos stakeholders, mas no caso deste projeto vamos considerar estes como os desenvolvedores e os professores do curso de Fisioterapia envolvidos. Para este projeto, no entanto quase todas as técnicas puderam ser utilizadas e tiveram um retorno positivo na aquisição dos requisitos com exceção dos questionários que não se mostraram necessários para a elaboração da interface uma vez que a presença dos stakeholders foi constante no desenvolvimento.

Primeiramente foi utilizada a técnica da observação, pela qual, muitos dados foram obtidos no começo do desenvolvimento a partir do acompanhamento dos pacientes no ambiente de tratamento e executando o exercício proposto. Após foi utilizada a técnicas de entrevista, na qual as bolsistas do projeto da área de fisioterapia esclareceram todas as dúvidas pertinentes ao funcionamento com o paciente. Os workshops, nos quais participaram os

professores e participantes do projeto que expressaram suas opiniões e deram dicas de como a interface poderia ser implementada, aconteceram durante todo o processo de concepção da interface. O estudo da documentação aconteceu pela leitura do material elaborado pelas bolsistas do curso de fisioterapia que descrevia as técnicas e os padrões utilizados pela fisioterapia no tratamento dos pacientes além da pesquisa sobre o funcionamento do jogo desenvolvido.

3.2 Modelagem do sistema

O projeto da interface principal, que encapsula os jogos gira em torno do projeto físico de um “totem” (Imagem 3.1), que faz parte do sistema em desenvolvimento. O totem é um móvel que abrigará uma televisão, um mini-computador com mouse e teclado para armazenamento do sistema implementado e de um aparelho Microsoft kinect. Para o projeto, dois totens ainda serão construídos. O primeiro totem deve ser colocado na área de exercícios fisioterápicos do HUSM para atender aos pacientes e o segundo será utilizado na sala do LaCA para testes dos módulos de jogos implementados posteriormente.

Figura 3.1: Modelo de um totem de visualizaçao de conteudo (MAXIMIDIA, 2009).

O totem será utilizado pelos fisioterapeutas para escolher jogos e escolher o paciente, além de ser usado pelos próprios pacientes para executar os exercícios com base nestes jogos. Baseado então neste totem, o sistema é projetado para ser de fácil entendimento e abrigar as seguintes funcionalidades (Figura 3.2):

- Fazer login/logoff do fisioterapeuta que utilizará o dispositivo.
- Buscar pacientes em um banco de dados
- Escolher paciente que realizará exercícios
- Escolher jogos e/ou exercícios a serem realizados pelo paciente

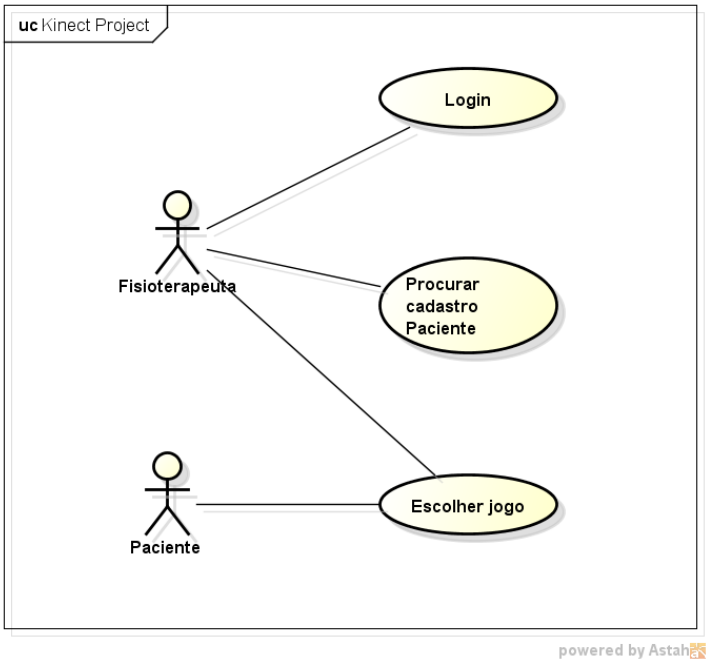

Figura 3.2: Diagrama de casos de uso do sistema principal

Além disso, o sistema deve fornecer informações existentes dos pacientes para os jogos desenvolvidos para que uma analise de desempenho e qualidade possa ser realizada posteriormente durante o jogo com base nos dados coletados. Estas informações devem poder ser armazenadas no servidor para futura análise através de um site que será desenvolvido para a aplicação. A finalidade do totem é ser um instrumento de auxílio aos fisioterapeutas para o tratamento dos pacientes.

3.2.1 Desenvolvimento do sistema

A interface principal do sistema abrigará o jogo criado neste projeto, além de outros possíveis jogos que serão desenvolvidos posteriormente. Sua função é interagir como um intermediador entre o banco de dados e os jogos, buscando as informações dos pacientes e

repassando para os jogos. A interface principal é importante, pois cada jogo desenvolvido na Unity3d é criado na forma *stand-alone* e contém apenas as especificações deste jogo. Ela permite assim que cada jogo posterior seja criado como se desejar, sem precisar alterar os que já estão integrados ao sistema e ao banco de dados e permitindo também que apenas as informações relevantes ao jogo selecionado sejam buscadas no banco de dados.

Utilizando WPF com o *Microsoft Visual Studio* a interface principal foi criada seguindo as especificações do totem e requisitos obtidos. Primeiramente foi criada a tela de login que contém os componentes *TextBox* e *Button*, após digitar o nome no campo e clicar em entrar, o programa verifica se o usuário existe no banco de dados do sistema. Se o usuário existe a próxima tela aparece também com um *TextBox* mas que agora serve para buscar o nome do paciente no banco de dados além de conter um botão de Logoff para o usuário poder desconectar-se caso necessário. O usuário do programa então deve digitar o nome do paciente na *TextBox* e este é procurado no banco de dados. Quando encontrado, suas informações são carregadas e o usuário pode escolher o jogo a ser utilizado pelo paciente ou ainda o vídeo de explicação que servirá como um tutorial para ver o funcionamento do jogo. O posicionamento dos elementos na tela foi realizado utilizando a ferramenta de visualização do *preview*. Através do componente *Toolbox*, é possível arrastar para a tela o elemento desejado

O jogo é escolhido quando o usuário clica em cima do botão com a imagem e nome do jogo que deseja acessar (Figura 3.3).

```
private void Start_app1(int patient_ID)
       //inicia app da tela equivalente
        {
            string argument = Convert.ToString(patient_ID);
            Process p = new Process();
            p.StartInfo.UseShellExecute = false;
            p.StartInfo.RedirectStandardOutput = true;
            p.StartInfo.FileName = "C:/Users/Kinect01/Desktop/pasta_teste/teste1.exe";
            p.StartInfo.Arguments = argument;
            p.Start();
        }
```

Figura 3.3: Função que inicia cada jogo.

Neste projeto a conexão com o banco de dados acontece apenas como caráter de teste, uma vez que uma database grande e estruturada será criada para o projeto final, contendo as informações de vários pacientes. Porém, o acesso ao banco foi implementado para buscar as informações referentes ao jogo desenvolvido neste projeto e mostrar o funcionamento da interface e do jogo. Utilizando um banco em MySQL com apenas uma tabela, as informações utilizadas foram o nome, os tempos e ângulos de um paciente fictício.

As interações com o banco de dados para a requisição de informações estão demonstradas na figura 3.4.

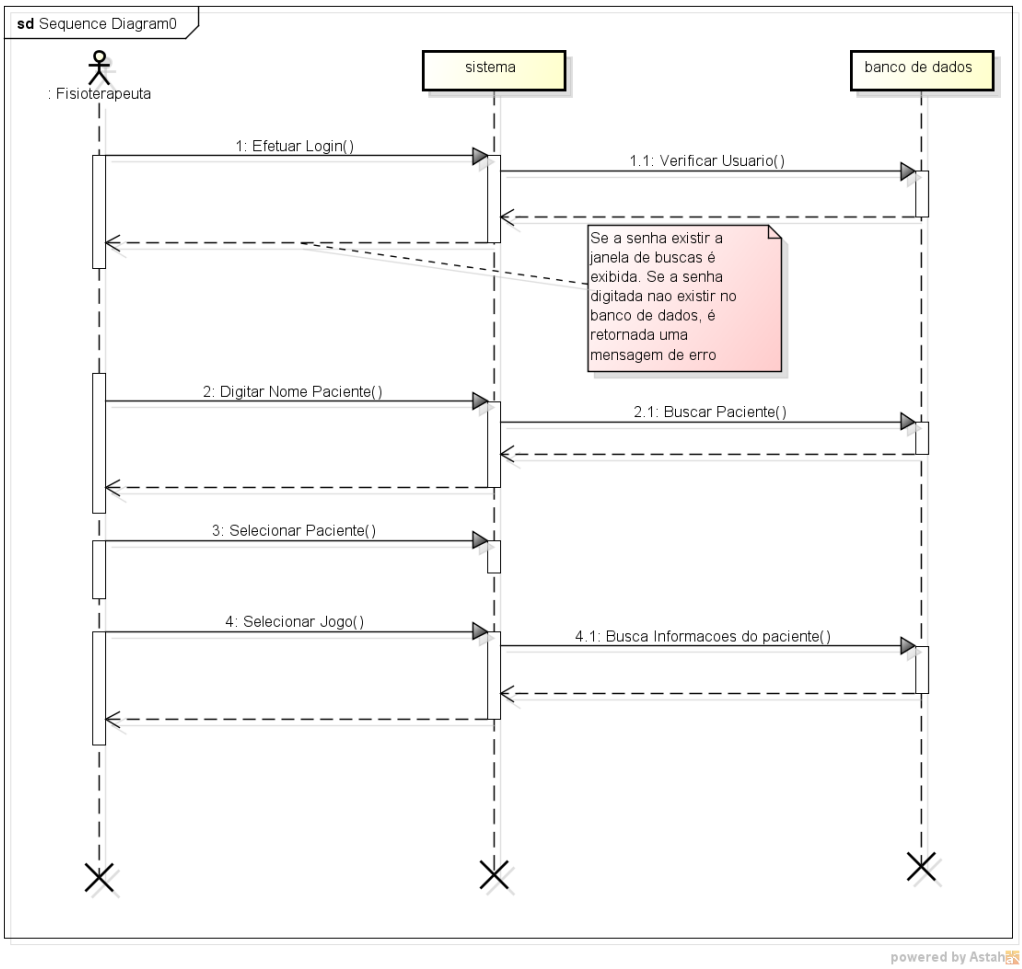

Figura 3.4: Diagrama de seqüência da interface principal.

3.3 Modelagem do Jogo das Estacas

O aparelho de teste *Nine Hole Peg Test* foi utilizado como base para o desenvolvimento de um jogo 3D denominado aqui do Jogo das Estacas. Este jogo tem como objetivo apoiar os fisioterapeutas nas atividades de recuperação dos pacientes.

A motivação ao aplicar-se a realidade virtual de jogos na reabilitação de pacientes com problemas como a hemiplegia, encontra-se na necessidade de exercícios intensos e repetitivos. A realidade virtual é ideal para a criação de um ambiente atraente para o paciente desenvolver o exercício sem se entediar ou perder logo o interesse. Além disso, a reabilitação utilizando a realidade virtual permite um bom controle sobre os exercícios e a capacidade de se obter informações e medições confiáveis sobre os exercícios realizados. (BOIAN, 2002).

A iteração do usuário com o jogo foi feita utilizando o dispositivo Microsoft Kinect que reconhece os movimentos dos pacientes, para realizar o exercício no jogo e para as medições necessárias, no caso a angulação da abdução.

3.3.1 Requisitos do Jogo baseado no teste *Nine Hole Peg Test*:

- Objeto móvel que deve seguir a mão do paciente;
- Lugar de criação do objeto móvel
- Mão virtual na tela de fácil visualização;
- Nove objetos-caixa (dropoffs) a serem depositados os objetos;
- Lugar para criar objetos a serem depositados nos nove objetos-caixa;
- Deve calcular o tempo levado para levar todos os nove objetos a fim de completar a caixa, calcular o tempo de levar todos os objetos de volta para o lugar inicial alem do tempo total das duas fases;
- Deve calcular o ângulo máximo de abdução do braço usado para mover os objetos durante cada vez na fase de levar o objeto até o dropoff e na fase de levá-los até o ponto inicial.
- O jogo deve mostrar os resultados obtidos em uma tela final
- Deve ter opção de troca de braço principal
- Ter opção de distancia
- O sistema deve então armazenar os tempos calculados e os ângulos obtidos em um banco de dados;
- Paciente não pode se deslocar do ponto ideal durante o jogo;
- Ambiente amplo sem interferência de terceiros perante o aparelho;
- Usar roupas justas de preferência (roupa colada ao corpo);
- O sistema necessita ser simples, para que os novos usuários possam utilizá-lo imediatamente e que os freqüentes lembrem como fazê-lo. Precisa ser eficiente e estar apto a lidar facilmente com os erros dos usuários;

Para este jogo um cenário foi elaborado (figura 3.5). Os cenários consistem em uma descrição narrativa informal que descrevem o que se espera do programa e deve ser feito e permite aos usuários participar do processo de desenvolvimento do software.

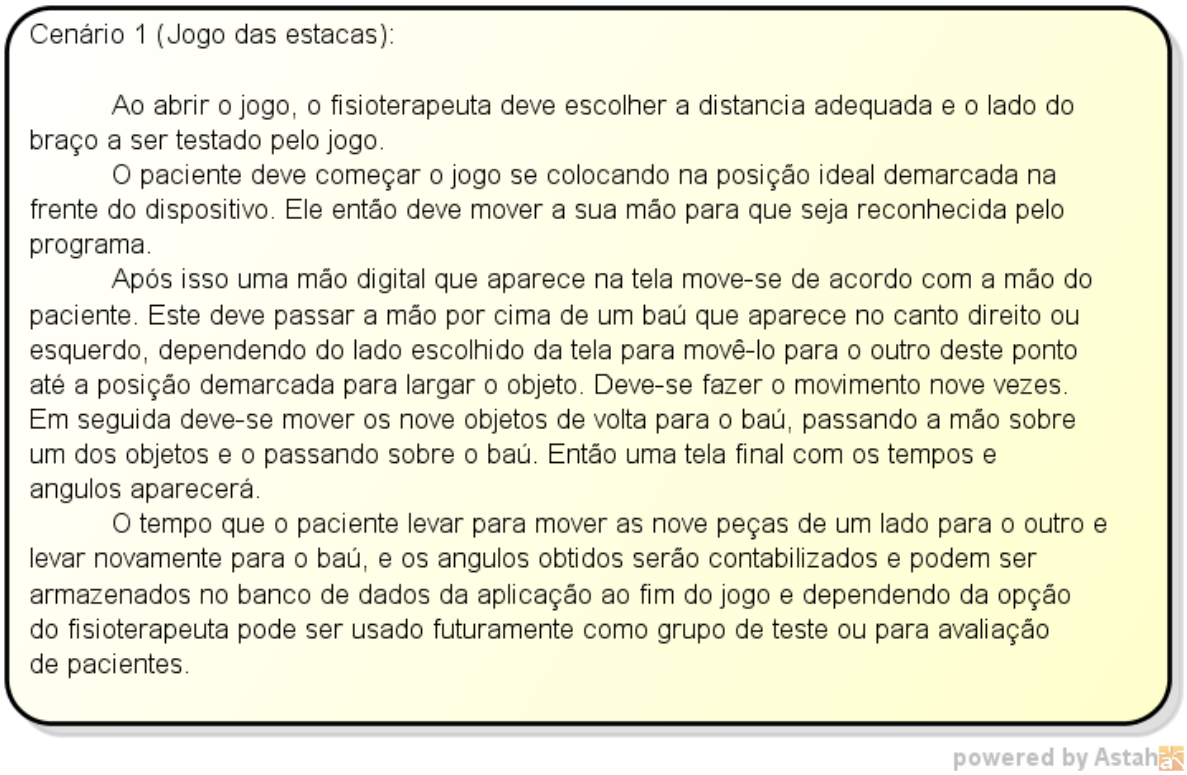

Figura 3.5: Cenário do jogo das estacas

3.3.2 Desenvolvimento do Jogo

Para começar o desenvolvimento do jogo, um protótipo de baixa fidelidade foi criado para tornar a aquisição de requisitos e a modelagem do jogo mais fácil. Este protótipo foi realizado seguindo um modelo de desenvolvimento interativo com ciclo de vida simples, onde um modelo era criado e avaliado a cada ciclo, então os novos requisitos eram adicionados e os problemas sanados. (Figura 3.6).

Este processo se repetiu até que o protótipo estivesse razoavelmente pronto para ganhar forma na Unity3d. Este conceito de utilização de um protótipo de baixa fidelidade juntamente com o ciclo de vida simples, foi utilizado, pois torna o custo de desenvolvimento baixo e facilita a obtenção de requisitos (PREECE, 2011). Este protótipo foi construído com a plataforma WPF e utilizando a linguagem C#, a mesma a ser utilizada na criação dos scripts da Unity3d, assim códigos criados como a angulação dos membros superiores poderiam ser reutilizados com uma devida adaptação. (Figura 3.7).

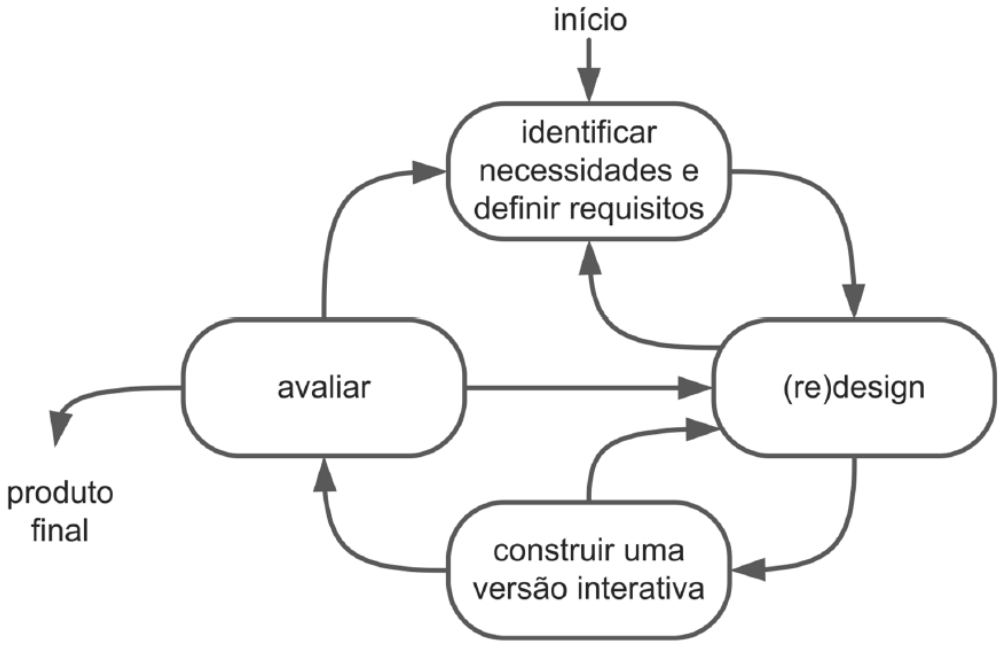

Figure 3.6: Ciclo de vida simples.

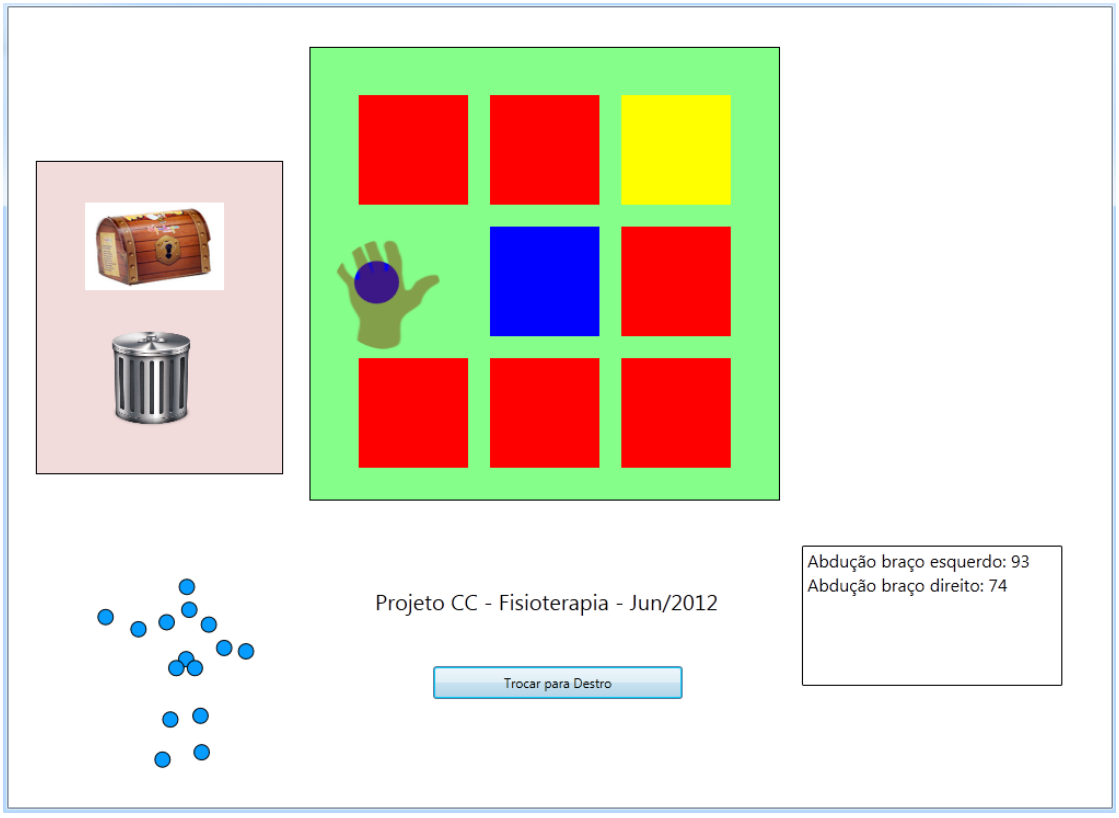

Figura 3.7: Imagem do protótipo de baixa fidelidade.

Após a aquisição de informações suficientes o próximo passo foi à criação de um protótipo de alta fidelidade incremental que já possuía todas as características necessárias do jogo. Utilizando a Unity3d para a criação de um modelo em um mundo tri dimensional que representava o modelo construído. O jogo foi construído levando tomando como base as três partes do jogo que são levar os objetos de um baú até um armário, devolver os objetos ao baú, e mostrar a tela de resultados.

O primeiro passo na construção do jogo foi a obtenção de modelos dos objetos e texturas para a construção de um cenário amigável. Todos os modelos utilizados no jogo foram obtidos na internet de forma gratuita, seguindo a licença de utilização dos mesmos. Muitos destes objetos foram criados para o Blender, que é um programa de modelagem de objetos, da forma .blend. Assim uma conversão para o modelo padrão .FBX, utilizado pela Unity3d, foi necessária. Com os objetos no formato correto eles foram adicionados ao projeto pela sua adição na pasta *Assets* do projeto. Cada objeto adicionado no projeto gera um *prefab* na tela de exibição dos elementos adicionados no projeto. Cada *prefab* representa um objeto pré-fabricado, contendo a Mesh_Renderer do objeto, que é o modelo do objeto que vai ser colocado na tela. Cada novo *prefab* pode ser adicionado ao jogo arrastando-o para a tela de hierarquia criando um novo *gameObject*.

Após a aquisição de todos os modelos, a próxima etapa foi colocar os objetos na cena em uma posição inicial. O jogo foi criado utilizando apenas uma cena por se tratar de um jogo simples. Cada novo objeto contém o script que possui a classe criada para controlar as iterações com outros e também o seu comportamento na fase. Cada script representa uma classe para o controle dos objetos, programados em C#. A programação das classes dos scripts é realizada utilizando-se o *MonoDevelop*, que é uma ferramenta embutida na Unity3d que funciona como um editor específico para os códigos dos projetos da Unity3d.

O objeto Zigfu, é mantido na cena do jogo para conter o script *Zig*, do plugin Zigfu, que é responsável por carregar as funções da SDK do Kinect e tornar disponíveis para a sua utilização pelos scripts aqui criados. Alguns scripts de detecção de movimento são disponibilizados pelo Zigfu com tipos diferenciados de detecção. O escolhido foi o script *ZigEngageSinlgeUse* que detecta o movimento de um usuário na frente do dispositivo e lhe atribui as propriedades do corpo virtual. Também são adicionado ao objeto Zigfu, os scripts *ZigDepthViewer*, que mostra no canto da tela a visão do radar de profundidade do Kinect e o ZigUserRadar, que mostra um radar vertical do posicionamento dos usuários detectados pelo Kinect.

O objeto principal do jogo é o objeto *FollowHandPoint.* Nele está incluído o script *ZigSkeleton* que é responsável pelo posicionamento dos objetos em relação a parte do corpo que o representa no Microsoft Kinect. Ele pode conter todas as partes do corpo, que são os pontos de controle do Kinect. Cada ponto de controle representa uma junta do corpo de uma pessoa real presente no campo de visão do Kinect. No script, cada junta é representada por uma variável do tipo *gameObject*, que tem a posição do seu *transform* atualizada a cada atualização da tela que acontece em média 60 vezes por segundo. O componente *transform* de cada objeto representa apenas a sua posição no espaço. No jogo o objeto *ActiveHand*, representa a mão ativa do paciente e é posicionada de acordo com os movimentos do mesmo. O cálculo da posição da junta no espaço tridimensional é abstraído pela utilização do plugin Zigfu, porém podemos utilizá-la e modificá-la como desejarmos. Neste jogo, o *ZigSqueleton* foi modificado para operar em apenas duas dimensões, facilitando a visualização da "mão" virtual e das operações com a mesma. Foi adicionada ao script a função de medição do ângulo de abdução do braço utilizado no exercício.

Para a medição do ângulo do braço utilizado, o cálculo da medida do ângulo leva em consideração um triangulo formado pela mão, ombro e lado da cintura. O ombro e o lado da cintura formam uma reta assim como o ombro e a mão formam outro. O ângulo que aparece na tela é calculado pelo ângulo entre essas duas retas. A 3.8 mostra as funções com o cálculo da angulação que é obtido através do arco-cosseno dos dois vetores.

```
float angulo(Vector3 vetor,Vector3 vetor2)
{
     //normalizacao dos vetores

      double norma = (Math.Sqrt(Math.Pow(vetor.x,2)+ Math.Pow(vetor.y,2)));
      vetor.x = (float)(vetor.x/norma);
      vetor.y = (float)(vetor.y/norma);

      norma = (Math.Sqrt(Math.Pow(vetor2.x,2)+ Math.Pow(vetor2.y,2)));
      vetor2.x = (float)(vetor2.x/norma);
      vetor2.y = (float)(vetor2.y/norma);

      // calcula o angulo das articulacoes

      float  ang = (float)((Math.Acos(vetor.x * vetor2.x + vetor.y * vetor2.y)*(180/Math.PI)));
      return ang;

}
float goniometria(int id_exercicio, int lado_avaliado)
{
    |
    float angulo_retorno=0;
    switch (lado_avaliado)
    {
    case 0:
        switch (id_exercicio)
        {
        case 0:
            angulo_retorno=angulo(gera_vetor(transforms[(int)ZigJointId.LeftShoulder],transforms[(int)ZigJointId.LeftHip]),
                gera_vetor(transforms[(int)ZigJointId.LeftShoulder],transforms[(int)ZigJointId.LeftWrist]));

        break;
```

Figura 3.8: Função que implementa o ângulo do abdução.

O ângulo obtido é utilizado pelo script *AngleTextScript* que guarda apenas o ângulo máximo obtido a cada vez que o movimento de abdução acontece. Os ângulos obtidos são salvos no *array string[] angulosArray* e são visualizados durante o jogo em um monitor virtual que mostra o ângulo atual máximo e também posteriormente na tela final do jogo onde todos os ângulos máximos são visualizados. O monitor virtual também mostra o tempo atual e a distancia do baú apenas para referencia do fisioterapeuta. Adicional a classe *AngleScript* no monitor virtual, a classe *timer* é responsável por salvar o tempo decorrido em cada fase.

O baú é controlado pelo script *VaultBoxBehaviour*, que controla a criação de novos objetos, que seriam os pinos do *Nine Hole Peg Test*. A criação destes objetos depende da colisão do objeto *ActiveHand,* que é a mão ativa, com o baú. Quando o componente *Rigidbody*, que representa o corpo físico, do objeto *ActiveHand* e do baú se encontram, a função *OnCollisionEnter*() é ativada e um objeto *Crate* é gerado. Este tem o mesmo posicionamento da *ActiveHand.* O objeto crate deixa de seguir a *ActiveHand* quando ocorre a colisão com um dos nove objetos *EmptySpot* que representam os buracos vazios como no *Nine Hole Peg Test*, posicionados em um armário virtual. Quando todos os nove objetos estiverem colocados no armário, que contem os *EmptySpots*, o jogo passa para a nova fase onde todos os objetos devem ser removidos e colocados novamente no baú. Novamente utilizando a colisão entre os objetos *ActiveHand*, com os objetos *Crate* colocados no armário, leva o objeto a se movimentar junto com a *ActiveHand* até que haja a colisão entre *Crate* e o baú, onde *Crate* é deletado. A 3.9 mostra a criação de um novo *Crate* quando há colisão.

```
void OnCollisionEnter(Collision collision) {

    if(GameStageTwoSpawner == false){
        if(collision.gameObject.name == "ActiveHand")
        {
            //cria o objeto Crate
            myCube = (GameObject)Instantiate(Resources.Load("Crate"));
            //define mesma posicao da ActiveHand
            myCube.transform.position = target.transform.position;
            BoxCoverTrigger = true;
```

Figura 3.9: Função da detecção de colisão.

A classe *GameBehaviour*, controla o menu virtual e as interações com o mouse. O menu contém as opções de distancia do baú em relação ao armário e do lado escolhido para o exercício. A ativação do menu é tratada como a colisão de um raio fictício criado pelo clique

do mouse com um botão do menu. Este raio é criado utilizando o recurso de *raycast* disponível na Unity3d como uma propriedade física. Ele atinge qualquer objeto que esteja na frente deste raio e ele se move apenas no eixo z da profundidade. Quando a opção de troca de lado é ativada, o objeto baú e o monitor virtual são trocados de lado para se adequar a utilização do usuário canhoto ou destro. O componente *ActiveHand,* bem como o objeto correspondente na cintura e ombro são trocados de lado na classe *ZigSkeleton* para que o calculo do ângulo de abdução não se comprometa.

A iteração entre os objetos da cena ocorre através do nome que o objeto possui. Os objetos considerados alvos, são acessados através da função *Find(),* presente no gameObject, na qual o nome do objeto a ser acessado é passado como parâmetro na forma de *string*. Assim permite-se que os componentes de outros objetos possam ser controlados pelo script de outras classes.

A tela final da aplicação é acessada apenas quando o exercício é completado. Nela as informações de todos os ângulos e tempos são acessadas e mostradas para o fisioterapeuta que tem a possibilidade de salvá-las no banco de dados como uma nova entrada. As informações carregadas pela interface principal do sistema são disponibilizadas através de um arquivo texto, criado anteriormente com as informações pertinentes a este jogo e são utilizados para a comparação com os novos resultados obtidos por parte do fisioterapeuta.

4 EXEMPLO DE USO DO JOGO

Este capítulo mostra telas das tarefas de como seria o uso do jogo por um usuário.

4.1 Login, Busca e opções

O usuário começa a iteração com a aplicação acessando o totem, que estará em stand-by. No totem, a primeira tela que aparece é a de login, na qual o fisioterapeuta digitará a sua senha. Após o Login a tela de busca de pacientes é exibida onde o fisioterapeuta deve digitar o nome do paciente a ser avaliado pelo jogo escolhido. O programa então busca pelo nome do paciente digitado e exibe na janela abaixo da busca. Em seguida quando o fisioterapeuta seleciona o paciente com 2 cliques sobre seu nome, a terceira é exibida. Nela, os botões para selecionar o jogo ou exibir o vídeo tutorial aparecem. As figuras 4.1 e 4.2 mostram as telas como apresentado.

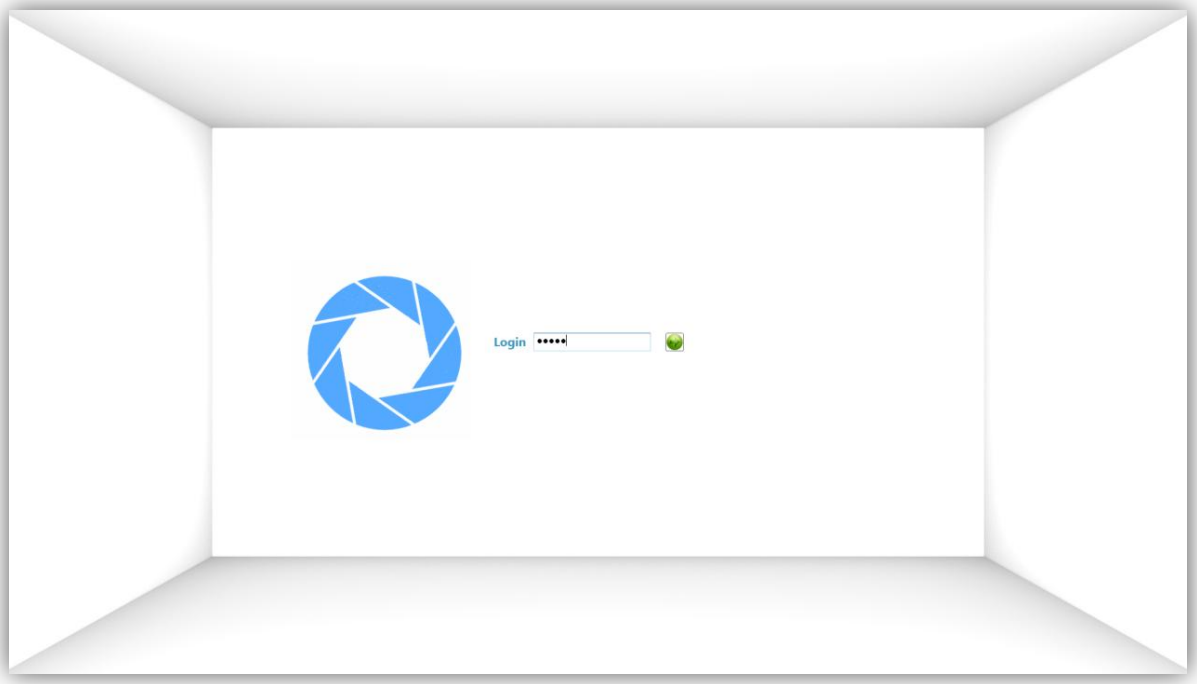

Figure 4.1: Tela de Login da interface

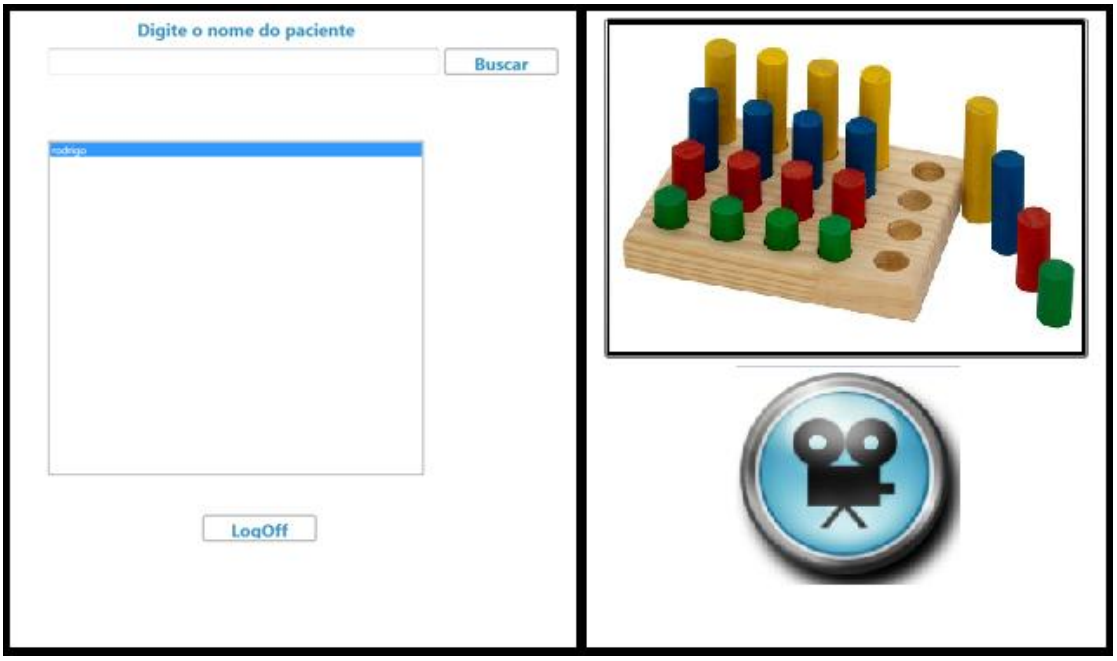

Figure 4.2: Telas de busca e escolha do jogo.

4.2 Exemplo de execução do jogo das Estacas

Após clicar no botão do jogo, a Unity3d inicia o *stand-alone* e o jogo começa como pode ser visto na figura, o paciente é reconhecido e mostrado pelo user radar que está no canto direito superior da tela. O tempo e ângulo são marcados no monitor em tempo real. O menu pode ser acessado pelo fisioterapeuta para decidir o lado do exercício, a distância do baú além de ser necessária a iteração com o botão continuar para que comece a segunda fase do jogo. Após o término do jogo a ultima tela mostra os resultados antigos obtidos pelo paciente no jogo através do banco de dados e também mostra todos os ângulos obtidos pela novo tentativa no jogo. As figuras 4.3 e 4.4 mostram um exemplo de tela do jogo e dos resultados respectivamente

Figure 4.3: Tela do jogo das estacas

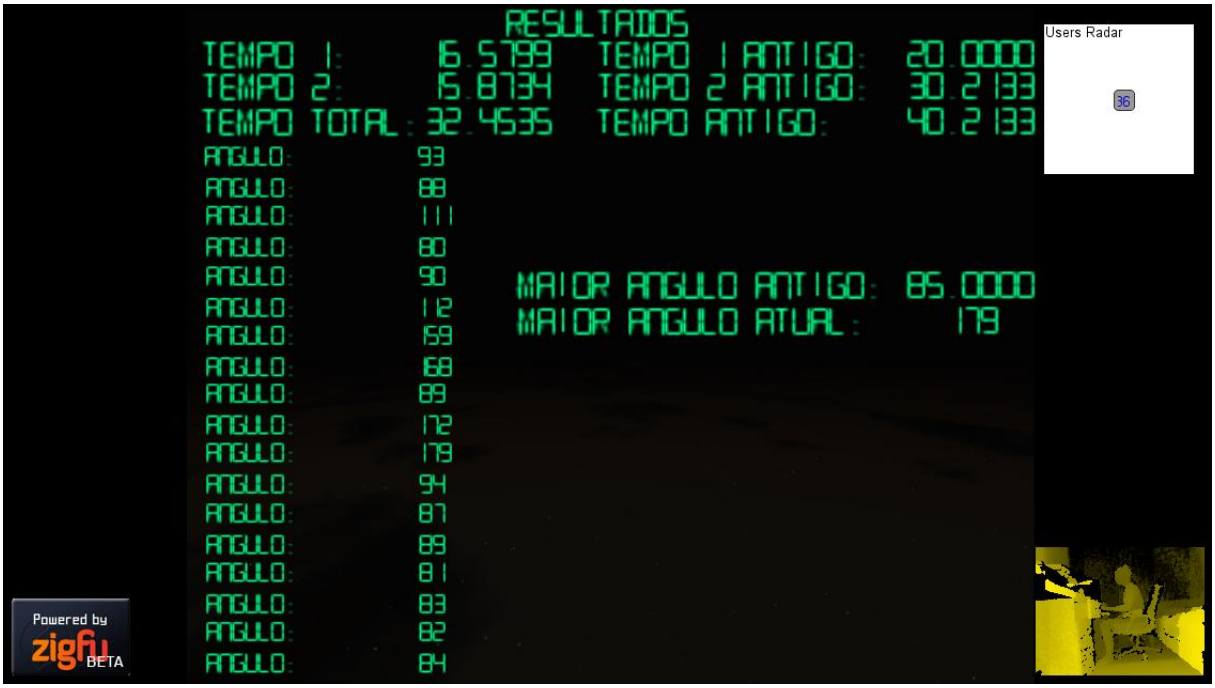

Figure 4.4: Tela de resultados do jogo das estacas.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho descreveu o processo de desenvolvimento de uma interface com jogos para o auxílio no tratamento de pacientes que apresentam deficiência na mobilidade motora. Abordaram-se os conceitos de análise e modelagem de requisitos para o desenvolvimento de aplicativos utilizando os novos dispositivos de interação homem-máquina, e o desenvolvimento de aplicações 3D. Para o processo de implementação da interface principal da aplicação, utilizou-se o conceito WPF de programação com base em XML que facilita a criação de menus e a conexão com o banco de dados. Para o jogo, foi utilizada a ferramenta Unity3d de criação de jogos, que permitiu a utilização do plugin Zigfu, sem o qual inviabilizaria a construção de um sistema de jogo em tempo hábil. O acesso ao plugin Zigfu permitiu acesso as funcionalidades do dispositivo Microsoft Kinect que antes só era possível através da SDK. Desta forma o jogo foi criado através da obtenção de requisitos e da implementação incremental do sistema com base no *feedback* dos participantes do projeto e pacientes.

A habilidade de reconhecimento do corpo do dispositivo e a capacidade do jogo de medir o desenvolvimento do paciente demonstram a capacidade destas tecnologias de agregar positivamente no tratamento de pacientes.

Espera-se que o trabalho desenvolvido sirva como base para a criação de novos jogos que utilizem dispositivos de reconhecimento de movimentos para o tratamento de deficiências de mobilidade motora e que auxiliem os profissionais da área.

Como sugestão para trabalhos futuros é possível citar o melhoramento nas funções que tem relacionamento com o banco de dados. A criação de um banco de dados com todas as informações pertinentes de fisioterapia possibilitaria a avaliação de vários tipos de exercícios que se desejar reproduzir em jogo. Também há a possibilidade da criação de novas medições no mesmo exercício como o ângulo em relação ao braço e antebraço. Pode-se ainda criar outros tipos de jogos tomando como base as mesmas classes desenvolvidas para este projeto, como por exemplo, o cálculo de todas as medidas goniométricas. Também há a possibilidade de melhoramento quanto à identificação do usuário pelo SDK, tornando possível a analise de pacientes sentados, deitados ou dispostos lateralmente em relação ao Kinect. Isto ampliaria a quantidade de movimentos disponíveis para as medições da goniometria.

6 REFERÊNCIAS

BENYON, David. **Interação humano-computador.** 2ª edição. São Paulo: Pearson, 2011

BOIAN, R.F.; LEE, C.S.; DEUTSCH, J.E.; BURDEA, G.C.; LEWIS, J.A. **Virtual Reality-based System for Ankle Rehabilitation Post Stroke** Disponível em: http://yu.ac.kr/~cvpr/paper/2002/vrmhr_anklerehab.pdf, 2002

CHAVES.M.L.F. **Acidente vascular encefálico: conceituação e fatores de risco.** Rev Bras Hipertensão. RS, 2000.

CRAWFORD, S. **How Microsoft kinect Works.** Disponivel em: < http://electronics.howstuffworks.com/microsoft-kinect2.htm> Acesso em 18 Mai. 2012.

ECMA. **C# Language Specification.** Disponível em: < http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf> 4th Edition / June 2006.

GOMES.B.M; NARDONI.G.C.G; LOPES.P.G; DE GODOY.E**. O efeito da técnica de reeducação postural global em um paciente com hemiparesia após acidente vascular encefálico**. REV. ACTA FISIATRIA. SP, 2006.

JOÃO.S.M.A. **Avaliação Fisioterapêutica do Ombro. Departamento de Fisioterapia, Fonoaudiologia e Terapia Ocupacional-FMUSP, 2011.** Disponível em: <http://www.fm.usp.br/fofito/fisio/pessoal/isabel/biomecanicaonline/articulacoes/ombro/PDF/avalombro.pdf>. Acesso em 02 de maio de 2012.

LEITE,H.R; NUNES,A.P.N.; CORREA,C.L .**Epidemiological profile of stroke survivors registered at the Health Family Strategy of Diamantina, MG** Rev. Fisioterapia e Pesquisa, SP, 2009**.**

LIMA, E.P.; RODRIGUES, J.P.; VASCONCELOS, A.G.; LANA-PEIXOTO, M.A.; HAASE, V.G. **Heterogeneidade dos déficits cognitivo e motor na esclerose múltipla: um estudo com a MSFC.** Disponível em: <

http://revistaseletronicas.pucrs.br/ojs/index.php/revistapsico/article/viewFile/3909/3381> *v. 39*, n. 3, pp. 371-381, jul./set. 2008.

MATHIOWETZ V.; WEBER K.; KASHMAN N.; VOLLAND G. **Adult Norms for the Nine Hole Peg Test of Finger Dexterity.** The Occupational Therapy Journal of Research. 1985;5:24-33.

MICROSOFT. **Microsoft Kinect for Windows.** Disponível em: <http://www.kinectforwindows.org> Acesso em: 15 Set. 2011.

MICROSOFT. **Getting Started with WPF.** Disponível em: <http://msdn.microsoft.com/en-us/library/

MINUTOLI.V.P; DELFINO.M;.DE FREITAS M;S.T.T;LIMA. M.O.;TORTOZA.C;.DOS SANTOS.C.A **Efeito do movimento passivo contínuo isocinético na hemiplegia espástica**. Rev.Acta Fisiatria. SP, 2007**.**

OAK M; **Importance of Computers in Medicine.** Disponível em: < http://www.buzzle.com/articles/importance-of-computers-in-medicine.html> Acessado 09 jun. 2012

OPENKINECT. **OpenKinect project**. Disponível em: < http://openkinect.org/wiki/Main_Page> Acesso em: 10 Mai. 2012.

PREECE, Jennifer, ROGERS, Yvonne and SHARP, Helen **Design de Interação: Além da interação homem-computador.** São Paulo: Bookman, 2011

PRIMESENSE. **Primesense - Natural Interaction**. Disponível em: <http://primesense.com> Acesso em: 25 Mai.2012.

UNITY3D. **Unity: Documentation**. Disponível em: < http://unity3d.com/support/documentation/> Acesso em: 1 Jun. 2012.

ZIGFU. **ZDK for Unity 3D.** Disponível em: <http://www.zigfu.com/en/zdk/unity3d> Acesso em: 1 Jun. 2012.

WARD, J. **What is Game Engine?** Disponível em: < http://www.gamecareerguide.com/features/529/what_is_a_game_.php> 2 Jun. 2012.