UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
COORDENAÇÃO DE PESQUISA E PÓS-GRADUAÇÃO

PROPOSTA DE UM AMBIENTE VIRTUAL DE APOIO À REABILITAÇÃO FUNCIONAL DE MEMBROS SUPERIORES UTILIZANDO O SENSOR DE CAPTURA DE MOVIMENTOS DA MICROSOFT KINECT

FABRÍCIO LEONARD LEOPOLDINO

Goiânia
Setembro 2013

FABRÍCIO LEONARD LEOPOLDINO

PROPOSTA DE UM AMBIENTE VIRTUAL DE APOIO À REABILITAÇÃO FUNCIONAL DE MEMBROS SUPERIORES UTILIZANDO O SENSOR DE MOVIMENTO DA MICROSOFT KINECT

Dissertação de Mestrado apresentada à Comissão de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação - UFG, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica e de Computação.

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. Leonardo da Cunha Brito.

Co-orientador: Prof. Dr. Marcus Fraga Vieira

Goiânia
Setembro 2013

FABRÍCIO LEONARD LEOPOLDINO

PROPOSTA DE UM AMBIENTE VIRTUAL DE APOIO À REABILITAÇÃO FUNCIONAL DE MEMBROS SUPERIORES UTILIZANDO O SENSOR DE CAPTURA DE MOVIMENTOS DA MICROSOFT KINECT

Esta dissertação foi julgada adequada à obtenção do título de Mestre em Engenharia Elétrica e de Computação e aprovada em sua forma final pela Escola de Engenharia Elétrica e de Computação da Universidade Federal de Goiás.

Goiânia, 11 de Setembro de 2013.

_____

Professor Leonardo da Cunha Brito, Dr.
UFG, Orientador.

_____

Professor Gelson da Cruz Júnior, Dr.
UFG, Examinador Interno

_____

Professor Leandro Luís Galdino de Oliveira, Dr.
UFG, Examinador Interno

Dedico este trabalho a toda minha família e principalmente aos meus pais, por terem me dado todo o carinho e a melhor educação possível. Em especial, este trabalho é dedicado a minha esposa, que é uma pessoa muito especial em minha vida, e que sempre esteve ao meu lado me apoiando em todos os momentos.

AGRADECIMENTOS

Este trabalho não teria acontecido sem a ajuda de diversas pessoas: algumas pela contribuição técnica e outras pela motivação.

Agradeço a Deus, Pai misericordioso, pela força e saúde necessárias para que este trabalho fosse concluído.

Aos meus orientadores Prof. Dr. Leonardo da Cunha Brito e Prof. Dr. Marcus Fraga Vieira, pela amizade, paciência, incentivo e por estarem sempre disponíveis e solícitos às discussões que fizeram parte da elaboração deste trabalho.

Ao colega Ademir Alves de Brito Junior que junto com o Professor Leonardo da Cunha Brito elaboraram o algoritmo de suavização da trajetória.

Aos terapeutas ocupacionais, fisioterapeutas e pacientes do Centro de Reabilitação e Readaptação Dr. Henrique Santillo, que me deram suporte e fundamentação para o desenvolvimento deste trabalho.

À minha esposa que tanto me apoiou e incentivou para que mais esta etapa de minha vida fosse concluída com êxito.

Aos meus pais e à minha irmã, exemplos de dignidade, que sempre me apoiaram e a quem devo quaisquer virtudes que porventura tenha conquistado.

A todos os meus amigos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Agradecimento à FAPEG (Fundação de Amparo à Pesquisa do Estado de Goiás) pelo apoio dado ao projeto.

“Se Deus criou as pessoas para amar, e as coisas para cuidar, por que amamos as coisas e usamos as pessoas!”

Bob Marley

RESUMO

O presente trabalho propõe um ambiente virtual de apoio à reabilitação funcional de membros superiores, utilizando o sensor de captura de movimento da Microsoft, Kinect. Está dividido em cinco capítulos: 1) Utilização de Ambientes Virtuais no Processo de Reabilitação Funcional de Pessoas Acometidas por um Acidente Vascular Encefálico; 2) Tecnologias Utilizadas; 3) Desenvolvimento do AVARFMS; 4) Discussão e 5) Conclusões. O Acidente Vascular Encefálico (AVE) deixa sequelas que comprometem o desenvolvimento das Atividades de Vida Diária. A reabilitação não consiste em um processo que busca apenas a adaptação do paciente, mas em um esforço concreto, com o objetivo de promover o desenvolvimento máximo da funcionalidade criadora e inclusiva das pessoas. Com o surgimento dos Ambientes Virtuais (AVs) e a maior integração entre as áreas da medicina de reabilitação e sistemas de informação, bem como o aparecimento de Sensores de Movimento (Microsoft Kinect, por exemplo) de baixo custo, um novo paradigma começa a ser postulado entre pesquisadores destas duas áreas: o estudo e desenvolvimento de AVs para auxiliar no processo de reabilitação. Com o objetivo de fornecer uma ferramenta divertida aos pacientes que foram acometidos por um AVE e oferecer aos terapeutas maneiras de medir (qualitativamente e quantitativamente) a evolução do tratamento de um paciente, é proposto um Ambiente Virtual que utiliza o sensor de movimentos Kinect, criado pela Microsoft. As conclusões apresentadas neste trabalho mostram que os objetivos propostos pelo ambiente foram atingidos.

Palavras-chave: Acidente Vascular Encefálico, Reabilitação Funcional, Ambientes Virtuais de Reabilitação, Sensor de Movimento, Microsoft Kinect®.

ABSTRACT

This study proposes a Virtual Environment to support the functional rehabilitation of the upper limbs, using the Microsoft Kinect sensor motion capture. It is organized in five chapters: 1) Use of Virtual Environments in the process of functional rehabilitation of people affected by an Encephalic Vascular Accident; 2) Technologies used, 3) AVARFMS development; 4) Discussion; 5) Conclusions. Sequelae of Encephalic Vascular Accident impair the development of the Activities of Daily Living. Rehabilitation does not consist in a process that seeks only the adaptation of the patient, but in a real effort with the goal of promoting the development of people creative and inclusive functionality. With the emergence of Virtual Environments and greater integration between the areas of rehabilitation medicine and information systems as well as the appearance of low cost motion sensors (Microsoft Kinect, for example), a new paradigm begins to be postulated between researchers in these two areas: the study and development of virtual environments to assist in the rehabilitation process. With the goal of providing a fun tool to patients who have been affected by an encephalic vascular accident and offering therapists ways to measure (qualitatively and quantitatively) the evolution of the patient treatment, it is proposed a virtual environment that uses the motion sensor Kinect, created by Microsoft. The findings presented in this study show that the proposed objectives were achieved.

Keywords: Encephalic Vascular Accident, Functional Rehabilitation, Virtual Environments for Rehabilitation, Motion Sensor, Microsoft Kinect®.

LISTA DE ILUSTRAÇÕES

FIGURA 1.1: AVE CAUSADO POR OBSTRUÇÃO..... 21
FIGURA 1.2: AVE ISQUÊMICO E HEMORRÁGICO. 22
FIGURA 1.3: CRUZAMENTO DAS VIAS DE CONDUÇÃO NO SISTEMA NERVOSO CENTRAL. 22
FIGURA 1.4: PRINCIPAIS CONSOLES DE VÍDEO GAME UTILIZADOS NO PROCESSO DE REABILITAÇÃO. 25
FIGURA 1.5: PACIENTE UTILIZANDO O AMBIENTE VIRTUAL FORNECIDO PELO NINTENDO WII..... 27
FIGURA 2.1: KINECT PARA XBOX. 32
FIGURA 2.2: KINECT PARA WINDOWS. 32
FIGURA 2.3: KINECT PARA WINDOWS FECHADO. 33
FIGURA 2.4: KINECT PARA WINDOWS ABERTO..... 33
FIGURA 2.5: KINECT AC ADAPTER. 34
FIGURA 2.6: AMBIENTE PARA MELHOR USO DO KINECT. 35
FIGURA 2.7: INTERAÇÃO ENTRE O KINECT E O APLICATIVO. 35
FIGURA 2.8: ESTRUTURA DO KINECT FOR WINDOWS SDK. 36
FIGURA 2.9: ESTRUTURA DO PROJETO CAPTURADADOS. 37
FIGURA 2.10: DLL INCORPORADA AO PROJETO..... 38
FIGURA 2.11: CÓDIGO DA CLASSE MAINWINDOW. 38
FIGURA 2.12: MÉTODO INICIARKINECT 39
FIGURA 2.13: MÉTODO FINALIZARKINECT. 39
FIGURA 2.14: CÓDIGO XAML..... 40
FIGURA 2.15: INTERFACE GRÁFICA COM USUÁRIO GERADA PELO CÓDIGO DA FIGURA 2.14..... 40
FIGURA 2.16: INTERFACE DO PROJETO CAPTURADADOS..... 41
FIGURA 2.17: APLICAÇÃO CAPTURADADOS EM FUNCIONAMENTO. 41
FIGURA 2.18: MÉTODO INICIARKINECT() COM SUPORTE A CÂMERA 42
FIGURA 2.19: MÉTODO MYKINECTCOLORFRAMEREADY..... 43
FIGURA 2.20: PONTOS RECONHECIDOS PELO KINECT. 45
FIGURA 2.21: PLANO CARTESIANO PELO VÍDEO DO COMPUTADOR. 46
FIGURA 2.22: TELA DA APLICAÇÃO CAPTURA ESQUELETO. 47
FIGURA 2.23: MÉTODO INICIARKINECT( ). 47
FIGURA 2.24: MÉTODO MYKINECTSKELETONFRAMEREADY. 48
FIGURA 2.25: APLICAÇÃO CAPTURA ESQUELETO EM FUNCIONAMENTO 49
FIGURA 2.26: PONTOS RECONHECIDOS PELO KINECT 49
FIGURA 2.27: TRECHO DE CÓDIGO RESPONSÁVEL PELA CAPTURA DA POSIÇÃO X,Y DA MÃO DIREITA..... 50
FIGURA 3.1: UMA *SPRITE* 53
FIGURA 3.2: SEM COLISÃO ENTRE SPRITES 54
FIGURA 3.3: COM COLISÃO ENTRE SPRITES 55
FIGURA 3.4: PONTO OMBRO CENTRAL DO PACIENTE DENTRO DO LIMITE..... 56
FIGURA 3.5: PONTO OMBRO CENTRAL DO PACIENTE FORA DO LIMITE 56

FIGURA 3.6: TELA DE MANUTENÇÃO DE ATIVIDADES DO AVARFMS 57
FIGURA 3.7: REPRESENTAÇÃO GRÁFICA DO COLISÃOPORBOUNDINGBOXESADAPTPOSICIONAMENTO PARA OS EIXOS X,Y. 58
FIGURA 3.8: REPRESENTAÇÃO GRÁFICA DO COLISÃOPORBOUNDINGBOXESADAPTPOSICIONAMENTO PARA O EIXO Z. 58
FIGURA 3.9: GRÁFICO COMPARATIVO ENTRE 80 PONTOS 61
FIGURA 3.10: ERROS DO GRÁFICO EM DESTAQUE. 62
FIGURA 3.11: DEMONSTRAÇÃO DO SEGMENTO DE LINHA E DO PONTO. 62
FIGURA 3.12: DIAGRAMA DE CASO DE USO DO AVARFMS. 65
FIGURA 3.13: TELA PRINCIPAL DO AVARFMS 66
FIGURA 3.14: TELA DE MANUTENÇÃO DE PACIENTES DO AVARFMS. 66
FIGURA 3.15: TELA DE MANUTENÇÃO DE ATIVIDADES DO AVARFMS. 68
FIGURA 3.16: JANELA DE INFORMAÇÃO. 69
FIGURA 3.17: TELA TERAPEUTA DESENVOLVE ATIVIDADE PARA PACIENTE 70
FIGURA 3.18: JANELA DE ERRO. 70
FIGURA 3.19: CRIAÇÃO DE UMA ATIVIDADE. 71
FIGURA 3.20: TELA CONFIGURAR ATIVIDADES DO PACIENTE. 73
FIGURA 3.21: CONFIGURAR ATIVIDADES DO PACIENTE COM OPÇÕES DE CONFIGURAÇÃO DA LINHA. 74
FIGURA 3.22: JANELA DE INFORMAÇÃO. 74
FIGURA 3.23: JANELA DE ERRO. 74
FIGURA 3.24: TELA PACIENTE DESENVOLVE ATIVIDADE COM A OPÇÃO MOSTRAR PACIENTE MARCADA COM MENSAGEM DE AVISO. 75
FIGURA 3.25: PACIENTE DESENVOLVE ATIVIDADE COM A OPÇÃO MOSTRAR PACIENTE DESMARCADA COM MENSAGEM DE AVISO. 75
FIGURA 3.26: TELA PACIENTE DESENVOLVE ATIVIDADE MOSTRANDO PACIENTE SEM MENSAGEM DE AVISO 76
FIGURA 3.27: TELA PACIENTE DESENVOLVE ATIVIDADE MOSTRANDO ESQUELETO SEM MENSAGEM DE AVISO 76
FIGURA 3.28: TELA PACIENTE DESENVOLVE ATIVIDADE MOSTRANDO PACIENTE COM MENSAGEM DE POSICIONAMENTO. 77
FIGURA 3.29: TELA PACIENTE DESENVOLVE ATIVIDADE MOSTRANDO ESQUELETO COM MENSAGEM DE POSICIONAMENTO. 77
FIGURA 3.30: POSICIONAMENTO DO PACIENTE EM PORCENTAGEM DENTRO DO LIMITE DO KINECT. 78
FIGURA 3.31: CARREGANDO E INICIANDO A ATIVIDADE MOSTRANDO PACIENTE. 78
FIGURA 3.32: PRIMEIRO PONTO DA TRAJETÓRIA ALCANÇADO MOSTRANDO PACIENTE. 79
FIGURA 3.33: ATIVIDADE CONCLUÍDA MOSTRANDO PACIENTE. 79
FIGURA 3.34: CARREGANDO E INICIANDO A ATIVIDADE MOSTRANDO ESQUELETO. 80
FIGURA 3.35: PRIMEIRO PONTO DA TRAJETÓRIA ALCANÇADO MOSTRANDO ESQUELETO. 80
FIGURA 3.36: ATIVIDADE CONCLUÍDA MOSTRANDO ESQUELETO 81
FIGURA 3.37: JANELA DE INFORMAÇÃO. 81
FIGURA 3.38: TELA HISTÓRICO DE ATIVIDADES. 82
FIGURA 3.39: TELA RESULTADO 82

FIGURA 3.40: GRÁFICO EVOLUÇÃO DO TEMPO. 83
FIGURA 3.41: GRÁFICO EVOLUÇÃO DA DISTÂNCIA DOS TRAJETOS PERCORRIDOS PELO PACIENTE. 84
FIGURA 3.42: GRÁFICO EVOLUÇÃO DO ERRO NA TRAJETÓRIA. 84
FIGURA 3.43: GRÁFICO TRAJETÓRIA 85
FIGURA 3.44: GRÁFICO COMPARATIVO DE TRAJETÓRIAS. 85
FIGURA 3.45: DIAGRAMA DE ENTIDADES. 87
FIGURA 5.1: JOGO SPACE INVADERS 91
FIGURA 7.1: SOLUTION EXPLORER DO AVARFMS 102

LISTA DE TABELAS

TABELA 2.1**:** VERSÕES DO SDK KINECT PARA WINDOWS 31
TABELA 2.2: VERSÕES DO KINECT FOR WINDOWS DEVELOPER TOOLKIT 31
TABELA 3.1: REQUISITOS FUNCIONAIS DO AVARFMS 52
TABELA 3.2: REQUISITOS NÃO FUNCIONAIS DO AVARFMS 52
TABELA 4.1: RESULTADO DO QUESTIONÁRIO DE SATISFAÇÃO DOS TERAPEUTAS. 88
TABELA 4.2: RESULTADO DO QUESTIONÁRIO DE SATISFAÇÃO DOS PACIENTES 89
TABELA 4.3: QUESTIONÁRIO APLICADO A 50 ALUNOS DO CURSO DE FISIOTERAPIA 89

LISTA DE CÓDIGOS

LISTAGEM 3.1: PSEUDOCÓDIGO COLISÃO POR BOUNDING BOXES 54
LISTAGEM 3.2: PSEUDOCÓDIGO COLISÃOPORBOUNDINGBOXESADAPTPOSICIONAMENTO 57
LISTAGEM 3.3: PSEUDOCÓDIGO COLISÃOPORBOUNDINGBOXESACERTOPONTO 59
LISTAGEM 3.4: PSEUDOCÓDIGO ALTERARESTADOPONTO 59
LISTAGEM 3.5: PSEUDOCÓDIGO MEDIAMOVEL 60
LISTAGEM 3.6: PSEUDOCÓDIGO SUAVIZARTRAJETORIA 60
LISTAGEM 3.7: PSEUDOCÓDIGO CALCULAERRO 62
LISTAGEM 3.8: PSEUDOCÓDIGO DISTANCIASEGUMENTOLINHA 63
LISTAGEM 7.1: CÓDIGO FONTE DO MÉTODO DESENHAESQUELETO 99
LISTAGEM 7.2: CÓDIGO FONTE DO MÉTODO GERAFIGURA 100
LISTAGEM 7.3: CÓDIGO FONTE DO MÉTODO GETJOINTPOINT 100
LISTAGEM 7.4: CÓDIGO FONTE DO MÉTODO OBTERINFORMAÇÕES 101
LISTAGEM 7.5: CÓDIGO FONTE DO ARQUIVO APPATIVIDADE. 102
LISTAGEM 7.6: CÓDIGO FONTE DO ARQUIVO APPPONTOSTRAJETORIA. 103
LISTAGEM 7.7: CÓDIGO FONTE DO ARQUIVO APPPONTO 104
LISTAGEM 7.8: CÓDIGO FONTE DO ARQUIVO APPPACIENTE. 105
LISTAGEM 7.9: CÓDIGO FONTE DO ARQUIVO APPATIVIDADEPACIENTE 106
LISTAGEM 7.10: CÓDIGO FONTE DO ARQUIVO ATIVIDADE. 107
LISTAGEM 7.11: CÓDIGO FONTE DO ARQUIVO HISTORICOATIVIDADEPACIENTE 108
LISTAGEM 7.12: CÓDIGO FONTE DO ARQUIVO OPCOES 108
LISTAGEM 7.13: CÓDIGO FONTE DO ARQUIVO PACIENTE. 108
LISTAGEM 7.14: CÓDIGO FONTE DO ARQUIVO PONTO. 108
LISTAGEM 7.15: CÓDIGO FONTE DO ARQUIVO PONTOSTRAJETORIO. 109
LISTAGEM 7.16: CÓDIGO FONTE DO ARQUIVO PONTOSSUAVIZADO 109
LISTAGEM 7.17: CÓDIGO FONTE DO ARQUIVO METRICAS 109
LISTAGEM 7.18: CÓDIGO FONTE DO ARQUIVO BANCODADOSAMBIENTEVIRTUAL. 116
LISTAGEM 7.19: CÓDIGO FONTE DO ARQUIVO MANTERATIVIDADE.XAML 117
LISTAGEM 7.20: CÓDIGO FONTE DO ARQUIVO MANTERATIVIDADE.CS 118
LISTAGEM 7.21: CÓDIGO FONTE DO ARQUIVO MANTERFAZERATIVIDADEPACIENTE.XAML 123
LISTAGEM 7.22: CÓDIGO FONTE DO ARQUIVO MANTERFAZERATIVIDADEPACIENTE.CS 124
LISTAGEM 7.23: CÓDIGO FONTE DO ARQUIVO MANTERRELATORIO.XAML 128
LISTAGEM 7.24: CÓDIGO FONTE DO ARQUIVO MANTERRELATORIO.CS 128
LISTAGEM 7.25: CÓDIGO FONTE DO ARQUIVO MANTERPACIENTE.XAML 130
LISTAGEM 7.26: CÓDIGO FONTE DO ARQUIVO MANTERPACIENTE.CS 131
LISTAGEM 7.27: CÓDIGO FONTE DO ARQUIVO DESENVOLVERATIVIDADE.XAML 133
LISTAGEM 7.28: CÓDIGO FONTE DO ARQUIVO DESENVOLVERATIVIDADE.CS 134
LISTAGEM 7.29: CÓDIGO FONTE DO ARQUIVO PACIENTEJOGAATIVIDADE.XAML 140
LISTAGEM 7.30: CÓDIGO FONTE DO ARQUIVO PACIENTEJOGAATIVIDADE.CS 141
LISTAGEM 7.31: CÓDIGO FONTE DO ARQUIVO RESULTADOS.XAML 156

LISTAGEM 7.32: CÓDIGO FONTE DO ARQUIVO RESULTADOS.CS 156

LISTA DE ABREVIATURAS E SIGLAS

AV	Ambiente Virtual
AVARFMS	Ambiente Virtual de Apoio à Reabilitação Funcional de Membros Superiores
AVD	Atividade de Vida Diária
AVE	Acidente Vascular Encefálico
CRER	Centro de Reabilitação e Readaptação Dr. Henrique Santillo
DLL	Dynamic-link library (Biblioteca de Ligação Dinâmica)
FNF	Requisitos não Funcionais
FPS	Frames por Segundo
MCU	Modelo de Caso de Uso
RF	Requisitos Funcionais
RV	Realidade Virtual
SBDCV	Sociedade Brasileira de Doenças Cerebrovasculares
SDK	Software Development Kit (Conjunto de Desenvolvimento de Software)
SGBD	Sistema Gerenciador de Banco de Dados
UML	Linguagem de Modelagem Unificada
XAML	eXtensible Application Markup Language

SUMÁRIO

1. UTILIZAÇÃO DE AMBIENTES VIRTUAIS NO PROCESSO DE REABILITAÇÃO FUNCIONAL DE PESSOAS ACOMETIDAS POR UM ACIDENTE VASCULAR ENCEFÁLICO 16

1.1 INTRODUÇÃO 16
1.2 TEMA 18
1.3 OBJETIVOS 19
1.3.1 Objetivos Gerais 19
1.3.2 Objetivos Específicos 19
1.4 JUSTIFICATIVA 20
1.4.1 Acidente Vascular Encefálico (AVE) 21
1.4.2 Tratamento e Reabilitação 23
1.4.3 Treinamento nas Atividades de Vida Diárias 24
1.4.4 Ambientes virtuais na reabilitação 25
1.5 ESTADO DA ARTE 27

2. TECNOLOGIAS UTILIZADAS 30

2.1 SENSOR DE MOVIMENTO MICROSOFT KINECT 30
2.2 VERSÕES DO MICROSOFT KINECT 32
2.3 VISÃO GERAL DO KINECT PARA WINDOWS 33
2.4 VISÃO GERAL DO SDK PARA DESENVOLVIMENTO PARA KINECT 35
2.5 ENTENDENDO O FUNCIONAMENTO DO KINECT EM UMA APLICAÇÃO 37
2.5.1 Aplicação “CapturaDados” 37
2.5.2 Aplicação “CapturaEsqueleto” 44

3. DESENVOLVIMENTO DO AMBIENTE VIRTUAL DE APOIO À REABILITAÇÃO FUNCIONAL DE MEMBROS SUPERIORES (AVARFMS) 51

3.1 INTRODUÇÃO 51
3.2 ESPECIFICAÇÕES DE REQUISITOS DO AVARFMS 51
3.2.1 Especificações de requisitos funcionais do AVARFMS 52
3.2.2 Especificações de requisitos não funcionais do AVARFMS 52
3.3 ALGORITMOS 53
3.3.1 Algoritmo Bounding Boxes 53
3.3.2 Algoritmo “Bounding Boxes” adaptado para posicionamento do paciente 55
3.3.3 Algoritmo “Bounding Boxes” adaptado para acerto dos pontos 59
3.3.4 Algoritmo para suavização da trajetória do paciente 59
3.4 MODELOS DE CASO DE USO DO AVARFMS 64
3.4.1 Caso de uso: Manter Paciente 65
3.4.2 Caso de uso: Manter Atividade 67
3.4.3 Caso de uso: Criar atividade 69
3.4.4 Caso de uso: Desenvolver atividade 72
3.4.5 Caso de uso: Consultar Histórico 81
3.5 MODELOS DE DIAGRAMA DE CLASSE AVARFMS 86

4. DISCUSSÃO 88

OBJETIVO 01: FORNECER MEIOS QUALITATIVOS E QUANTITATIVOS PARA QUE PROFISSIONAL DE SAÚDE POSSA ACOMPANHAR A EVOLUÇÃO DO TRATAMENTO DE REABILITAÇÃO DE UMA VÍTIMA DO AVE. 88
OBJETIVO 02: FORNECER UMA FERRAMENTA LÚDICA E DIVERTIDA ÀS VÍTIMAS DO AVE, EM SEU PROCESSO DE REABILITAÇÃO. 88

5. CONCLUSÕES 90

5.1 TRABALHOS FUTUROS 90
5.1.1 Gravação da sessão do paciente 90
5.1.2 Cálculo dos ângulos de movimento 90
5.1.3 Alterar o jogo 90
5.1.4 Reabilitar os membros inferiores 91

6. REFERÊNCIAS 92

7. ANEXOS 99

7.1 – ANEXOS DA APLICAÇÃO CAPTURA ESQUELETO 99
7.1.1 - Método DesenhaEsqueleto 99
7.1.2 - Método ObterInformações 100
7.2 – ANEXOS DO AVARFMS 101
7.2.1- O projeto ao AVARFMS 101

1. UTILIZAÇÃO DE AMBIENTES VIRTUAIS NO PROCESSO DE REABILITAÇÃO FUNCIONAL DE PESSOAS ACOMETIDAS POR UM ACIDENTE VASCULAR ENCEFÁLICO

1.1 INTRODUÇÃO

A perspectiva do aumento acentuado da longevidade que ocorre nos países em desenvolvimento tem determinado uma mudança no perfil demográfico de toda a população, evidenciando-se um aumento de doenças crônico-degenerativas e, dentre estas, as cerebrovasculares. No Brasil, os índices de mortalidade decorrentes de um acidente vascular encefálico (AVE), popularmente conhecido como derrame cerebral, vêm diminuindo. No entanto, o AVE tem deixado sequelas que comprometem a qualidade de vida das pessoas acometidas, principalmente nas atividades realizadas no dia a dia, como: alimentação, higiene pessoal, vestimenta, movimento e etc. Essas atividades são denominadas Atividades de Vida Diária (AVDs) pelos terapeutas e fisioterapeutas (PERLINI e FARO, 2005).

Segundo a Sociedade Brasileira de Doenças Cerebrovasculares (SBDCV, 2013), o AVE é a principal causa de mortes entre os brasileiros, sendo a principal causa de incapacidade no mundo. Aproximadamente 70% das pessoas não retornam ao trabalho após um AVE devido às sequelas e 50% ficam dependentes de outras pessoas no dia a dia. Apesar de atingir com mais frequência indivíduos acima de 60 anos, o AVE pode ocorrer em qualquer idade, inclusive em crianças. O AVE vem crescendo cada vez mais entre os jovens, ocorrendo em 10% de pacientes com menos de 55 anos e a Organização Mundial de Saúde (WHO, 2013) prevê que uma a cada seis pessoas no mundo terá um AVE ao longo de sua vida.

Atualmente, pode-se observar que a utilização de novas tecnologias da área da computação aplicadas à saúde tem por objetivo auxiliar os profissionais de saúde, em seu trabalho de diagnósticos, e os profissionais de reabilitação, no desenvolvimento de atividades de avaliações e meios de tratamento de seus pacientes. Segundo Jerônimo e Lima (2006), a utilização dessas tecnologias tem se destacado por inúmeros fatores, por exemplo:

- Repetição de exercícios;

- Adaptação do nível de dificuldade de acordo com o desempenho do paciente;
- Realimentação visual e imediata ao paciente e ao profissional de saúde;
- Motivação ao paciente, pois o mesmo se concentra na atividade e por alguns instantes esquece sua condição física.

O processo de reabilitação inspira-se em principios fundamentais: totalidade, individualidade, fundamentação científica, enfoque multidisciplinar, permanência, universalidade e utilidade social. Esses princípios determinam as ações a serem desenvolvidas pelo paciente com o objetivo de alcançar o seu potencial máximo. A reabilitação não visa apenas à adaptação pessoal do paciente, mas a um esforço ideológico e concreto no sentido de promover o desenvolvimento máximo da funcionalidade efetiva, criadora e inclusiva (JERÔNIMO e LIMA, 2006).

Posto isso, esta dissertação tem como objetivo propor um ambiente virtual de apoio à reabilitação funcional de membros superiores (AVARFMS), utilizando o sensor de movimento Microsoft Kinect®. A utilização de Ambientes Virtuais (AVs) vem se mostrando útil no processo de reabilitação de pessoas acometidas por um AVE, principalmente nos processos cognitivos, devido às características multissensoriais apresentadas por um AV (PANTELIDIS, 1995).

A exploração da Realidade Virtual (RV) apresenta diversas vantagens em relação a outras tecnologias, sendo a participação ativa do paciente no processo de reabilitação a principal delas. Além disso, a utilização de Ambientes e Realidades Virtuais por pessoas com diversas deficiências, comum em pessoas acometidas por uma AVE, oferece novas abordagens que são difíceis de serem realizadas com processos comuns (PANTELIDIS, 1995).

Os resultados apresentados neste trabalho demonstram a eficiência e a aplicabilidade do ambiente virtual, utilizando como referência o Centro de Reabilitação e Readaptação Dr. Henrique Santillo (CRER), localizado em Goiânia, estado de Goiás. Fundado em 2002, o CRER oferece atendimento humanizado e especializado em reabilitação a pessoas com deficiência física e/ou sensorial, exclusivamente pelo Sistema Único de Saúde (SUS).

1.2 TEMA

A reabilitação compreende todo um programa, com o objetivo de capacitar pacientes acometidos por um AVE a melhorarem as suas funções físicas, intelectuais, psicológicas e/ou sociais, para obter o máximo grau de independência na realização de suas AVDs. Por meio do processo de reabilitação, o paciente pode readquirir capacidades e/ou reaprender novas formas de realizar tarefas. Segundo Silva (2010), em um programa de reabilitação, a prática direta bem orientada e repetitiva é o elemento primordial para o sucesso da vítima de um AVE. O processo de reabilitação envolve seis princípios:

1. Prevenção, reconhecimento e gestão das complicações e comorbilidades[1];
2. Buscar na terapia o máximo de independência;
3. Auxílio na adaptação do paciente e da família com a situação atual;
4. Prevenção do déficit secundário por meio da reintegração social, incluindo o regresso ao lar, à família e às atividades lúdicas e vocacionais;
5. Reforço da qualidade de vida levando em conta o déficit residual;
6. Prevenção de um segundo AVE ou outros eventos vasculares, como enfarte agudo do miocárdio, que ocorrem com mais frequência em pacientes que já foram vítimas de um AVE.

A recuperação funcional de um paciente pode ser iniciada quando a sua situação clínica for estabilizada, sendo que, na primeira etapa, deve se concentrar na promoção da independência motora. Para isso, o paciente é solicitado a realizar um amplo conjunto de exercícios ativos ou passivos, com o objetivo de fortalecer os membros debilitados. Os profissionais de saúde indicados para auxiliar o paciente nesse primeiro momento são os enfermeiros, terapeutas ocupacionais e fisioterapeutas, que também ajudam o paciente a desenvolver tarefas mais complexas, como tomar banho, colocar roupa e utilizar o banheiro. Nesse momento, o paciente começa a readquirir a capacidade para desenvolver as suas AVDs, o que representa o primeiro passo na independência funcional (DUNCAN, 2005).

A reabilitação poderá ser um processo contínuo de aquisição, manutenção e aperfeiçoamento das capacidades básicas necessárias para a independência do paciente. Nesse

[1]Comorbidade é um termo usado para descrever a ocorrência simultânea de dois ou mais problemas de saúde em um mesmo indivíduo. Esse é um fenômeno frequente na prática clínica, e sua identificação é um fator importante que afeta tanto o prognóstico dos pacientes como a conduta terapêutica do médico (MARQUES, 1994).

caso, a participação de diferentes profissionais, com a intervenção da comunidade durante meses ou anos em seu processo de reabilitação pode ser necessária (CATHY, 1997).

A utilização de tecnologias ligadas à área da computação aplicadas à saúde vem se consolidando a cada dia, por meio do compartilhamento de experiências de sucesso em diversas áreas: na área médica, por exemplo, promove auxílio a diagnósticos; na área de reabilitação, contribui para o desenvolvimento de programas de avaliação e meios de tratamento. A colaboração entre especialistas das áreas de tecnologia e de saúde vem crescendo na última década com grandes progressos, favorecendo o uso de tecnologias integradas a programas de reabilitação, destacando-se a utilização de ambientes virtuais para treinamento e educação de pessoas com algum tipo de deficiência (JERÔNIMO; LIMA, 2006).

Para o planejamento de um tratamento de reabilitação funcional, cabe ao profissional de saúde (terapeuta ocupacional, fisioterapeuta, etc) uma avaliação objetiva acerca das limitações do paciente. Atualmente, são utilizados, como ferramentas de apoio, os ambientes virtuais fornecidos pelos consoles de vídeo game como Nintendo Wii®, Playstation® Move e Xbox 360® em sessões fisioterapêuticas de reabilitação (SARI; SCHUSTER; ALVARENGA, 2008).

1.3 OBJETIVOS

1.3.1 Objetivos Gerais

O presente trabalho tem por objetivo propor um ambiente virtual de apoio à reabilitação funcional de membros superiores de pacientes acometidos por AVE, utilizando o sensor de movimento Microsoft Kinect como interface homem-máquina ou dispositivo de entrada ou dispositivo de captura de movimento.

1.3.2 Objetivos Específicos

- Fornecer uma ferramenta computacional lúdica e divertida às vítimas de AVE, em seu processo de reabilitação.

- Fornecer, por meio da ferramenta computacional, meios qualitativos e quantitativos para que o profissional de saúde possa acompanhar a evolução do tratamento de reabilitação de uma vítima de AVE.

1.4 JUSTIFICATIVA

A reabilitação não consiste em um processo que busca apenas a adaptação do paciente, mas em um esforço concreto, com o objetivo de promover o desenvolvimento máximo da funcionalidade criadora e inclusiva das pessoas. O processo de reabilitação busca a sua inspiração em princípios gerais que norteiam suas ações, pois busca a eficiência máxima do potencial das pessoas com funções motoras prejudicadas por um AVE (JERÔNIMO; LIMA, 2006).

Por muito tempo, a condição motora de uma pessoa era monitorada por palpação ou observações visuais. Com os avanços de técnicas de utilização de neuroimagem (tomografia computadorizada, por exemplo), que fazem o mapeamento das áreas do cérebro, alterações cerebrais durante a aprendizagem motora podem ser avalidas. Infelizmente, os exames de neuroimagem possuem um alto custo e não podem ser utilizados diariamente na verificação e análise de tratamentos (GONÇALVES, 2008).

Com o surgimento dos Ambientes Virtuais, maior integração entre as áreas da medicina de reabilitação e de sistemas de informação e o aparecimento de sensores de movimento (Microsoft Kinect, por exemplo) de baixo custo, um novo paradigma começa a ser postulado entre pesquisadores dessas duas áreas: o estudo e desenvolvimento de AVs para auxiliar no processo de reabilitação.

Os AVs, também conhecidos como realidade artificial, são considerados tecnologias inovadoras que irão produzir impactos consideráveis sobre a neurorreabilitação nos próximos anos. Pode-se observar que, durante os últimos anos, o número de grupos de pesquisa e projetos em desenvolvimento começou a aumentar e até mesmo o número de produtos comercias (KAUTZ, 2004).

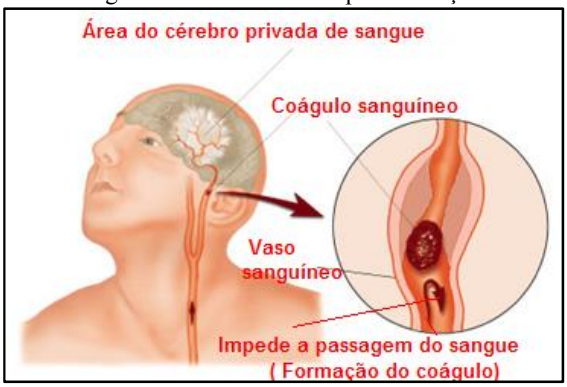
Os AVs são constituídos de uma combinação de *software* e *hardware* especializados, que permitem criar ambientes gráficos de aparência realística, nos quais o paciente pode se locomover em duas ou três dimensões. A simulação de objetos virtuais, criados por *softwares*,

fornece uma realimentação visual, funcionando como ferramenta de aprendizagem motora e cognitiva para pacientes, utilizada por profissionais responsáveis pelo processo de reabilitação. A utilização de AVs fornece uma forma de se obter constantes autocorreções durante o desenvolvimento de uma atividade motora na qual as habilidades de planejamento motor são estimuladas, o que traz benefícios à plasticidade neural (SARDI; SCHUSTER; ALVARENGA, 2012).

1.4.1 Acidente Vascular Encefálico (AVE)

O AVE pode ser descrito como uma síndrome clínica que ocorre devido à obstrução ou rompimento dos vasos sanguíneos do sistema nervoso central, causando déficit neurológico e motor em suas vítimas (BARBOSA, 2008). A Figura 1.1 demonstra a ocorrência de um AVE por obstrução:

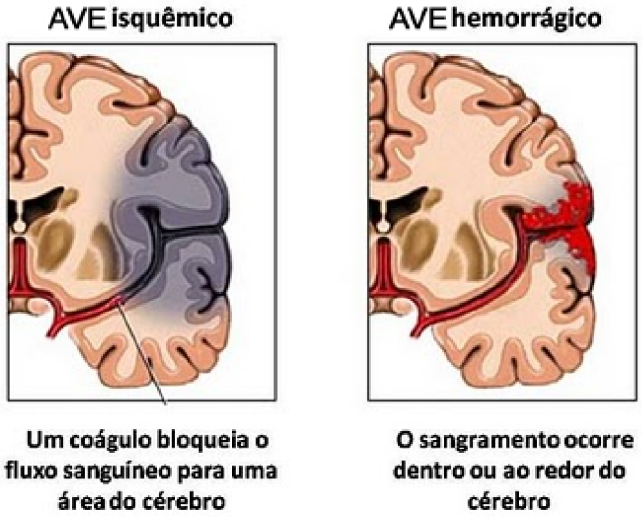
Figura 1.1: AVE causado por obstrução



Fonte: CAIADO, 2013.

O AVE pode ser de origem isquêmica ou hemorrágica. No de origem isquêmica, ocorre alguma obstrução que reduz subitamente o fluxo sanguíneo em uma artéria do cérebro, provocando falta de circulação e diminuição da função neurológica. No de origem hemorrágica, ocorre o rompimento de um vaso sanguíneo que provoca um sangramento no interior ou ao redor do cérebro (SBDCV, 2013). Estima-se que cerca de 80% dos acidentes vasculares cerebrais sejam isquêmicos. A Figura 1.2 demonstra os dois tipos de AVE:

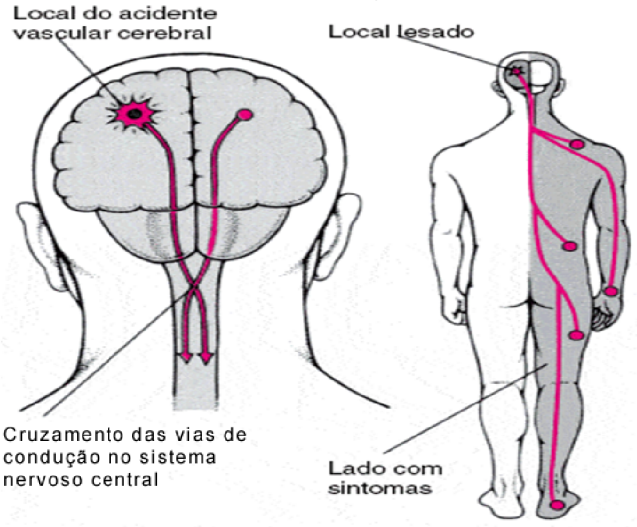
Figura 1.2: AVE isquêmico e hemorrágico.



Fonte: Adaptado de ZANETTI, 2013.

Usualmente, os acidentes vasculares cerebrais lesam apenas um lado do cérebro. Como as vias de condução no sistema nervoso central cruzam em direção ao outro lado do corpo, os sintomas ocorrem no lado do corpo oposto ao lado lesado do cérebro (SBDCV, 2013), como mostra a figura 1.3:

Figura 1.3: Cruzamento das vias de condução no sistema nervoso central.



Fonte: Adaptado de FALCÃO, 2012.

Tudo aquilo que possa facilitar a ocorrência do AVE é considerado fator de risco. É imprescindível a sua caracterização e devida correção, pois quase toda prevenção do AVE é baseada no combate aos fatores de risco (WHO, 2013). Os principais fatores de risco são:

- Pressão Arterial;

- Doença Cardíaca;
- Colesterol;
- Tabagismo;
- Uso excessivo de bebidas alcoólicas;
- Diabetes;
- Histórico de doença vascular anterior;
- Obesidade;
- Anticoncepcionais;
- Sedentarismo.

Durante as visitas ao Centro de Reabilitação e Readaptação Dr. Henrique Santillo (CRER) para o levantamento das informações necessárias para o desenvolvimento deste trabalho, foi observado que as pessoas acometidas por um AVE demonstravam as seguintes deficiências: paralisia e alterações da motricidade, alterações sensoriais, alterações de comunicação, alterações de cognição e distúrbios emocionais (PINHEIRO, 2013). Com base nessas observações, pode-se concluir que o tratamento e a reabilitação da pessoa vitimada por um AVE dependerá sempre das particularidades que envolvam cada caso (WHO, 2013).

1.4.2 Tratamento e Reabilitação

O tratamento para vítimas de um AVE deve ser baseado em programas holísticos, desenvolvidos por equipes inter e multidisciplinares (fisioterapeutas, psicólogos, nutricionistas, professores, professores de educação física, fonoaudiólogos, assistentes sociais, médicos clínicos e neurologistas, além dos profissionais da enfermagem) e, evidentemente, o paciente e sua família (LELIS, 2010). O tratamento deve contar com validade ecológica[2], com a utilização de metodologias qualitativas e quantitativas, pois seja qual for o tipo do acidente, as consequências são bastante danosas. Os programas de reabilitação incluem as terapias física e ocupacional e as reabilitações cognitivas e da linguagem. Estes programas integram um grande número de profissionais (SILVA, 2010).

[2] A validade ecológica pode ser entendida como um fenômeno transitório caracterizado por tentativas fundamentadas e sistemáticas para análise de comportamento atual dentro de contextos ambientais específicos, utilizando método de investigação discreto, realista e fidedigno (DAVIDS, 1988) e, ainda, a medida de aproximação da pesquisa ao mundo real (BARREIROS, 2008).

Conforme a região do cérebro atingida, bem como a extensão das lesões, o AVE pode oscilar entre dois opostos. Os de menor intensidade praticamente não deixam sequelas; os mais graves, todavia, podem levar as pessoas à morte ou a um estado de absoluta dependência, sem condições, por vezes, de nem mesmo sair da cama. Os danos são consideravelmente maiores quando o atendimento demora mais de três horas para ser iniciado (Biblioteca Virtual em Saúde, 2013).

Para as pessoas que sobrevivem a um AVE, a reabilitação é uma das fases mais importantes do tratamento. O processo de reabilitação deve ser iniciado nos primeiros dias após o AVE, de preferência no hospital, acompanhado da equipe médica, e, posteriormente, ter continuidade numa unidade especializada em reabilitação de pacientes com doenças vasculares cerebrais (SILVA, 2010; BARBOSA, 2008), como por exemplo, o Centro de Reabilitação e Readaptação Dr. Henrique Santillo (CRER), localizado na cidade de Goiânia, estado de Goiás.

O principal objetivo da reabilitação é auxiliar o paciente a adaptar-se às suas deficiências, favorecer sua recuperação funcional, motora e neuropsicológica, promovendo a sua reintegração no seio familiar, social e profissional. Um programa de reabilitação adequado deve contribuir para a recuperação da autoestima (SARTI, 2013; SILVA, 2010).

Um AVE envolve não somente o paciente, mas também sua família e pessoas próximas. A vítima de um AVE deve perceber que familiares amigos e pessoas próximas são inseridos, na medida de suas possibilidades, ao programa de reabilitação (LELIS, 2010).

1.4.3 Treinamento nas Atividades de Vida Diárias

As Atividades de Vida Diárias (AVDs) são aquelas realizadas no dia a dia de uma pessoa, tais como: alimentação, vestuário, toalete, banho, transferências da cadeira fixa para a cadeira de rodas, locomoção, comunicação e interação social. O treinamento nas AVDs implica a adaptação à condição atual e às sequelas que o indivíduo apresenta (hemiplegia[3], déficit de coordenação e equilíbrio, alterações cognitivas e visuoespaciais, entre outras) (SILVA, 2010; BARBOSA, 2008).

[3]Perda total da capacidade de executar movimentos voluntários em metade do corpo (CHASA, 2013).

O profissional capacitado para realizar o acompanhamento das pessoas acometidas por um AVE é o terapeuta ocupacional ou o fisioterapeuta, que, com uma observação detalhada das atividades desenvolvidas pelo paciente, pode determinar o melhor treinamento nas AVDs (SILVA, 2010). Os indivíduos são estimulados a realizar as atividades da forma mais independente possível, dentro de sua nova condição motora e cognitiva. A interação social pode ser estimulada por meio de atividades de vida prática, ou seja, atividades relacionadas à capacidade do indivíduo de interagir com o ambiente e solucionar problemas comuns à vida em sociedade, tais como: fazer compras, limpar a casa, administrar dinheiro, utilizar transporte público e resolver problemas de lógica simples (SILVA, 2010; LELIS, 2010; BARBOSA, 2008).

O planejamento de uma proposta de tratamento funcional para as AVDs deve levar em consideração as limitações do paciente. Para auxiliar os profissionais na proposta de tratamento, a computação vem se destacando como uma ferramenta de apoio, por meio dos Ambientes Virtuais (AVs) fornecidos pelos consoles de vídeo games que utilizam controles de captura de movimentos. Os principais consoles de vídeo game que utilizam controles de captura de movimentos são: "Nintendo Wii®", com o seu controle sem fio "Wii Remote®"; "Xbox 360®", com o sensor de movimento Kinect e "Playstation 3®", com o controle sem fio, o "Playstation Move" (figura 1.4).

Figura 1.4: Principais consoles de vídeo game utilizados no processo de reabilitação.

Fonte: AANABATHULA, 2009

1.4.4 Ambientes virtuais na reabilitação

Durante a fase de levantamento de requisitos para o Ambiente Virtual de Apoio à Reabilitação Funcional de Membros Superiores (AVARFMS) proposto neste trabalho, foram observadas as dificuldades que os pacientes têm no desenvolvimento das atividades propostas pelos terapeutas ocupacionais, além da desmotivação por não conseguirem realizar tarefas

simples no seu dia a dia. Muitas vezes, a recuperação é bastante dolorosa, pois envolve força no processo de movimentação de seus membros inferiores e superiores (SILVA, 2010).

O AVE causa sequelas sensitivas, motoras e cognitivas, acarretando incapacidade funcional, o que provoca dependência e diminuição da qualidade de vida do indivíduo, causando problemas na integração social do mesmo (ASSIS, 2010; VANDERLINE, 2010). As pessoas acometidas por um AVE tornam-se sedentárias e isoladas socialmente, devido ao comprometimento físico, mental e social após o AVE. Esse tipo de problema atinge não só o paciente, como também a sua família, os serviços de assistência à saúde e a sociedade como um todo. Nesta fase da vida do paciente, os déficits advindos do AVE causam uma necessidade de reaprendizagem das tarefas funcionais (SARDI; SCHUSTER; ALVARENGA, 2012). Segundo estes autores, a aprendizagem

> é um processo comum a todos os animais, por meio do qual novas informações são adquiridas (aquisição) pelo sistema nervoso, que armazena memórias (consolidação) e evoca essas informações quando necessárias à meta da tarefa motora. A capacidade para apreender seria então aquisição e/ou desenvolvimento de habilidades para alterar comportamento com base na experiência (SARDI; SCHUSTER; ALVARENGA, 2012, p.30).

O processo de reintrodução do paciente começa com a realização de uma tarefa que requer, no caso de uma tarefa motora, uma ação da musculatura esquelética para atingir seu objetivo. O desenvolvimento dessa atividade é feito após a apresentação da tarefa pelo profissional responsável pelo tratamento, no caso, o Terapeuta Ocupacional. O processo deve ser repetido de forma extensiva pelo paciente, com o objetivo de melhorar e organizar a habilidade motora. Quanto mais complexa for a atividade a ser desenvolvida, os padrões que surgem durante a experiência do paciente são memorizados e automatizados (SARDI; SCHUSTER; ALVARENGA, 2012; AULD; PANTELIDIS, 2012).

Os Ambientes Virtuais (AVs) para a área de saúde vêm conseguindo obter uma maior participação dos pacientes, provendo interfaces visuais (Figura 1.5) que geram grandes níveis de motivação e participação por parte dos pacientes. Com a utilização dos controles sem fio que capturam os movimentos de uma pessoa em jogos digitais, como o “Wii remote”, da Nintendo®, o “Playstation® Move”, da Sony® e o “Kinect®”, da Microsoft, os profissionais de saúde vislumbram novas possibilidades interessantes para o uso terapêutico (VANDERLINE, 2010).

Figura 1.5: Paciente utilizando o ambiente virtual fornecido pelo Nintendo Wii®

Fonte: DOUTOR SHOPFISIO, 2013

A exploração dos Ambientes Virtuais (AVs) vem promovendo avanços significativos no processo de reabilitação física. Os AVs fornecem interfaces que geram um grande nível de motivação por parte dos pacientes, propiciando recursos que permitem a observação de cenários em ângulos e distâncias distintas, que oferecem situações únicas, de maneira individualizada, forçando a participação ativa do paciente (VANDERLINE; 2010). Esses tipos de ambientes permitem a participação de pessoas com incapacidades mentais ou físicas, disponibilizando recursos para prática das Atividades de Vida Diárias (AVDs) do mundo real, além de fornecerem um ambiente motivador para despertar a necessidade de conhecimento e aprendizagem por meio da diversão e entretenimento (SARDI; SCHUSTER; ALVARENGA, 2012; AULD; PANTELIDIS, 2012).

A reabilitação com o apoio dos ambientes virtuais proporciona benefícios físicos, mentais e de desenvolvimento de aprendizagem para os pacientes. Os benefícios físicos são caracterizados pelo alívio do estresse e distração das questões relacionadas à deficiência; os mentais são caracterizados pelo alívio do estresse e elevação do humor, enquanto que os de desenvolvimento e aprendizagem são caracterizados pela maior concentração e melhoria da coordenação e destreza manual (SARDI; SCHUSTER ; ALVARENGA, 2012).

1.5 ESTADO DA ARTE

Neste Capítulo, serão mostrados os avanços da pesquisa relacionada à utilização dos ambientes virtuais e suas várias nuances, especialmente em relação à reabilitação de pessoas

vitimadas por AVE bem como suas contribuições para a abordagem proposta por esta dissertação.

Em seu trabalho, Davaasambuu al (2012) avalia o sensor de movimentos Microsoft Kinect como um artefato tecnológico de grande auxílio no processo de reabilitação, pois permite que o paciente possa se mover livremente em frente ao sensor, e possibilita que o processo de reabilitação seja feito também em casa. Além disso, segundo este autor, a utilização do sensor torna a reabilitação mais flexível, pois possibilita a repetição frequente dos exercícios. A prática das atividades por uma pessoa que utilizada o sensor torna-se divertida, o que agrega valor ao processo de reabilitação, por conseguir que o paciente participe ativamente do processo.

Labelle (2011) também avalia a aplicação do sensor de movimento Microsoft Kinect como uma ferramenta potencial no processo de reabilitação de pessoas que sofreram um AVE. Segundo esta autora, a aplicação do sensor apresentou um grande potencial de uso por sua capacidade de captura de dados em três dimensões (x, y, z), apresentação de imagens em tempo real e gravação da sessão de exercícios para avaliação posterior. Labelle também faz uma rápida comparação entre a biblioteca "OpenNI" e o SDK[4] fornecido pela Microsoft. No presente trabalho foi utilizado apenas o SDK fornecido pela própria Microsoft.

Jerônimo e Lima (2006) apresentam um estudo no qual são utilizados ambientes virtuais no processo terapêutico de reabilitação, demonstrando que o campo da reabilitação possibilita grande interação entre a realidade virtual e a medicina, destacando-se as seguintes características: fornecimento imediato de realimentação, adaptação ao paciente, participação ativa do paciente no processo, além de ser mais atrativo e menos doloroso do que os métodos tradicionais.

O trabalho de Gonçalves (2008) busca validar a utilização de *software* de aprendizagem e controle motor como instrumento para avaliação de habilidades motoras finas de membros superiores de indivíduos que apresentam quadro clínico de hemiplegia, com idade entre 45 e 75 anos dos sexos masculinos e femininos, após um acidente vascular encefálico que foram diagnosticados com hemiparesia.

[4] SDK é a sigla para Software Development Kit (ou Software Developers Kit – pacote de desenvolvimento de software). É esse pacote que permite à programadores elaborarem aplicativos para rodarem em uma plataforma específica (ILEX; 2008).

O estudo de Barbosa (2008) apresentou a utilização dos ambientes virtuais como ferramenta de apoio ao tratamento de pacientes com diagnóstico de hemiplegia. Este trabalho forneceu informações preciosas para esta dissertação, como por exemplo, os exercícios organizados para pacientes que sofreram um acidente vascular encefálico (AVE) e os elementos reforçadores fornecidos pelos ambientes virtuais (AVs) ao processo de reabilitação.

2. TECNOLOGIAS UTILIZADAS

Nesta seção, são apresentadas as tecnologias utilizadas para o desenvolvimento do Ambiente Virtual de Apoio à Reabilitação Funcional de Membros Superiores (AVARFMS), direcionado aos pacientes que sofreram um AVE atendidos pelo Sistema Único de Saúde (SUS) no Centro de Reabilitação e Readaptação Dr. Henrique Santillo (CRER). Será apresentado também o projeto de desenvolvimento e funcionamento do ambiente proposto.

2.1 Sensor de Movimento Microsoft Kinect

A história do sensor de movimentos Microsoft Kinect teve seu início muito antes de sua criação. O Kinect possui suas raízes no tempo em que se sonhava com a utilização de interfaces baseadas em gestos. O filme *Minority Report* (A Nova Lei, em português), lançado em 2002, no qual o policial John Anderton (Tom Cruise) acessa informações diante de uma grande tela curva de computador apenas movimentando suas mãos, instigou ainda mais a imaginação das pessoas (LANDIN, 2010).

O Kinect teve o seu primeiro anúncio no dia 01 de junho de 2009 na E3[5], sob o codinome "Projeto Natal", seguindo a tradição da Microsoft de nomear seus projetos com nome de cidade, como uma homenagem à cidade natal do diretor geral da área de incubação do Xbox, Alex Kipman (E3, 2009).

Existem duas versões do sensor de movimentos Kinect: uma lançada em 04 de novembro de 2010, a primeira versão do Kinect para Xbox 360 (CHEN, 2012); e outra em 01 de fevereiro de 2012, a primeira versão para o sistema operacional Windows (EISLER, 2012). Os responsáveis pela tecnologia utilizada pelo Kinect são Zeev Zalevsky, Shpunt Alexander, Maizels Aviad e Garcia Javier (WIPO-PATENT SCOPE, 2012).

Em janeiro de 2012, a Microsoft liberou a versão Beta do Kit de Desenvolvimento (SDK) para Kinect, para projetos de pesquisas e aplicações não comerciais. Os aplicativos desenvolvidos com essa versão funcionavam no Kinect para Xbox 360. A tabela 2.1 apresenta as versões do SDK do Kinect. No momento em que este trabalho estava sendo escrito,

[5] A E3 2013 (Electronic Entertainment Expo) é uma feira internacional de jogos eletrônicos que reúne lançamentos e novidades das melhores empresas do mercado games (BELLO, 2011).

encontrava-se na versão 1.7. O SDK é livre para o desenvolvimento de aplicativos (Microsoft Corporation, 2013).

Tabela 2.1**:** Versões do SDK Kinect para Windows

Data	Nomenclatura	Versão
05/01/2012	Kinect for Windows SDK Beta 2	1.0.0.45
05/02/2012	Kinect for Windows SDK v1.0	1.0.3.191
18/05/2012	Kinect for Windows SDK v1.5	1.5.2.331
04/10/2012	Kinect for Windows SDK v1.6	1.6.0.505
12/03/2013	Kinect for Windows SDK v1.7	1.7.0.529

Fonte: MICROSOFT 2013

Para iniciar o desenvolvimento de aplicativos que utilizem o Kinect, além do SDK, deve ser instalado também o "Kinect for Windows Developer Toolkit". A tabela 2.2 apresenta as versões do kit de ferramentas (Toolkit).

Tabela 2.2: Versões do Kinect for Windows Developer Toolkit

Data	Nomenclatura	Versão
18/05/2012	Kinect for Windows Developer Toolkit v1.5.0	1.5.0.145
16/06/2012	Kinect for Windows Developer Toolkit v1.5.1	1.5.1.180
09/10/2012	Kinect for Windows Developer Toolkit v1.5.2	1.5.2.223
26/09/2012	Kinect for Windows Developer Toolkit v1.6	1.6.0.309
12/03/2013	Kinect for Windows Developer Toolkit v1.7	1.7.0.510

Fonte: MICROSOFT 2013

O Ambiente de Desenvolvimento Integrado (IDE) sugerido pela Microsoft para o desenvolvimento de aplicações que irão utilizar o sensor de movimento é o *Visual Studio* 2012, em qualquer uma de suas versões (*Express* - Gratuito para estudos, *Professional*, *Premium* ou *Ultimate*) (VISUAL STUDIO 2013). Quanto mais completa for a versão utilizada, mais recursos que facilitarão o processo de desenvolvimento do aplicativo serão oferecidos.

2.2 Versões do Microsoft Kinect

O sensor de movimento Microsoft Kinect é oferecido em duas versões: Microsoft Kinect para Xbox 360 e Microsoft Kinect para Windows (figuras 2.1 e 2.2 respectivamente), cada uma voltada para um tipo de aplicações. A versão para Xbox 360 é voltada para os jogos digitais, a versão para Windows é voltada para o desenvolvimento de aplicações que nos dão a capacidade de interagir naturalmente com computadores, simplesmente gesticulando ou falando (KINECT, 2013).

Figura 2.1: Kinect para Xbox.

Fonte: TW Games, 2012

Figura 2.2: Kinect para Windows.

Fonte: Brasoftware, 2012

Mesmo sendo da mesma empresa, cada versão possui suas funcionalidades e recursos. A versão para Windows possui muitos recursos que não estão habilitados na versão para Xbox, como (MILLER, 2013; KERDKHOVE, 2012; CATUHE, 2012):

- Modo perto: permite que a câmera capture objetos no limite de 40 centímetros na frente do dispositivo. A versão para Xbox posssui limite de 80 centímetros;
- Sentado ou em pé: rastreamento esquelético que oferece a capacidade de controlar a cabeça, o pescoço e os braços de qualquer usuário sentado ou em pé;
- Cabo USB: garante a confiabilidade em uma ampla gama de computadores e melhora a convivência com outros periféricos USB. Existe o mesmo cabo para a versão para Xbox que é vendida separadamente;

- Alargada configurações da câmera: fornece configurações extras, como a exposição, brilho, etc, para um melhor ajuste;
- *KinectFusion* (vNext): mapas de ambiente 3D em tempo real que permitem o uso ou a substituição dos objetos em tempo real;
- *Handgrip* (vNext): na detecção da mão permite a execução de gestos como "pinch-to-zoom", garra, etc, o que pode melhorar seus aplicativos e possibilitar a construção de novos tipos de aplicações.

No quesito licenciamento dos produtos, na versão para Windows é permitido o uso do produto para fins comerciais, enquanto que para Xbox 360 não. Quando é adquirida a versão para Windows, também são adquiridas as licenças por seu uso. Logo, se o aplicativo em desenvolvimento vai ao público em geral, o uso do Kinect para Windows é permitido, pois no momento de sua aquisição, muitos benefícios são fornecidos pelos termos de sua licença (KERDKHOVE, 2012). Para o desenvolvimento deste trabalho, a versão utilizada foi a para Windows, por causa de suas questões legais e recursos.

2.3 Visão geral do Kinect para Windows

As Figuras 23 e 2.4, a seguir mostram a representação esquemática de todo os principais componentes de hardware do Kinect. Observando partir da frente, a partir do lado de fora é possível identificar o projeto infravermelho (1), led (2), a câmara RGB (3), a cameria de profundidade (4). Um conjunto de quatro microfones (5 e 7), e o motor de inclinação (6) dispostas dentro da caixa de plástico (CATHUE, 2012; MILES, 2012, ASHLEY, 2012).

Figura 2.3: Kinect para Windows fechado.

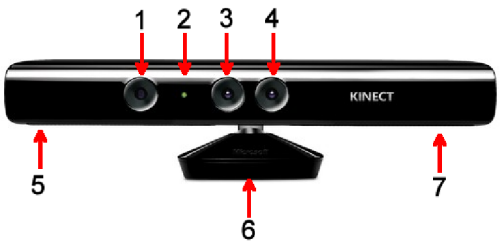
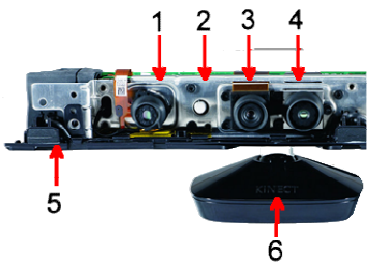


Figura 2.4: Kinect para Windows aberto.



Fonte: CATUHE, 2012

Para utilizar o Kinect em um computador, tanto na versão para Windows ou Xbox, o computador deve possuir uma porta USB 2.0 e uma fonte de energia adicional (Adaptador

Kinect AC (figura 2.5)), pois a porta USB não consegue fornecer a energia necessária para o sensor. Essa fonte acompanha a versão para Windows (CATUHE, 2012; MILES, 2012; ASHLEY, 2012).

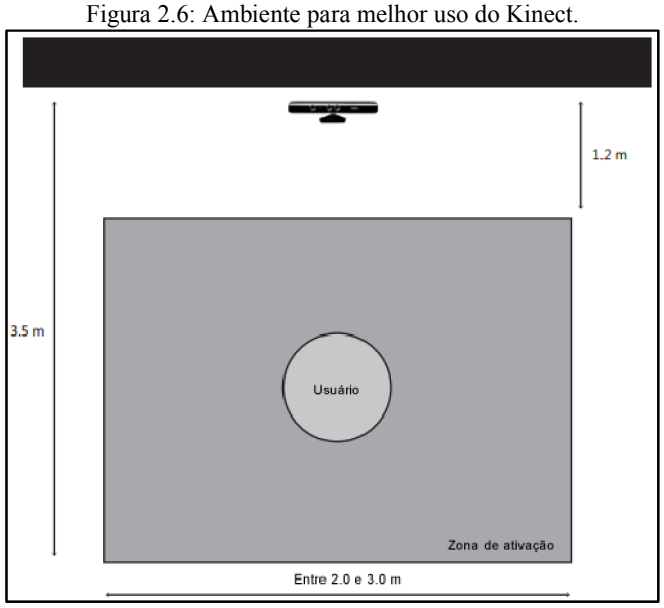
Figura 2.5: Kinect AC Adapter.

Fonte: LOWENSOHN, 2010

O Kinect utiliza lentes ópticas e possui algumas limitações, mas funciona bem sob a configuração apresentada na figura 2.6. Nessa configuração, o Kinect deve ser centralizado no ambiente onde será utilizado e o usuário deve ficar entre 1,2m e 3,5m de distância do Kinect. Essa distância é conhecida como área de ativação. Deve seguir também as seguintes recomendações de uso do aparelho (CATUHE 2012):

- Não coloque o sensor sobre ou na frente de um alto-falante ou em uma superfície que vibre ou faça ruído;
- Mantenha o sensor fora de luz direta do sol;
- Não use o sensor próximo de fontes de calor.

Figura 2.6: Ambiente para melhor uso do Kinect.



Fonte: Adaptado de CATUHE, 2012

2.4 Visão geral do SDK para Desenvolvimento para Kinect

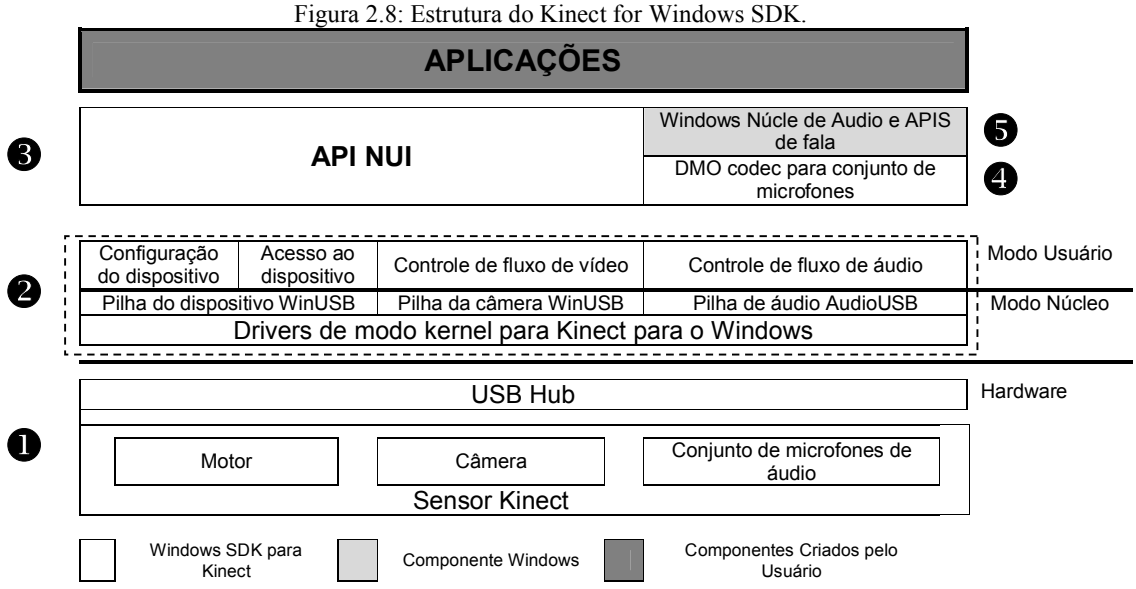
O SDK oferece uma biblioteca de *software* (Natural User Interface - NUI) e ferramentas para auxiliar os desenvolvedores de aplicativos para Kinect. A interação entre o aplicativo, a biblioteca e o Kinect é mostrada na Figura 2.7. (CATUHE, 2012; MILES, 2012; ASHLEY, 2012)

Figura 2.7: Interação entre o Kinect e o aplicativo.

Fonte: MSDN, 2012

A estrutura do Kinect for Windows SDK (CATUHE, 2012; MILES, 2012; ASHLEY, 2012) é mostrada na figura 2.8, e explicada logo a seguir.

Figura 2.8: Estrutura do Kinect for Windows SDK.



Fonte: MSDN, 2012

1. *Hardware*: os componentes de *hardware*, incluindo o sensor de movimentos Kinect e a porta USB, na qual o sensor é ligado ao computador.
2. *Drivers*: os *drivers* do Windows para o Kinect que são adicionados durante a instalação do SDK. Estes *drivers* fornecem suporte aos seguintes componentes:
    - A matriz de microfones do Kinect como dispositivo de áudio no modo kernel que podem ser acessadas por meio de APIs de áudio padrão do Windows;
    - Controles de fluxo de áudio e vídeo (cor, profundidade, esqueleto);
    - Funções do dispositivo de enumeração que permitem que um aplicativo possa usar mais de um sensor Kinect;
3. Componentes de áudio e vídeo;
    - Kinect Interface de Usuário Natural para rastreamento de esqueleto, de áudio, de cor e de imagem e profundidade;
4. DirectX Media Object (DMO) para microfone e localização da fonte de áudio;
5. APIS Padrão para Windows 7: áudio, fala e mídia APIs do Windows 7, e do SDK do Microsoft Speech. Essas APIs também estão disponíveis para aplicações desktop para Windows 8.

2.5 Entendendo o funcionamento do Kinect em uma aplicação

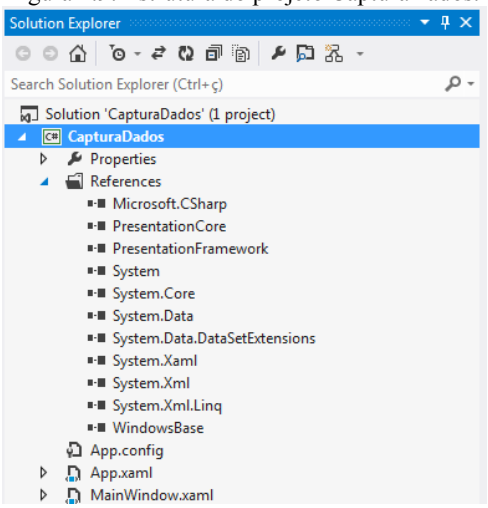
Com o *software* necessário para o desenvolvimento de aplicativos e com o suporte ao Kinect instalados e configurados, será demonstrado o seu funcionamento em duas aplicações: “CapturaDados” e “CapturaEsqueleto”, aplicações desenvolvidas durante o aprendizado da programação para o Kinect, necessário para o desenvolvimento do Ambiente Virtual de Apoio à Reabilitação Funcional de Membros Superiores (AVARFMS).

Para as aplicações, foi utilizado o *Visual Studio 2012 Ultimate*, com projeto do tipo *Windows Presentation Foundation (WPF) Application*, com a linguagem de programação C#, versão 4.0. Para esses aplicativos, foi utilizada a versão 1.5 do SDK, a mesma utilizada no AVARFMS.

2.5.1 Aplicação “CapturaDados”

A aplicação “CapturaDados” funciona de modo semelhante a uma câmera comum, que captura as imagens e as mostra na tela do computador. Essa aplicação foi bastante útil, pois possibilitou o entendimento de como o sensor de movimento é integrado aos aplicativos, como é inicializado e encerrado, e como sua câmera é ativada. Essa aplicação foi a primeira desenvolvida durante o aprendizado da programação para o sensor Microsoft Kinect. A figura 2.9 apresenta o "Solution Explorer" com a estrutura do projeto CapturaDados criada.

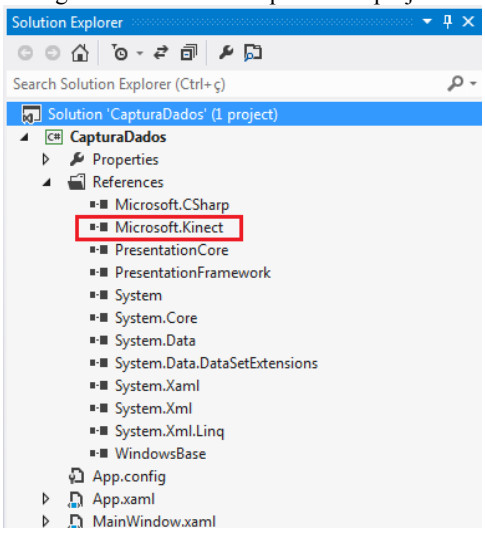
Figura 2.9: Estrutura do projeto CapturaDados.



Com o SDK do “Kinect for Windows” instalado, deve-se adicionar o arquivo “Microsoft.Kinect.dll”, que consiste em uma DLL com as funcionalidades para fazer o uso do

sensor. A figura 2.10 apresenta o "Solution Explorer" com a DLL incorporada ao projeto. Essa biblioteca encontra-se, para uma instalação padrão, no seguinte diretório "C:\Program Files\Microsoft SDKs\Kinect\v1.5\Assemblies\ Microsoft.Kinect.dll".

Figura 2.10: DLL incorporada ao projeto.



Com a DLL incorporada ao projeto, deve-se utilizar a diretiva "using" da linguagem C#, para fazer a importação da classe "Microsoft.Kinect", que se encontra na DLL “Microsoft.Kinect.dll”, para a classe desejada. Nesse caso, foi utilizada a classe "MainWindows". A figura 2.11 apresenta a classe “MainWindow” com a classe importada e um objeto de instância com visibilidade *private* do tipo “KinectSensor” declarado com o nome de “ _myKinect”.

Figura 2.11: Código da classe MainWindow.

```
inWindow.xaml    MainWindow.xaml.cs*
CapturaDados.MainWindow                MainWindow
 1  using System.Windows;
 2  using Microsoft.Kinect;
 3
 4  namespace CapturaDados
 5  {
 6      /// <summary>
 7      /// Interaction logic for MainWindow.xaml
 8      /// </summary>
 9      public partial class MainWindow : Window
10      {
11          private KinectSensor _myKinect;
12
13          public MainWindow()
14          {
15              InitializeComponent();
16          }
17      }
18  }
```

Para utilizar o sensor propriamente dito, deve-se definir qual o número do sensor que será utilizado. O SDK fornece suporte à utilização de vários sensores Kinect ao mesmo tempo. Para isso, a coleção "KinectSensor[N]" é utilizada. Neste caso, “N” representa o número do sensor que será utilizado e, após sua definição, deve-se iniciar o sensor. Como o sensor é um dispositivo sensível, deve-se também encerrá-lo pelo aplicativo (CATUHE, 2012; MILES, 2012; ASHLEY, 2012). Para isso, foram criados dois métodos: um para iniciar o sensor (InicializaKinect( ) { ... }) (Figura 2.12), e outro para finalizar o sensor (FinalizarKinect( ) { ... }) (Figura 2.13), implementados na classe “MainWindow” do projeto “CapturaDados”.

Figura 2.12: Método IniciarKinect

```
private void IniciarKinect()
{
    try
    {
        _myKinect = KinectSensor.KinectSensors[0];
        _myKinect.Start();
    } catch (Exception e)
    {
        MessageBox.Show("Erro ao iniciar o Kinect");
    }
}
```

Figura 2.13: Método FinalizarKinect.

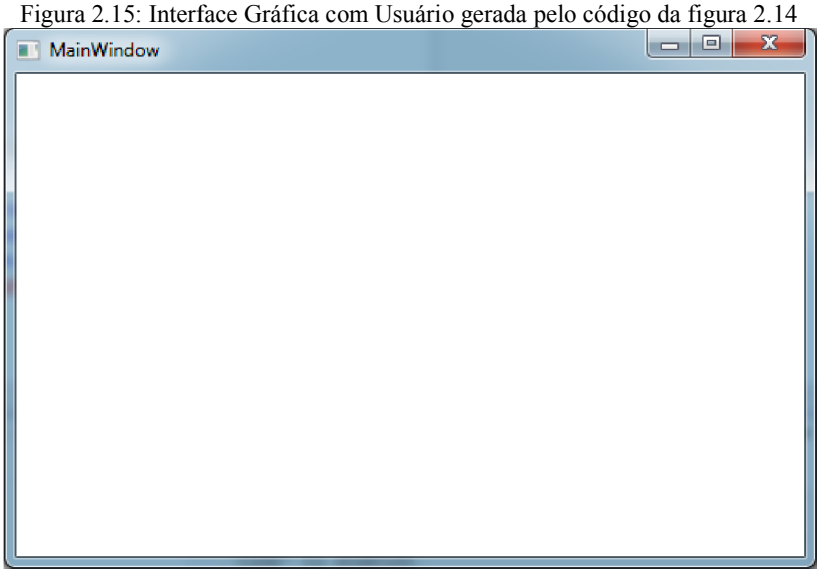
```
private void FinalizaKinect()
{
    try
    {
        if (_myKinect == null) return;
        _myKinect.Stop();
        _myKinect.Dispose();
        _myKinect = null;
    } catch (Exception e)
    {
        MessageBox.Show("Erro ao finalizar o Kinect");
    }
}
```

A tecnologia WPF, utilizada para o desenvolvimento das interfaces com o usuário do AVARFMS, utiliza uma linguagem de marcação “eXtensible Application Markup Language (XAML)” (Figura 2.14) para o desenvolvimento de Interfaces Gráficas com Usuário (GUI) Ricas (Figura 2.15), oferecendo um modelo consistente de programação para o desenvolvimento de aplicações, com uma clara separação entre interface com o usuário e lógica de negócios (NATHAN 2010).

Figura 2.14: Código XAML.

```
esign   XAML
1  <Window x:Class="CapturaDados.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="MainWindow" Height="350" Width="525">
5      <Grid>
6
7      </Grid>
8  </Window>
9
```

Figura 2.15: Interface Gráfica com Usuário gerada pelo código da figura 2.14

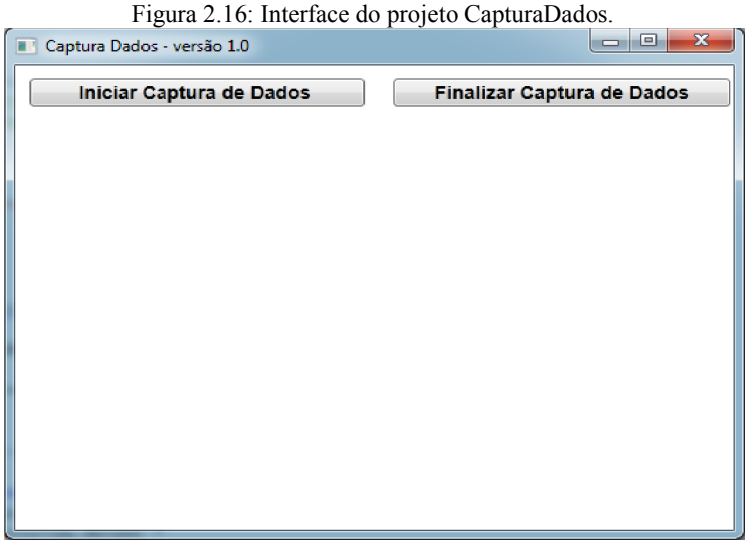


A lógica de negócio, que representa a parte pesada da codificação, ou seja, a utilização de estruturas condicionais, de repetição, de variáveis, de método, de cálculos matemáticos e etc (NATHAN 2010), foi escrita utilizando a linguagem C# versão 4.0.

O projeto exemplo contém os componentes WPF: “Button” e “Image”, sendo o componente “Button” responsável pela interação entre o usuário e a aplicação e o “Image”

pela interação entre os dados capturados pelo sensor Microsoft Kinect e a aplicação (Figura 2.16).

Figura 2.16: Interface do projeto CapturaDados.



O botão "Iniciar Captura de Dados" é responsável por disparar as ações que executam os métodos responsáveis por iniciar o sensor e pela captura das imagens que são adicionadas no componente “Image” (Figura 2.17).

Figura 2.17: Aplicação CapturaDados em funcionamento.

Para permitir que o sensor capture as imagens, é necessário habilitar a fluxo de dados para cores *(ColorStream.Enable())* e permitir que esse fluxo seja manuseado

(ColorFrameReady). Para isso, uma alteração no método "IniciarKinect() { .. }" foi feita (Figura 2.18).

Figura 2.18: Método IniciarKinect() com suporte a câmera

```
private void IniciarKinect()
{
    try
    {
        _myKinect = KinectSensor.KinectSensors[0];
        _myKinect.ColorStream.Enable();
        _myKinect.ColorFrameReady += MyKinectColorFrameReady;
        _myKinect.Start();
    } catch (Exception e)
    {
        MessageBox.Show("Erro ao iniciar o Kinect");
    }
}
```

O sensor utiliza um manipulador de eventos para gerenciar a captura das imagens (*ColorFrameReady += MyKinectColorFrame*), que consiste em um método que recebe dois argumentos: o primeiro (*object sender*) consiste em um objeto simples e o segundo (*ColorImageFrameReadyEventargs* e) é um objeto com as funcionalidades desejadas no momento (Figura 2.19). O método "MyKinectColorFrameReady" é responsável por controlar e mostrar as imagens capturadas pelo sensor.

Figura 2.19: Método MyKinectColorFrameReady.

```
37  void MyKinectColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
38  {
39      using (var imagemColorida = e.OpenColorImageFrame())
40      {
41          if (imagemColorida == null)
42              return;
43
44          if (_dadosImagem == null)
45          {
46              _dadosImagem = new byte[
47                  imagemColorida.Width *
48                  imagemColorida.Height *
49                  4
50              ];
51          }
52          imagemColorida.CopyPixelDataTo(_dadosImagem);
53
54          imgDadosImagem.Source = BitmapSource.Create(
55              imagemColorida.Width, imagemColorida.Height,
56              96, 96,
57              PixelFormats.Bgr32,
58              null,
59              _dadosImagem,
60              imagemColorida.Width * imagemColorida.BytesPerPixel
61          );
62      }
63  }
```

O sensor Kinect captura 30 fps[6]. Para garantir a precisão do sensor, o método "MyKinectColorFrameReady" é chamado para gerar uma nova imagem para essa captura. Como o processo depende de fatores externos (ex. sensor em perfeito funcionamento) e consome bastantes recursos, é aconselhável que ocorra dentro de um bloco *using*. Após o processamento, os recursos serão liberados e, caso ocorra algum erro, ele será tratado sem ocasionar danos ao funcionamento da aplicação (CATUHE, 2012; MILES, 2012; ASHLEY, 2012).

Para evitar um processo não desejado, primeiro é verificado se no parâmetro "e" já existe alguma imagem (*OpenCororImageFrame*) para ser tratada (linha 41, figura 2.19). Caso não exista o comando *return,* é chamado o que ocasiona a não execução do restante do método, caso contrário a execução do método continua verificando se já existe um conjunto de *bytes* do tamanho da imagem em memória (linha 44). Caso não exista, esse conjunto de *bytes* será criado. Esse processo serve para otimizar o processamento das imagens pelo sensor. Caso já exista, o processo continua copiando todos os *bytes* da imagem para a variável "_dadosImagem", utilizada para compor a imagem a ser mostrada na tela do computador. Com todos os dados (bytes) da imagem obtidos, o processo de criação da imagem será

[6] Quadros por segundo

iniciado (linhas 54 a 61). Para o exemplo, foi definida uma imagem com a utilização da classe “BitmapSource”, com as seguintes características:

- Dimensões: 640 * 480
- Resolução de 96 dpi por frame
- Formato do vídeo Bgr32
- Nenhuma paleta de cores

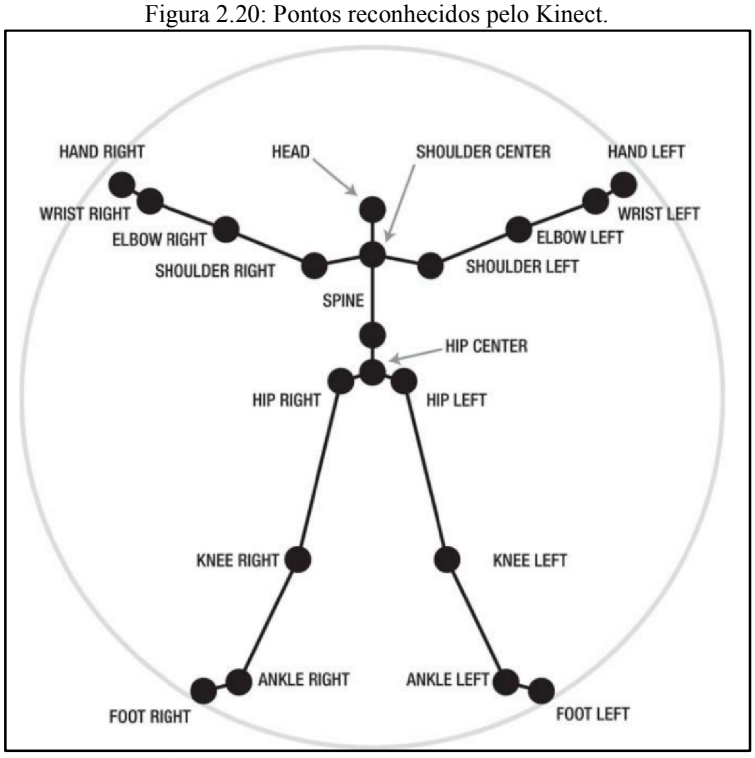
Para a próxima aplicação, “CapturaEsqueleto”, o processo de adicionar a DLL “Microsoft.Kinect.dll”, assim como a importação da classe “Microsoft.Kinect” e o método “FinalizarKinect( )” são os mesmos. Já o método para iniciar, “InicializaKinect( )”, irá sofrer algumas alterações que serão explicadas.

2.5.2 Aplicação “CapturaEsqueleto”

A aplicação “CapturaEsqueleto” demonstra como o sensor de movimento faz a captura do esqueleto de uma pessoa que se encontra em sua área de captura.

O Kinect reconhece 20 pontos de junção e 19 ossos (Figura 2.20) em um espaço 3D, capturando a posição nos eixos X, Y e Z de cada um dos pontos. Por meio das funcionalidades oferecidas pelo SDK, a aplicação “CapturaEsqueleto” pode manipular os valores fornecidos pelo sensor, obtendo, dessa forma, a coordenada do ponto na tela do computador (CATUHE, 2012; MILES, 2012; ASHLEY, 2012).

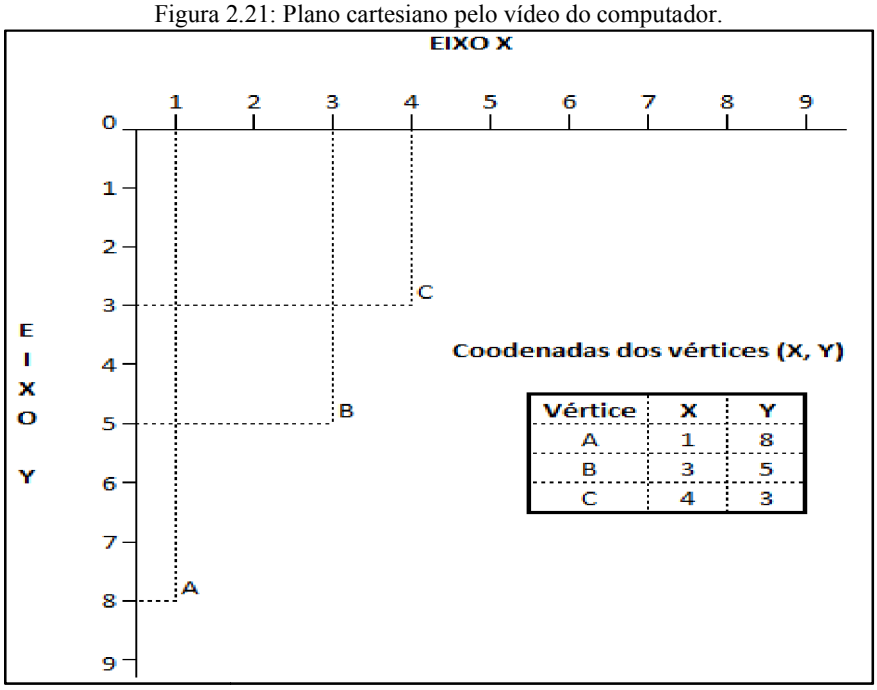
Figura 2.20: Pontos reconhecidos pelo Kinect.



Fonte: CATUHE, 2012

Para um melhor entendimento das coordenadas obtidas pelo Kinect, é necessário entender o sistema de coordenadas da tela do computador. A principal diferença entre o sistema de coordenadas tradicional e o utilizado pelo computador (figura 2.21) é que a origem dos eixos X e Y encontra-se no canto superior esquerdo. O sistema de coordenadas do computador é conhecido como coordenada de tela (LOBÃO 2010).

Figura 2.21: Plano cartesiano pelo vídeo do computador.



As coordenadas de tela são associadas à resolução de tela. Isso que dizer que se o monitor estiver configurado para uma resolução de 640x480, significa que o eixo X terá 640 pixels enquanto que o eixo Y terá 480 pixels. Para o projeto de exemplo, foi utilizada uma resolução de 640x480 pixels. Dependendo do modo que for habilitado pelo sensor, o nível de resolução e cores é alterado (CATUHE, 2012; MILES, 2012; ASHLEY, 2012).

Para o modo *ColorStream*:

- RgbResolution640x480Fps30 (Padrão)
- RgbResolution1280x960Fps12
- RawYuVResolution640x480Fps15
- YuvResolustion640x480Fps

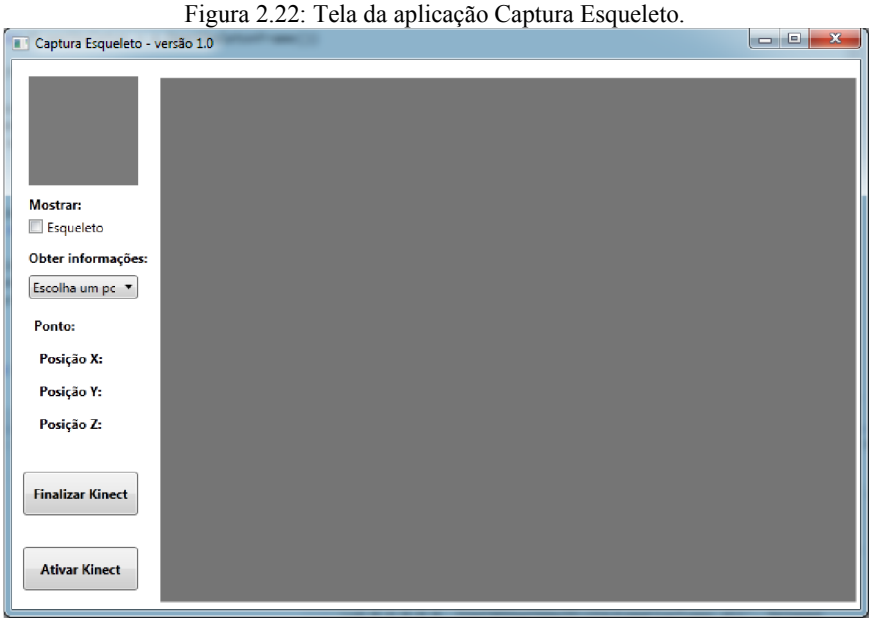
Para o modo *DepthImageFormat*:

- Resolution640x480Fps30
- Resolution320x240Fps30
- Resolution80x60Fps30

A Figura 2.22 mostra a tela principal da aplicação “CapturaEsqueleto”. Essa tela apresenta os seguintes componentes: duas caixas de imagem (quadros cinza); um componente de escolha (o usuário pode escolher se a imagem do esqueleto será mostrada ou não); um

componente Combo Box (o usuário escolhe qual ponto do corpo deseja obter informações); dois botões (um para iniciar e outro para finalizar o Kinect).

Figura 2.22: Tela da aplicação Captura Esqueleto.



A aplicação "CapturaEsqueleto" utiliza o modo *ColorStream* (linha 42, Figura 2.23) padrão. Também foi habilitado o modo *SkeletonStream* (linha 46), para poder fazer a captura do pontos reconhecidos. Quando o usuário ativa o sensor, os modos são habilitados pelo método "IniciaKinect()", além de configurar um vetor (linha 43,44) para armazenar todos os 6 esqueletos que o sensor pode administrar. A figura 2.23 mostra o método "IniciarKinect()".

Figura 2.23: Método IniciarKinect( ).

```
37  private void IniciarKinect()
38  {
39      try
40      {
41          _myKinect = KinectSensor.KinectSensors[0];
42          _myKinect.ColorStream.Enable();
43          _skeletonsFrames =
44              new Skeleton[_myKinect.SkeletonStream.FrameSkeletonArrayLength];
45          _myKinect.ColorFrameReady += MyKinectColorFrameReady;
46          _myKinect.SkeletonStream.Enable();
47          _myKinect.SkeletonFrameReady += MyKinectSkeletonFrameReady;
48          _myKinect.Start();
49      }
50      catch (Exception)
51      {
52          MessageBox.Show("Erro ao iniciar o Kinect");
53      }
54  }
```

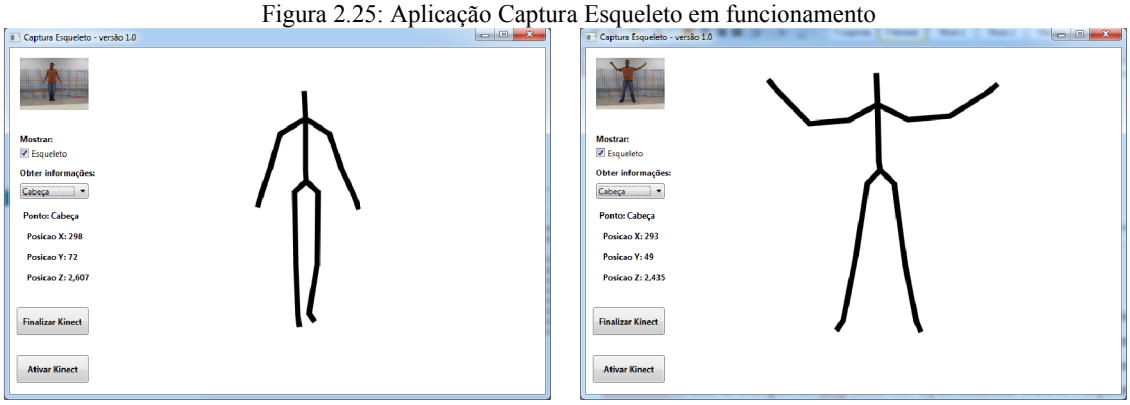
Com os modos necessários habilitados, o sensor inicia o seu processamento de captura dos movimentos do usuário, que é representado pela variável “esqueletoAtual”. O método responsável por esse processo é o “MyKinectSkeletonFrameReady()”, que recebe como argumento um objeto *sender* e outro do tipo “SkeletonFrameReadyEventArgs”, que representam as informações capturadas pelo Kinect e enviadas para a aplicação. Nesse método, é verificado se existe algum “esqueletoAtual” em frente ao Kinect. Caso exista, limpa-se a tela onde o “esqueletoAtual” será mostrado, utilizando uma consulta com a linguagem “Linq” (LIBERTY, 2011), no “array de esqueletos (_skeletonsFrames)”, criado no método “IniciarKinect”. Em seguida, realiza-se uma busca ao esqueleto mais próximo ao Kinect, armazenando suas informações na variável “esqueletoAtual”. Se esta variável não estiver com valor *null*, são iniciados os processos: mostrar o esqueleto na tela, usando o método “DesenhaEsqueleto()” e capturar as coordenadas X,Y,Z, usando o método “ObterInformações()” da variável “esqueletoAtual”. A Figura 2.24 mostra o método “MyKinectSkeletonFrameReady()”.

Figura 2.24: Método MyKinectSkeletonFrameReady.

```
56  void MyKinectSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
57  {
58      using (var frameSkeleton = e.OpenSkeletonFrame())
59      {
60          if (frameSkeleton == null) return;
61
62          LayoutRoot.Children.Clear();
63          frameSkeleton.CopySkeletonDataTo(_skeletonsFrames);
64          var esqueletoAtual = (from s in _skeletonsFrames
65                                where s.TrackingState == SkeletonTrackingState.Tracked
66                                select s).FirstOrDefault();
67
68          if (esqueletoAtual == null)
69              return;
70
71          if (ckbShowEsqueleto.IsChecked != true) return;
72          DesenhaEsqueleto(esqueletoAtual);
73          ObterInformacoes(esqueletoAtual);
74      }
75  }
```

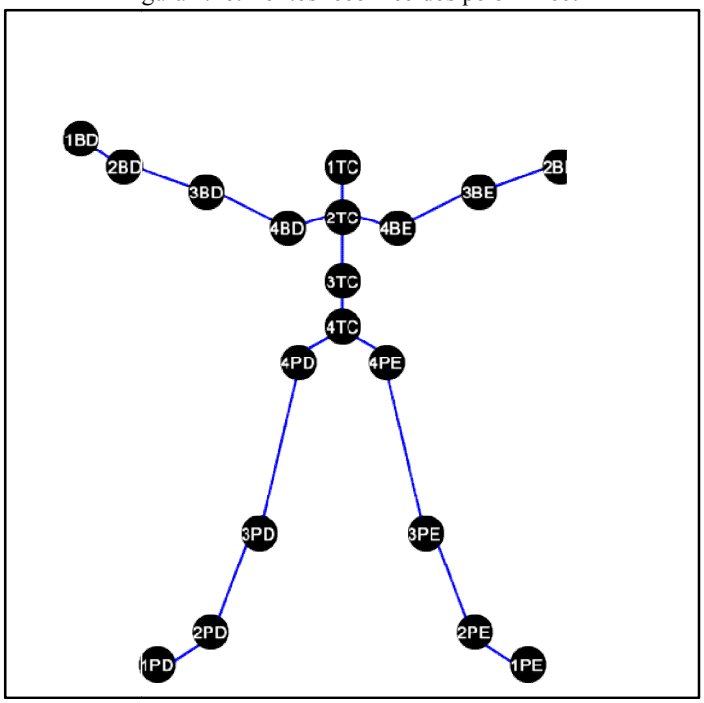
O método “DesenhaEsqueleto()” é responsável pela organização dos pontos que o Kinect reconhece, como mostrado na Figura 2.20. Com base na organização dos pontos, uma imagem será desenhada. A Figura 2.25 mostra a imagem que representa o usuário.

Figura 2.25: Aplicação Captura Esqueleto em funcionamento



O método “DesenhaEsqueleto()” faz a composição da figura da seguinte forma: pontos centrais (1TC, 2TC, 3TC, 4TC); pontos superiores esquerdos (1BE, 2BE, 3BE, 4BE); pontos superiores direitos (1BD, 2BD, 3BD, 4BD); pontos inferiores esquerdos (1PE, 2PE, 3PE, 4PE); pontos inferiores direitos (1PD, 2PD, 3PD, 4PD). A Figura 2.26 mostra o desenho completo. O método “DesenhaEsqueleto()”, encontra-se no anexo 7.1.1.

Figura 2.26: Pontos reconhecidos pelo Kinect



O método “ObterInformações()” é responsável por obter as coordenadas X, Y, Z do usuário atual. Com base nas informações do usuário atual e no ponto escolhido pelo usuário, os valores das coordenadas do ponto são obtidos por meio do mapeamento do esqueleto do usuário Kinect. Para facilitar a lógica desse método, é utilizada a instrução *switch case* do C#.

A Figura 2.27 mostra o trecho de código responsável por capturar as coordenadas da mão direita do usuário.

Figura 2.27: Trecho de código responsável pela captura da posição x,y da mão direita.

```
254  case "Mão direita":
255      posicao = _myKinect.MapSkeletonPointToColor(
256          esqueletoAtual.Joints[JointType. HandRight].Position,
257          ColorImageFormat.RgbResolution640x480Fps30);
258      posicaoX = posicao.X;
259      posicaoY = posicao.Y;
260      posicaoZ = esqueletoAtual.Joints[JointType.HandRight].Position.Z;
261      break;
```

O trecho de código da Figura 2.27 funciona da seguinte forma: com base no ponto *HandRight* (mão direita) do esqueleto corrente – representado pela variável "esqueletoAtual", com uma resolução de 640x480 pixels, o mapeamento do esqueleto é feito pelo método "MapSkeletonPointToColor", fornecido pelo SDK, e retorna as coordenadas X e Y em pixel. A coordenada Z é obtida através do atributo "Position.Z" do vetor de pontos (Joints). Nesse caso, o ponto desejado é passado como índice para o vetor de pontos e o valor Z em mm é retornado. O método "*ObterInformações*()" encontra-se no anexo 7.1.2.

O conhecimento adquirido com estas duas aplicações proporcionou o entendimento necessário para dar início ao desenvolvimento do AVARFMS. A parte mais complexa do desenvolvimento foi deixar de utilizar o teclado e o mouse para a interação com a aplicação, substituindo-os pelos movimentos do corpo.

3. Desenvolvimento do Ambiente Virtual de Apoio à Reabilitação Funcional de Membros Superiores (AVARFMS)

3.1 Introdução

O paciente acometido por AVE demonstra reduzida tolerância à realização de exercícios. Isso acarreta o comprometimento motor, social e funcional gerando um círculo vicioso, ocasionando o sedentarismo e isolamento social. Os pacientes sofrem de possíveis déficits advindos do AVE, o que traz uma a necessidade de (re)aprendizado de tarefas funcionais após o acidente (SARDI 2012).

Com o objetivo de adicionar uma nova ferramenta e avaliar os efeitos da realidade virtual no processo de reabilitação, o AVARFMS é proposto, trazendo estímulos motivacionais e lúdicos, de acordo com as necessidades do paciente e com os objetivos deste trabalho.

O AVARFMS destaca-se, dentre os ambientes virtuais utilizados, por manter o histórico das atividades desenvolvidas pelos pacientes. Este histórico poderá ser utilizado para acompanhar a evolução do paciente no processo de reabilitação, pois algumas métricas serão oferecidas pelo AVARFMS, além de permitir que o terapeuta monte o ambiente a ser utilizado pelo paciente.

3.2 Especificações de requisitos do AVARFMS

Segundo Bezerra (BEZERRA, 2007), o levantamento de requisitos de software consiste em uma técnica utilizada no processo de desenvolvimento, com o objetivo de auxiliar na definição das funcionalidades que o sistema computacional deve ter para atender às necessidades dos usuários. Os requisitos de *software* são divididos em dois grupos: requisitos funcionais e não funcionais.

3.2.1 Especificações de requisitos funcionais do AVARFMS

Os requisitos funcionais (RF) definem as funcionalidades que o sistema deve oferecer aos clientes (BEZERRA, 2007). Durante a fase de levantamento de requisitos, foi obtida a lista de requisitos funcionais listados na tabela 3.1:

Tabela 3.1: Requisitos funcionais do AVARFMS

RF 1.	O ambiente deve manter informações cadastrais sobre os pacientes.
RF 2.	O ambiente deve manter informações cadastrais sobre as atividades desenvolvidas para um paciente.
RF 3.	O ambiente deve manter o histórico das atividades desenvolvidas pelos pacientes.
RF 4.	O ambiente deve emitir gráficos de acompanhamento para: a trajetória; o tempo; o tamanho do trajeto percorrido; o erro em relação à trajetória desenvolvida pelo paciente e a trajetória perfeita; e um comparativo entre as trajetórias.

3.2.2 Especificações de requisitos não funcionais do AVARFMS

Os requisitos não funcionais (RNF) definem os aspectos, as condições de comportamento ou as restrições que um software deve prover, com o objetivo de apoiar os requisitos funcionais. Os requisitos não funcionais contemplam requisitos de interface homem-máquina, sistema operacional, sistema gerenciador de banco de dados (SGBD), plataforma de software, usabilidade, portabilidade, segurança e desempenho (BEZERRA 2007). Durante a fase de levantamento de requisitos, foi obtida a lista de requisitos não funcionais listados na tabela 3.2:

Tabela 3.2: Requisitos não funcionais do AVARFMS

RNF 1.	O ambiente deve utilizar um sistema gerenciador de banco de dados para manter as informações.
RNF 2.	A interação entre o paciente e o ambiente deve ser feita através de um sensor de movimento Microsoft Kinect®.
RNF 3.	A interação entre profissional de saúde e ambiente deve ser feita através de mouse, teclado e sensor de movimento Microsoft Kinect®.
RNF 4.	O ambiente deve funcionar sobre o sistema operacional Microsoft Windows®.

3.3 Algoritmos

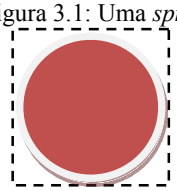
Segundo Cormen (2002), um algoritmo consiste em um procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída. Em outras palavras, um algoritmo é definido como uma sequência de passos computacionais que transformam os dados de entrada em informações úteis na saída.

Para que o AVARFMS fosse flexível e se adaptar às necessidades do paciente – sempre levando em consideração a sua condição física e cognitiva – e fornecesse, ao mesmo tempo, informações necessárias para a conclusão da tarefa, foi utilizada a técnica de detecção e colisão de *sprites*[7], utilizada no desenvolvimento de jogos eletrônicos (MADERA, 2011).

3.3.1 Algoritmo Bounding Boxes

Buscando um algoritmo com desempenho adequado, foi utilizado o "Bounding Boxes", que aproxima o formato da *sprite* com uma ou mais caixas (2D ou 3D). A Figura 3.1 mostra uma *sprite* que representa um círculo dentro de uma caixa 2D.

Figura 3.1: Uma *sprite*



Para implementar o teste de detecção e colisão entre duas *sprites* contidas em caixa, utilizando este algoritmo, deve-se verificar se a posição x, y do canto superior esquerdo de uma *sprite* A está contida dentro da caixa da *sprite* B. A listagem 3.1 apresenta o pseudocódigo deste algoritmo.

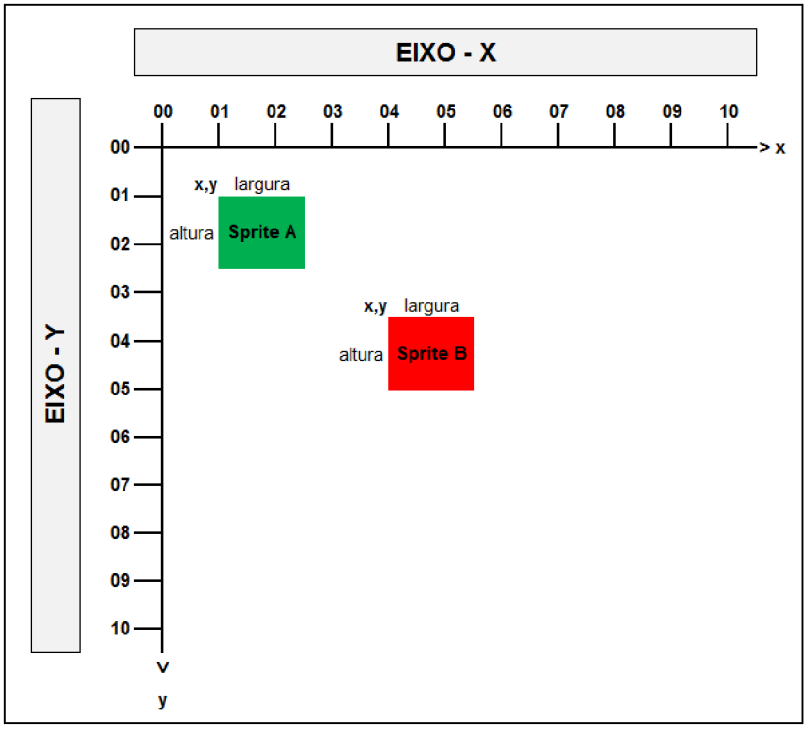
[7] Sprites é uma imagem bidimensional que pode ser manipulada de forma independente do resto da cena de um jogo. Este termo é tanto usado para descrever a imagem em si quanto para descrever a classe no programa do jogo que desenha a imagem (e que inclui outras propriedades, como velocidade, posição, largura, altura, etc.). Uma vez que computadores sempre desenham imagens 2D como retângulos, usualmente uma sprite inclui partes transparentes para dar a ilusão de um desenho não retangular. (LOBÃO 2010)

Listagem 3.1: Pseudocódigo Colisão por Bounding Boxes

```
01. função ColisãoPorBoundingBoxes(Sprite A, Sprite B)
02. inicio
03.     se posição x, y superior de qualquer um das Sprites A ou B esta
          contida dentro da outra então
04.         "Houve colisão"
05.     senão
06.         "Não houve colisão"
06. fim
```

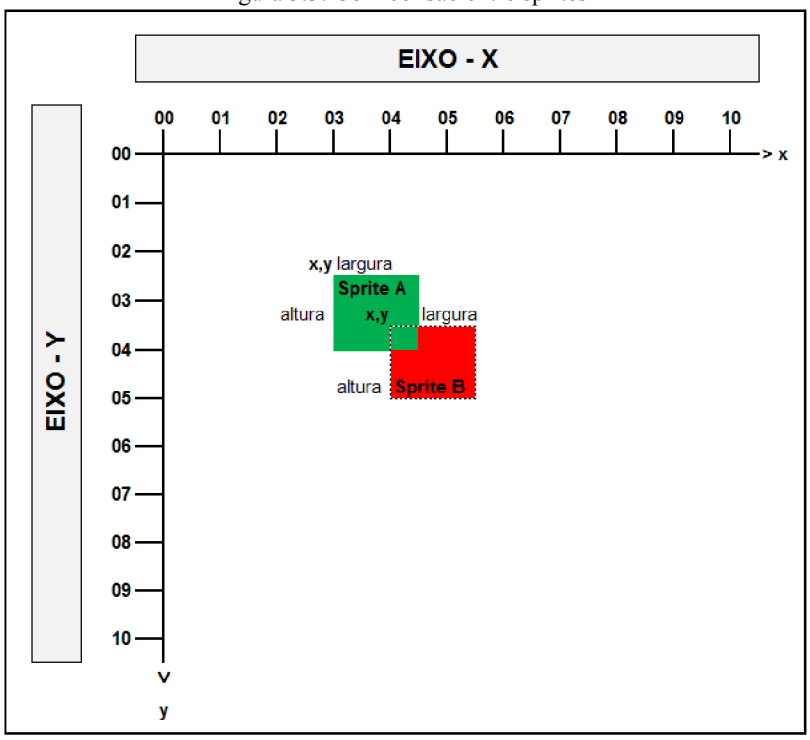
Para entender melhor o funcionamento do método "ColisãoPorBoundingBoxes" da listagem 3.1, observe a ilustração (Figura 3.2), que mostra que não ocorreu colisão entre a Sprite A (X = 1, Y = 1, Altura = 1,5, Largura = 1,5) e a Sprite B (X = 4, Y = 3,5, Altura = 1,5, Largura = 1,5). Nesse caso, as *sprites* são representadas por um quadrado.

Figura 3.2: Sem colisão entre sprites



A Figura 3.3 ilustra que ocorreu colisão entre a Sprite A (X = 3, Y = 2,5, Altura = 1,5, Largura = 1,5) e a Sprite B (X = 4, Y = 3,5, Altura = 1,5, Largura = 1,5). Neste exemplo, a posição x, y do canto superior da Sprite B está contida dentro da área da Sprite A.

Figura 3.3: Com colisão entre sprites



O ambiente virtual utiliza duas adaptações do algoritmo de detecção e colisão "Bounding Boxes": uma, com o objetivo de colocar o paciente na mesma posição em que o profissional de saúde responsável pela atividade estava, quando a atividade foi criada, e outra, com o objetivo de executar determinadas ações, quando a mão do paciente estivesse dentro de um dos pontos determinados pela atividade. Essas duas adaptações serão explicadas a seguir.

3.3.2 Algoritmo "Bounding Boxes" adaptado para posicionamento do paciente

Na sessão 3.4.3 – Caso de uso: Criar atividade –, será explicado como a atividade para o paciente deve ser criada pelo profissional de saúde. Nesse momento, as posições x, y, z do ponto ombro central, reconhecido pelo sensor Kinect, são armazenadas no banco de dados. Essas posições são utilizadas para auxiliar o posicionamento do paciente em frente ao sensor, como será explicado na sessão 3.4.4 – Caso de uso: Desenvolver atividade.

Levando em conta que, apesar do profissional de saúde responsável pelo tratamento do paciente conhecer bem a sua condição clínica, o seu biótipo físico é diferente do paciente, foi necessária uma adaptação do algoritmo "Bounding Boxes", para auxiliar no posicionamento do paciente na mesma posição (X, Y, Z) do profissional de saúde quando a atividade foi criada.

No momento em que a atividade é escolhida, o ambiente virtual mostra um quadrado com as bordas em vermelho (Figura 3.4). O ponto ombro central do paciente não pode estar fora deste quadrado; caso isso ocorra, uma mensagem será mostrada (Figura 3.5). Para fins de demonstração, as opções “Mostrar Paciente”, “Mostrar ponto” e “Mostrar linha” não foram marcadas na tela de manutenção de atividades do AVARFMS (Figura 3.6).

Figura 3.4: Ponto ombro central do paciente dentro do limite

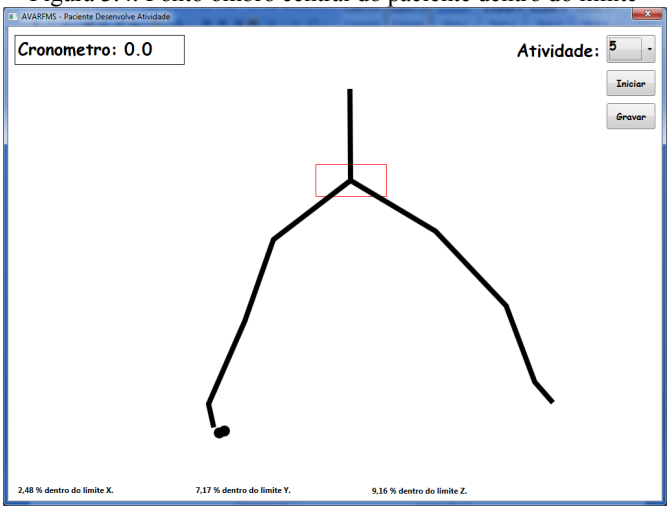


Figura 3.5: Ponto ombro central do paciente fora do limite

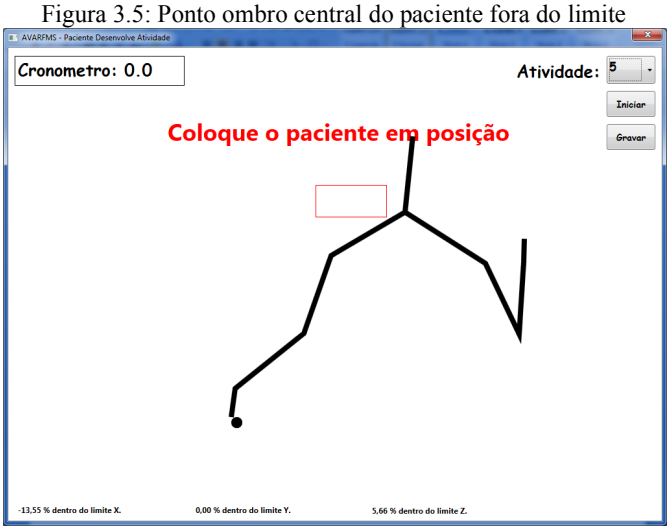
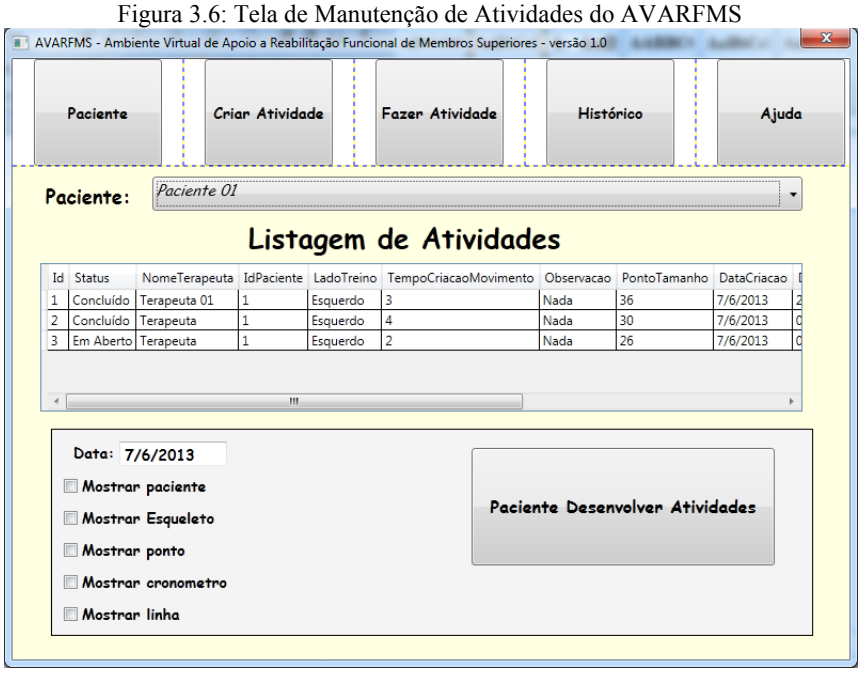


Figura 3.6: Tela de Manutenção de Atividades do AVARFMS



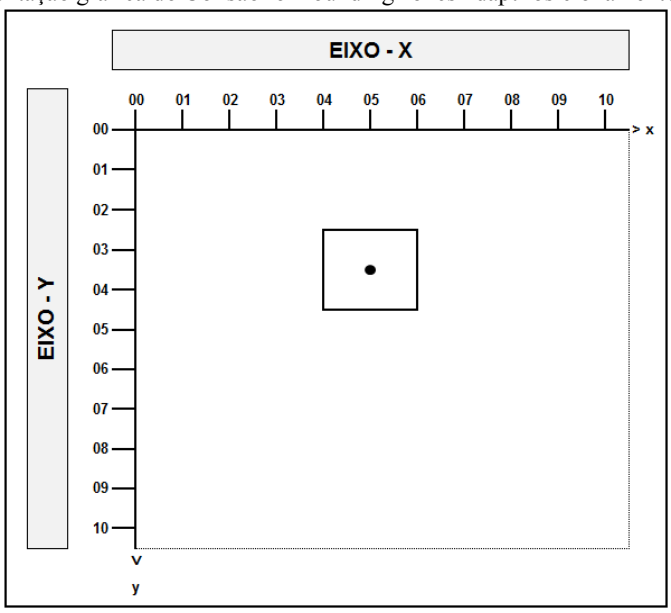
Considere que o retângulo com bordas vermelhas seja a *sprite A* e o ponto ombro central do paciente seja a *sprite B*. Para implementar o teste de detecção e colisão com o Algoritmo "Bounding Boxes" adaptado para posicionamento do paciente entre estas duas *sprites*, deve-se verificar se toda a *sprite B* está contida dentro da *sprite A*. São consideradas as posições X e Y do terapeuta para a criação do retângulo com bordas vermelhas. A posição Z também é considerada no posicionamento do paciente. A listagem 3.2 apresenta o pseudocódigo deste algoritmo.

Listagem 3.2: Pseudocódigo ColisãoPorBoundingBoxesAdaptPosicionamento

```
01. função ColisãoPorBoundingBoxesAdaptPosicionamento(Sprite A, Sprite B)
02. inicio
03.     se Sprites B esta contida dentro da Sprite A então
04.         "Dentro dos limites"
05.     senão
06.         "Fora dos limites"
06. fim
```

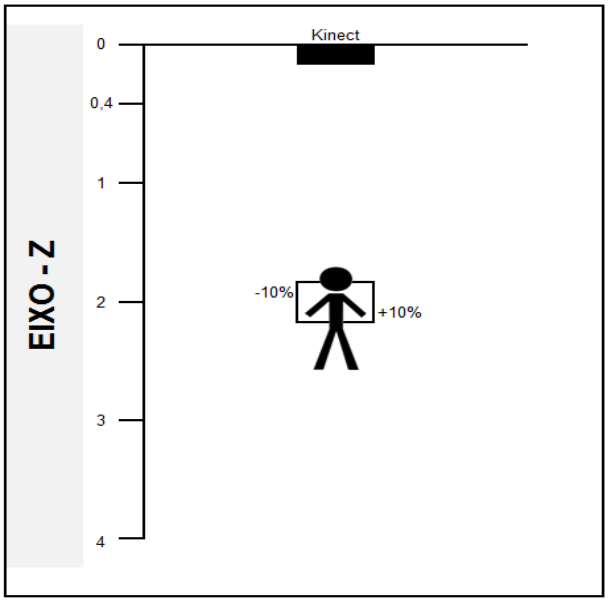
Para entender melhor o funcionamento do método "ColisãoPorBoundingBoxesAdaptPosicionamento" da listagem 3.2, compare o código com a ilustração da figura 3.7, que mostra que não ocorreu colisão entre a *Sprite A* (X = 4, Y = 2,5, Altura = 3, Largura = 3) e a *Sprite B* (X = 5, y = 3,5, Altura = 1, Largura = 1),

Figura 3.7: Representação gráfica do ColisãoPorBoundingBoxesAdaptPosicionamento para os eixos x,y.



O posicionamento do paciente leva em conta um plano em 3 dimensões (X, Y, Z). O posicionamento das posições X e Y foi explicado acima. Em relação ao ponto Z, o paciente deve estar dentro de um limite pré-estabelecido do ambiente virtual, de 10% para mais ou para menos, em relação à posição do profissional de saúde responsável pela criação da atividade. A figura 3.8 demonstra esse posicionamento. No exemplo, o paciente pode estar a 1,80m ou 2,20m da localização original do profissional de saúde que, no caso, era de 2,00m exatos.

Figura 3.8: Representação gráfica do ColisãoPorBoundingBoxesAdaptPosicionamento para o eixo z.



3.3.3 Algoritmo "Bounding Boxes" adaptado para acerto dos pontos

Com o paciente posicionado, o mesmo deve executar a atividade desenvolvida para ele, como será explicado na sessão 3.4.4 – Caso de uso: Desenvolver atividade. Quando o profissional de saúde clicar no botão iniciar, o ambiente virtual ficará em um estado de prontidão, aguardando que o paciente leve o ponto que se encontra sobre a sua mão na tela, até o ponto com o número zero ou em vermelho. Nesse momento, o Algoritmo "Bounding Boxes", adaptado para acerto dos pontos, entra em funcionamento.

Desse modo, considere que o ponto que aparece na tela seja a *Sprite A* e o ponto que se encontra sobre a mão do paciente seja a *Sprite B* para implementar o teste de detecção da colisão entre duas *sprites* contidas em caixa. Utilizando este algoritmo deve-se verificar se a *Sprite B* está contida dentro da *Sprite A*. A listagem 3.3 apresenta o pseudocódigo deste algoritmo.

Listagem 3.3: Pseudocódigo ColisãoPorBoundingBoxesAcertoPonto

```
01. função ColisãoPorBoundingBoxesAcertoPonto(Sprite A, Sprite B)
02. inicio
03.     se Sprite B está contida na Sprite A então
04.         "executa função AlterarEstadoPonto"
05. fim
```

Obseve que, caso a *Sprite B* esteja contida na *Sprite A*, a função "AlterarEstadoBola" é executada. Essa função altera a cor do ponto, de vermelho para verde, e ativa o próximo ponto que o paciente deve alcançar, fazendo, dessa forma, que ele faça o trajeto desejado (Listagem 3.4).

Listagem 3.4: Pseudocódigo AlterarEstadoPonto

```
01. função AlterarEstadoPonto ()
02. inicio
03.     Alterar cor do ponto corrente de vermelho para verde.
04.     Altera cor do próximo ponto de cinza para vermelho.
05.     Posiciona como próxima alvo o ponto em vermelho.
06. fim
```

3.3.4 Algoritmo para suavização da trajetória do paciente

Com o objetivo de melhorar a confiabilidade dos dados capturados durante a execução da atividade pelo paciente, é aplicado um filtro, utilizando uma média móvel na trajetória. Foi necessário fazer a suavização da trajetória por existir uma variação da localição do ponto que se encontra sobre uma das mãoes do paciente (esquerda/direita) quanto a mão a passada

próximo de outro ponto, cabeça por exemplo, o sensor perde o foco. A listagem 3.5 apresenta o pseudocódigo deste algoritmo. Isso é feito por meio da função “SuavizarTrajétoria”, listagem 3.6, que será explicada logo a seguir.

Listagem 3.5: Pseudocódigo MediaMovel

```
01. função MediaMovel (pontosBrutos, pontoOrigem, pontoDestino, j)
02. inicio
03.     somaX = 0.0;
04.     somaY = 0.0;
05.     i = pontoOrigem;
06.     enquanto i for menor que pontoDestino faça
07.     inicio
08.        somaX = somaX + pontosBrutos[i].X;
09.        somaY = somaY + pontosBrutos[i].Y;
10.        incrementar i em um
11.     fim
12.     se pontoOrigem for igual a zero então
13.            gerar novo ponto com valor de X igual a somaX / i e Y =
somaY/i
14.     senão
15.         gerar novo ponto com valor de X igual a somaX/j e Y = somaY/j}
16.     fim
17. fim
```

A função “MediaMovel” funciona da seguinte forma: recebe, como dados de entrada, os pontos da trajetória do paciente, ponto de origem, destino e o tamanho da janela. São iniciados os valores das somas X e Y com zero, e a variável “i”, auxiliar, com o valor do ponto de origem. Uma estrutura de repetição é executada até que o valor da variável “i”, que é incrementada de um em um, seja maior que o ponto de destino. Dentro da estrutura de repetição é executada a soma de todos os valores de X e Y dos pontos da trajetória do paciente. Ao finalizar o laço, é feita a verificação do ponto de origem: se for igual a zero, a média móvel é calculada, levando em conta a quantidade de pontos do intervalo, determinada pela variável “i”; caso contrário, a média móvel é calculada levando em conta o tamanho da janela.

Listagem 3.6: Pseudocódigo SuavizarTrajetoria

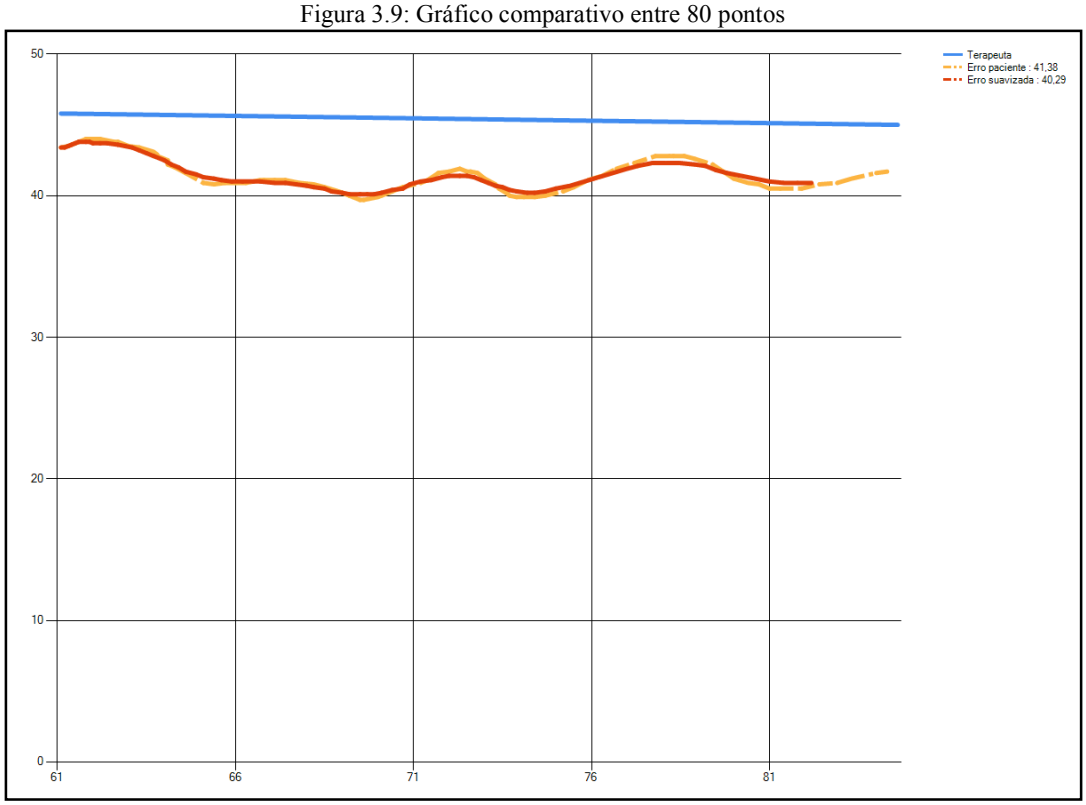
```
01. Função SuavizarTrajetoria(PontosTrajetoria)        {
02.   janela = 10;
03.   p1 = 0;
04.   p2 = 0;
05.   equanto existir pontos da trajetória faça
06.   inicio
07.     se o ponto da trajetória for um dos dez primeiros
08.          calcule a média móvel para o ponto suavizado do ponto p1 até
p2.
09.        incremente p2 em um
10.     senão
11.        incremente p1 em um
```

```
12.         calculo a média móvel para o ponto suavizado do ponto p1 até
            p2, usando a janela
13.         incremente p2 em um
14.       fim se
15.     fim equanto
16.     retorne o ponto suavizado
17. fim
```

A função “SuavizarTrajetoria” funciona da seguinte forma: recebe, como dados de entrada, os pontos da trajetória do paciente; são iniciados os valores da janela com 10, o ponto origem p1 com zero e o ponto destino p2 com zero; é executada uma estrutura de repetição enquanto existirem pontos da trajetória. Dentro da estrutura de repetição são executados os seguintes testes: para os 10 primeiros pontos da trajetória, calcule a média móvel para o ponto suavizado do ponto origem p1 até o destino, utilizando p1 como fator determinante da média; para os outros pontos da trajetória, calcule a média utilizando o valor da janela como fator determinando da média.

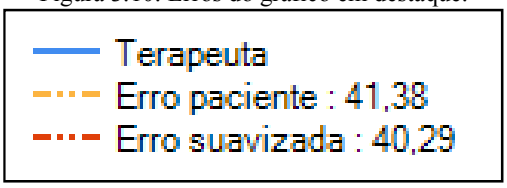
A Figura 3.9 ilustra um gráfico comparativo entre 80 pontos obtidos de uma trajetória executada e a trajetória suavizada.

Figura 3.9: Gráfico comparativo entre 80 pontos



Pode-se observar que o gráfico (Figura 3.9) ilustra a suavização da trajetória do paciente (linha de cor laranja), com erro 40,29 e a trajetória não suavizada (linha da cor amarela), com erro 41,38, em relação à trajetória do terapeuta (linha da cor azul). Para uma melhor apresentação dos resultados, observe a figura 3.10, que mostra os erros. Esse erro indica a diferença entre a trajetória real obtida com o Kinect e a trajetória Suavizada.

Figura 3.10: Erros do gráfico em destaque.



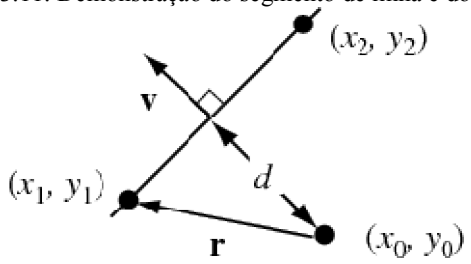
A função "CalcularErro" possui o objetivo de calcular o erro entre as trajetórias do terapeuta e do paciente não suavizada, e entre as trajetórias do terapeuta e do paciente suavizada. A listagem 3.7 apresenta o pseudocódigo desta função.

Listagem 3.7: Pseudocódigo CalculaErro

```
01. função CalcularErro(Trajetoria)
02. inicio
03.     para todos os pontos da trajetória faça
04.         armazena a distância calcula com a função
                DistanciaSegmentoLinha.
05.     fim para
06.     erro = Normalizar distâncias / quantidade de distâncias
07. fim
```

A função "CalcularErro" funciona da seguinte forma: recebe, como dados de entrada, os pontos da trajetória desejada (paciente não suavizada, paciente suavizada); para todos os pontos da trajetória, calcula e armazena a distância entre a linha formada pela trajetória criada pelo profissional de saúde e os pontos da trajetória desejada (Figura 3.11).

Figura 3.11: Demonstração do segmento de linha e do ponto.



A distância (d) entre um ponto e uma reta é calculada, unindo o próprio ponto (x0, y0) à reta, através de um segmento (x1, y1) a (x2, y2), que deverá formar com a reta um ângulo reto (90º) (Figura 3.11). Para estabelecer a distância entre os dois, foi utilizada a equação geral da reta e da coordenada do ponto (Equação: 4.1).

$$d = \left|\hat{\mathbf{v}} \cdot \mathbf{r}\right| = \frac{\left|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1)\right|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}. \qquad \text{Equação: 4.1}$$

Esses pontos são utilizados também pela função “DistanciaSegmentoLinha” (Listagem 3.8).

Listagem 3.8: Pseudocódigo DistanciaSegumentoLinha

```
01. função DistanciaSegumentoLinha(Origem, Destino, Ponto)
02. inicio
03.     vx = origemX – pontoX
04.     vy = origemY - pontoY
05.     ux = destinoX - origemX
06.     uy = destinoy – origemY
08.     uy = destinoy – origemY
09.     lenSqr = (ux * ux + uy * uy)
10.     detP = -vx * ux + -vy * uy;
11.     se detP < 0 então
12.         r = Normalizar(origem, destino)
13.     senão se detP > lenSqr então
14.         r = Normalizar(destino, trajeto)
15.     senão
16.         r = |uv * vy – uy * vx| / raiz(lenSqr)
17.     fim se
07. fim
```

A função “DistanciaSegmentoLinha” funciona da seguinte forma:

1) recebe, como dados de entrada, os pontos que representam a origem, o destino e um ponto qualquer do tipo da trajetória desejada (não suavizada ou suavizada);
2) calcula a diferença entre os valores de X e Y (vx, e vy), para o ponto que representa a origem ao ponto da trajetória;
3) calcula a diferença entre os valores de X e Y (ux e uy) entre o ponto que representa o destino e o ponto que representa a origem;
4) calcula o comprimento do segmento de reta ao quadrado (lenSqr) e a deteminante da matriz (detP) composta por vx, ux, vy, uy; caso o valor da determinante seja

menor que zero, é necessário normalizar os pontos de origem e destino da trajetória; caso o valor da determinante seja maior que o comprimento do segmento de reta ao quadrado (lenSqr), é necessário normalizar os pontos de destino e origem da trajetória; caso contrário, normalizar o ponto da trajetória.

A normalização é feita utilizando a equação 4.2, que é serve para calcular a distância entre dois pontos.

$$D = \sqrt{(xd - x0)^2 + (yd - y0)^2} \qquad \text{Equação: 4.2}$$

3.4 Modelos de caso de uso do AVARFMS

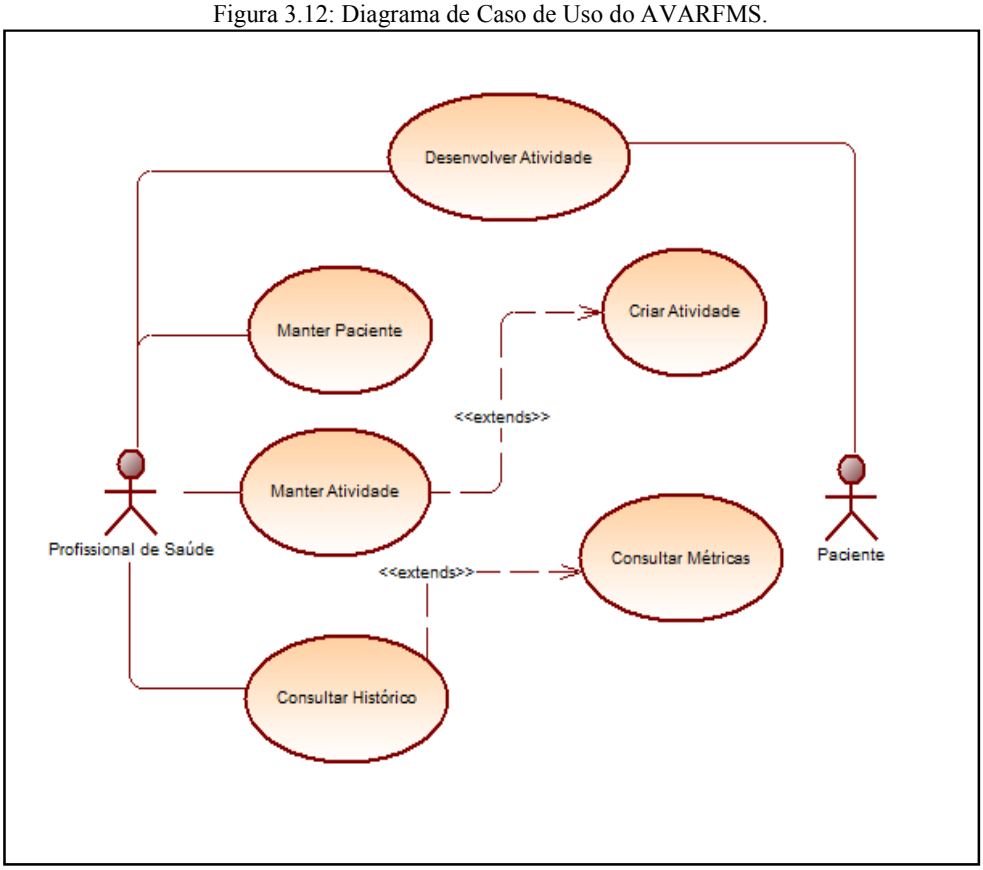
A modelagem de casos de uso (MCU) consiste em uma representação das funcionalidades externamente observáveis do sistema em desenvolvimento e dos elementos externos que irão interagir com ele. A MCU é um modelo de análise que representa o refinamento dos requisitos funcionais do sistema. A ferramenta utilizada pela Linguagem de Modelagem Unificada (UML) no processo de modelagem de casos de uso é o diagrama de caso de uso. Esse modelo é utilizado para direcionar as diversas tarefas posteriores do desenvolvimento do sistema, além de forçar o desenvolvedor a manter o foco nas necessidades do sistema (BEZERRA, 2007).

Casos de uso descrevem cenários que demonstram as funcionalidades de um *software*, levando em consideração as interações entre os usuários (atores) e o sistema em desenvolvimento, atendendo os requisitos levantados anteriormente (BEZERRA, 2007). Durante o processo de análise do ambiente virtual, os seguintes atores foram identificados:

- Profissional de saúde: indíviduo responsável pelo tratamento de um paciente, pode ser um terapeuta ocupacional, fisioterapeuta ou qualquer profissional envolvido no processo de reabilitação de pessoas vitimadas por um AVE, que deseje utilizar o AVARFMS como ferramenta de apoio para uma vítima de um AVE.
- Paciente: pessoa acometida por um AVE, interessada em utilizar o AVARFMS como ferramenta de apoio ao seu tratamento.

Durante a fase de levantamento de requisitos, foi obtido o diagrama de caso de uso (Figura 3.12), que apresenta a interação entre os casos de usos e seus respectivos atores.

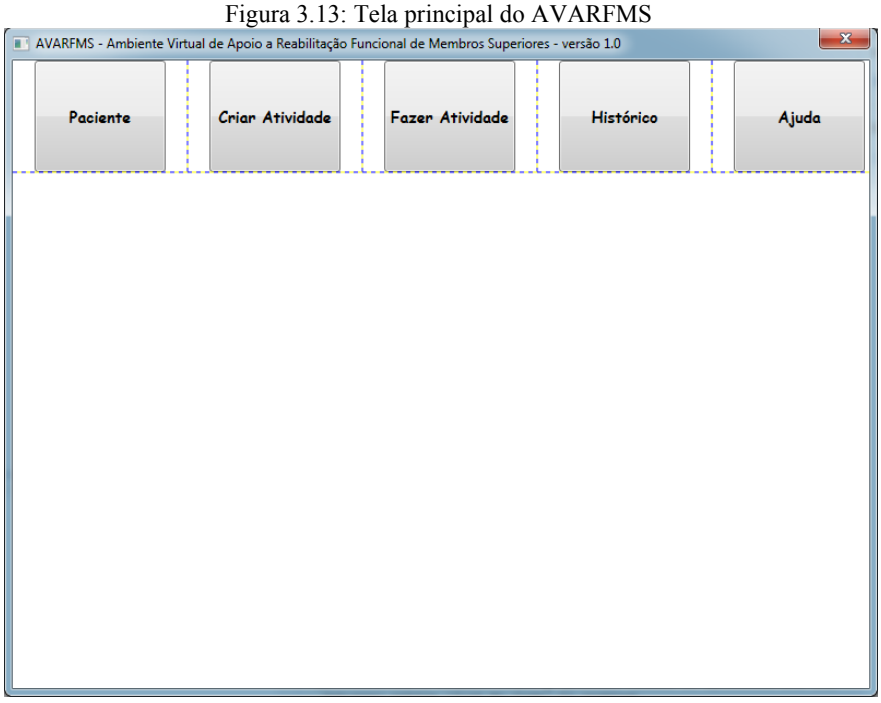
Figura 3.12: Diagrama de Caso de Uso do AVARFMS.



3.4.1 Caso de uso: Manter Paciente

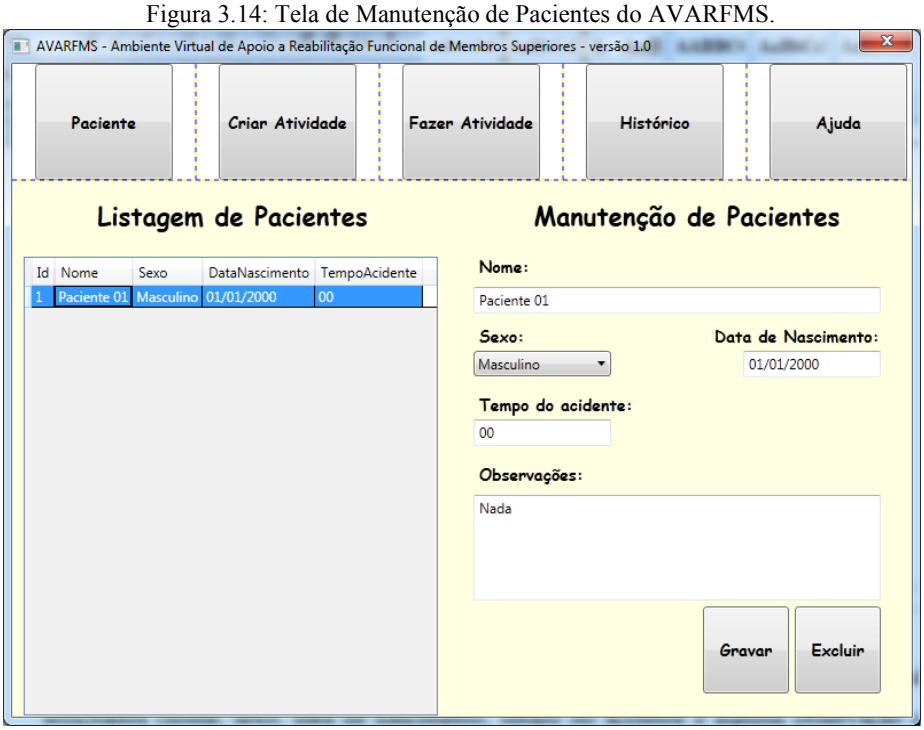
O caso de uso “manter paciente” inicia-se quando se clica no botão “Paciente” da tela Principal (Figura 3.13) do AVARFMS. Nesse momento, o ambiente verifica se existe um banco de dados pré-configurado; caso não exista, o sistema cria um novo banco de dados, vazio, e carrega a tela “Manutenção dos Pacientes” (Figura 3.14). No carregamento dessa tela, os dados de pacientes já cadastrados no AVARFMS serão carregados e mostrados.

Figura 3.13: Tela principal do AVARFMS



Na tela "Manutenção dos Pacientes" (Figura 3.14), o profissional de saúde pode fazer as seguintes ações: adicionar um novo paciente, apagar um paciente existente e alterar os dados de um paciente.

Figura 3.14: Tela de Manutenção de Pacientes do AVARFMS.



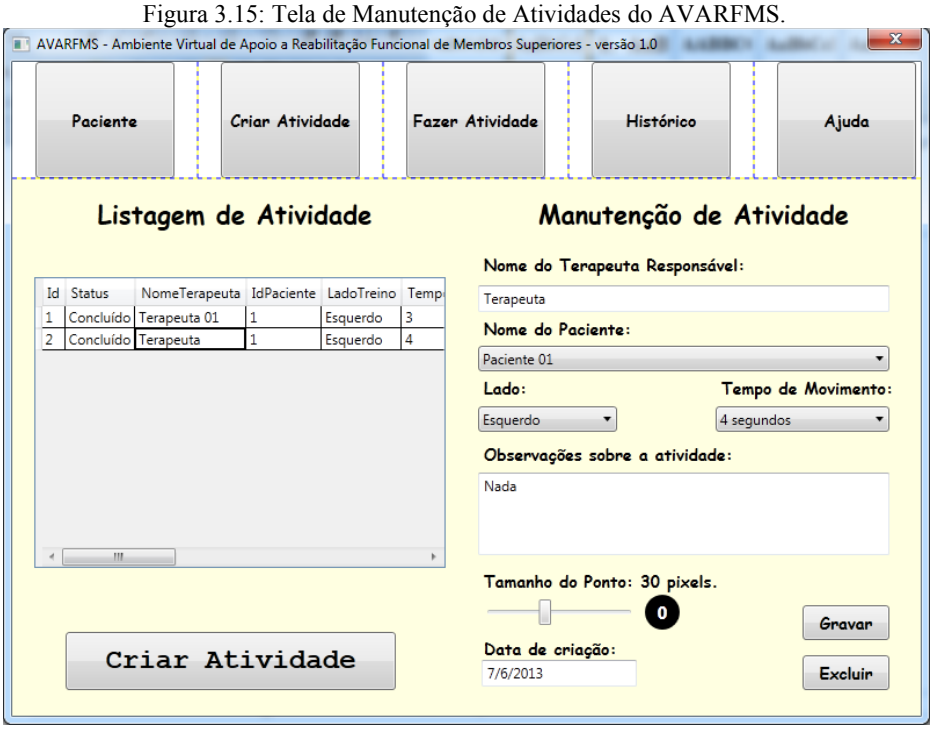
Para adicionar um novo paciente, o profissional de saúde deve informar os dados solicitados (nome, sexo, data de nascimento, tempo do acidente e alguma observação sobre o paciente) na parte de “Manutenção de Pacientes” e clicar no botão “Gravar”. Nesse momento, os dados são salvos no banco de dados e um código de identificação sequencial para o paciente é criado.

Para apagar ou alterar os dados de um paciente, o profissional de saúde deve dar um duplo clique sobre qualquer informação da “Listagem de Paciente”. Quando os dados forem carregados na parte de “Manutenção de Pacientes”, basta o profissional de saúde executar a operação desejada. Por exemplo: para apagar algum dado, basta clicar no botão “Excluir”, e os dados do paciente serão excluídos do banco de dados; para alterar algum dado, basta fazer as alterações desejadas, clicar no botão “Gravar” e as alterações serão salvas no banco de dados. Após a execução de qualquer uma dessas operações, a listagem com os dados será carregada novamente, refletindo as ações feitas.

3.4.2 Caso de uso: Manter Atividade

O caso de uso Manter Atividade inicia-se quando se clica no botão “Criar Atividade da tela Principal” (Figura 3.13) do ambiente. Nesse momento, o ambiente carrega a tela de Manutenção das Atividades (Figura 3.15), na qual as atividades existentes serão listadas e os nomes dos pacientes cadastrados serão carregados. O profissional de saúde pode fazer as seguintes ações: adicionar uma nova atividade, apagar uma atividade existente, alterar os dados de uma atividade e iniciar a criação de uma atividade.

Figura 3.15: Tela de Manutenção de Atividades do AVARFMS.



Para adicionar uma nova atividade, o profissional de saúde deve informar os dados solicitados (nome do terapeuta, nome do paciente, lado para treino, tempo para criação da atividade, alguma observação sobre a atividade, o tamanho do ponto a ser mostrado na tela e a data da criação da atividade) na parte de "Manutenção de Atividades" e clicar no botão "Gravar". Nesse momento, os dados são salvos no banco de dados e um código de identificação sequencial para a atividade será gerado. Essas informações serão utilizadas pelo profissional de saúde no momento da criação da atividade, conforme será explicado no "Caso de Uso: Criar Atividade" (Sessão 3.4.3).

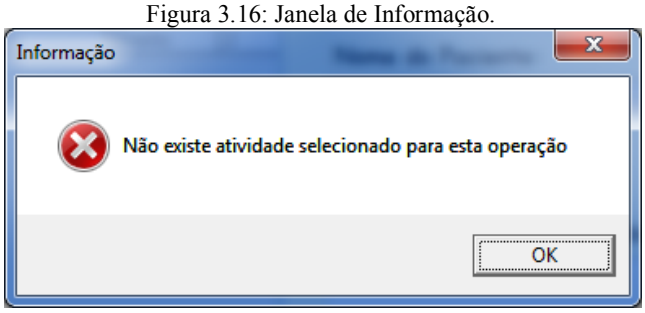
As atividades cadastradas irão ficar com o status "ABERTO". Quando a atividade for efetivamente criada, esse status será alterado para "CONCLUÍDO". Caso o status esteja "ABERTO", as informações de ajuste do paciente em frente ao sensor de movimento da Microsoft Kinect® (posição x, y e z do ponto ombro central do profissional de saúde) e a quantidade de vezes que essa atividade foi desenvolvida pelo paciente estarão com o valor padrão zero. Caso o status esteja "CONCLUÍDO", esses dados estarão com os valores utilizados pelo profissional de saúde no momento da criação da atividade. O número de vezes somente será alterado após a execução da atividade pelo paciente, conforme será explicado no "Caso de Uso: Desenvolver Atividade" (Sessão 3.4.4).

Para apagar ou alterar os dados de uma atividade, o profissional de saúde deve dar um duplo clique sobre qualquer informação da “Listagem de Atividades”. Quando os dados forem carregados na parte de “Manutenção de Atividades”, basta o profissional de saúde executar a operação desejada: para apagar os dados, deve-se clicar no botão “Excluir” e os dados da atividade serão excluídos do banco de dados; para alterar, deve-se fazer as alterações desejadas, clicar no botão “Gravar” e as alterações serão salvas no banco de dados. Após a execução de qualquer uma dessas operações, a listagem com os dados será carregada novamente, refletindo as ações feitas.

3.4.3 Caso de uso: Criar atividade

O caso de uso “Criar Atividade” inicia-se quando o profissional de saúde seleciona uma atividade na “Listagem de Atividades” da tela “Manutenção de Atividades” (Figura 3.15) e clica no botão “Criar Atividade”. Caso nenhuma atividade seja solicitada, uma janela de informação com a mensagem “Não existe atividade selecionada para esta operação” (Figura 3.16) será mostrada.

Figura 3.16: Janela de Informação.

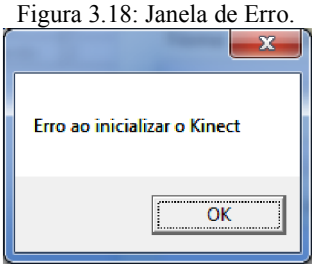


Caso a atividade esteja selecionada, inicia-se o carregamento da tela “Terapeuta Desenvolve Atividade para Paciente” (Figura 3.17).

Figura 3.17: Tela Terapeuta Desenvolve Atividade para Paciente

Nesse momento, o ambiente verifica a existência de um sensor de movimento Microsoft Kinect®. Caso encontre algum problema com o sensor, uma janela de erro com a mensagem "Erro ao inicializar o Kinect" (Figura 3.18) é mostrada; caso contrário, o sensor é inicializado e a tela "Terapeuta Desenvolve Atividade Para Paciente" é carregada (Figura 3.17).

Figura 3.18: Janela de Erro.



Nesse momento, as informações definidas na tela "Manutenção das Atividades" (Figura 3.15) serão utilizadas da seguinte forma:

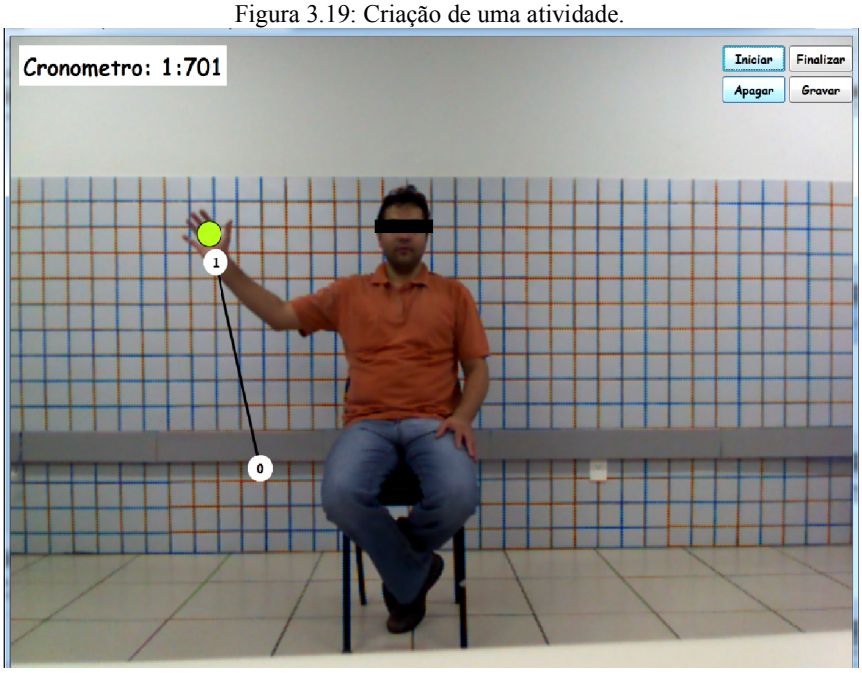
- O tempo em segundo: é utilizado para determinar o tempo que o profissional de saúde deve ficar com a mão parada em algum lugar da tela para que um ponto apareça na tela. Após o ponto aparecer na tela, o profissional deve mover a mão para o outro ponto para que um novo ponto apareça. Os pontos recebem uma numeração sequencial;

- O lado (esquerdo ou direito): é utilizado para determinar para qual lado do corpo a atividade será criada. Nesse caso, um ponto aparece sobre a mão do profissional de saúde;
- O tamanho do ponto: é utilizado para determinar o tamanho do ponto que será mostrado na tela para o paciente.

Observe que na Figura 3.17 aparece a mensagem “Terapeuta não encontrado", em vermelho. Essa mensagem aparece quando o sensor de movimento Microsoft Kinect® não consegue reconhecer alguma pessoa à sua frente. Quando o sensor detectar alguma pessoa à sua frente, um ponto da cor verde é colocado no centro da mão configurada anteriormente na tela “Manutenção das Atividades” (Figura 3.15). Isso indica que o sensor de movimento está funcionando e capturando as posições (X, Y) da mão. Quando se clica no botão “Iniciar”, o cronômetro é disparado, e o profissional de saúde tem o tempo pré-configurado para colocar a mão em algum lugar da tela, tendo por objetivo montar uma tarefa que será repetida pelo paciente.

A Figura 3.19 monstra a criação de uma atividade pelo profissional de saúde. As configurações para essa atividade foram definidas no caso de uso “Manter Atividade” (Sessão 3.3.2).

Figura 3.19: Criação de uma atividade.



Após um tempo pré-determinado, um ponto branco e numerado é adicionado na tela, no mesmo local onde o ponto verde estava, e o cronômetro é zerado, reiniciando o processo. Observe que os pontos são numerados sequencialmente, iniciando em zero. Para finalizar a colocação dos pontos, basta clicar no botão "Finalizar" e aguardar que a posição (X, Y) dos pontos, brancos e numerados, seja armazenada no SGBD.

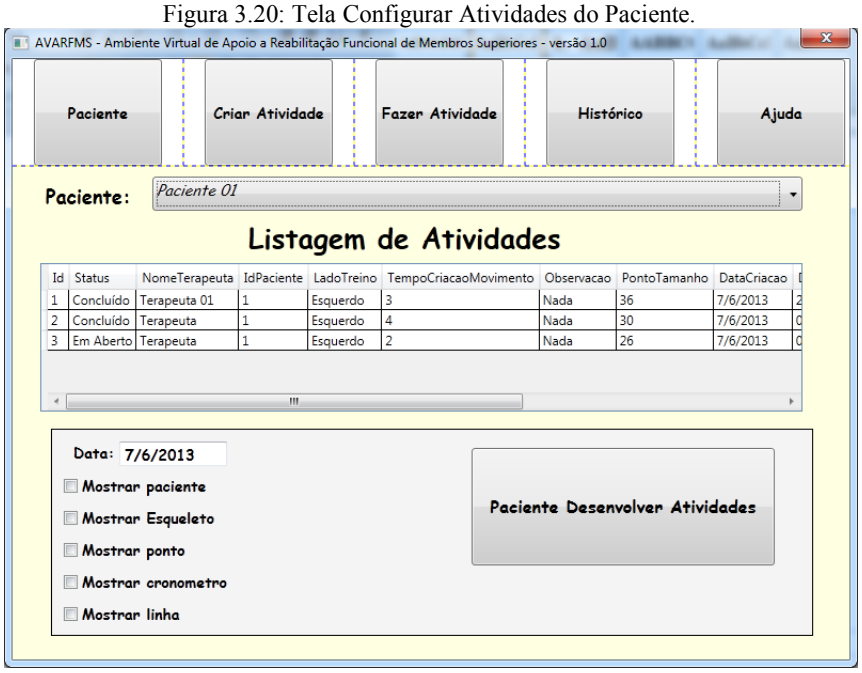
Os botões localizados no canto superior direito da tela possuem as seguintes utilidades:

- Botão Iniciar: inicia o processo de criação da atividade; o cronômetro é disparado e o processo de captura das posições (X, Y) dos pontos na tela é iniciado;
- Botão Finalizar: finaliza o processo de criação da atividade; o cronômetro é parado e o processo de captura das posições (X, Y) dos pontos na tela é finalizado;
- Botão Limpar: limpa a tela e as posições capturadas;
- Botão Gravar: grava, no banco de dados as seguintes informações:
  a) a posição x, y e z do ponto reconhecido como ombro central pelo sensor Microsoft Kinect do profissional de saúde;
  b) a posição (X,Y) dos pontos mostrados na tela;
  c) o código do paciente para o qual a atividade foi criada.

3.4.4 Caso de uso: Desenvolver atividade

O caso de uso "Desenvolver Atividade" inicia-se quando o botão "Fazer Atividade", da tela Principal (Figura 3.13) do AVARFMS, é clicado. Nesse momento, o ambiente carrega a tela "Configurar Atividades do Paciente" (Figura 3.20), na qual os dados dos pacientes cadastrados são carregados. O profissional de saúde escolhe o paciente, e as atividades desenvolvidas para ele serão listadas. De acordo com as necessidades que o paciente possui, o profissional de saúde configura o ambiente.

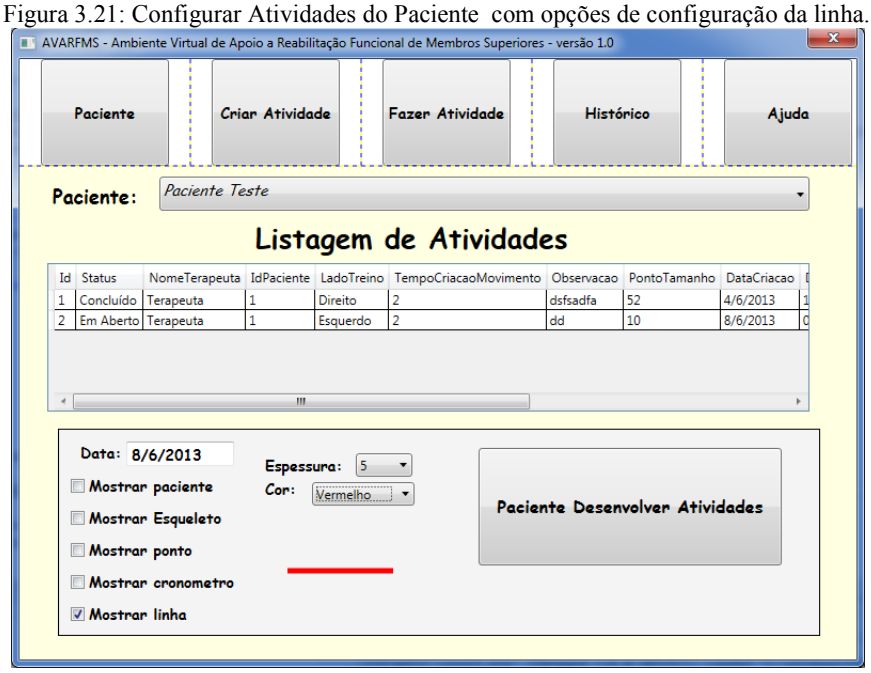
Figura 3.20: Tela Configurar Atividades do Paciente.



O profissional de saúde escolhe a(s) atividade(s) a ser(em) desenvolvidas pelo paciente. Para escolher mais de uma atividade, basta manter a tecla “CTRL” pressionada enquanto clica com o cursor do mouse sobre a atividade desejada. As seguintes configurações podem ser feitas:

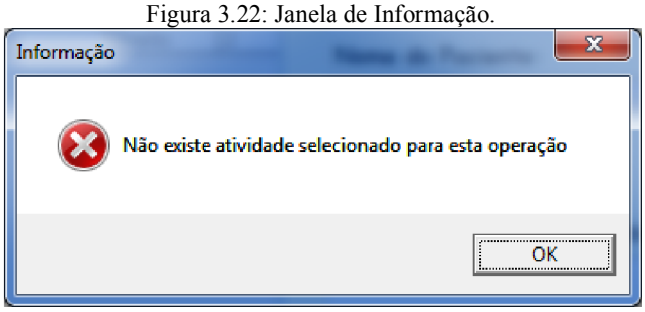
- Mostrar paciente: caso deseje que o paciente se veja na tela;
- Mostrar esqueleto: caso deseje que seja mostrado um esqueleto, semelhante ao mostrado na aplicação “CapturaEsqueleto” (Figura 2.25) na (Sessão 2.5.2);
- Mostrar ponto: caso deseje que os pontos sejam mostrados na tela;
- Mostrar cronômetro: caso deseje que um cronômetro seja mostrado na tela;
- Mostrar linha: caso deseje que as linhas que fazem a união dos pontos sejam mostradas. Quando essa opção for marcada, as opções “espessura” e “cor da linha” serão mostradas na tela “Configurar Atividades do Paciente” (Figura 3.21).

Figura 3.21: Configurar Atividades do Paciente com opções de configuração da linha.



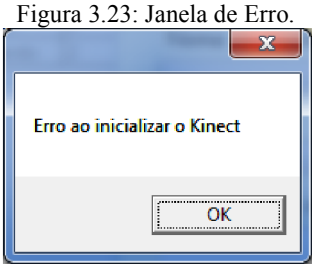
Após escolher as opções, clica-se no botão "Paciente Desenvolver Atividades". Caso alguma atividade não esteja selecionada, será mostrada uma janela de informação, como mostra a Figura 3.22.

Figura 3.22: Janela de Informação.



Nesse momento, o ambiente verifica a existência de um sensor de movimento Microsoft Kinect®. Caso encontre algum problema com o sensor, uma janela de erro com a mensagem "Erro ao inicializar o Kinect" (Figura 3.23) é mostrada.

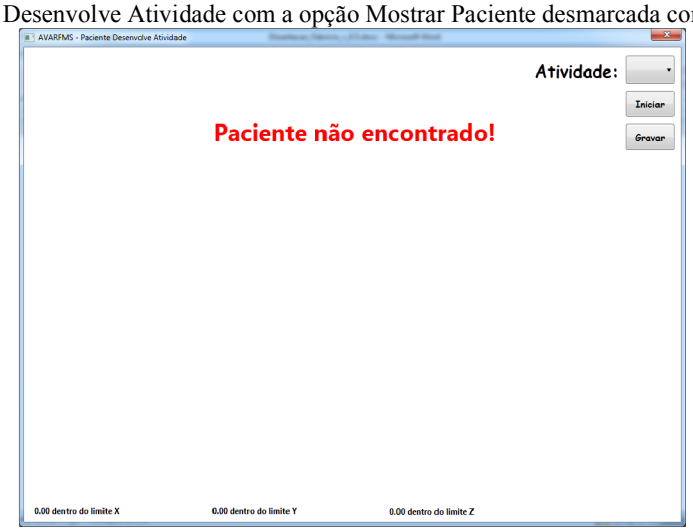
Figura 3.23: Janela de Erro.



Caso não ocorra nenhum problema com o sensor, a tela “Paciente Desenvolve Atividade” é mostrada. A figura 3.24 mostra a tela com a opção “Mostrar Paciente” marcada e a opção “Mostrar Esqueleto” desmarcada, enquanto que a figura 3.25 mostra a tela com a opção “Mostrar Paciente” desmarcada e a opção “Mostrar Esqueleto” marcada.

Figura 3.24: Tela Paciente Desenvolve Atividade com a opção Mostrar Paciente marcada com mensagem de aviso.

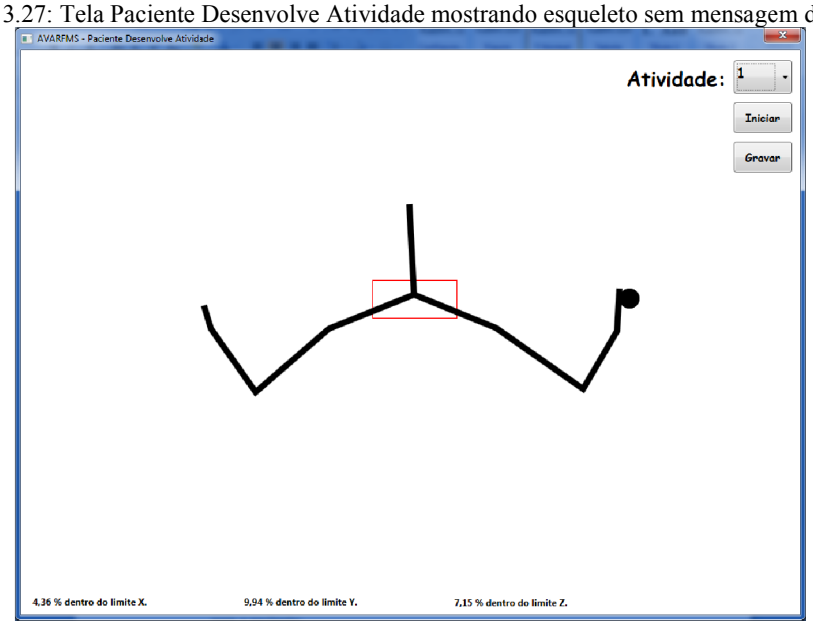
Figura 3.25: Paciente Desenvolve Atividade com a opção Mostrar Paciente desmarcada com mensagem de aviso.



Pode ser observardo nas figuras 3.24 e 3.25, a mensagem “Paciente não encontrado", em vermelho. Essa mensagem é mostrada quando o sensor de movimento Microsoft Kinect® não consegue reconhecer alguma pessoa à sua frente. Quando o sensor detectar alguma pessoa à sua frente, a mensagem desaparece (Figuras 3.26 e 3.27).

Figura 3.26: Tela Paciente Desenvolve Atividade mostrando paciente sem mensagem de aviso

Figura 3.27: Tela Paciente Desenvolve Atividade mostrando esqueleto sem mensagem de aviso



Na tela "Paciente Desenvolve Atividade" existe uma Combo Box[8] e dois botões localizados no canto superior direito da tela que possuem as seguintes utilidades:

- Combo Box: lista todas as atividades selecionadas para o paciente, na tela "Configurar Atividades do Paciente" (Figura 3.20);
- Botão Iniciar: coloca em vermelho o ponto com número zero, preparando o AVARFMS para iniciar a captura dos pontos;
- Botão gravar: grava os pontos (X, Y) que compõem o trajeto executado pelo paciente no banco de dados.

[8] Uma combo-box é uma interface de usuário que traz uma lista de opção associada a um texto (NATHAN 2010).

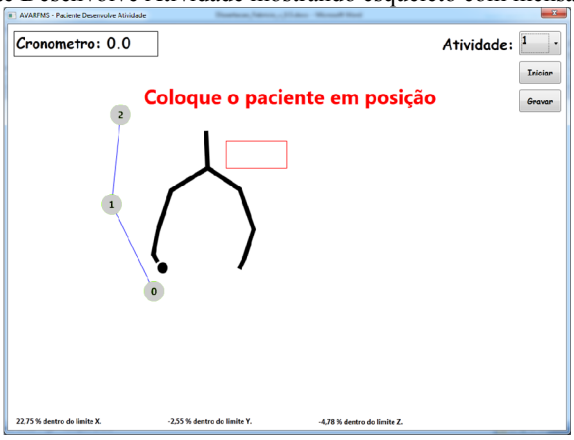
Quando uma atividade for escolhida no “Combo Box Atividade”, as seguintes informações serão carregadas:

- Posição X, Y e Z do ponto ombro central do profissional que desenvolveu a atividade para o paciente;
- A posição X, Y e numeração dos pontos que o paciente deve seguir.

Caso o ponto ombro central do paciente, ponto branco, não esteja dentro do retângulo vermelho, a mensagem “Coloque o paciente em posição” será mostrada (Figuras 3.28 e 3.29).

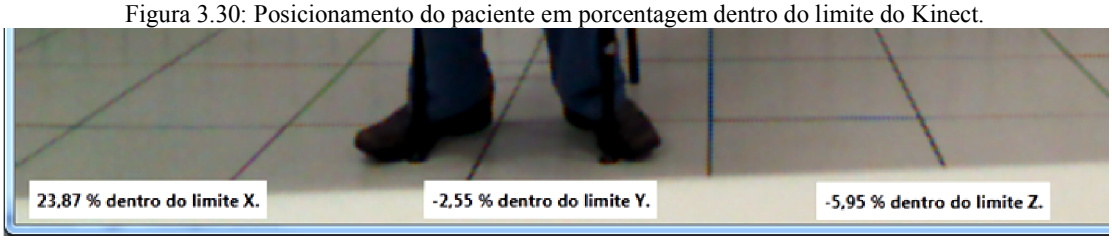
Figura 3.28: Tela Paciente Desenvolve Atividade mostrando paciente com mensagem de posicionamento.

Figura 3.29: Tela Paciente Desenvolve Atividade mostrando esqueleto com mensagem de posicionamento.



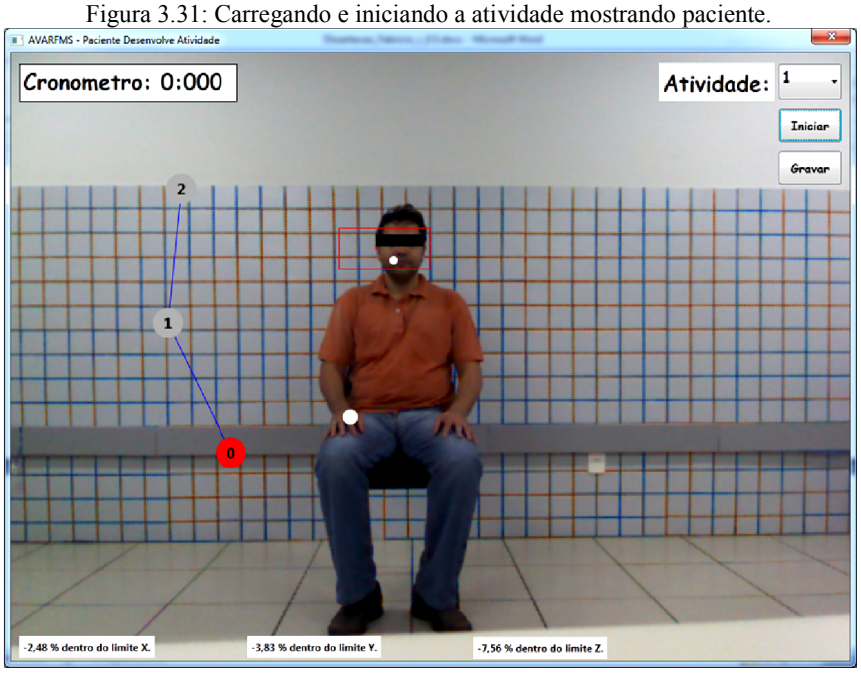
Na parte inferior da tela (Figura 3.30), é mostrado, em porcentagem, quando o paciente está dentro do limite. Esse recurso auxilia no posicionamento do paciente dentro do limite do Kinect. Para essa primeira versão do AVARFMS, foi adotado um limite de 20% para o posicionamento do paciente, conforme explicado na sessão 3.3.2.

Figura 3.30: Posicionamento do paciente em porcentagem dentro do limite do Kinect.



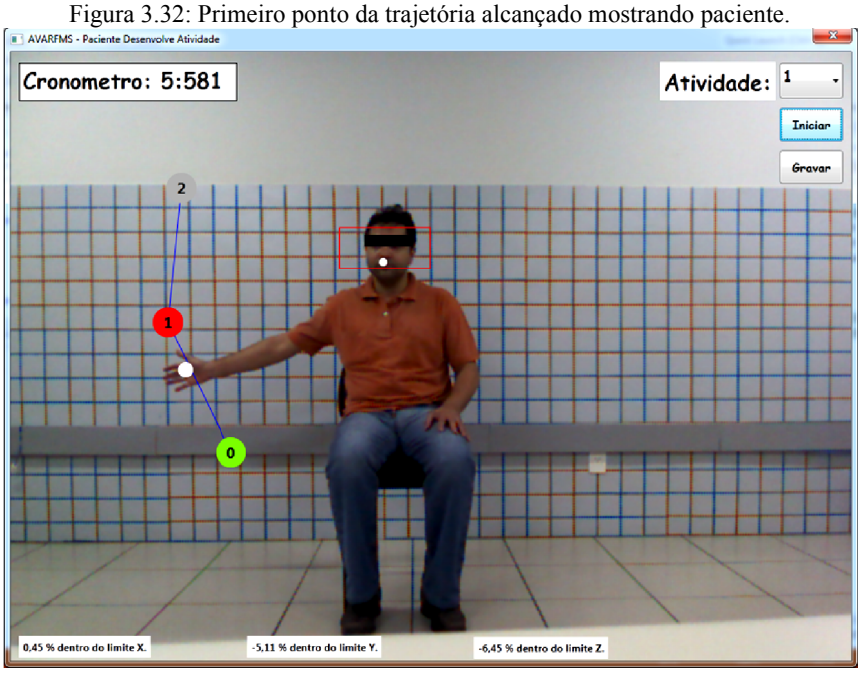
Por questões didáticas, será mostrada a execução do desenvolvimento de uma atividade por uma pessoa. Para isso, será demonstrado o desenvolvimento da atividade com as opções: “Mostrar Paciente”, “Mostrar Linha”, “Mostrar Ponto” e “Mostrar Cronômetro”, marcadas; e a opção “Mostrar Esqueleto”, desmarcada. Com o paciente devidamente posicionado, quando o botão “Iniciar” é clicado, o AVARFMS altera a cor do ponto com número zero, de cinza para vermelho, e aguarda que o paciente posicione o ponto branco, no meio de sua mão direita, dentro do ponto vermelho (Figura 3.31). O objetivo da atividade é seguir a trajetória desenvolvida.

Figura 3.31: Carregando e iniciando a atividade mostrando paciente.



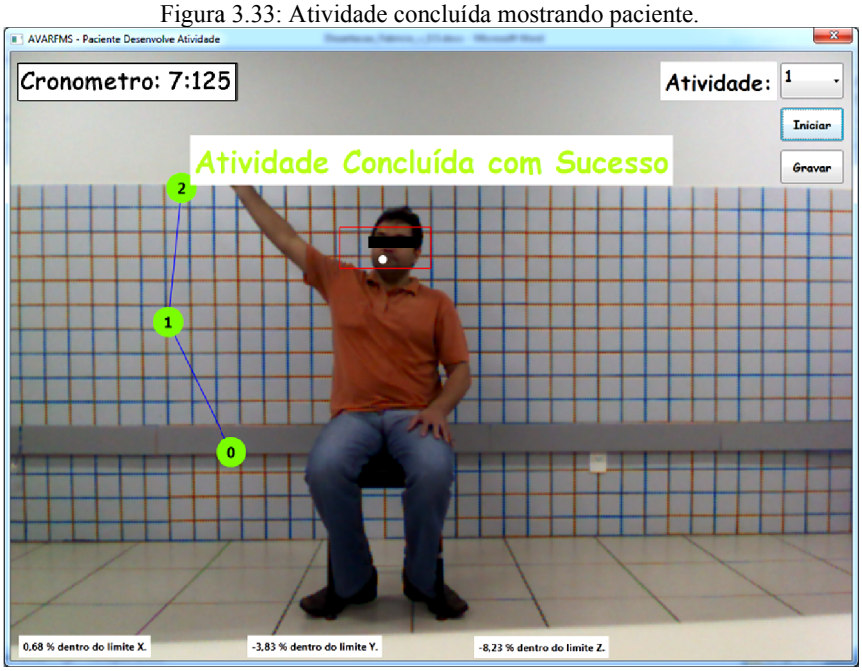
Quando o paciente colocar o ponto branco dentro do ponto vermelho, a sua cor muda para verde, o cronômetro é iniciado e a cor do próximo ponto é alterada de cinza para vermelho (Figura 3.32).

Figura 3.32: Primeiro ponto da trajetória alcançado mostrando paciente.



Quando o último ponto for atingido, o cronômetro para, todos os pontos ficam com a cor verde, e a mensagem “Atividade Concluída com Sucesso” aparece (Figura 3.33).

Figura 3.33: Atividade concluída mostrando paciente.



Abaixo, será mostrado o desenvolvimento da mesma atividade, com as opções: “Mostrar Esqueleto”, “Mostrar Linha”, “Mostrar Ponto” e “Mostrar Cronômetro”, marcadas; e a opção “Mostrar Paciente”, desmarcada (Figuras 3.34, 3.35 e 3.36).

Figura 3.34: Carregando e iniciando a atividade mostrando esqueleto.

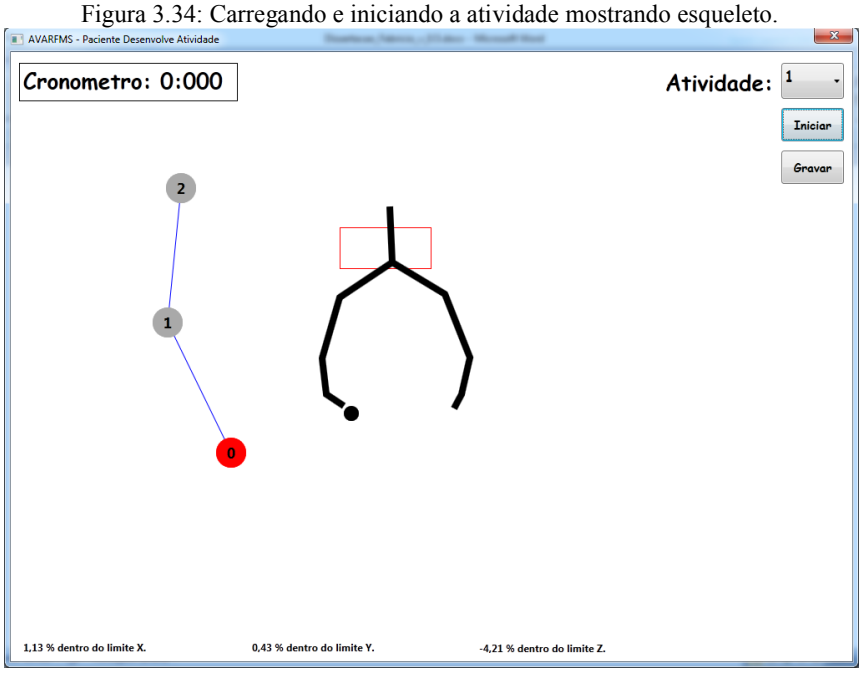


Figura 3.35: Primeiro ponto da trajetória alcançado mostrando esqueleto.

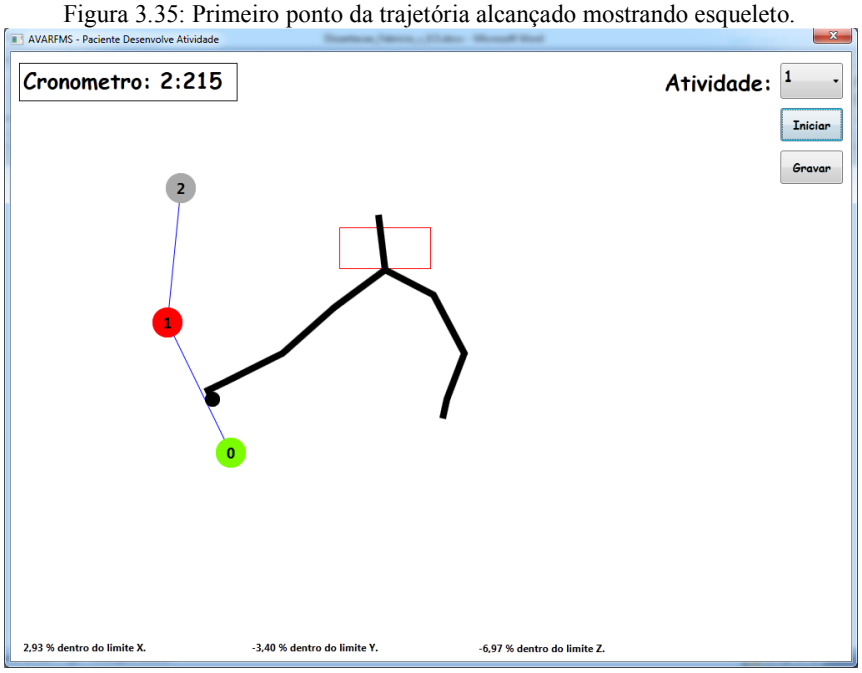
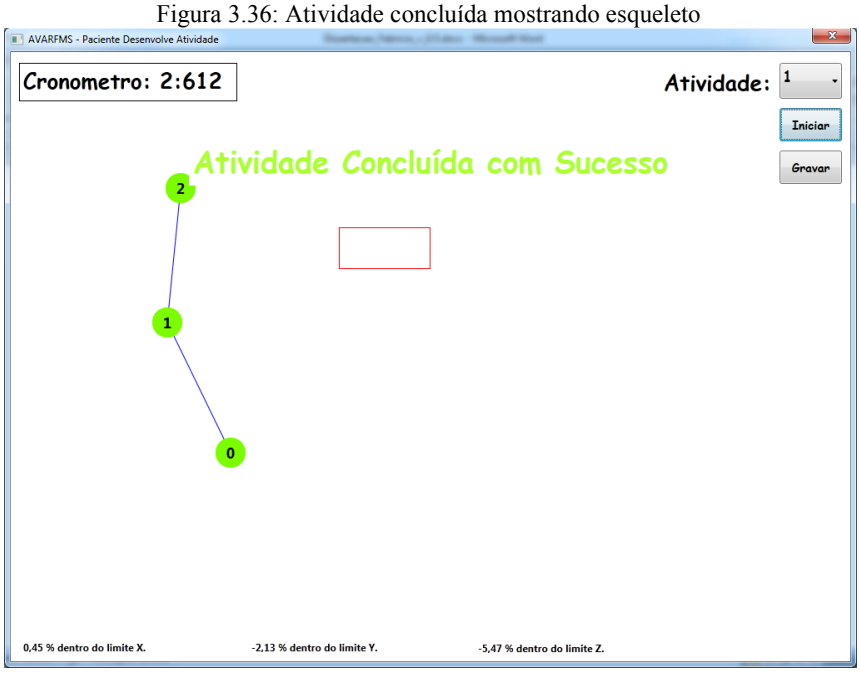
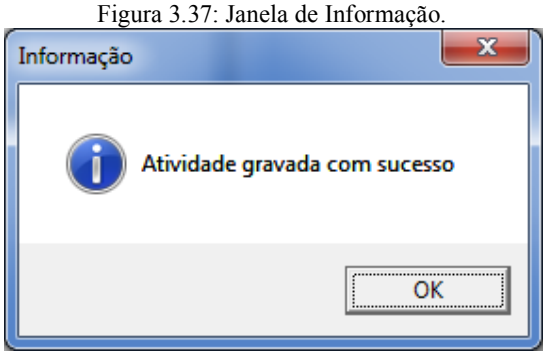


Figura 3.36: Atividade concluída mostrando esqueleto



Quando a atividade é concluída, o esqueleto desaparece da tela (Figura 3.36). Para guardar todos os pontos da trajetória executada pelo paciente, basta clicar no botão "Gravar" e as posições (X, Y) dos pontos capturados pelo sensor Microsoft Kinect® serão armazenadas no Sistema Gerenciador de Banco de Dados (SGBD). Quando o processo de gravação for finalizado, uma janela de informação com a mensagem "Atividade gravada com sucesso" (Figura 3.37) será mostrada.

Figura 3.37: Janela de Informação.

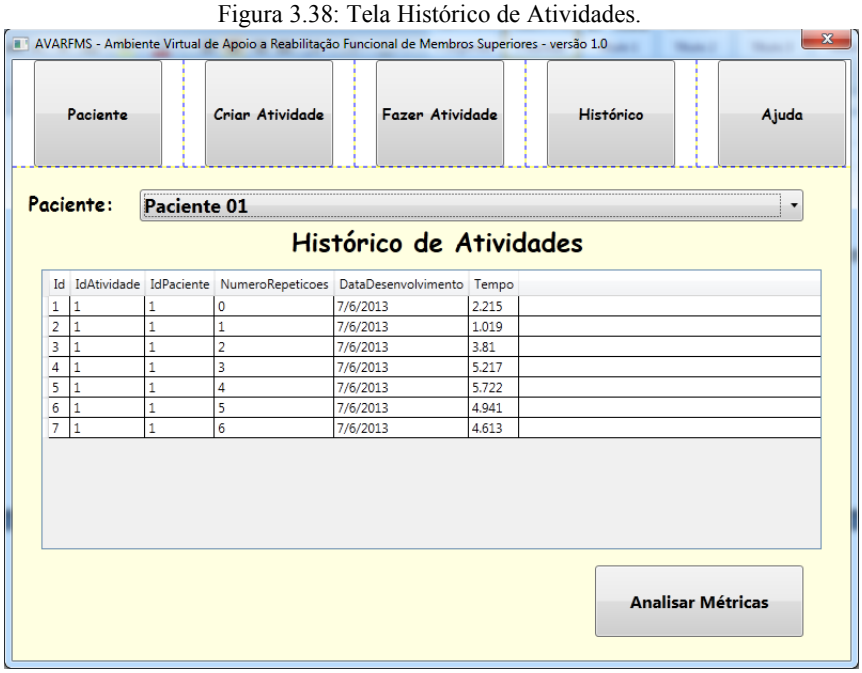


3.4.5 Caso de uso: Consultar Histórico

O caso de uso "Consultar Histórico" inicia-se quando o botão "Histórico" da tela "Principal" (Figura 3.13) do AVARFMS for clicado. Nesse momento, o ambiente carrega a tela de "Histórico de Atividade" (Figura 3.38), na qual os dados dos pacientes existentes serão

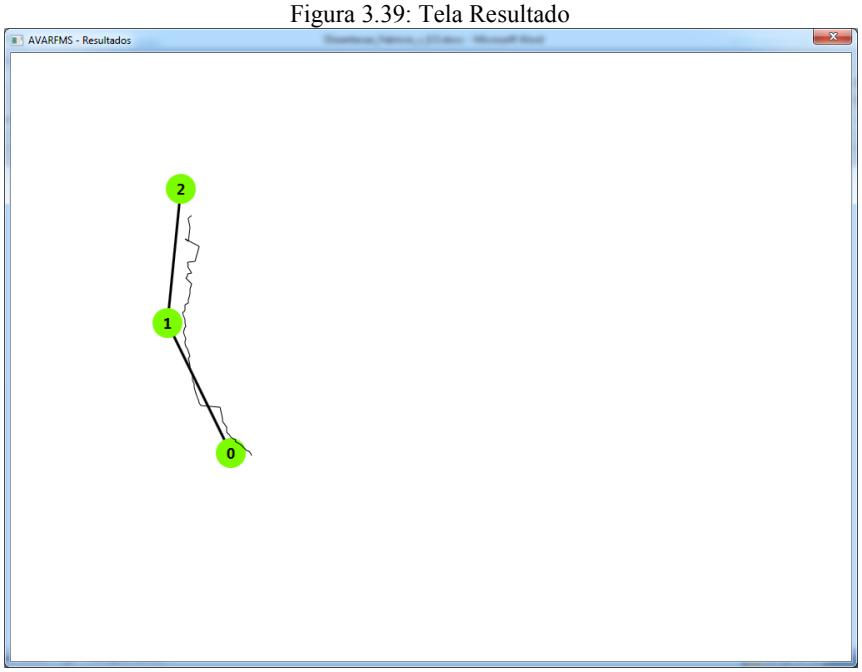
carregados. O profissional de saúde escolhe o paciente, e as atividades desenvolvidas por ele serão listadas.

Figura 3.38: Tela Histórico de Atividades.



Caso um duplo clique seja dado sobre uma das atividades, o ambiente virtual apresenta a tela “Resultado” (Figura 3.39), que mostra o resultado da trajetória desenvolvida pelo paciente. A linha fina representa o resultado da trajetória do paciente.

Figura 3.39: Tela Resultado

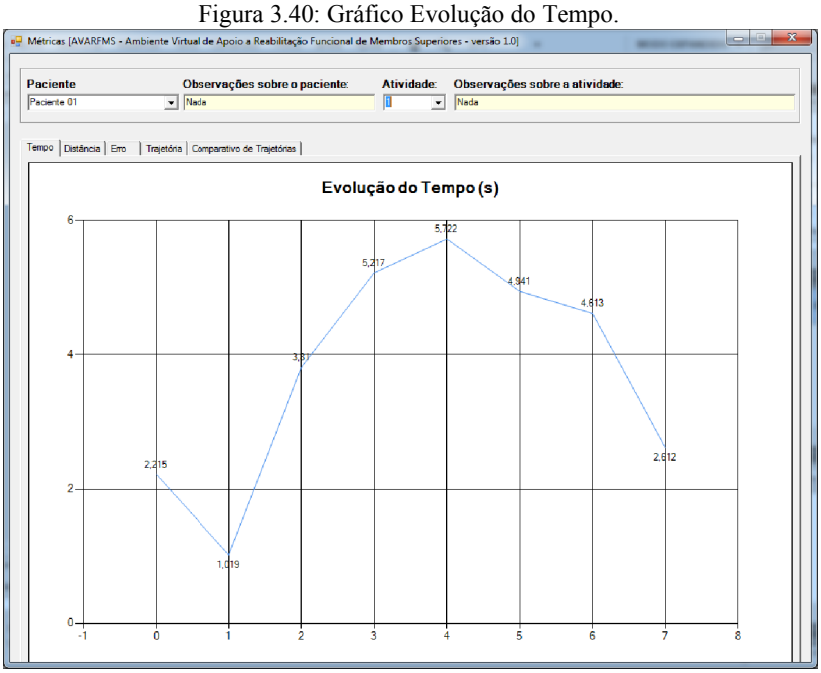


Caso o botão “Analisar Métricas” seja clicado, a tela “Métricas” (Figura 3.39) será carregada. Nessa janela, o profissional de saúde pode acompanhar o desenvolvimento das atividades realizadas por um paciente por meio dos seguintes gráficos:

- Evolução do tempo
- Evolução da distância
- Evolução do erro na trajetória
- Trajetórias
- Comparativo de trajetórias

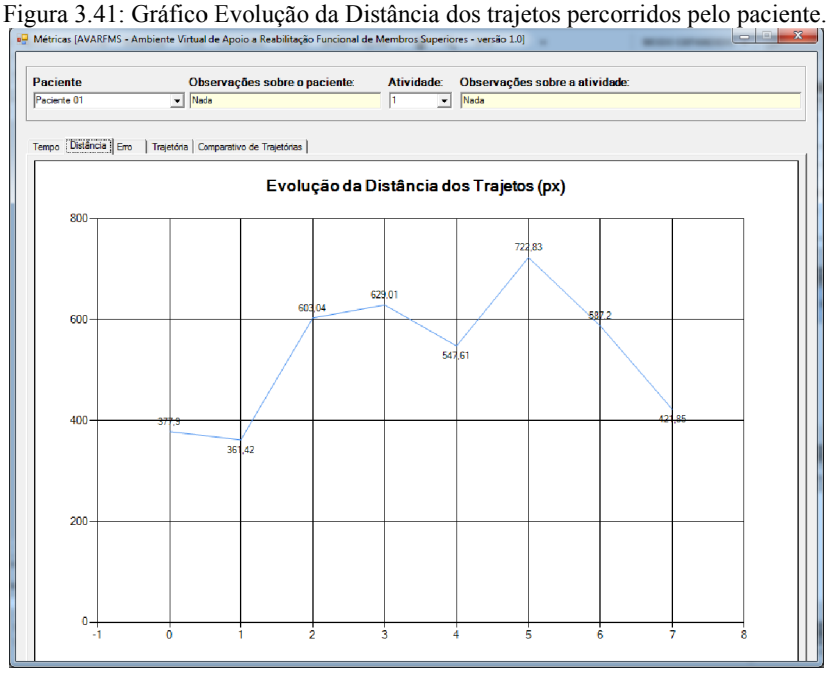
O gráfico da “Evolução do Tempo” (figura 3.40) será útil para que o profissional de saúde acompanhe a evolução do tempo que o paciente leva para desenvolver a atividade.

Figura 3.40: Gráfico Evolução do Tempo.



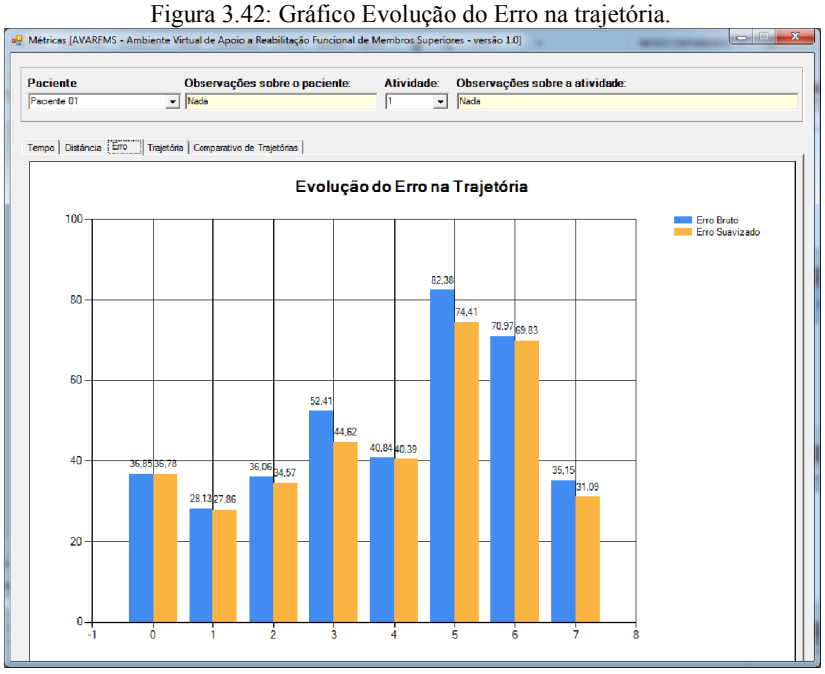
O gráfico da “Evolução da Distância” (Figura 3.41) será útil para o profissional de saúde acompanhar a distância percorrida pelo paciente em cada sessão. A distância é baseada na quantidade de pontos capturados pelo sensor Microsoft Kinect® durante o desenvolvimento da atividade.

Figura 3.41: Gráfico Evolução da Distância dos trajetos percorridos pelo paciente.



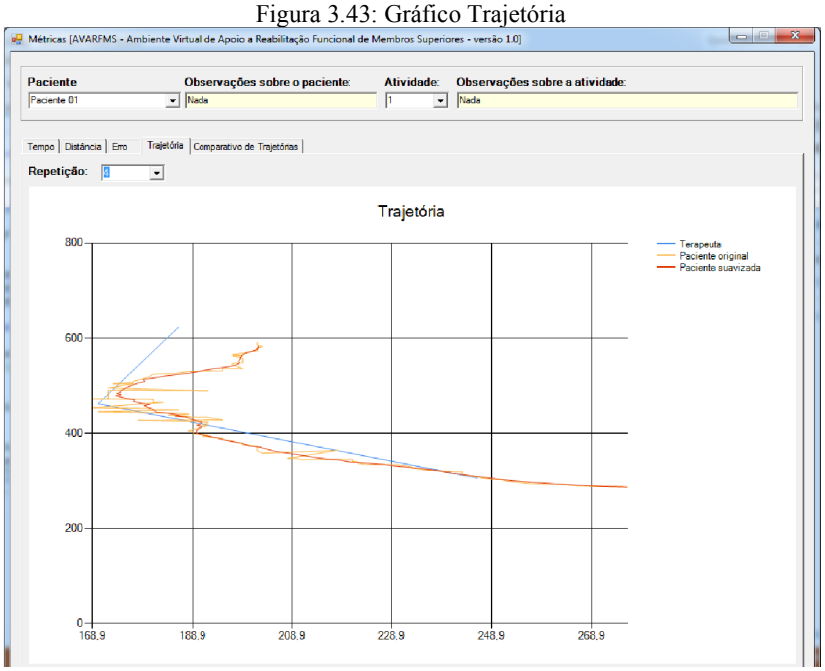
O gráfico da "Evolução do Erro" na trajetória (Figura 3.42) será útil para que o profissional de saúde acompanhe a evolução do erro que o paciente está fazendo durante o desenvolvimento de suas atividades. Esse gráfico faz um comparativo entre os erros da trajetória desenvolvida pelo paciente e a mesma trajetória suavizada, conforme explicado na sessão (3.3.4).

Figura 3.42: Gráfico Evolução do Erro na trajetória.



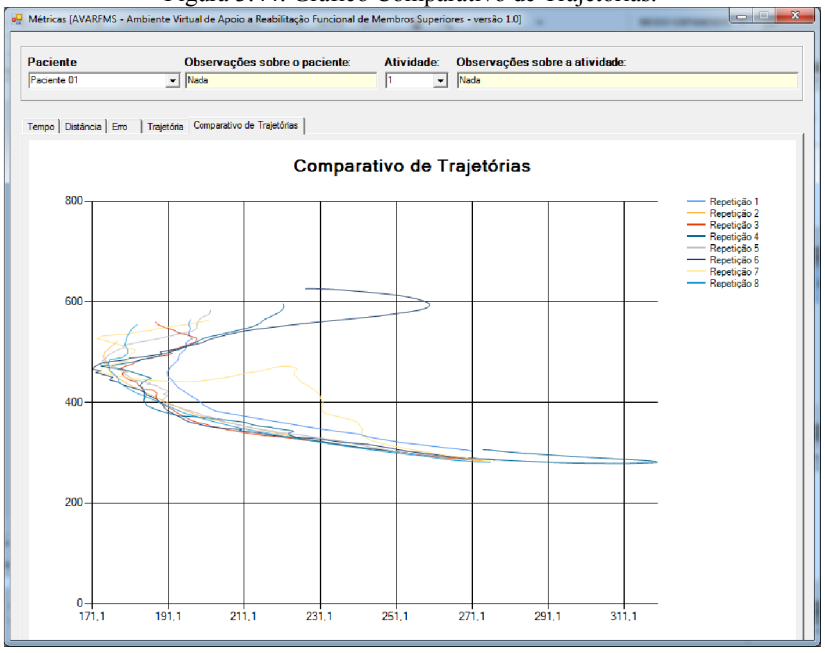
O “Gráfico Trajetória” (Figura 3.43) é útil para que o profissional de saúde faça um comparativo das trajetórias original e suavizada, desenvolvidas pelo paciente durante a atividade.

Figura 3.43: Gráfico Trajetória



O gráfico “Comparativo de Trajetórias” (Figura 3.44) será útil para que o profissional de saúde faça um comparativo das trajetórias suavizadas do paciente ao longo de diferentes sessões.

Figura 3.44: Gráfico Comparativo de Trajetórias.



3.5 Modelos de diagrama de classe AVARFMS

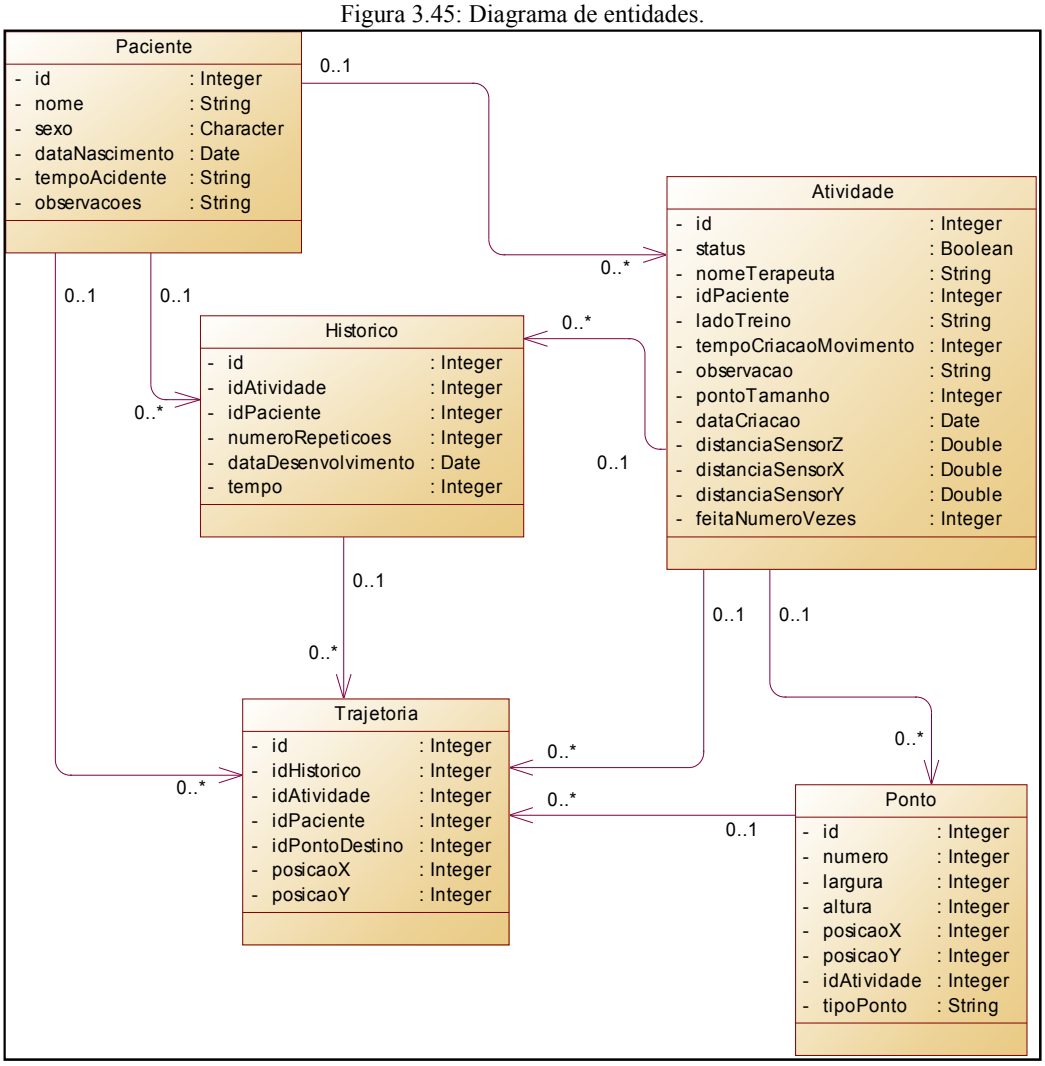
O modelo de diagrama de classes é utilizado na construção do modelo de classes, do nível de análise até o nível de especificação. Este diagrama é responsável por descrever a estrutura da informação (classes e suas relações) que são utilizadas no sistema. Neste trabalho, serão apresentadas apenas as classes que representam os objetos de entidades que são utilizados como repositório para as informações que são manipuladas pelo ambiente virtual (BEZERRA, 2007). As classes de entidades são utilizadas para criar as tabelas no sistema gerenciador de banco de dados utilizado pelo ambiente virtual.

Durante o processo de desenvolvimento do ambiente virtual as seguintes classes de entidades foram criadas:

- Paciente: classe que representa as informações sobre o paciente;
- Atividade: classe que representa as atividades criadas pelo terapeuta para serem desenvolvidas pelo paciente;
- Ponto: classe que representa os pontos da atividade criada pelo terapeuta; esses pontos serão mostrados na tela para o paciente;
- Histórico: classe que representa o histórico das atividades desenvolvidas pelos pacientes; com base nas informações dessa classe, o terapeuta poderá acompanhar a evolução do tratamento;
- Trajetória: classe que representa todos os pontos (x, y) capturados pelo sensor de movimento durante a atividade executada pelo paciente.

A figura 3.45 apresenta o diagrama das classes de entidade e seus respectivos relacionamentos. Os métodos foram suprimidos por representarem apenas as ações de gravação, alteração e exclusão dos dados.

Figura 3.45: Diagrama de entidades.



Todo o código fonte do projeto AVARFMS e apresentado no anexo 7.2.1.

4. DISCUSSÃO

Sendo o principal objetivo deste trabalho a proposição de um ambiente virtual de apoio à reabilitação funcional de membros superiores utilizando o sensor de movimento Microsoft Kinect, são discutidos abaixo alguns objetivos específicos:

Objetivo 01: Fornecer meios qualitativos e quantitativos para que profissional de saúde possa acompanhar a evolução do tratamento de reabilitação de uma vítima do AVE.

Com a utilização do sensor de movimentos Microsoft Kinect – que forneceu ao ambiente virtual a capacidade de capturar a posição x, y da mão do paciente durante uma atividade criada por um terapeuta – e a utilização de um sistema gerenciador de banco de dados – que permitiu o armazenamento dessas posições (x, y) –, os profissionais de saúde, por meio de gráficos, podem acompanhar a evolução do tratamento, como apresentados na sessão "Caso de uso: Consultar Métricas" (4.3.5). Os gráficos gerados pelo sistema permitem ao profissional de saúde acompanhar a evolução do paciente na execução das atividades destinadas a ele, fornecendo, dessa forma, meios qualitativos e quantitativos de avaliar a evolução do paciente.

Objetivo 02: Fornecer uma ferramenta lúdica e divertida às vítimas do AVE, em seu processo de reabilitação.

Após algumas sessões de testes com pacientes, acompanhadas por profissionais de saúde do CRER, foi solicitado aos terapeutas que respondessem a um pequeno questionário de satisfação, do qual foram obtidos os resultados apresentados na tabela 4.1.

Tabela 4.1: Resultado do questionário de satisfação dos terapeutas.

	Medidas	***TO 01**	***TO 02**	***TO 03**	***TO 04**	***TO 05**	***TO 06**
Acharam a utilização do ambiente virtual complicada?	Nada 0 1 2 3 4 5 Muito	1	1	2	2	0	1
Utilizaria o ambiente em como ferramenta de apoio?	Nada 0 1 2 3 4 5 Muito	5	5	4	4	5	5
Os resultados fornecidos pelo ambiente são úteis na avaliação da evolução do tratamento de um paciente?	Nada 0 1 2 3 4 5 Muito	4	5	5	4	5	3
O ambiente prende a atenção do paciente?	Nada 0 1 2 3 4 5 Muito	4	4	5	4	5	5
A capacidade em montar a atividade a ser desenvolvida pelo paciente pelo profissional responsável pelo tratamento é importante?	Nada 0 1 2 3 4 5 Muito	5	5	5	4	5	5

*** TO = Terapeuta Ocupacional**

Foi solicitado aos pacientes que participaram dos testes que respondessem a um pequeno questionário de satisfação, do qual foram obtidos os resultados da tabela 4.2.

Tabela 4.2: Resultado do questionário de satisfação dos pacientes

	Medidas	*PA 01	*PA 02
Realizar a atividade foi complicado?	Nada 0 1 2 3 4 5 Muito	1	2
O entendimento do que fazer foi difícil?	Nada 0 1 2 3 4 5 Muito	0	0
O ambiente prendeu a sua atenção no desenvolvimento da atividade?	Nada 0 1 2 3 4 5 Muito	4	4
Gostaria de utilizar o ambiente mais vezes no processo de reabilitação?	Nada 0 1 2 3 4 5 Muito	4	4

*** PA = Paciente**

Após palestras sobre o ambiente virtual, foi proposto que os alunos de fisioterapia respondessem a um pequeno questionário de satisfação, cujo resultado é mostrado na tabela 4.3.

Tabela 4.3: Questionário aplicado a 50 alunos do curso de fisioterapia

Pergunta	Medidas	Sim	Não
Conhece o uso de ambientes virtuais em processos de reabilitação?	Sim / Não	30	20
Acharam a utilização do ambiente virtual complicada?	Sim / Não	10	40
Utilizaria o ambiente como ferramenta de apoio?	Sim / Não	40	10
A capacidade em montar a atividade a ser desenvolvida pelo paciente pelo profissional responsável pelo tratamento é importante?	Sim / Não	40	10

Os resultados obtidos através dos três questionaram demonstram claramento que os profissifionais, pacientes e os futuros profissionais da área de reabilitação utilizariam o ambiente virtual como ferramenta de apoio ao processo de reabilitação por ser uma ferramenta de fácil uso, divertida e fornece formas quantitativas e qualitativas de acompanhar a evolução do tratamento do paciente.

5. CONCLUSÕES

O objetivo principal deste trabalho foi a criação de um ambiente virtual de apoio à reabilitação funcional de membros superiores, utilizando o sensor de movimento Microsoft Kinect para reabilitação de vítimas de um AVE. Na introdução, foram discutidos alguns trabalhos sobre o tema. Apesar da utilização desses tipos de ambientes, foi observado que não existia nenhum que proporcionasse ao profissional de saúde, montar a atividade a ser desenvolvida pelo paciente, quesito este que, segundo os profissionais de saúde consultados durante o desenvolvimento do ambiente, foi o grande diferencial do ambiente proposto em relação aos outros, além do fornecimento das métricas para o acompanhamento da evolução do tratamento do paciente.

5.1 Trabalhos futuros

Durante o desenvolvimento do trabalho, tanto de pesquisa quanto de desenvolvimento, várias ideias foram levantadas:

5.1.1 Gravação da sessão do paciente

Uma das sugestões pelos profissionais de saúde seria a gravação da sessão do paciente, de forma que todos os movimentos que o paciente executasse fossem gravados para uma futura análise pelos profissionais de saúde.

5.1.2 Cálculo dos ângulos de movimento

Essa primeira versão do ambiente virtual trabalha com o objetivo de verificar trajetória e tempo no movimento. Seria interessante que os ângulos entre mão, cotovelo e ombro fossem calculados, o que poderia ser utilizado como mais um dado qualitativo e quantitativo para o profissional de saúde avaliar a evolução do paciente.

5.1.3 Alterar o jogo

Essa primeira versão do ambiente virtual traz apenas um jogo do estilo de ligar os pontos, no qual se pode verificar a qualidade da trajetória desenvolvida pelo paciente. Como o

ambiente proporciona que o profissional de saúde determine a posição dos pontos na tela, seria interessante alterar o cenário. Por exemplo, um jogo do estilo *Space Invaders*, figura 6.1, no qual o profissional de saúde determine o local onde as naves irão aparecer, com o objetivo de exercitar ombros e cotovelo.

Figura 5.1: Jogo Space Invaders

Fonte: GOTHZ, 2010

5.1.4 Reabilitar os membros inferiores

Essa primeira versão do ambiente virtual possui como objetivo apenas a reabilitação dos membros superiores. Seria interessante acrescentar também os membros inferiores no desenvolvimento das atividades. Por exemplo, adicionar um jogo do estilo chute ao gol, no qual o terapeuta determine o local onde a bola irá aparecer e o paciente tem que simular um chute na bola. Com isso, o terapeuta poderia avaliar a amplitude da perna do paciente.

6. REFERÊNCIAS

AANABATHULA, Akshay. **November 2009 Game Releases**. Novembro, 2009. Disponível em: <http://skattertech.com/2009/11/november-2009-game-releases/>. Acesso em 12/02/2013.

ASHLEY, James; WEBB, Jarret. **Beginning Kinect Programming with the Microsoft Kinect SDK**. New York: Apress, 2012.

ASSIS, Gilda Aparecida. **NEUROR - Sistema de Apoio à Reabilitação dos Membros Superiores de Pacientes Vítimas de Acidentes Vasculares Encefálicos** (Doutorado). Escola Politécnica da Universidade de São Paulo, São Paulo, 2010

AULD, Dr. Lawrence Auld; PANTELIDIS Dr. Veronica. **The Virtual Reality and Education Laboratory at East Carolina University**. The JornalSchool of Education at East Carolina University, Greenville, North Carolina, 01 nov. 1999. Disponível em: <http://thejournal.com/Articles/1999/11/01/The-Virtual-Reality-and-Education-Laboratory-at-East-Carolina-University.aspx?p=1>. Acesso em: 12/12/2012.

BARBOSA, Dagoberto Miranda**. Utilização de um sistema de realidade virtual não imersiva como ferramenta para a reabilitação de membros superiores de indivíduos hemiparéticos**. Dissertação (Mestrado em Engenharia Elétrica e de Computação) – Universidade do Federal de Goiás, Goiânia, 2008.

BARREIROS, J. **Metodologia da investigação científica**. Lisboa: Faculdade de Motricidade Humana da Universidade Técnica de Lisboa, 2008. Disponível em: <http://home.fmh.utl.pt/~jbarreiros/MIC-pdf>. Acessoem: 27 nov. 2012.

BELLO, Gabriel Soto. **Aquecimento BJ: tudo que você precisa saber sobre a E3 2011!**, 11 de maio de 2011. Disponivel em: < http://www.baixakijogos.com.br/noticias/14615-aquecimento-bj-tudo-que-voce-precisa-saber-sobre-a-e3-2011 >. Acesso em 30/06/2013.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistema com UML**. Rio de Janeiro: Elsevier, 2007.

Biblioteca Virtual em Saúde - Ministério da Saúde. **AVC - Acidente Vascular Cerebral**.

Disponível em: http://bvsms.saude.gov.br/bvs/dicas/105avc.html. Acesso em 12/03/2013 às 15:00 h.

Brasoftware. **KINECT FOR WINDOWS**. Disponível em: < http://www.brasoftware.com.br/ch/prod/37071/kinect-for-windows.aspx>. Acesso em 11/11/2012.

CAIADO, Elen Cristine. **A relação da Fonoaudiologia com o AVC (Acidente Vascular Cerebral)**. Brasil Escola. Disponível em: < http://www.brasilescola.com/fonoaudiologia/a-relacao-fonoaudiologia-com-avc-acidente-vascular-.htm >. Acesso em: 12/01/2013

CALIN, Alina. et al. **Upper limb rehabilitation system using Microsoft Kinect**. STUDIA UNIV. BABES-BOLYAI, INFORMATICA, Volume LVI, Number 4, 2011.

CATHY, M. Helgason; PHILIP A. Wolf. **American Heart Association Prevention Conference IV: Prevention and Rehabilitation of Stroke**. 1997;28(7):1522-6

CATUHE, David**. Programming with the Kinect for Windows - Software Development Kit**. Redmond, Washington: Microsoft Press, 2012.

CHASA: **Children´s Hemiplegia and Stroke Association.** Disponível em: <http://www.chasa.org/medical/hemiplegia/>. Acesso em 12/03/2013 às 16:00h.

CHEN, Jason. **Microsoft Xbox 360 Kinect Launches November** 4. GIZMODO, junho de 2010. Disponível em: < http://gizmodo.com/5563148/microsoft-xbox-360-kinect-launches-november-4>. Acesso em 12/12/2012 às 16:00.

DAVAASAMBUU, Erdenestsogt. et al. **A Microsoft Kinect based virtual rehabilitation system**. The 5th International Conference FIFAT 2012.

DAVIDS, K. **Ecological validity in understanding sport performance: some problems of definition**. Quest, Champaign, v.40, p.126-36, 1988.

DOUTOR SHOPFISIO. **Faça do videogame um aliado na luta contra o AVC**. Disponível em: < http://blog.shopfisio.com.br/?doutor=faca-do-videogame-um-aliado-na-luta-contra-o-avc>.Acesso em 14/02/2013.

DUNCAN, W. Pamela et al. **Management of Adult Stroke Rehabilitation Care: a clinical**

practice guideline. Stroke 2005 Sep;36(9):e100-43.

E3: **Microsoft shows off gesture control technology for Xbox 360**. Los Angeles Time, julho de 2009. Disponível em: <http://latimesblogs.latimes.com/technology /2009 /06/microsofte3.html >. Acesso em 12/12/2012 às 15:35 h.

EISLER , Craig. KINECt™ for Windows BLOG. Starting February 1, 2012: **Use the Power of Kinect for Windows to Change the World**, janeiro de 2012. Disponível em: < http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx >. Acesso em 12/12/2012 às 16:10.

FALCÃO, Leandra Teixeira. **TUDO SOBRE O AVC - ACIDENTE VASCULAR CEREBRAL**. Med Foco. Disponível em: < http://leandrafono.blogspot.com.br/2012/10/entenda-o-avc-e-previna-se.html >. Acesso em 12/12/2012.

Fast Company. **ALEX KIPMAN / Microsoft.** Imagine Cup. Disponível em: <http://www.fastcompany.com/most-creative-people/2011/alex-kipman-microsoft>. Acesso em 10/12/2012.

FERRAZ, Leonardo Tórtora Devienne; YAMASHITA, Renato Kenichi Sakata. **Desenvolvimento de jogo eletrônico para reabilitação utilizando um sensor de som e movimento (Kinect).** Trabalho de conclusão de curso. Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de sistemas mecânicos.

FOLDOC – Free On-Line Dictionary Of Computing. **Application Program Interface**, 15 de fevereiro de 1995. Disponível em: < http://foldoc.org/Application+Program+Interface >. Acesso em 30/06/2013.

FREITAS, Daniel Q. et al. D**evelopment and Evaluatin of Kinect Based Motor Rehabilitation Game**. SBC – Procceding of SBGames 2012. Computing Track Full Papers.

GONÇALVES, Viviane Pacheco. **Software de aprendizagem e controle motor para avaliação de indivíduos hemiparéticos: validade e confiabilidade.** Dissertação (Mestrado em Ciências do movimento humano) – Universidade do Estado de Santa Catarina, Florianópolis, 2008.

GOTHZ, Alisson. **Space Invaders vai virar filme!**. Setembro, 2010. Disponível em: < http://www.trash80s.com.br/tag/space-invaders/page/2/ >. Acesso em 10/10/2012

ILEX. **O que é SDK?**, 19 de fevereiro de 2008. Disponível em: < http://blogdoiphone.com/2008/02/o-que-e-afinal-o-sdk/ >. Acesso em 25/05/2013.

JERÔNIMO, Rosimeire Aparecida; LIMA, Simone Maria Puresa Fonseca. **Tecnologias computacionais e ambientes virtuais no processo terapêutico de reabilitação**. Revista O mundo da Saúde são Paulo: 2006; jan/mar 30 (1) 96-106.

KAUTZ, H, Etzioni. O. **Foundations of assisted cognition systems departamento of computer sciente& engineering**. Seattle: University of Washington; 2002.

KERKHOVE**, Tom What is the difference between Kinect for Windows & Kinect for Xbox360?**. Kinecting for Windows, setembro de 2012. .Disponível em: <http://www.kinectingforwindows.com/2012/09/07/what-is-the-difference-between-kinect-for-windows-kinect-for-xbox360/>. Acesso em 20/02/2013 às 02:00 h.

KINECT. **Kinect for Windows Features**. Kinect for Windows, 2013. Disponível em: <http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx>. Acesso em 19/02/2013 às 02:00 h.

KLEINA, Nilton. **Como o Kinect vai revolucionar o computador**. Novembro, 2011. Disponível em: < http://www.tecmundo.com.br/kinect/15242-como-o-kinect-vai-revolucionar-o-computador.htm#ixzz1dgFdxREs>. Acesso em 15/01/2013.

LABELLE, Kathryn. **Evaluation of Kinect Joint Tracking for clinical and in-home stroke rehabilitation tools**. University of Notre Dame, Indiana, 2011.

LANDIN, Wikerson. Da ficção para a realidade**: MinorityReport - A Nova Lei**. TECMUNDO, julho de 2010. Disponível em:<http://www.tecmundo.com.br/cinema/ 4637-da-ficcao-para-a-realidade-minority-report-a-nova-lei.htm>. Acesso em 12/12/ 2012, às 14:12 h.

LELIS, Cheila Maíra. **Reabilitação de Pacientes com AVC**. Disponível em: http://terapiaocupacionalnobrasil.blogspot.com.br/2010/01/reabilitacao-do-paciente-com-avc.html. Acesso em 12/03/2013 às 15:00h.

LIBERTY, Jesse. BETTS, Paul. **Programming Reactive Extensions and LINQ**. New York: Apress, 2011.

LOBÃO, Alexandre Santos; EVANGELISTA, Bruno Pereira; FARIAS, José Antônio leal de; OLIVEIRA Patrick Pabllo Borgers de. **XNA 3.0 para desenvolvimento de jogos no Windows, Zune e Xbox 360**. Rio de Janeiro: Brasport, 2010.

LOWENSOHN, Josh. **Kinect power adapter appears in Microsoft's store**. Novembro, 2010. Disponível em: < http://news.cnet.com/8301-10797_3-20022236-235.html >. Acesso em 11/11/2012.

MADERA, Francisco. **An Introduction to the Collision Detection Algorithms**. Facultad de Matemáticas, UADY. Abstraction & Application 5 (2011) 7-18

MARQUES, Carla ET al; **Comorbidade: conceito e implicações na pesquisa clínica em psiquiatria**. J. bras. psiquiatr; 43(3):117-21, mar. 1994.

Microsoft Corporation. **KINECT for Windows.** Microsoft Kinect for Windows Software Development Kit (SDK), 2013. Disponível em: http://www.microsoft.com/en-us/kinectforwindows/develop/sdk-eula.aspx; Acesso em 12/01/2013.

MICROSOFT. **Kinect for Windows SDK**. Microsoft Download Center, maio de 2012. Disponível em: <http://www.microsoft.com/en-us/download/details.aspx?id=28782>. Acesso em 20/03/2013, às 21:00h

MILES, Rob. **Start Here! Learn the Kinect API**. Sebastopol, California: O´Reilly, 2012

MILLER, Ross**. Kinect for Windows coming February 1st with 'near mode' — not for use with Xbox 360**. THE VERGE, fevereiro de 2013. Disponível em: <http://www.theverge.com/2012/1/9/2695734/kinect-for-windows-official-availability>. Acesso em 20/02/2013 às 01:10 h.

MSDN. **Kinect for Windows Architecture**. Disponível em: <http://msdn.microsoft.com/en-us/library/jj131023.aspx>. Acesso em 10/10/2012.

NATHAN, Adam. **WPF 4 - Unleashed**. Idiana: Sams, 2010.

PANTELIDIS, Dr.ª Veronica S. **VR in the School. Volume 1**, Number 1, June 1995.

Disponível em: < http://vr.coe.ecu.edu/vrits/vris1-1.htm >. Acessoem: 28 nov. 2012.

PERLINI, Nara Marilene Oliveira Girardon; FARO, Ana Cristina Mancussi. **Cuidar de pessoa incapacidade por acidente vascular cerebral no domicílio: o fazer do cuidador familar**. Revista da Escola de Enfermagem da USP. 2005; 39(2): 154-63.

PINHEIRO, Dr. Pedro. **7 sintomas do AVC**. MD Saúde, São Paulo, outubro de 2011. Disponível em: <http://www. mdsaude.com/2011/10/sintomas-avc.html>. Acesso em 12/03/2013 às 14:00 h.

SARDI, Marcelo Durigon; SCHUSTER, Rodrigo Costa; ALVARENGA, Luiz Fernando Calage. **Efeitos da realidade virtual em hemiparéticos crônicos pós-acidente vascular encefálico**. Revista Brasileira de Saúde, ano 10, nº 32, abr/jun 2012;

SARDI, Marcelo Durigon; SCHUSTER, Rodrigo Costa; ALVARENGA; Luiz Fernando Calage. **Efeitos da Realidade Virtual em Hemiparéticos Crônicos Pós-acidente Vascular Encefálico**. Revista Brasileira de Ciências da Saúde, ano 10, nº 32, abr/jun 2012.

SARTI C, et al. **International trends in mortality from stroke**, 1968 to 1994. Disponível em: http://stroke.ahajournals.org/content/31/7/1588.full. Acesso em 12/03/2013 às 14:00 h.

SBDCV. Sociedade Brasileira de Doenças Cerebrovasculares**. Acidente Vascular Cerebral**. Disponível em: http://www.sbdcv.org.br/publica_avc.asp. Acesso em 12/03/2013, às 14:12 h

SILVA, Emanuel de Jesus Alves. **Reabilitação após o AVC** (Mestrado). Universidade do Porto - Faculdade de Medicina, Portugal, 2010

Tw Game. **Xbox 360 - Kinect Sensor com jogo Kinect Adventures**. Disponível em: < http://www.twgame.com.br/products.php?product=Xbox-360-%252d-Kinect-Sensor-com-jogo-Kinect-Adventures >. Acesso em 11/11/2012.

VANDERLINE, Fernando. **Vídeo Games na Saúde e Reabilitação - FISIOGAMES Trazendo o sorriso de volta a reabilitação**. 1ª Ed. São Paulo: Editora Schoba, 2010.

VISUAL STUDIO. **Comparação de Capacidades**. Visual Studio, 2013. Disponível em: < http://www.microsoft.com/visualstudio/ptb/products/compare>. Acesso em 20/03/2013, às 21:10 h.

WHO. Health topics: Stroke, **Cerebrovascular accident. World Health Organization**. Disponível em: <http://www.who.int/topics/cerebrovascular_accident/en/>. Acesso em 12/01/2013, às 15:20 h.

WIPO-PATENTSCOPE. **(WO2007043036) Method And System For Object Reconstruction**. Disponível em: <http://patentscope.wipo.int/search/en/ WO200704303 6>. Acesso em 12/12/2012, às 15:30 h.

ZANETTI, Marina. **Acidente Vascular Cerebral – AVC | Isquêmico, Hemorrágico, Sintomas**. Med Foco. Disponível em: < http://www.medfoco.com.br/acidente-vascular-cerebral-avc-isquemico-hemorragico-sintomas/ >. Acesso em 12/01/2013.

7. ANEXOS

7.1 – Anexos da Aplicação Captura Esqueleto

7.1.1 - Método DesenhaEsqueleto

Com base nos pontos reconhecidos pelo Kinect o método faz o desenho do esqueleto da pessoa que se encontra em frente ao sensor, esse método utiliza ainda dois método de apoio GerarFigura (Listagem) e GetJointPoint (Listagem).

Listagem 7.1: Código Fonte do Método DesenhaEsqueleto

```
01.     private void DesenhaEsqueleto(Skeleton skeleton)
02.     {
03.         var corLinhaSkeleton = Brushes.Black;
04.         //Desenhando o esqueleto
05.         var joints = new[]
06.                      {
07.                          JointType.Head, JointType.ShoulderCenter,
08.                          JointType.Spine, JointType.HipCenter
09.                      };
10.         var figura = GerarFigura(skeleton, corLinhaSkeleton, joints);
11.         LayoutRoot.Children.Add(figura);
12.         joints = new[]
13.                  {
14.                      JointType.ShoulderCenter, JointType.ShoulderLeft,
15.                      JointType.ElbowLeft, JointType.WristLeft,
16.                      JointType.HandLeft
17.                  };
18.         figura = GerarFigura(skeleton, corLinhaSkeleton, joints);
19.         LayoutRoot.Children.Add(figura);
20.         joints = new[]
21.                  {
22.                      JointType.ShoulderCenter, JointType.ShoulderRight,
23.                      JointType.ElbowRight, JointType.WristRight,
24.                      JointType.HandRight
24.                  };
26.         figura = GerarFigura(skeleton, corLinhaSkeleton, joints);
27.         LayoutRoot.Children.Add(figura);
28.         joints = new[]
29.                  {
30.                      JointType.HipCenter, JointType.HipLeft,
31.                      JointType.KneeLeft, JointType.AnkleLeft,
32.                      JointType.FootLeft
33.                  };
34.         figura = GerarFigura(skeleton, corLinhaSkeleton, joints);
35.         LayoutRoot.Children.Add(figura);
36.         joints = new[]
37.                  {
38.                      JointType.HipCenter, JointType.HipRight,
39.                      JointType.KneeRight, JointType.AnkleRight,
40.                      JointType.FootRight
```

```
41.                 };
42.         figura = GerarFigura(skeleton, corLinhaSkeleton, joints);
43.         LayoutRoot.Children.Add(figura);
44.     }
```

O método DesenhaEsqueleto recebe como parâmetro de entrada o esqueleto (skeleton), um tipo de dado gerado pelo sensor Microsoft Kinect®, com todos os pontos do esqueleto, baseado nesses pontos o método gera um vetor com os pontos da parte desejada e passa como parâmetro esse vetor para o método GeraFigura (Listagem 6.2), o qual possui a responsabilidade de gerar uma linha ligando os pontos.

Listagem 7.2: Código Fonte do Método GeraFigura

```
01.         private Polyline GerarFigura(Skeleton skeleton, Brush corLinhaSkeleton, IEnumerable<JointType>
joints)
02.     {
03.         var linha = new Polyline {StrokeThickness = 8, Stroke = corLinhaSkeleton};
04.         foreach (var t in joints)
05.         {
06.             linha.Points.Add(GetJointPoint(skeleton.Joints[t]));
07.         }
08.         return linha;
09.     }
```

O método GetPointJoint (Listagem 6.3), recebe como parâmetro de entrada um determinado ponto, e faz o calculo da posição desse ponto na tela retornando para o método GeraFigura (Lisagem 6.2) um dado do tipo Point que contem as posições x, y desse ponto na tela.

Listagem 7.3: Código Fonte do Método GetJointPoint

```
01.     private Point GetJointPoint(Joint joint)
02.     {
03.         var point = _myKinect.MapSkeletonPointToDepth(
04.             joint.Position,
05.             _myKinect.DepthStream.Format
06.             );
07.         point.X *= (int) LayoutRoot.ActualWidth/_myKinect.DepthStream.FrameWidth;
08.         point.Y *= (int) LayoutRoot.ActualHeight/_myKinect.DepthStream.FrameHeight;
09.         return new Point(point.X, point.Y);
10.     }
```

7.1.2 - Método ObterInformações

O método ObterInformações recebe como parâmetro de entrada o esqueleto reconhecido pelo sensor Microsoft Kinect®, e faz o calculo da posição x, y e z dos pontos.

Para o ponto escolhido pela pessoa na tela Principal da Aplicação Captura Esqueleto (Figura 2.26), as informações da posição do ponto serão mostradas no lado esquerdo da tela.

Listagem 7.4: Código Fonte do Método ObterInformações

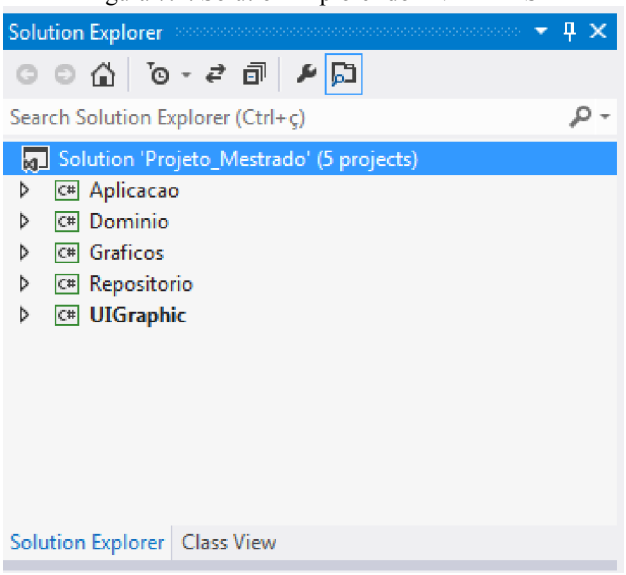
```
01.    private void ObterInformacoes(Skeleton esqueletoAtual)
02.    {
03.    var ponto = cboPontos.Text;
04.     var posicaoX = 0;
05.    var posicaoY = 0;
06.    var posicaoZ = 0.0;
07.    ColorImagePoint posicao;
08.    switch (ponto)
09.     {
10.     case "Cabeça":
11.          posicao = _myKinect.MapSkeletonPointToColor(esqueletoAtual.Joints
                 [JointType.Head].Position, ColorImageFormat.RgbResolution640x480Fps30);
12.         posicaoX = posicao.X;
13.         posicaoY = posicao.Y;
14.         posicaoZ = esqueletoAtual.Joints[JointType.Head].Position.Z;
15.         break;
17.     case "Ombro Central":
18.         posicao = _myKinect.MapSkeletonPointToColor(esqueletoAtual.Joints
              [JointType.ShoulderCenter].Position, ColorImageFormat.
                 RgbResolution640x480Fps30);
19                posicaoX = posicao.X;
20                posicaoY = posicao.Y;
21                posicaoZ = esqueletoAtual.Joints[JointType.ShoulderCenter].Position.Z;
22                break;
//Repete o processo de calculo para os outros pontos (Espinha, quadril Central e etc)
23.          }
24.          lblPonto.Content = "Ponto: " + ponto;
25.          lblPosicaoX.Content = string.Format("Posicao X: {0}", posicaoX);
26.          lblPosicaoY.Content = string.Format("Posicao Y: {0}", posicaoY);
27.          lblPosicaoZ.Content = string.Format("Posicao Z: {0:0.000}", posicaoZ);
28.       }
```

7.2 – Anexos do AVARFMS

7.2.1- O projeto ao AVARFMS

O Ambiente Virtual de Reabilitação Funcional de Membros Superiores foi desenvolvido utilizando o Microsoft Visual Studio 2012 (Versão Ultimate), o projeto é composto por cinco subprojetos (Figura 7.1) : Aplicação, Dominio, Graficos, Repositório e UIGraphic. O projeto foi divido dessa forma buscando uma melhor separação das responsabilidades de cada camada.

Figura 7.1: Solution Explorer do AVARFMS



O projeto Aplicação e responsável pela parte de persitência de dados do aplicativo o código fonte desse projeto e mostrado nas listagens 7.5, 7.6, 7.7, 7.8 e 7.9.

Listagem 7.5: Código fonte do arquivo AppAtividade.

```
01  using System.Collections.Generic;
02  using System.Linq;
03  using Dominio;
04  using Repositorio;
05
06  namespace Aplicacao
07  {
08      public class AppAtividade
09      {
10          private BancoDadosAmbienteVirtual banco;
11
12          public AppAtividade()
13          {
14              banco = new BancoDadosAmbienteVirtual();
15          }
16
17          public void Salvar(Atividade atividade)
18          {
19              banco.Atividades.Add(atividade);
20              banco.SaveChanges();
21          }
22
23          public IEnumerable<Atividade> Lista()
24          {
25              return banco.Atividades.ToList();
26          }
27
28          public void Alterar(Atividade atividade)
29          {
30              var atividadeAlterar = banco.Atividades.First(a => a.Id ==
31  atividade.Id);
32              atividadeAlterar.NomeTerapeuta = atividade.NomeTerapeuta;
33              atividadeAlterar.IdPaciente = atividade.IdPaciente;
```

```
34              atividadeAlterar.LadoTreino = atividade.LadoTreino;
35              atividadeAlterar.TempoCriacaoMovimento =
36  atividade.TempoCriacaoMovimento;
37              atividadeAlterar.Observacao = atividade.Observacao;
38              atividadeAlterar.DataCriacao = atividade.DataCriacao;
39              atividadeAlterar.PontoTamanho = atividade.PontoTamanho;
40              atividadeAlterar.DistanciaSensorZ = atividade.DistanciaSensorZ;
41              atividadeAlterar.DistanciaSensorX = atividade.DistanciaSensorX;
42              atividadeAlterar.DistanciaSensorY = atividade.DistanciaSensorY;
43              banco.SaveChanges();
44          }
45
46          public void Excluir(Atividade atividade)
47          {
48              var atividadeExcluir = banco.Atividades.First(a => a.Id ==
50  atividade.Id);
51              banco.Set<Atividade>().Remove(atividadeExcluir);
52              banco.SaveChanges();
53          }
54
55          public Atividade ConsultarAtividade(int id)
56          {
57              return banco.Atividades.ToList().First(p => p.Id == id);
58          }
59      }
60  }
```

Listagem 7.6: Código fonte do arquivo AppPontosTrajetoria.

```
01  using System.Collections.Generic;
02  using System.Linq;
03  using Dominio;
04  using Repositorio;
05
06  namespace Aplicacao
07  {
08      public class AppPontosTrajetoria
09      {
10          private BancoDadosAmbienteVirtual banco;
11
12          public AppPontosTrajetoria()
13          {
14              banco = new BancoDadosAmbienteVirtual();
15          }
16
17          public void Salvar(PontosTrajetoria ponto)
18          {
19              banco.PontosTrajetorias.Add(ponto);
20              banco.SaveChanges();
21          }
22
23          public IEnumerable<PontosTrajetoria> Lista()
24          {
25              return banco.PontosTrajetorias.ToList();
26          }
27
28          public void Alterar(PontosTrajetoria ponto)
29          {
30              var pontoAlterar = banco.PontosTrajetorias.First(p => p.Id ==
31  ponto.Id);
```

```
32              pontoAlterar.IdAtividade = ponto.IdAtividade;
33              pontoAlterar.IdPaciente = ponto.IdPaciente;
34              pontoAlterar.X = ponto.X;
35              pontoAlterar.Y = ponto.Y;
36              banco.SaveChanges();
37          }
38
39          public void Excluir(PontosTrajetoria ponto)
40          {
41              var pontoExcluir = banco.PontosTrajetorias.First(p => p.Id ==
42  ponto.Id);
43              banco.Set<PontosTrajetoria>().Remove(pontoExcluir);
44              banco.SaveChanges();
45          }
46
47          public PontosTrajetoria ConsultarPonto(int id)
48          {
49              return banco.PontosTrajetorias.ToList().First(p => p.Id == id);
50          }
51
52          public List<PontosTrajetoria> ConsultarPontoPorAtividade(int
53  idHistorico)
54          {
55              return banco.PontosTrajetorias.Where(p => p.IdHistorico ==
56  idHistorico).ToList();
57          }
58      }
59
60  }
```

Listagem 7.7: Código fonte do arquivo AppPonto

```
01  using System.Collections.Generic;
02  using System.Linq;
03  using Dominio;
04  using Repositorio;
05
06  namespace Aplicacao
07  {
08      public class AppPonto
09      {
10           private BancoDadosAmbienteVirtual banco;
11
12           public AppPonto()
13          {
14              banco = new BancoDadosAmbienteVirtual();
15          }
16
17          public void Salvar(Ponto ponto)
18          {
19              banco.Pontos.Add(ponto);
20              banco.SaveChanges();
21          }
22
23          public IEnumerable<Ponto> Lista()
24          {
25              return banco.Pontos.ToList();
26          }
27
28          public void Alterar(Ponto ponto)
29          {
```

```
30              var pontoAlterar = banco.Pontos.First(p => p.Id == ponto.Id);
31              pontoAlterar.Numero = ponto.Numero;
32              pontoAlterar.Largura = ponto.Largura;
33              pontoAlterar.Altura = ponto.Altura;
34              pontoAlterar.PosicaoX = ponto.PosicaoX;
35              pontoAlterar.PosicaoY = ponto.PosicaoY;
36              pontoAlterar.IdAtividade = ponto.IdAtividade;
37              banco.SaveChanges();
38          }
39
40          public void Excluir(Ponto ponto)
41          {
42              var pontoExcluir = banco.Pontos.First(p => p.Id == ponto.Id);
43              banco.Set<Ponto>().Remove(pontoExcluir);
44              banco.SaveChanges();
45          }
46
47          public Ponto ConsultarPonto(int id)
48          {
49              return banco.Pontos.ToList().First(p => p.Id == id);
50          }
51
52          public List<Ponto> ConsultarPontoPorAtividade(int idAtividade)
53          {
54              return banco.Pontos.Where(p => p.IdAtividade ==
55  idAtividade).ToList();
56          }
57      }
58  }
```

Listagem 7.8: Código fonte do arquivo AppPaciente.

```
01  using System.Collections.Generic;
02  using System.Linq;
03  using Dominio;
04  using Repositorio;
05
06  namespace Aplicacao
07  {
08      public class AppPaciente
09      {
10          private BancoDadosAmbienteVirtual banco;
11
12          public AppPaciente()
13          {
14              banco = new BancoDadosAmbienteVirtual();
15          }
16
17          public void Salvar(Paciente paciente)
18          {
19              banco.Pacientes.Add(paciente);
20              banco.SaveChanges();
21          }
22
23          public IEnumerable<Paciente> Lista()
24          {
25              return banco.Pacientes.ToList();
26          }
27
28          public void Alterar(Paciente paciente)
29          {
```

```
            var pacienteAlterar = banco.Pacientes.First(p => p.Id ==
paciente.Id);
            pacienteAlterar.Nome = paciente.Nome;
            pacienteAlterar.Sexo = paciente.Sexo;
            pacienteAlterar.DataNascimento = paciente.DataNascimento;
            pacienteAlterar.TempoAcidente = paciente.TempoAcidente;
            pacienteAlterar.Observacoes = paciente.Observacoes;
            banco.SaveChanges();
        }

        public void Excluir(Paciente paciente)
        {
            var pacienteExcluir = banco.Pacientes.First(p => p.Id ==
paciente.Id);
            banco.Set<Paciente>().Remove(pacienteExcluir);
            banco.SaveChanges();
        }

        public Paciente ConsultarPaciente(int id)
        {
            return banco.Pacientes.ToList().First(p => p.Id == id);
        }

    }
}
```

Listagem 7.9: Código fonte do arquivo AppAtividadePaciente

```
using System.Collections.Generic;
using System.Linq;
using Dominio;
using Repositorio;

namespace Aplicacao
{
    public class AppHistoricoAtividadePaciente
    {
         private BancoDadosAmbienteVirtual banco;

         public AppHistoricoAtividadePaciente()
        {
            banco = new BancoDadosAmbienteVirtual();
        }

         public void Salvar(HistoricoAtividadePaciente historicoAtividadePaciente)
        {
            banco.HistoricoAtividadePacientes.Add(historicoAtividadePaciente);
            banco.SaveChanges();
        }

        public IEnumerable<HistoricoAtividadePaciente> Lista()
        {
            return banco.HistoricoAtividadePacientes.ToList();
        }

        public void Alterar(HistoricoAtividadePaciente historicoAtividadePaciente)
        {
            var historicoAtividadePacienteAlterar =
banco.HistoricoAtividadePacientes.First(a => a.Id ==
historicoAtividadePaciente.Id);
```

```
33              historicoAtividadePacienteAlterar.IdAtividade =
34  historicoAtividadePaciente.IdAtividade;
35              historicoAtividadePacienteAlterar.IdPaciente =
36  historicoAtividadePaciente.IdPaciente;
37              historicoAtividadePacienteAlterar.NumeroRepeticoes =
38  historicoAtividadePaciente.NumeroRepeticoes;
39              historicoAtividadePacienteAlterar.DataDesenvolvimento =
40  historicoAtividadePaciente.DataDesenvolvimento;
41              historicoAtividadePacienteAlterar.Tempo =
42  historicoAtividadePaciente.Tempo;
43              banco.SaveChanges();
44          }
45
46          public void Excluir(HistoricoAtividadePaciente historicoAtividadePaciente)
47          {
48              var historicoAtividadePacienteExcluir =
49  banco.HistoricoAtividadePacientes.First(a => a.Id ==
50  historicoAtividadePaciente.Id);
51
52  banco.Set<HistoricoAtividadePaciente>().Remove(historicoAtividadePacienteExcluir);
53              banco.SaveChanges();
54          }
55
56          public HistoricoAtividadePaciente ConsultarAtividade(int id)
57          {
58              return banco.HistoricoAtividadePacientes.ToList().First(p => p.Id ==
59  id);
60          }
61      }
62  }
```

O projeto Domino e responsável pelas classes de persitência, ou objetos POCO no C#, de dados do aplicativo o código fonte desse projeto e mostrado nas listagens 7.10, 7.11, 7.12, 7.13, 7.14, 7.15 e 7.16.

Listagem 7.10: Código fonte do arquivo Atividade.

```
01  namespace Dominio
02  {
03      public class Atividade
04      {
05          public int Id { get; set; }
06          public string Status { get; set; }
07          public string NomeTerapeuta { get; set; }
08          public int IdPaciente { get; set; }
09          public string LadoTreino { get; set; }
10          public int TempoCriacaoMovimento { get; set; }
11          public string Observacao { get; set; }
12          public int PontoTamanho { get; set; }
13          public string DataCriacao { get; set; }
14          public float DistanciaSensorZ { get; set; }
15          public int DistanciaSensorX { get; set; }
16          public int DistanciaSensorY { get; set; }
17          public int FeitaNumeroVezes { get; set; }
18      }
```

Listagem 7.11: Código fonte do arquivo HistoricoAtividadePaciente

```
01  namespace Dominio
02  {
03      public class HistoricoAtividadePaciente
04      {
05          public int Id { get; set; }
06          public int IdAtividade { get; set; }
07          public int IdPaciente { get; set; }
08          public int NumeroRepeticoes { get; set; }
09          public string DataDesenvolvimento { get; set; }
10          public double Tempo { get; set; }
11      }
12  }
```

Listagem 7.12: Código fonte do arquivo Opcoes

```
01  namespace Dominio
02  {
03      public class Opcoes
04      {
05          public bool ShowPaciente { get; set; }
06          public bool ShowEsqueleto { get; set; }
07          public bool ShowPonto { get; set; }
08          public bool ShowLinha { get; set; }
09          public bool ShowCronometro { get; set; }
10          public int EspessuraLinha { get; set; }
11          public int CorLinha { get; set; }
12      }
13  }
```

Listagem 7.13: Código fonte do arquivo Paciente.

```
01  namespace Dominio
02  {
03      public class Paciente
04      {
05          public int Id { get; set; }
06          public string Nome { get; set; }
07          public string Sexo { get; set; }
08          public string DataNascimento { get; set; }
09          public string TempoAcidente { get; set; }
10          public string Observacoes { get; set; }
11      }
12  }
```

Listagem 7.14: Código fonte do arquivo Ponto.

```
01  namespace Dominio
02  {
03      public class Ponto
04      {
05          public int Id { get; set; }
06          public int Numero { get; set; }
07          public int Largura { get; set; }
08          public int Altura { get; set; }
09          public double PosicaoX { get; set; }
10          public double PosicaoY { get; set; }
11          public int IdAtividade { get; set; }
12          public string tipoPonto { get; set; }
```

```
13      }
14  }
```

Listagem 7.15: Código fonte do arquivo PontosTrajetorio.

```
01  namespace Dominio
02  {
03      public class PontosTrajetoria
04      {
05          public int Id { get; set; }
06          public int IdHistorico { get; set; }
07          public int IdAtividade { get; set; }
08          public int IdPaciente { get; set; }
09          public int IdPontoDestino { get; set; }
10          public double X { get; set; }
11          public double Y { get; set; }
12      }
13  }
```

Listagem 7.16: Código fonte do arquivo PontosSuavizado

```
01  namespace Dominio
02  {
03      public class PontoSuavizado
04      {
05          public double X { get; set; }
06          public double Y { get; set; }
07      }
08  }
```

O projeto Graficos e responsável pela apresentação dos gráficos das métricas, esse projeto do tipo Windows Forms, o código fonte desse projeto e mostrado na listagem 7.17.

Listagem 7.17: Código fonte do arquivo Metricas

```
01  using System;
02  using System.Collections.Generic;
03  using System.Linq;
04  using System.Windows.Forms;
05  using System.Windows.Forms.DataVisualization.Charting;
06  using Aplicacao;
07  using Dominio;
08
09  namespace Graficos
10  {
11      public partial class Metricas : Form
12      {
13
14          private List<Paciente> _listaPacientes;
15          private List<HistoricoAtividadePaciente> _listHistorico;
16          private readonly AppPaciente _bdPaciente = new AppPaciente();
17          private readonly AppHistoricoAtividadePaciente _bdHistorico = new
18  AppHistoricoAtividadePaciente();
19          private readonly AppAtividade _bdAtividade = new AppAtividade();
20          private readonly AppPonto _bdPonto = new AppPonto();
21          private readonly AppPontosTrajetoria _bdPontosTrajetoria = new
22  AppPontosTrajetoria();
23
24          public Metricas()
25          {
```

```
            InitializeComponent();
            CarregarPacientes();
            cboAtividade.Enabled = false;
        }

        private void CarregarPacientes()
        {
            _listaPacientes = _bdPaciente.Lista().ToList();
            //Populando o cboPaciente
            cboPaciente.DataSource = _listaPacientes;
            cboPaciente.DisplayMember = "nome";
            cboPaciente.ValueMember = "id";
            cboPaciente.SelectedIndex = -1;
        }

        private void CarregarDadosPaciente(object sender, EventArgs e)
        {
            txtObservacaoPaciente.Text = ((Paciente)
cboPaciente.SelectedItem).Observacoes;
            _listHistorico =
                _bdHistorico.Lista().Where(h => h.IdPaciente == ((Paciente)
cboPaciente.SelectedItem).Id).ToList();
            var atividades = new List<HistoricoAtividadePaciente>();
            var aux = 0;
            foreach (var historico in _listHistorico)
            {
                if (aux == historico.IdAtividade) continue;
                aux = historico.IdAtividade;
                atividades.Add(historico);
            }


            cboAtividade.DataSource = atividades;
            cboAtividade.DisplayMember = "idAtividade";
            cboAtividade.ValueMember = "id";
            cboAtividade.SelectedIndex = -1;
            txtObservacaoAtividade.Clear();
            cboAtividade.Enabled = true;
        }

        private void CarregarDadosAtividade(object sender, EventArgs e)
        {
            if (cboAtividade.SelectedItem == null) return;
            var atividade =
                _bdAtividade.ConsultarAtividade(((HistoricoAtividadePaciente)
cboAtividade.SelectedItem).IdAtividade);
            txtObservacaoAtividade.Text = atividade.Observacao;
            cboRepeticao.Enabled = true;

            var paciente = ((Paciente) cboPaciente.SelectedItem).Id;

            PopularGraficoTempo(atividade.Id, paciente);
            PopularGraficoDistancia(atividade.Id, paciente);
            PopularGraficoErro(atividade.Id, paciente);
            PopularComboRepeticao();
            PopularGraficoComparativoTrajetoria(atividade.Id, paciente);
        }

        private void PopularGraficoComparativoTrajetoria(int id, int paciente)
        {
            var historicos = _bdHistorico.Lista().Where(h => h.IdPaciente ==
paciente).Where(a => a.IdAtividade == id).ToList();
```

```
88
89              //Removendo as séries
90              for (int s = graficoComparativo.Series.Count - 1; s >= 0; s--)
91              {
92                  graficoComparativo.Series.RemoveAt(s);
93              }
94
95              foreach (var historico in historicos)
96              {
97                  var pontosTrajetoria = _bdPontosTrajetoria.Lista().Where(p =>
98  p.IdHistorico == historico.Id).ToList();
99                  var pontosSuavizados = SuavizarTrajetoria(pontosTrajetoria);
100
101                 var serie = new Series
102                                 {
103                                     ChartType = SeriesChartType.Line,
104                                     Name = string.Format("Repetição {0}",
105 historico.Id)
106                                 };
107                 foreach (var ponto in pontosSuavizados)
108                 {
109                     serie.Points.AddXY(ponto.X, 768 - ponto.Y);
110                     //serie.Points.AddXY(ponto.X - ponto.Y);
111                 }
112                 graficoComparativo.Series.Add(serie);
113             }
114         }
115
116
117         private void PopularComboRepeticao()
118         {
119             var idPaciente = ((Paciente) cboPaciente.SelectedItem).Id;
120             var idAtividade = ((HistoricoAtividadePaciente)
120 cboAtividade.SelectedItem).IdAtividade;
121
122             var historico =
123                 _bdHistorico.Lista().Where(h => h.IdPaciente == idPaciente &&
124 h.IdAtividade == idAtividade).ToList();
125
126             cboRepeticao.DataSource = historico;
127             cboRepeticao.DisplayMember = "NumeroRepeticoes";
128             cboRepeticao.ValueMember = "id";
129             cboRepeticao.SelectedIndex = -1;
130
131         }
132
133         private void CboRepeticaoDropDownClosed(object sender, EventArgs e)
134         {
135             if (cboRepeticao.SelectedItem == null) return;
136             var idPaciente = ((Paciente) cboPaciente.SelectedItem).Id;
137             var idAtividade = ((HistoricoAtividadePaciente)
138 cboAtividade.SelectedItem).IdAtividade;
139             var numRepeticao = ((HistoricoAtividadePaciente)
140 cboRepeticao.SelectedItem).NumeroRepeticoes;
141             PopularGraficoTrajetória(idPaciente, idAtividade, numRepeticao);
142         }
143
144         private void PopularGraficoTrajetória(int idPaciente, int idAtividade,
145 int numRepeticao)
146         {
147             //Removendo as séries
148             for (int s = grTrajetoria.Series.Count - 1; s >= 0; s--)
```

```
149             {
150                 grTrajetoria.Series.RemoveAt(s);
151             }
152             var pontos = _bdPonto.Lista().Where(p => p.IdAtividade ==
153 idAtividade).ToList();
154             //grTrajetoria.Series[0].Points.Clear();
155             //grTrajetoria.Series[1].Points.Clear();
156             //grTrajetoria.Series[2].Points.Clear();
157             var serieTrajetoriaTO = new Series { ChartType =
158 SeriesChartType.Line,  Name = "Terapeuta" };
159             var serieTrajetoriaPAOriginal = new Series { ChartType =
160 SeriesChartType.Line, Name = "Paciente original" };
161             var serieTrajetoriaPASuavizada = new Series { ChartType =
162 SeriesChartType.Line, Name = "Paciente suavizada" };
163 
164             foreach (var ponto in pontos)
165             {
166                 serieTrajetoriaTO.Points.AddXY(ponto.PosicaoX, 768 -
167 ponto.PosicaoY);
168                 //grTrajetoria.Series[0].Points.AddXY(ponto.PosicaoX,
169 ponto.PosicaoY);
170             }
171 
172             var historico = _bdHistorico.Lista().FirstOrDefault(h => h.IdPaciente
173 == idPaciente &&
174 
175 h.IdAtividade == idAtividade &&
176 
177 h.NumeroRepeticoes == numRepeticao);
178             if (historico == null) return;
179             var pontosTrajetoria = _bdPontosTrajetoria.Lista().Where(p =>
180 p.IdHistorico == historico.Id).ToList();
181 
182             foreach (var ponto in pontosTrajetoria)
183             {
184                 serieTrajetoriaPAOriginal.Points.AddXY(ponto.X, 768 - ponto.Y);
185                 //grTrajetoria.Series[1].Points.AddXY(ponto.X, ponto.Y);
186             }
187 
188             var pontosSuavizados = SuavizarTrajetoria(pontosTrajetoria);
189             foreach (var ponto in pontosSuavizados)
190             {
191                 double x = Convert.ToDouble(string.Format("{0:00}", ponto.X));
192                 double y = Convert.ToDouble(string.Format("{0:00}", ponto.Y));
193                 serieTrajetoriaPASuavizada.Points.AddXY(x, 768 - y);
194                 //grTrajetoria.Series[2].Points.AddXY(ponto.X, ponto.Y);
195             }
196 
197             grTrajetoria.Series.Add(serieTrajetoriaTO);
198             grTrajetoria.Series.Add(serieTrajetoriaPAOriginal);
199             grTrajetoria.Series.Add(serieTrajetoriaPASuavizada);
200 
201         }
202 
203         private IEnumerable<PontoSuavizado>
204 SuavizarTrajetoria(List<PontosTrajetoria> pontosBrutos)
205         {
206             const int janela = 10;
207             var contJanela = 0;
208             var pontoSuavizado = new PontoSuavizado();
209             var pontosSuavizado = new List<PontoSuavizado>();
210             int p1 = 0;
```

```
211             int p2 = 0;
212             foreach (var ponto in pontosBrutos)
213             {
214                 if (contJanela < janela)
215                 {
216                     pontoSuavizado = CalculoMediaMovel(pontosBrutos, p1, p2,
217 janela);
218                     p2++;
219                     contJanela++;
220                     //Console.WriteLine("{0:0.0000}, {1:0.0000}",
221                     //    pontoSuavizado.X,
222                     //    pontoSuavizado.Y);
223                 } else if (contJanela < pontosBrutos.Count)
224                 {
225                     p1++;
226                     pontoSuavizado = CalculoMediaMovel(pontosBrutos, p1, p2,
227 janela);
228                     p2++;
229                     contJanela++;
230                     //Console.WriteLine("{0:0.0000}, {1:0.0000}",
231                     //    pontoSuavizado.X,
232                     //    pontoSuavizado.Y);
233                 }
234                 pontosSuavizado.Add(pontoSuavizado);
235             }
236             return pontosSuavizado;
237         }
238 
239         private PontoSuavizado CalculoMediaMovel(List<PontosTrajetoria>
240 pontosBrutos, int p1, int p2, int janela)
241         {
242             var somaX = 0.0;
243             var somaY = 0.0;
244             var i = p1;
245             for (; i <= p2; i++)
246             {
247                 somaX += pontosBrutos[i].X;
248                 somaY += pontosBrutos[i].Y;
249             }
250 
251             return p1 == 0 ? new PontoSuavizado {X = somaX/i, Y = somaY/i} : new
252 PontoSuavizado { X = somaX / janela, Y = somaY / janela };
253 
254         }
255 
256         private void PopularGraficoErro(int idAtividade, int idPaciente)
257         {
258             var historico = _bdHistorico.Lista().Where(h => h.IdAtividade ==
259 idAtividade && h.IdPaciente == idPaciente).ToList();
260             var pontoTO = _bdPonto.Lista().Where(t => t.IdAtividade ==
261 idAtividade).ToList();
262 
263             grErro.Series[0].Points.Clear();
264             grErro.Series[1].Points.Clear();
265 
266             foreach (var h in historico)
267             {
268                 var pontoTrajetoriaPA =
269                     _bdPontosTrajetoria.Lista().Where(t => t.IdAtividade ==
270 idAtividade && t.IdPaciente == idPaciente && t.IdHistorico == h.Id).
271                         ToList();
272                var erroBruto = CalcularErro(pontoTrajetoriaPA, pontoTO);
```

```
273
274                 var pontosSuavizados =
275 SuavizarTrajetoria(pontoTrajetoriaPA).ToList();
276                 var pontos = new List<PontosTrajetoria>();
277                 var listPontosPA = pontosSuavizados as List<PontoSuavizado> ??
278 pontosSuavizados.ToList();
279                 for( int i = 0; i < pontoTrajetoriaPA.Count; i++)
280                 {
281                     var p = new PontosTrajetoria
282                                 {
283                                     Id = pontoTrajetoriaPA[i].Id,
284                                     IdAtividade =
285 pontoTrajetoriaPA[i].IdAtividade,
286                                     IdHistorico =
287 pontoTrajetoriaPA[i].IdHistorico,
288                                     IdPaciente = pontoTrajetoriaPA[i].IdPaciente,
289                                     IdPontoDestino =
290 pontoTrajetoriaPA[i].IdPontoDestino,
291                                     X = listPontosPA[i].X,
292                                     Y = listPontosPA[i].Y
293                                 };
294                      pontos.Add(p);
295                 }
296                 var erroSuavizado = CalcularErro(pontos, pontoTO);
297
298                 grErro.Series[0].Points.Add(new DataPoint(h.NumeroRepeticoes,
299 erroBruto));
300                 grErro.Series[1].Points.Add(new DataPoint(h.NumeroRepeticoes,
301 erroSuavizado));
302             }
303
304
305
306
307         }
308
309         private double CalcularErro(IEnumerable<PontosTrajetoria> listPontosPA,
310 IReadOnlyList<Ponto> listPontosTO )
311         {
312             var distancias = new List<double>();
313             var erro = 0.0;
314             foreach (var pontoPA in listPontosPA)
315             {
316                 distancias.Add(DistanciaSegmentoLinha(
317                     listPontosTO[pontoPA.IdPontoDestino - 1],
318                     listPontosTO[pontoPA.IdPontoDestino],
319                     pontoPA
320                     ));
321             }
322
323            erro = Convert.ToDouble(string.Format("{0:0.00}",
324 Normalizar(distancias)/Math.Sqrt(distancias.Count)));
325             return erro;
326         }
327
328         private double DistanciaSegmentoLinha(Ponto origem, Ponto destino,
329 PontosTrajetoria trajeto)
330         {
331             double r;
332             var vx = origem.PosicaoX - trajeto.X;
333             var vy = origem.PosicaoY- trajeto.Y;
334             var ux = destino.PosicaoX - origem.PosicaoX;
```

```
335             var uy = destino.PosicaoY - origem.PosicaoY;
336             var lenSqr = (ux*ux + uy*uy);
337             var detP = -vx*ux + -vy*uy;
338             if (detP < 0)
339             {
340                 r = Normalizar(
341                     new PontoSuavizado {X = origem.PosicaoX, Y =
342 origem.PosicaoY},
343                     new PontoSuavizado {X = trajeto.X, Y = trajeto.Y}
344                     );
345             } else if (detP > lenSqr)
346             {
347                 r = Normalizar(
348                     new PontoSuavizado { X = destino.PosicaoX, Y =
349 destino.PosicaoY },
350                     new PontoSuavizado { X = trajeto.X, Y = trajeto.Y }
351                     );
352             }
353             else
354             {
355                 r = Math.Abs(ux*vy - uy*vx)/Math.Sqrt(lenSqr);
356             }
357 
358             return r;
359 
360         }
361 
362         private double Normalizar(PontoSuavizado p1, PontoSuavizado p2)
363         {
364             return Math.Sqrt(Math.Pow((p1.X - p2.X),2) + Math.Pow((p1.Y -
365 p2.Y),2));
366         }
367 
368         private double Normalizar(IEnumerable<double> listDistancia)
369         {
370             var soma = listDistancia.Sum(valor => Math.Pow(valor, 2));
371             var normal = Math.Sqrt(soma);
372             return normal;
373         }
374 
375         private void PopularGraficoDistancia(int idAtividade, int idPaciente)
376         {
377             //Recuperandos os históricos do paciente.
378             var historicos = _bdHistorico.Lista().Where(h => h.IdAtividade ==
379 idAtividade && h.IdPaciente == idPaciente).ToList();
380             var distancias = new List<double>();
381 
382             foreach (var historico in historicos)
383             {
384                 var pontosTrajetoria = _bdPontosTrajetoria.Lista().Where(id =>
385 id.IdHistorico == historico.Id).ToList();
386                 if (pontosTrajetoria.Count <= 1) continue;
387                 var somaDistancia = 0.0;
388                 for (var i = 0; i < pontosTrajetoria.Count - 1; i++)
389                 {
390                     somaDistancia += CalcularDistancia(pontosTrajetoria[i],
391 pontosTrajetoria[i + 1]);
392                 }
393                 distancias.Add(Double.Parse(string.Format("{0:0.00}",
394 somaDistancia)));
395                 Console.WriteLine("Soma: {0:0.00}", somaDistancia);
396             }
```

```
397
398             grDistancia.Series[0].Points.Clear();
399             var atividade = 0;
400             foreach (var distancia in distancias)
401             {
402                 grDistancia.Series[0].Points.AddXY(atividade++, distancia);
403             }
404
405         }
406
407         private double CalcularDistancia(PontosTrajetoria p1, PontosTrajetoria
408 p2)
409         {
410             return Math.Sqrt(Math.Pow((p2.X - p1.X), 2) + Math.Pow((p2.Y - p1.Y),
411 2));
412         }
413
414
415
416         private void PopularGraficoTempo(int idAtividade, int idPaciente)
417         {
418             var atividades = _bdHistorico.Lista().Where(a => a.IdAtividade ==
419 idAtividade && a.IdPaciente == idPaciente).ToList();
421             grTempo.Series[0].Points.Clear();
422             foreach (var historico in atividades)
423                 grTempo.Series[0].Points.Add(new
424 DataPoint(historico.NumeroRepeticoes, historico.Tempo));
425         }
426     }
427 }
428
```

O projeto Respositorio e responsável pelo controle entre a aplicação e o SGBD o código fonte desse projeto e mostrado na listagem 7.18.

Listagem 7.18: Código fonte do arquivo BancoDadosAmbienteVirtual.

```
01  using System.Data.Entity;
02  using Dominio;
03
04  namespace Repositorio
05  {
06      public class BancoDadosAmbienteVirtual : DbContext
07      {
08          public DbSet<Paciente> Pacientes { get; set; }
09          public DbSet<Atividade> Atividades { get; set; }
10          public DbSet<Ponto> Pontos { get; set; }
11          public DbSet<HistoricoAtividadePaciente> HistoricoAtividadePacientes {
12  get; set; }
13          public DbSet<PontosTrajetoria> PontosTrajetorias { get; set; }
14      }
15  }
```

O projeto UIGraphic e responsável pela Interface Gráfica com o Usuário, esse e um projeto do Tipo WPF. Projetos do tipo WPF são compostos de dois tipos de arquivos os xaml que compõem a interface gráfico com o usuário e os cs que são as classes C# onde são

implementadas as funcionalidades da tela, código fonte desse projeto e mostrado na listagem 7.18 ao 7.32.

Listagem 7.19: Código fonte do arquivo ManterAtividade.xaml

```
<UserControl x:Class="UIGraphic.Formularios.ManterAtividade"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="750"
Loaded="ManterAtividadeLoad">
    <Grid Background="{DynamicResource {x:Static SystemColors.InfoBrushKey}}">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Label Grid.Column="0" Content="Listagem de Atividade"
HorizontalAlignment="Left" Margin="68,10,0,0" VerticalAlignment="Top"
FontSize="22" FontWeight="Bold" FontFamily="Comic Sans MS" Width="248"
HorizontalContentAlignment="Center"/>
        <DataGrid x:Name="gridListagemAtividade"  Grid.Column="0"
Margin="10,60,10,140" Width="355" Height="250"
MouseDoubleClick="SelecionarAtividadeClick"
MouseEnter="GridMouseEnterAtualizarAlteracoes"/>
        <Button x:Name="btnCriarAtividade" Content="Criar Atividade"
HorizontalAlignment="Left" Margin="46,390,0,0" VerticalAlignment="Top" Width="285"
Height="50" FontFamily="Courier New" FontWeight="Bold" FontSize="24"
Click="btnCriarAtividade_Click"/>
        <Label Grid.Column="1" Content="Nome do Terapeuta Responsável:"
HorizontalAlignment="Left" Margin="10,60,0,0" VerticalAlignment="Top"
Style="{DynamicResource labelManutencao}"/>
        <Label Grid.Column="1" Content="Lado:" HorizontalAlignment="Left"
Margin="10,166,0,0" VerticalAlignment="Top" Style="{DynamicResource
labelManutencao}"/>
        <Label Grid.Column="1" Content="Tempo de Movimento:"
HorizontalAlignment="Left" Margin="215,166,0,0" VerticalAlignment="Top"
Style="{DynamicResource labelManutencao}"/>
        <TextBox x:Name="txtAtividadeTerapeutaNome" Grid.Column="1"
HorizontalAlignment="Left" Height="23" Margin="10,92,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="355"/>
        <Label Grid.Column="1" Content="Manutenção de Atividades"
HorizontalAlignment="Left" Margin="57,10,0,0" VerticalAlignment="Top"
FontSize="22" FontWeight="Bold" FontFamily="Comic Sans MS" Width="278"
HorizontalContentAlignment="Center"/>
        <ComboBox x:Name="cboAtividadeLado" Grid.Column="1"
HorizontalAlignment="Left" Margin="10,196,0,0" VerticalAlignment="Top"
Width="120">
            <ComboBoxItem Content="Esquerdo"/>
            <ComboBoxItem Content="Direito"/>
        </ComboBox>
        <Button x:Name="pacienteGravar" Content="Gravar" Grid.Column="1"
HorizontalAlignment="Left" Margin="290,367,0,0" VerticalAlignment="Top"
Style="{DynamicResource buttonAtividadeStyle}" Click="AtividadeGravarClick" />
        <Button x:Name="pacienteExcluir" Content="Excluir" Grid.Column="1"
HorizontalAlignment="Left" Margin="290,410,0,0" VerticalAlignment="Top"
Style="{DynamicResource buttonAtividadeStyle}" Click="AtividadeExcluirClick" />
        <ComboBox x:Name="cboAtividadeTempo" Grid.Column="1"
HorizontalAlignment="Left" Margin="215,196,0,0" VerticalAlignment="Top"
Width="150"/>
```

```
58          <Label Grid.Column="1" Content="Nome do Paciente:"
59  HorizontalAlignment="Left" Margin="10,115,0,0" VerticalAlignment="Top"
60  Style="{DynamicResource labelManutencao}"/>
61          <ComboBox Name="cboListaPacientes"  Grid.Column="1"
62  HorizontalAlignment="Left" Margin="10,144,0,0" VerticalAlignment="Top" Width="355"
63  DropDownClosed="cboListaPacientes_DropDownClosed"/>
64          <Label Grid.Column="1" Content="Observações sobre a atividade:"
65  HorizontalAlignment="Left" Margin="10,223,0,0" VerticalAlignment="Top"
66  Style="{DynamicResource labelManutencao}"/>
67          <TextBox Name="txtAtividadeObservacao" Grid.Column="1"
68  HorizontalAlignment="Left" Height="71" Margin="10,253,0,0" TextWrapping="Wrap"
69  VerticalAlignment="Top" Width="355"/>
70          <Label x:Name="lblTamanhoPonto" Grid.Column="1" Content="Tamanho do Ponto:
71  10 pixels." HorizontalAlignment="Left" Margin="10,332,0,0" VerticalAlignment="Top"
72  Style="{DynamicResource labelManutencao}"/>
73          <Slider x:Name="sliderTamanhoPonto" Grid.Column="1"
74  HorizontalAlignment="Left" Margin="13,362,0,0" VerticalAlignment="Top" Width="134"
75  Minimum="10" Maximum="60" ValueChanged="AlterarTamanhoPonto"/>
76          <Canvas x:Name="canvasAreaModelo" Grid.Column="1"
77  HorizontalAlignment="Left" Height="60" Margin="154,358,0,0"
78  VerticalAlignment="Top" Width="60">
79              <Ellipse Height="10" Canvas.Left="0" Stroke="Black" Canvas.Top="0"
80  Width="10" HorizontalAlignment="Center" VerticalAlignment="Center" Fill="Black"/>
81          </Canvas>
82          <Label Grid.Column="1" Content="Data de criação:"
83  HorizontalAlignment="Left" Margin="10,390,0,0" VerticalAlignment="Top"
84  Style="{DynamicResource labelManutencao}"/>
85          <TextBox x:Name="txtDataCriacao" Grid.Column="1"
86  HorizontalAlignment="Left" Height="23" Margin="13,414,0,0" TextWrapping="Wrap"
87  VerticalAlignment="Top" Width="134"/>
88
89
90      </Grid>
91  </UserControl>
92
```

Listagem 7.20: Código fonte do arquivo ManterAtividade.cs

```
01  using System;
02  using System.Collections.Generic;
03  using System.Linq;
04  using System.Windows;
05  using System.Windows.Controls;
06  using System.Windows.Input;
07  using System.Windows.Media;
08  using System.Windows.Shapes;
09  using Dominio;
10  using UIGraphic.Janela;
11
12  namespace UIGraphic.Formularios
13  {
14      /// <summary>
15      /// Interaction logic for ManterAtividade.xaml
16      /// </summary>
17      public partial class ManterAtividade : UserControl
18      {
19          private bool eNovoAtividade;
20          private Atividade atividadeGeral;
21          private int larguraBola;
22
23          private bool criandoAtividade = false;
24
```

```
25          public ManterAtividade()
26          {
27              InitializeComponent();
28              eNovoAtividade = true;
29          }
30
31          private void AtividadeGravarClick(object sender, RoutedEventArgs e)
32          {
33              try
34              {
35                  var idAG = 0;
36                  var stStatus = "Em Aberto";
37                  if (atividadeGeral != null)
38                  {
39                      idAG = atividadeGeral.Id;
40                      stStatus = atividadeGeral.Status;
41                  }
42                  atividadeGeral = new Atividade
43                      {
44                          Id = idAG,
45                          Status = stStatus,
46                          NomeTerapeuta = txtAtividadeTerapeutaNome.Text,
47                          IdPaciente = (int) cboListaPacientes.SelectedValue,
48                          LadoTreino = cboAtividadeLado.Text,
49                          Observacao = txtAtividadeObservacao.Text,
50                          TempoCriacaoMovimento =
51  (int)cboAtividadeTempo.SelectedValue,
52                          PontoTamanho = (int)sliderTamanhoPonto.Value,
53                          DataCriacao = txtDataCriacao.Text,
54                          DistanciaSensorZ = 0.0f
55                  };
56
57                  if (eNovoAtividade)
58                  {
59                      Principal.appAtividade.Salvar(atividadeGeral);
60                      MessageBox.Show(
61                          "Atividade gravado com sucesso",
62                          "Informação",
63                          MessageBoxButton.OK,
64                          MessageBoxImage.Information);
65                  }
66                  else
67                  {
68                      Principal.appAtividade.Alterar(atividadeGeral);
69                      MessageBox.Show(
70                          "Atividade alterado com sucesso",
71                          "Informação",
72                          MessageBoxButton.OK,
73                          MessageBoxImage.Information);
74                  }
75
76                  ShowListaAtividades();
77                  eNovoAtividade = true;
78                  atividadeGeral = null;
79                  LimparCampos();
80              }
81              catch (Exception ex)
82              {
83                  MessageBox.Show(ex.Message);
84              }
85          }
86
```

```
87          private void LimparCampos()
88          {
89              txtAtividadeTerapeutaNome.Text = string.Empty;
90              cboListaPacientes.SelectedIndex = -1;
91              cboAtividadeLado.SelectedIndex = -1;
92              cboAtividadeTempo.SelectedIndex = 0;
93              txtAtividadeObservacao.Text = string.Empty;
94              cboAtividadeLado.SelectedIndex = -1;
95              sliderTamanhoPonto.Value = 10;
96              txtDataCriacao.Text = string.Empty;
97              AddCiculoAreaModelo();
98          }
99
100         private void ShowListaAtividades()
101         {
102             gridListagemAtividade.ItemsSource = Principal.appAtividade.Lista();
103             gridListagemAtividade.IsReadOnly = true;
104
105         }
106
107         private void CarregarListaPacientes()
108         {
109             var pacientes = Principal.appPaciente.Lista();
110             var dictionaryPaciente = pacientes.ToDictionary(paciente =>
111 paciente.Id, paciente => paciente.Nome);
112             cboListaPacientes.ItemsSource = dictionaryPaciente;
113             cboListaPacientes.DisplayMemberPath = "Value";
114             cboListaPacientes.SelectedValuePath = "Key";
115         }
116
117         private void ManterAtividadeLoad(object sender, RoutedEventArgs e)
118         {
119             CarregarListaPacientes();
120             CarregarListaTempo();
121             ShowListaAtividades();
122         }
123
124         private void CarregarListaTempo()
125         {
126             //Populando tempo
127             var dictionaryTempo = new Dictionary<int, string>();
128             for (var i = 2; i <= 10; i++)
129             {
130                 dictionaryTempo.Add(i, string.Format("{0} segundos", i));
131             }
132             cboAtividadeTempo.ItemsSource = dictionaryTempo;
133             cboAtividadeTempo.DisplayMemberPath = "Value";
134             cboAtividadeTempo.SelectedValuePath = "Key";
135             cboAtividadeTempo.SelectedIndex = 0;
136         }
137
138         private void AddCiculoAreaModelo()
139         {
140
141             larguraBola = (int) sliderTamanhoPonto.Value;
142
143             //Grid---------------------------------------
144             var pontoArea = new Grid
145             {
146                 Width = larguraBola,
147                 Height = larguraBola
148             };
```

```
149             pontoArea.Children.Add(new Ellipse() { Fill = Brushes.Black });
150
151             //Texto------------------------------
152             var idNumero = new TextBlock
153             {
154                 Text = "0",
155                 Foreground = Brushes.White,
156                 FontSize = larguraBola / 2,
157                 FontWeight = FontWeights.Bold,
158                 HorizontalAlignment = System.Windows.HorizontalAlignment.Center,
159                 VerticalAlignment = System.Windows.VerticalAlignment.Center
160             };
161
162             pontoArea.Children.Add(idNumero);
163
164             //Centralizando o idNumero dentro do pontoArea
165             Canvas.SetTop(pontoArea, 0);
166             Canvas.SetLeft(pontoArea, 0);
167
168             if (canvasAreaModelo != null)
169             {
170                 canvasAreaModelo.Children.Clear();
171                 canvasAreaModelo.Children.Add(pontoArea);
172             }
173
174             lblTamanhoPonto.Content = string.Format("Tamanho do Ponto: {0}
175 pixels.", larguraBola);
176
177         }
178
179         private void SelecionarAtividadeClick(object sender, MouseButtonEventArgs
180 e)
181         {
182             if (gridListagemAtividade.SelectedItem == null)
183                 return;
184
185             var atividade = (Atividade)gridListagemAtividade.SelectedItem;
186
187             if (atividade != null)
188             {
189                 eNovoAtividade = false;
190                 PreencheCampos(atividade.Id);
191             }
192             gridListagemAtividade.UnselectAll();
193
194         }
195
196         private void PreencheCampos(int id)
197         {
198             atividadeGeral = Principal.appAtividade.ConsultarAtividade(id);
199             txtAtividadeTerapeutaNome.Text = atividadeGeral.NomeTerapeuta;
200             cboListaPacientes.Text =
201 Principal.appPaciente.ConsultarPaciente(atividadeGeral.IdPaciente).Nome;
202             cboAtividadeLado.Text = atividadeGeral.LadoTreino;
203             cboAtividadeTempo.Text = string.Format("{0} segundos",
204 atividadeGeral.TempoCriacaoMovimento);
205             txtAtividadeObservacao.Text = atividadeGeral.Observacao;
206             txtDataCriacao.Text = atividadeGeral.DataCriacao;
207             sliderTamanhoPonto.Value = atividadeGeral.PontoTamanho;
208             AddCiculoAreaModelo();
209         }
210
```

```
211         private void AlterarTamanhoPonto(object sender,
212 RoutedPropertyChangedEventArgs<double> e)
213         {
214             AddCiculoAreaModelo();
215         }
216 
217         private void AtividadeExcluirClick(object sender, RoutedEventArgs e)
218         {
219             try
220             {
221 
222                 if (atividadeGeral != null)
223                 {
224                     Principal.appAtividade.Excluir(atividadeGeral);
225                     LimparCampos();
226                     ShowListaAtividades();
227                     MessageBox.Show(
228                         "Atividade excluida com sucesso!",
229                         "Informação",
230                         MessageBoxButton.OK,
231                         MessageBoxImage.Information);
232                     eNovoAtividade = true;
233                 }
234                 else
235                 {
236                     MessageBox.Show(
237                         "Não existe atividade selecionado para esta operação",
238                         "Informação",
239                         MessageBoxButton.OK,
240                         MessageBoxImage.Information);
241                 }
242             }
243             catch (Exception ex)
244             {
245                 MessageBox.Show(ex.Message);
246             }
247 
248         }
249 
250 
251         private void btnCriarAtividade_Click(object sender, RoutedEventArgs e)
252         {
253             if (gridListagemAtividade.SelectedItem == null)
254             {
255                 MessageBox.Show(
256                     "Não existe atividade selecionado para esta operação",
257                     "Informação",
258                     MessageBoxButton.OK,
259                     MessageBoxImage.Error);
260             }
261             else
262             {
263                 var atividade = (Atividade)gridListagemAtividade.SelectedItem;
264                 if (atividade.Status.Equals("Concluído"))
265                 {
266                     MessageBox.Show(
267                     "Atividade concluída",
268                     "Informação",
269                     MessageBoxButton.OK,
270                     MessageBoxImage.Error);
271                 }
272                 else
```

```
273                 {
274                     var desenvolverAtividade = new
275 DesenvolverAtividade(atividade);
276                     desenvolverAtividade.Show();
277                     criandoAtividade = true;
278                 }
279             }
280         }
281 
282         private void GridMouseEnterAtualizarAlteracoes(object sender,
283 MouseEventArgs e)
284         {
285             if (criandoAtividade)
286             {
287                 ShowListaAtividades();
288                 gridListagemAtividade.UnselectAll();
289                 criandoAtividade = false;
290             }
291         }
292 
293         private void cboListaPacientes_DropDownClosed(object sender, EventArgs e)
294         {
295             txtDataCriacao.Text = string.Format("{0}/{1}/{2}",
296                                                 DateTime.Now.Day,
297                                                 DateTime.Now.Month,
298                                                 DateTime.Now.Year);
299         }
300 
301     }
302 }
```

Listagem 7.21: Código fonte do arquivo ManterFazerAtividadePaciente.xaml

```
01  <UserControl x:Class="UIGraphic.Formularios.ManterFazerAtividadePaciente"
02               xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
03               xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
04               xmlns:mc="http://schemas.openxmlformats.org/markup-
05 compatibility/2006"
06               xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
07               mc:Ignorable="d"
08               d:DesignHeight="450" d:DesignWidth="750">
09     <Grid Background="{DynamicResource {x:Static SystemColors.InfoBrushKey}}">
10         <Label Content="Paciente:" HorizontalAlignment="Left" Margin="26,10,0,0"
11 VerticalAlignment="Top" FontFamily="Comic Sans MS" FontSize="18"
12 FontWeight="Bold"/>
13         <ComboBox x:Name="cboPacientes" HorizontalAlignment="Left"
14 Margin="131,10,0,0" VerticalAlignment="Top" Width="609" Height="32" FontSize="14"
15 FontFamily="Comic Sans MS" FontStyle="Italic"
16 DropDownClosed="CarregarAtividadesPacienteDropDownClosed"/>
17         <Label Content="Listagem de Atividades" HorizontalAlignment="Left"
18 Margin="216,43,0,0" VerticalAlignment="Top" Style="{DynamicResource
19 labelTelaCriarAtividade}"/>
20         <DataGrid x:Name="gridAtividades" HorizontalAlignment="Left"
21 Margin="26,91,0,0" VerticalAlignment="Top" Height="139" Width="714"/>
22         <Rectangle Fill="#FFF4F4F5" HorizontalAlignment="Left" Height="194"
23 Margin="36,246,0,0" Stroke="Black" VerticalAlignment="Top" Width="714"/>
24         <CheckBox x:Name="cbShowPaciente" Content="Mostrar paciente"
25 HorizontalAlignment="Left" Margin="48,290,0,0" VerticalAlignment="Top"
26 FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"/>
27         <CheckBox x:Name="cbShowEsqueleto" Content="Mostrar Esqueleto"
28 HorizontalAlignment="Left" Margin="48,320,0,0" VerticalAlignment="Top"
29 FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"/>
```

```
30          <CheckBox x:Name="cbShowLinha" Content="Mostrar linha"
31  HorizontalAlignment="Left" Margin="48,410,0,0" VerticalAlignment="Top"
32  FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"
33  Checked="ShowCanvasOpcoesLinha" Unchecked="HideCanvasOpcoesLinha"/>
34          <CheckBox x:Name="cbShowPonto" Content="Mostrar ponto"
35  HorizontalAlignment="Left" Margin="48,350,0,0" VerticalAlignment="Top"
36  FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"/>
37          <CheckBox x:Name="cbShowCronometro" Content="Mostrar cronometro"
38  HorizontalAlignment="Left" Margin="48,380,0,0" VerticalAlignment="Top"
39  FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"/>
40          <Button x:Name="btnDesenvolverAtividade" Content="Paciente Desenvolver
41  Atividades" HorizontalAlignment="Left" Margin="430,264,0,0"
42  VerticalAlignment="Top" Width="284" Height="110" FontFamily="Comic Sans MS"
43  FontSize="16" FontWeight="Bold" Click="BtnDesenvolverAtividadeClick"/>
44          <Label Content="Data:" HorizontalAlignment="Left" Margin="52,254,0,0"
45  VerticalAlignment="Top" FontFamily="Comic Sans MS" FontSize="14"
46  FontWeight="Bold"/>
47          <TextBox x:Name="txtData" HorizontalAlignment="Left" Height="23"
48  Margin="100,258,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="101"
49  FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"/>
50          <Canvas Name="canvasOpcoesLinha" Margin="225,266,370,19">
51              <Label Content="Espessura:" HorizontalAlignment="Left"
52  VerticalAlignment="Top" FontFamily="Comic Sans MS" FontSize="14"
53  FontWeight="Bold"/>
54              <Label Content="Cor:" HorizontalAlignment="Left"
55  VerticalAlignment="Top" FontFamily="Comic Sans MS" FontSize="14" FontWeight="Bold"
56  RenderTransformOrigin="0.487,1.5" Canvas.Top="22"/>
57              <ComboBox x:Name="cboEspessura" HorizontalAlignment="Left"
58  VerticalAlignment="Top" Width="53" Canvas.Left="90" Canvas.Top="3"
59  DropDownClosed="CboEspessuraAlterarEspessura"/>
60              <ComboBox x:Name="cboCor" HorizontalAlignment="Left"
61  VerticalAlignment="Top" Width="96" Canvas.Top="30" Canvas.Left="49"
62  DropDownClosed="CboCorAlterarCor"/>
63              <Canvas x:Name="canvasExemploLinha" Height="108"  Width="155"
64  Canvas.Top="57"/>
65
66          </Canvas>
67
68
69      </Grid>
70  </UserControl>
```

Listagem 7.22: Código fonte do arquivo ManterFazerAtividadePaciente.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Windows;
5   using System.Windows.Controls;
6   using System.Windows.Media;
7   using System.Windows.Shapes;
8   using Dominio;
9   using UIGraphic.Janela;
10
11  namespace UIGraphic.Formularios
12  {
13      /// <summary>
14      /// Interaction logic for ManterFazerAtividadePaciente.xaml
15      /// </summary>
16      public partial class ManterFazerAtividadePaciente : UserControl
17      {
```

```
18          public ManterFazerAtividadePaciente()
19          {
20              InitializeComponent();
21              CarregarListaPacientes();
22              CarregaOpcoesLinha();
23              canvasOpcoesLinha.Visibility = Visibility.Hidden;
24          }
25
26          private void CarregarListaPacientes()
27          {
28              var pacientes = Principal.appPaciente.Lista();
29              var dictionaryPaciente = pacientes.ToDictionary(paciente =>
30  paciente.Id, paciente => paciente.Nome);
31              cboPacientes.ItemsSource = dictionaryPaciente;
32              cboPacientes.DisplayMemberPath = "Value";
33              cboPacientes.SelectedValuePath = "Key";
34          }
35
36          private void CarregaOpcoesLinha()
37          {
38              var dictionaryEspessura = new Dictionary<int, int>();
39              for (var i = 1; i <= 100; i++)
40              {
41                  dictionaryEspessura.Add(i, i);
42              }
43              cboEspessura.ItemsSource = dictionaryEspessura;
44              cboEspessura.DisplayMemberPath = "Value";
45              cboEspessura.SelectedValuePath = "Key";
46              cboEspessura.SelectedIndex = 0;
47
48
49              var dictionaryCor = new Dictionary<int, string>
50                                      {
51                                          {0, "Azul"},
52                                          {1, "Preto"},
53                                          {2, "Verde"},
54                                          {3, "Vermelho"}
55                                      };
56
57              cboCor.ItemsSource = dictionaryCor;
58              cboCor.DisplayMemberPath = "Value";
59              cboCor.SelectedValuePath = "Key";
60              cboCor.SelectedIndex = 0;
61          }
62
63          private void CarregarAtividadesPacienteDropDownClosed(object sender,
64  EventArgs e)
65          {
66              if (cboPacientes.SelectedValue == null)
67                  return;
68
69              var idPaciente = (int) cboPacientes.SelectedValue;
70              var atividades = Principal.appAtividade.Lista().Where(a =>
71  a.IdPaciente == idPaciente);
72              gridAtividades.ItemsSource = atividades;
73
74              txtData.Text = string.Format("{0}/{1}/{2}",
75                                           DateTime.Now.Day,
76                                           DateTime.Now.Month,
77                                           DateTime.Now.Year);
78
79          }
```

```
80
81          private void BtnDesenvolverAtividadeClick(object sender, RoutedEventArgs
82  e)
83          {
84
85              //Parâmetros a serem passados para a janela FazerAtividades
86              var listaAtividades = gridAtividades.SelectedItems; //Lista com as
87  atividades selecionadas
88              var nomePaciente = cboPacientes.Text;
89              var opcoes = new Opcoes();
90
91              //Verificar se alguma atividade foi selecionada
92              if (listaAtividades.Count == 0)
93              {
94                  MessageBox.Show(
95                      "Pelo menos uma atividade deve ser selecionada",
96                      "Informação",
97                      MessageBoxButton.OK,
98                      MessageBoxImage.Error);
99                  return;
100             }
101
102
103
104             if (cbShowLinha.IsChecked == true)
105             {
106                 opcoes.ShowLinha = true;
107                 opcoes.EspessuraLinha = (int) cboEspessura.SelectedValue;
108                 opcoes.CorLinha = (int) cboCor.SelectedValue;
109             }
110
111             if (cbShowPonto.IsChecked == true)
112             {
113                 opcoes.ShowPonto = true;
114             }
115
116             if (cbShowPaciente.IsChecked == true)
117             {
118                 opcoes.ShowPaciente = true;
119             }
120             if (cbShowCronometro.IsChecked == true)
121             {
122                 opcoes.ShowCronometro = true;
123             }
124
125             if (cbShowEsqueleto.IsChecked == true)
126             {
127                 opcoes.ShowEsqueleto = true;
128             }
129
130             var data = txtData.Text;
131
132             var pacienteJogaAtividade = new
133 PacienteJogaAtividade(listaAtividades, opcoes, nomePaciente, data);
134             pacienteJogaAtividade.Show();
135
136         }
137
138         private void ShowCanvasOpcoesLinha(object sender, RoutedEventArgs e)
139         {
140             canvasOpcoesLinha.Visibility = Visibility.Visible;
141         }
```

```
142
143         private void HideCanvasOpcoesLinha(object sender, RoutedEventArgs e)
144         {
145             canvasOpcoesLinha.Visibility = Visibility.Hidden;
146         }
147
148         public void AddLinha()
149         {
150             var cor = new SolidColorBrush();
151             var opcoes = (int) cboCor.SelectedValue;
152             if (opcoes == 0)
153             {
154                 cor.Color = Colors.Blue;
155             }
156             else if (opcoes == 1)
157             {
158                 cor.Color = Colors.Black;
159             }
160             else if (opcoes == 2)
161             {
162                 cor.Color = Colors.Green;
163             }
164             else if (opcoes == 3)
165             {
166                 cor.Color = Colors.Red;
167             }
168
169
170             var linha = new Line
171                             {
172                                 X1 = 1,
173                                 Y1 = 1,
174                                 X2 = 100,
175                                 Y2 = 1,
176                                 StrokeThickness = (int)
177 cboEspessura.SelectedValue,
178                                 Stroke = cor,
179                                 HorizontalAlignment = HorizontalAlignment.Center,
180                                 VerticalAlignment = VerticalAlignment.Center
181                             };
182
183            Canvas.SetTop(linha, 55);
184            Canvas.SetLeft(linha, 25);
185             canvasExemploLinha.Children.Clear();
186            canvasExemploLinha.Children.Add(linha);
187         }
188
189         private void CboEspessuraAlterarEspessura(object sender, EventArgs e)
190         {
191             AddLinha();
192         }
193
194         private void CboCorAlterarCor(object sender, EventArgs e)
195         {
196             AddLinha();
197         }
198     }
199 }
```

Listagem 7.23: Código fonte do arquivo ManterRelatorio.xaml

```
<UserControl x:Class="UIGraphic.Formularios.ManterRelatorio"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
            d:DesignHeight="450" d:DesignWidth="750" Loaded="UserControl_Loaded">
    <Grid Background="{DynamicResource {x:Static SystemColors.InfoBrushKey}}">
        <Label Content="Paciente:" HorizontalAlignment="Left" Margin="10,16,0,0"
VerticalAlignment="Top" FontFamily="Comic Sans MS" FontSize="18"
FontWeight="Bold"/>
        <ComboBox Name="cboPaciente" HorizontalAlignment="Left"
Margin="119,21,0,0" VerticalAlignment="Top" Width="621" FontSize="18"
FontWeight="Bold" DropDownClosed="CboPacienteDropDownClosed"/>
        <Label Content="Histórico de Atividades" HorizontalAlignment="Left"
Margin="256,48,0,0" VerticalAlignment="Top" FontFamily="Comic Sans MS"
FontSize="24" FontWeight="Bold"/>
        <DataGrid Name="gridHistorico" Margin="10,96,10,0"
VerticalAlignment="Top" Height="262" Width="730"
MouseDoubleClick="SelecionaAtividade"/>
        <Button Content="Analisar Métricas" HorizontalAlignment="Left"
Margin="545,373,0,0" VerticalAlignment="Top" Width="195" Height="67"
FontSize="16" FontWeight="Bold" Click="ShowMetricas"/>

    </Grid>
</UserControl>
```

Listagem 7.24: Código fonte do arquivo ManterRelatorio.cs

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using Dominio;
using UIGraphic.Janela;
using Graficos;

namespace UIGraphic.Formularios
{
    /// <summary>
    /// Interaction logic for ManterRelatorio.xaml
    /// </summary>
    public partial class ManterRelatorio : UserControl
    {
        public ManterRelatorio()
        {
            InitializeComponent();
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
            CarregarListaPacientes();
        }

        private void CboPacienteDropDownClosed(object sender, EventArgs e)
        {
            CarregarHistoricoAtividadesPaciente();
        }

```

```
32          private void CarregarListaPacientes()
33          {
34              var pacientes = Principal.appPaciente.Lista();
35              var dictionaryPaciente = pacientes.ToDictionary(paciente =>
36  paciente.Id, paciente => paciente.Nome);
37              cboPaciente.ItemsSource = dictionaryPaciente;
38              cboPaciente.DisplayMemberPath = "Value";
39              cboPaciente.SelectedValuePath = "Key";
40
41          }
42
43          private void CarregarHistoricoAtividadesPaciente()
44          {
45              if (cboPaciente.SelectedValue == null)
46                  return;
47
48              var idPaciente = (int)cboPaciente.SelectedValue;
49              var historico =
50  Principal.appHistoricoAtividadePaciente.Lista().Where(h => h.IdPaciente ==
51  idPaciente);
52              gridHistorico.ItemsSource = historico;
53              gridHistorico.IsReadOnly = true;
54          }
55
56          private void SelecionaAtividade(object sender, MouseButtonEventArgs e)
57          {
58              if (gridHistorico.SelectedItem == null)
59                  return;
60
61              var historico =
62  (HistoricoAtividadePaciente)gridHistorico.SelectedItem;
63
64              if (historico != null)
65              {
66                  var resultado = new Resultados(historico);
67                  resultado.Show();
68              }
69              gridHistorico.UnselectAll();
70          }
71
72        private void AnalisarTrajetoria(object sender, RoutedEventArgs e)
73          {
74              if (cboPaciente.SelectedItem == null)
75              {
76                  MessageBox.Show(
77                          "Escolha um paciente",
78                          "Informação",
79                          MessageBoxButton.OK,
80                          MessageBoxImage.Information);
81                  return;
82              }
83              var idPaciente = (int)cboPaciente.SelectedValue;
84              var resultado = new Resultados(idPaciente);
85              resultado.Show();
86          }
87
88        private void ShowMetricas(object sender, RoutedEventArgs e)
89        {
90            var metricas = new Metricas();
91            metricas.Show();
92        }
93
```

```
94      }
95  }
```

Listagem 7.25: Código fonte do arquivo ManterPaciente.xaml

```
1   <UserControl x:Class="UIGraphic.Formularios.ManterPaciente"
2                xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3                xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4                xmlns:mc="http://schemas.openxmlformats.org/markup-
5   compatibility/2006"
6                xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7                mc:Ignorable="d"
8                d:DesignHeight="450" d:DesignWidth="750"
9   Loaded="ManterPacienteLoad">
10      <Grid Background="{DynamicResource {x:Static SystemColors.InfoBrushKey}}">
11          <Grid.ColumnDefinitions>
12              <ColumnDefinition />
13              <ColumnDefinition />
14          </Grid.ColumnDefinitions>
15          <Label Grid.Column="0" Content="Listagem de Pacientes"
16  HorizontalAlignment="Left" Margin="68,10,0,0" VerticalAlignment="Top"
17  FontSize="22" FontWeight="Bold" FontFamily="Comic Sans MS" Width="248"
18  HorizontalContentAlignment="Center"/>
19          <DataGrid x:Name="listagemPacientes"  Grid.Column="0"
20  Margin="10,66,21,39" MouseDoubleClick="SelecionarPacienteClick"/>
21          <Label Grid.Column="1" Content="Nome:" HorizontalAlignment="Left"
22  Margin="10,60,0,0" VerticalAlignment="Top" Style="{DynamicResource
23  labelManutencao}"/>
24          <Label Grid.Column="1" Content="Sexo:" HorizontalAlignment="Left"
25  Margin="10,120,0,0" VerticalAlignment="Top" Style="{DynamicResource
26  labelManutencao}"/>
27          <Label Grid.Column="1" Content="Data de Nascimento:"
28  HorizontalAlignment="Left" Margin="215,120,0,0" VerticalAlignment="Top"
29  Style="{DynamicResource labelManutencao}"/>
30          <Label Grid.Column="1" Content="Tempo do acidente:"
31  HorizontalAlignment="Left" Margin="10,179,0,0" VerticalAlignment="Top"
32  Style="{DynamicResource labelManutencao}"/>
33          <Label Grid.Column="1" Content="Observações:" HorizontalAlignment="Left"
34  Margin="10,239,0,0" VerticalAlignment="Top" Style="{DynamicResource
35  labelManutencao}"/>
36          <TextBox x:Name="pacienteNome" Grid.Column="1" HorizontalAlignment="Left"
37  Height="23" Margin="10,92,0,0" TextWrapping="Wrap" VerticalAlignment="Top"
38  Width="355"/>
39          <TextBox x:Name="pacienteTempoDoAcidente"  Grid.Column="1"
40  HorizontalAlignment="Left" Height="23" Margin="10,206,0,0" TextWrapping="Wrap"
41  VerticalAlignment="Top" Width="120"/>
42          <TextBox x:Name="pacienteObservacoes" Grid.Column="1"
43  HorizontalAlignment="Left" Height="91" Margin="10,271,0,0" TextWrapping="Wrap"
44  VerticalAlignment="Top" Width="355"/>
45          <Label Grid.Column="1" Content="Manutenção de Pacientes"
46  HorizontalAlignment="Left" Margin="57,10,0,0" VerticalAlignment="Top"
47  FontSize="22" FontWeight="Bold" FontFamily="Comic Sans MS" Width="278"
48  HorizontalContentAlignment="Center"/>
49          <ComboBox x:Name="pacienteSexo" Grid.Column="1"
50  HorizontalAlignment="Left" Margin="10,147,0,0" VerticalAlignment="Top"
51  Width="120">
52              <ComboBoxItem Content="Masculino"/>
53              <ComboBoxItem Content="Feminino"/>
54          </ComboBox>
55          <Button x:Name="pacienteGravar" Content="Gravar" Grid.Column="1"
56  HorizontalAlignment="Left" Margin="210,367,0,0" VerticalAlignment="Top"
57  Style="{DynamicResource buttonPacienteStyle}" Click="PacienteGravarClick" />
```

```
58          <Button x:Name="pacienteExcluir" Content="Excluir" Grid.Column="1"
59  HorizontalAlignment="Left" Margin="290,367,0,0" VerticalAlignment="Top"
60  Style="{DynamicResource buttonPacienteStyle}" Click="PacienteExcluirClick"/>
61          <TextBox x:Name="pacienteDataNascimento" Grid.Column="1"
62  HorizontalAlignment="Left" Height="23" Margin="245,147,0,0" TextWrapping="Wrap"
63  VerticalAlignment="Top" Width="120"/>
64
65      </Grid>
66  </UserControl>
```

Listagem 7.26: Código fonte do arquivo ManterPaciente.cs

```
1   using System;
2   using System.Windows;
3   using System.Windows.Controls;
4   using System.Windows.Input;
5   using Dominio;
6
7   namespace UIGraphic.Formularios
8   {
9       /// <summary>
10      /// Interaction logic for ManterPaciente.xaml
11      /// </summary>
12      public partial class ManterPaciente : UserControl
13      {
14          //private AppPaciente appPaciente;
15          private bool eNovoPaciente;
16          private Paciente pacienteGeral;
17          public ManterPaciente()
18          {
19              InitializeComponent();
20              eNovoPaciente = true;
21          }
22
23          private void PacienteGravarClick(object sender, RoutedEventArgs e)
24          {
25              try
26              {
27                  var idPG = 0;
28                  if (pacienteGeral != null)
29                  {
30                      idPG = pacienteGeral.Id;
31                  }
32
33                  pacienteGeral = new Paciente
34                                      {
35                                          Id = idPG,
36                                          Nome = pacienteNome.Text,
37                                          Sexo = pacienteSexo.Text,
38                                          DataNascimento =
39  pacienteDataNascimento.Text,
40                                          TempoAcidente =
41  pacienteTempoDoAcidente.Text,
42                                          Observacoes = pacienteObservacoes.Text
43                                      };
44
45                  if (eNovoPaciente)
46                  {
47                      Principal.appPaciente.Salvar(pacienteGeral);
48                      MessageBox.Show(
49                          "Paciente gravado com sucesso",
50                          "Informação",
```

```
51                          MessageBoxButton.OK,
52                          MessageBoxImage.Information);
53                  }
54                  else
55                  {
56                      Principal.appPaciente.Alterar(pacienteGeral);
57                      MessageBox.Show(
58                          "Paciente alterado com sucesso",
59                          "Informação",
60                          MessageBoxButton.OK,
61                          MessageBoxImage.Information);
62                  }
63
64                  ShowListaPacientes();
65                  eNovoPaciente = true;
66                  LimparCampos();
67              } catch(Exception ex)
68              {
69                  MessageBox.Show(ex.Message);
70              }
71          }
72
73
74          private void ShowListaPacientes()
75          {
76              listagemPacientes.ItemsSource = Principal.appPaciente.Lista();
77              listagemPacientes.IsReadOnly = true;
78              listagemPacientes.Columns[5].Visibility = Visibility.Hidden;
79          }
80
81          private void LimparCampos()
82          {
83              pacienteNome.Text = string.Empty;
84              pacienteSexo.SelectedIndex = -1;
85              pacienteDataNascimento.Text = string.Empty;
86              pacienteTempoDoAcidente.Text = string.Empty;
87              pacienteObservacoes.Text = string.Empty;
88          }
89
90          private void ManterPacienteLoad(object sender, RoutedEventArgs e)
91          {
92              ShowListaPacientes();
93          }
94
95          private void SelecionarPacienteClick(object sender, MouseButtonEventArgs
96  e)
97          {
98              if (listagemPacientes.SelectedItem == null)
99                  return;
100
101             var paciente = (Paciente) listagemPacientes.SelectedItem;
102
103             if (paciente != null)
104             {
105                 eNovoPaciente = false;
106                 PreencheCampos(paciente.Id);
107             }
108
109         }
110
111         private void PreencheCampos(int id)
112         {
```

```
113             pacienteGeral = Principal.appPaciente.ConsultarPaciente(id);
114             pacienteNome.Text = pacienteGeral.Nome;
115             pacienteSexo.Text = pacienteGeral.Sexo;
116             pacienteDataNascimento.Text = pacienteGeral.DataNascimento;
117             pacienteTempoDoAcidente.Text =
118 Convert.ToString(pacienteGeral.TempoAcidente);
119             pacienteObservacoes.Text = pacienteGeral.Observacoes;
120         }
121 
122         private void PacienteExcluirClick(object sender, RoutedEventArgs e)
123         {
124             try
125             {
126 
127                 if (pacienteGeral != null)
128                 {
129                     Principal.appPaciente.Excluir(pacienteGeral);
130                     LimparCampos();
131                     ShowListaPacientes();
132                     MessageBox.Show(
133                         "Paciente excluido com sucesso!",
134                         "Informação",
135                         MessageBoxButton.OK,
136                         MessageBoxImage.Information);
137                     eNovoPaciente = true;
138                 }
139                 else
140                 {
141                     MessageBox.Show(
142                         "Não existe paciente selecionado para esta operação",
143                         "Informação",
144                         MessageBoxButton.OK,
145                         MessageBoxImage.Information);
146                 }
147             } catch (Exception ex)
148             {
149                 MessageBox.Show(ex.Message);
150             }
151 
152         }
153     }
154 }
```

Listagem 7.27: Código fonte do arquivo DesenvolverAtividade.xaml

```
1   <Window x:Class="UIGraphic.Janela.DesenvolverAtividade"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           Title="AVARFMS - Terapeuta Desenvolve Atividade para Paciente"
5           Height="768" Width="1024" WindowStartupLocation="CenterScreen"
6   ResizeMode="NoResize" Loaded="CarregarJanela" Closed="FecharJanela">
7       <Grid x:Name="TelaJogo" Margin="0,0,0,0">
8               <Image Name="KinectVideo" Margin="0,0,0,0" Width="1024"
9   Height="768"/>
10              <Polyline x:Name="LinhaDeLigacao" Stroke="Black"
11  StrokeThickness="3"/>
12              <Canvas x:Name="paintCanvas"/>
13              <Canvas x:Name="FolhaDesenho"/>
14          <Canvas x:Name="Geral">
15              <Label Content="0.0" Name="lblCronometro" Canvas.Left="174"
16  Canvas.Top="10" Style="{DynamicResource labelTelaCriarAtividade}"
```

```
RenderTransformOrigin="0.868,0.5" Background="White"/>
            <Label Content="Cronometro:" Canvas.Left="10" Canvas.Top="10"
Style="{DynamicResource labelTelaCriarAtividade}" Background="White"/>
            <Label x:Name="lblMensagemTerapeuta" Content="Terapeuta não
Encontrado" Canvas.Left="220" Canvas.Top="337" FontSize="48" FontWeight="Bold"
Foreground="#FFDA0505"/>
            <Button x:Name="btnIniciar"  Content="Iniciar" Canvas.Left="853"
Canvas.Top="10" Width="75" Style="{DynamicResource buttonAtividadeStyle}"
Click="IniciarCriacaoAtividade"/>
            <Button x:Name="btnFinalizar" Content="Finalizar" Canvas.Left="933"
Canvas.Top="10" Width="75" Style="{DynamicResource buttonAtividadeStyle}"
Click="FinalizarCriacaoAtividade"/>
            <Button x:Name="btnLimparTela" Content="Apagar" Canvas.Left="853"
Canvas.Top="45" Width="75" Style="{DynamicResource buttonAtividadeStyle}"
Click="BtnLimparTelaClick"/>
            <Button x:Name="btnGravarAtividade" Content="Gravar"
Canvas.Left="933" Canvas.Top="45" Width="75" Style="{DynamicResource
buttonAtividadeStyle}" Click="BtnGravarAtividadeClick"/>
        </Canvas>
    </Grid>
</Window>
```

Listagem 7.28: Código fonte do arquivo DesenvolverAtividade.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using Dominio;
using Microsoft.Kinect;

namespace UIGraphic.Janela
{
    /// <summary>
    /// Interaction logic for DesenvolverAtividade.xaml
    /// </summary>
    public partial class DesenvolverAtividade : Window
    {

        private Point posicao = new Point(0, 0);
        private readonly Atividade _atividadeGlobal;
        private KinectSensor _myKinect;
        private bool _ocorreuErroKinect;

        private List<Ponto> listaPontos = new List<Ponto>();


        private int _contPontos = 0;
        private int _totalSegundos = 0;
        private Stopwatch cronometroFase = new Stopwatch();

        private bool iniciarCriacaoFase = false;


        public DesenvolverAtividade(Atividade atividade)
        {
            InitializeComponent();
            IniciarKinect();
```

```
39              _atividadeGlobal = atividade;
40
41          }
42
43          private void paintBola(Point currentPosition)
44          {
45              var bola = new Ellipse
46                              {
47                                  Width = _atividadeGlobal.PontoTamanho,
48                                  Height = _atividadeGlobal.PontoTamanho,
49                                  Fill = Brushes.GreenYellow,
50                                  Stroke = Brushes.Black
51                              };
52
53              Canvas.SetTop(bola, currentPosition.Y);
54              Canvas.SetLeft(bola, currentPosition.X);
55
56              int count = paintCanvas.Children.Count;
57              paintCanvas.Children.Add(bola);
58
59              if (count > 1)
60              {
61                  paintCanvas.Children.RemoveAt(0);
62              }
63
64          }
65
66          private void CarregarJanela(object sender, RoutedEventArgs e)
67          {
68              if (_ocorreuErroKinect)
69              {
70                  Close();
71              }
72              if (_skeletonTerapeuta != null)
73              {
74
75              }
76              var paciente =
77  Principal.appPaciente.ConsultarPaciente(_atividadeGlobal.IdPaciente);
78          }
79
80          private void FecharJanela(object sender, EventArgs e)
81          {
82              FinalizarKinect();
83          }
84
85
86
87
88          #region Kinect
89
90          private Skeleton _skeletonTerapeuta = null;
91
92          private void IniciarKinect()
93          {
94              try
95              {
96                  _myKinect = KinectSensor.KinectSensors[0];
97                  //Para pessoa sentada
98                  _myKinect.SkeletonStream.TrackingMode =
99  SkeletonTrackingMode.Seated;
100                 _myKinect.ColorStream.Enable();
```

```
101                 _myKinect.SkeletonStream.Enable();
102 
103                 _myKinect.ColorFrameReady += MyKinectColorFrameReady;
104                 _myKinect.SkeletonFrameReady += MyKinectSkeletonFrameReady;
105                 _myKinect.Start();
106                 _ocorreuErroKinect = false;
107             }
108             catch
109             {
110                 MessageBox.Show("Erro ao inicializar o Kinect");
111                 _ocorreuErroKinect = true;
112             }
113         }
114 
115         void MyKinectColorFrameReady(object sender, ColorImageFrameReadyEventArgs
116 e)
117         {
118             using (var colorFrame = e.OpenColorImageFrame())
119             {
120                 if (colorFrame != null)
121                 {
122                     var colorData = new byte[colorFrame.PixelDataLength];
123                     colorFrame.CopyPixelDataTo(colorData);
124                     var bitmap = BitmapSource.Create(
125                         colorFrame.Width, colorFrame.Height, //Dimensões da
126 imagem
127                         96, 96, //Resolução
128                         PixelFormats.Bgr32, //Formato do vídeo
129                         null, //Paleta de cores
130                         colorData, //dados do video
131                         colorFrame.Width * colorFrame.BytesPerPixel);
132                     KinectVideo.Source = bitmap;
133                 }
134             }
135         }
136 
137         private Ponto ponto;
138         void MyKinectSkeletonFrameReady(object sender,
139 SkeletonFrameReadyEventArgs e)
140         {
141             Skeleton[] skeletons = null;
142 
143             lblMensagemTerapeuta.Visibility = Visibility.Visible;
144             paintCanvas.Children.Clear();
145             using (var frame = e.OpenSkeletonFrame())
146             {
147                 if (frame != null)
148                 {
149                     skeletons = new Skeleton[frame.SkeletonArrayLength];
150                     frame.CopySkeletonDataTo(skeletons);
151                 }
152 
153             }
154 
155             if (skeletons == null)
156             {
157                 return;
158             }
159 
160             foreach (var skeleton in skeletons)
161             {
162                 if (skeleton.TrackingState == SkeletonTrackingState.Tracked)
```

```
163                 {
164                     _skeletonTerapeuta = skeleton;
165                     lblMensagemTerapeuta.Visibility = Visibility.Hidden;
166                     var maoJoint = _atividadeGlobal.LadoTreino.Equals("Esquerdo")
167 ? skeleton.Joints[JointType.HandLeft] : skeleton.Joints[JointType.HandRight];
168                     var maoPosition = maoJoint.Position;
169 
170                     var posicaoMaoTela = _myKinect.MapSkeletonPointToDepth(
171                         maoJoint.Position,
172 DepthImageFormat.Resolution640x480Fps30);
173                     //Calculando a posição X na tela
174                     var x = (int)
175 ((posicaoMaoTela.X*TelaJogo.ActualWidth/_myKinect.DepthStream.FrameWidth));
176                     x += _atividadeGlobal.PontoTamanho / 2;
177                     //Calculando a posição Y na tela
178                     var y = (int)
179 ((posicaoMaoTela.Y*TelaJogo.ActualHeight/_myKinect.DepthStream.FrameHeight));
180                     y += _atividadeGlobal.PontoTamanho/2;
181 
182 
183                     posicao = new Point(x, y);
184                     paintBola(posicao);
185 
186                     ponto = new Ponto
187                     {
188                         Altura = _atividadeGlobal.PontoTamanho,
189                         Largura = _atividadeGlobal.PontoTamanho,
190                         Numero = 0,
191                         PosicaoX = (int)posicao.X,
192                         PosicaoY = (int)posicao.Y,
193                         IdAtividade = _atividadeGlobal.Id,
194                         tipoPonto = "suspensao"
195                     };
196                     string message = string.Format("Mao Esquerda: X:{0:0.0},
197 Y:{1:0.0}",
198                                                    x,
199                                                    y);
200                     Console.WriteLine(message);
201                     MontarTela(ponto);
202                 }
203             }
204            // Console.WriteLine(message);
205         }
206 
207 
208         private void MontarTela(Ponto ponto)
209         {
210             if (iniciarCriacaoFase)
211             {
212                 _totalSegundos = (int)cronometroFase.Elapsed.TotalSeconds;
213 
214                 lblCronometro.Content = string.Format("{0:0}:{1:000}",
215 cronometroFase.Elapsed.TotalSeconds,
216 
217 cronometroFase.Elapsed.Milliseconds);
218 
219                 if (_totalSegundos >= _atividadeGlobal.TempoCriacaoMovimento)
220                 {
221                     ponto.Numero = _contPontos++;
222                     listaPontos.Add(ponto);
223                     cronometroFase.Restart();
224                     DesenharLinhasPontos();
```

```
225                         _totalSegundos = 0;
226                         cronometroFase.Start();
227                     }
228                 }
229             }
230 
231             private void FinalizarKinect()
232             {
233                 try
234                 {
235                     if (_myKinect == null) return;
236                     _myKinect.Stop();
237                     _myKinect.Dispose();
238                 }
239                 catch
240                 {
241                     MessageBox.Show("Erro ao finalizar o Kinect");
242                 }
243             }
244             #endregion
245 
246 
247 
248             public void DesenharLinhasPontos()
249             {
250                 for (var i = 0; i < listaPontos.Count - 1; i++)
251                 {
252                     AddLinha(listaPontos[i], listaPontos[i + 1]);
253                 }
254 
255                 foreach (var ponto in listaPontos)
256                 {
257                     AddPontoFolhaDesenho(ponto);
258                 }
259             }
260 
261             public void AddLinha(Ponto origem, Ponto destino)
262             {
263                 var cor = new SolidColorBrush
264                 {
265                     Color = Colors.Black
266                 };
267 
268                 var linha = new Line
269                 {
270                     X1 = origem.PosicaoX + _atividadeGlobal.PontoTamanho / 2,
271                     Y1 = origem.PosicaoY + _atividadeGlobal.PontoTamanho / 2,
272                     X2 = destino.PosicaoX + _atividadeGlobal.PontoTamanho / 2,
273                     Y2 = destino.PosicaoY + _atividadeGlobal.PontoTamanho / 2,
274                     StrokeThickness = 3,
275                     Stroke = cor
276                 };
277 
278 
279                 FolhaDesenho.Children.Add(linha);
280             }
281 
282             private void AddPontoFolhaDesenho(Ponto ponto)
283             {
284 
285                 //Grid---------------------------------------
286                 var pontoArea = new Grid
```

```
287             {
288                 Width = _atividadeGlobal.PontoTamanho,
289                 Height = _atividadeGlobal.PontoTamanho
290             };
291             pontoArea.Children.Add(new Ellipse { Fill = Brushes.White });
292 
293             //Texto-------------------------------
294             var idNumero = new TextBlock
295             {
296                 Text = "" + ponto.Numero,
297                 Foreground = Brushes.Black,
298                 FontSize = _atividadeGlobal.PontoTamanho / 2.0,
299                 FontWeight = FontWeights.Bold,
300                 HorizontalAlignment = HorizontalAlignment.Center,
301                 VerticalAlignment = VerticalAlignment.Center
302             };
303 
304             pontoArea.Children.Add(idNumero);
305 
306             //Centralizando o idNumero dentro do pontoArea
307             Canvas.SetTop(pontoArea, ponto.PosicaoY);// - (pontoArea.Height /
308 2));
309             Canvas.SetLeft(pontoArea, ponto.PosicaoX);// - (pontoArea.Width /
310 2));
311 
312             FolhaDesenho.Children.Add(pontoArea);
313 
314         }
315 
316         private void FinalizarCriacaoAtividade(object sender, RoutedEventArgs e)
317         {
318             iniciarCriacaoFase = false;
319         }
320 
321         private void IniciarCriacaoAtividade(object sender, RoutedEventArgs e)
322         {
323             if (_skeletonTerapeuta == null)
324             {
325                 MessageBox.Show(
326                     "Criação de atividade não pode ser iniciada",
327                     "Informação",
328                     MessageBoxButton.OK,
329                     MessageBoxImage.Error);
330             }
331             else
332             {
333                 iniciarCriacaoFase = true;
334                 DeterminandoDistanciaDoSensor();
335                 cronometroFase.Start();
336             }
337         }
338 
339         private void DeterminandoDistanciaDoSensor()
340         {
341             var ombroCentral =
342 _skeletonTerapeuta.Joints[JointType.ShoulderCenter];
343             var point = _myKinect.MapSkeletonPointToDepth(
344                     ombroCentral.Position,
345                     DepthImageFormat.Resolution640x480Fps30
346                     );
347 
348             point.X = (int)((point.X * TelaJogo.ActualWidth /
```

```
_myKinect.DepthStream.FrameWidth));
            point.Y = (int)((point.Y * TelaJogo.ActualHeight /
_myKinect.DepthStream.FrameHeight));


            _atividadeGlobal.DistanciaSensorZ = ombroCentral.Position.Z;
            _atividadeGlobal.DistanciaSensorX = point.X;
            _atividadeGlobal.DistanciaSensorY = point.Y;

        }

        private void BtnLimparTelaClick(object sender, RoutedEventArgs e)
        {
            FolhaDesenho.Children.Clear();
            cronometroFase.Reset();
            listaPontos = new List<Ponto>();
            _contPontos = 0;
        }

        private void BtnGravarAtividadeClick(object sender, RoutedEventArgs e)
        {
            try
            {
                var appPonto = Principal.appPonto;
                foreach (var ponto in listaPontos)
                {
                    appPonto.Salvar(ponto);
                }

                var atividade =
Principal.appAtividade.ConsultarAtividade(_atividadeGlobal.Id);
                atividade.Status = "Concluído";
                atividade.DistanciaSensorZ = _atividadeGlobal.DistanciaSensorZ;
                atividade.DistanciaSensorX = _atividadeGlobal.DistanciaSensorX;
                atividade.DistanciaSensorY = _atividadeGlobal.DistanciaSensorY;
                Principal.appAtividade.Alterar(atividade);

                MessageBox.Show(
                    "Atividade gravada com sucesso",
                    "Informação",
                    MessageBoxButton.OK,
                    MessageBoxImage.Information);
            } catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            Close();
        }

    }

}
```

Listagem 7.29: Código fonte do arquivo PacienteJogaAtividade.xaml

```
<Window x:Class="UIGraphic.Janela.PacienteJogaAtividade"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="AVARFMS - Paciente Desenvolve Atividade"
        Height="768" Width="1024" WindowStartupLocation="CenterScreen"
ResizeMode="NoResize" Closed="WindowClosed" Loaded="CarregarJanela">
    <Grid Name="TelaJogo">
```

```
8           <Image Name="KinectVideo" HorizontalAlignment="Stretch" Width="1024"
9   Height="768"/>
10          <Canvas x:Name="FolhaDesenho"/>
11          <Canvas Name="paintCanvas" HorizontalAlignment="Stretch" MaxWidth="1024"
12  MaxHeight="768"/>
13          <Canvas Name="canvasMensagens" Height="768" Width="1024"/>
14          <Canvas x:Name="canvasAjuste" Height="768" Width="1024"/>
15          <Canvas x:Name="canvasGeral" Height="768" Width="1024">
16              <Rectangle Name="fundoCronometro" Fill="White" Height="46"
17  Canvas.Left="10" Stroke="Black" Canvas.Top="13" Width="262"/>
18              <Label Content="0.0" Name="lblCronometro" Canvas.Left="174"
19  Canvas.Top="10" Style="{DynamicResource labelTelaCriarAtividade}"/>
20              <Label Content="Cronometro:" Name="lblCronometroInfo"
21  Canvas.Left="10" Canvas.Top="10" Style="{DynamicResource
22  labelTelaCriarAtividade}"/>
23              <ComboBox x:Name="cboAtividades" Canvas.Left="923" Canvas.Top="13"
24  Width="75" FontFamily="Comic Sans MS" FontSize="18" FontWeight="Bold"
25  DropDownClosed="CboAtividadesDropDownClosed" Height="43"/>
26              <Label Content="Atividade:" Canvas.Left="779" Style="{DynamicResource
27  labelTelaCriarAtividade}" Canvas.Top="12" Background="White"/>
28              <Button Content="Iniciar" Canvas.Left="923" Canvas.Top="67"
29  Width="75" Style="{DynamicResource buttonAtividadeStyle}"
30  Click="ButtonClickIniciar" Height="40"/>
31              <Button Content="Gravar" Canvas.Left="923" Canvas.Top="118"
32  Width="75" Style="{DynamicResource buttonAtividadeStyle}" Click="GravarAtividade"
33  Height="40"/>
34              <Label x:Name="lblPosicaoX" Content="0.00 dentro do limite X"
35  Canvas.Left="10" Canvas.Top="700" FontWeight="Bold" Background="White"/>
36              <Label x:Name="lblPosicaoY" Content="0.00 dentro do limite Y"
37  Canvas.Left="284" Canvas.Top="700" FontWeight="Bold" Background="White"/>
38              <Label x:Name="lblPosicaoZ" Content="0.00 dentro do limite Z"
39  Canvas.Left="556" Canvas.Top="701" FontWeight="Bold" Background="White"/>
40          </Canvas>
41          <Canvas Name="paintLinha" HorizontalAlignment="Stretch" MaxWidth="1024"
42  MaxHeight="768"/>
43          <Canvas Name="paintEsqueleto" HorizontalAlignment="Stretch"
44  MaxWidth="1024" MaxHeight="768"/>
45      </Grid>
46  </Window>
```

Listagem 7.30: Código fonte do arquivo PacienteJogaAtividade.cs

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using System.Diagnostics;
5   using System.Linq;
6   using System.Windows;
7   using System.Windows.Controls;
8   using System.Windows.Media;
9   using System.Windows.Media.Imaging;
10  using System.Windows.Shapes;
11  using Dominio;
12  using Microsoft.Kinect;
13
14  namespace UIGraphic.Janela
15  {
16      /// <summary>
17      /// Interaction logic for PacienteJogaAtividade.xaml
18      /// </summary>
19      public partial class PacienteJogaAtividade : Window
```

```
20      {
21
22          private readonly IList _listaAtividades;
23          private bool _ocorreuErroKinect;
24          private readonly Opcoes _opcoes;
25          private int _contPontos = 0;
26          private bool _iniciarFase;
27          private readonly string _dataAtividade = string.Empty;
28          private double _tempoDesenvolvimentoAtividade;
29          private readonly Stopwatch _cronometroFase = new Stopwatch();
30          private bool _iniciarCapturaPontosTrajetoria;
31          private List<PontosTrajetoria> _trajetoria;
32          private Atividade _atividadeGlobal = null;
33
34          public PacienteJogaAtividade(IList listaAtividades, Opcoes opcoes,
35  string nomePaciente, string data)
36          {
37              this._listaAtividades = listaAtividades;
38              _dataAtividade = data;
39              this._opcoes = opcoes;
40              InitializeComponent();
41              IniciarKinect();
42          }
43
44         private void CarregarJanela(object sender, RoutedEventArgs e)
45          {
46              if (_ocorreuErroKinect)
47              {
48                  Close();
49              }
50              CarregarComboBoxAtividades();
51
52              lblCronometro.Visibility = Visibility.Hidden;
53              lblCronometroInfo.Visibility = Visibility.Hidden;
54              fundoCronometro.Visibility = Visibility.Hidden;
55          }
56
57          private List<Ponto> _listaPontos = null;
58
59          private void CboAtividadesDropDownClosed(object sender, EventArgs e)
60          {
61              if (cboAtividades.SelectedValue == null) return;
62              GeraAtividade();
63              ColocarPontosEmSuspensao();
64          }
65
66
67
68          private void ButtonClickIniciar(object sender, RoutedEventArgs e)
69          {
70              if (cboAtividades.SelectedValue == null)
71              {
72                  MessageBox.Show(
73                      "Escolha uma atividade",
74                      "Informação",
75                      MessageBoxButton.OK,
76                      MessageBoxImage.Information);
77                  return;
78              }
79              _contPontos = 0;
80              if (_listaPontos != null)
81              {
```

```
82                  ColocarPontosEmSuspensao();
83                  DeterminarTipoPonto();
84                  GeraAtividade();
85                  _iniciarFase = true;
86                  _trajetoria = new List<PontosTrajetoria>();
87                  _iniciarCapturaPontosTrajetoria = false;
88                  _cronometroFase.Stop();
89                  _cronometroFase.Reset();
90              }
91
92      }
93
94          private void ColocarPontosEmSuspensao()
95          {
96              foreach (var ponto in _listaPontos)
97              {
98                  ponto.tipoPonto = "suspensao";
99              }
100         }
101
102         private void DeterminarTipoPonto()
103         {
104             if (_contPontos == 0)
105             {
106                 _listaPontos[_contPontos].tipoPonto = "objetivo";
107             }
108
109         }
110
111         private void GravarAtividade(object sender, RoutedEventArgs e)
112         {
113             try
114             {
115                 var appHistoricoAtividadePaciente =
116 Principal.appHistoricoAtividadePaciente;
117                 var historico = new HistoricoAtividadePaciente();
118                 var atividade =
119 Principal.appAtividade.ConsultarAtividade(_atividadeGlobal.Id);
120
121                 historico.IdAtividade = _atividadeGlobal.Id;
122                 historico.IdPaciente = _atividadeGlobal.IdPaciente;
123                 historico.NumeroRepeticoes = atividade.FeitaNumeroVezes++;
124                 historico.DataDesenvolvimento = _dataAtividade;
125                 historico.Tempo = _tempoDesenvolvimentoAtividade;
126
127                 Principal.appAtividade.Alterar(atividade);
128
129
130                 appHistoricoAtividadePaciente.Salvar(historico);
131
132                 var idHistorico =
133 Principal.appHistoricoAtividadePaciente.Lista().Last().Id;
134
135                 foreach (var p in _trajetoria)
136                 {
137                     p.IdHistorico = idHistorico;
138                     p.IdAtividade = _atividadeGlobal.Id;
139                     p.IdPaciente = _atividadeGlobal.IdPaciente;
140                     Principal.appPontosTrajetoria.Salvar(p);
141                 }
142
143                 MessageBox.Show(
```

```
144                     "Atividade gravada com sucesso",
145                     "Informação",
146                     MessageBoxButton.OK,
147                     MessageBoxImage.Information);
148             }
149             catch (Exception ex)
150             {
151                 MessageBox.Show(ex.Message);
152             }
153         }
154
155
156
157         private void GeraAtividade()
158         {
159             _atividadeGlobal =
160 Principal.appAtividade.ConsultarAtividade(Convert.ToInt32(cboAtividades.Text));
161
162             _listaPontos =
163 Principal.appPonto.ConsultarPontoPorAtividade(Convert.ToInt32(cboAtividades.Text
164 ));
165
166             FolhaDesenho.Children.Clear();
167
168             if (_listaPontos != null)
169             {
170                 if (_opcoes.ShowLinha)
171                 {
172                     DesenharLinhas();
173                     DesenharPontos();
174                 }
175                 if (_opcoes.ShowPonto)
176                 {
177                     DesenharPontos();
178                 }
179                 if (_opcoes.ShowCronometro)
180                 {
181                     lblCronometro.Visibility = Visibility.Visible;
182                     lblCronometroInfo.Visibility = Visibility.Visible;
183                     fundoCronometro.Visibility = Visibility.Visible;
184                 }
185             }
186         }
187
188         public void DesenharLinhas()
189         {
190             for (var i = 0; i < _listaPontos.Count - 1; i++)
191             {
192                 AddLinha(_listaPontos[i], _listaPontos[i + 1]);
193             }
194         }
195
196         public void DesenharPontos()
197         {
198             if (_opcoes.ShowPonto)
199             {
200                 foreach (var ponto in _listaPontos)
201                 {
202                     AddPontoFolhaDesenho(ponto);
203                 }
204             } else
205             {
```

```
206                 var pontoOrigem = _listaPontos.ElementAt(0);
207                 var pontoDestino = _listaPontos.ElementAt(_listaPontos.Count -
208 1);
209                 pontoDestino.Numero = 2;
210                 AddPontoFolhaDesenho(pontoOrigem);
211                 AddPontoFolhaDesenho(pontoDestino);
212             }
213         }
214 
215         public void AddLinha(Ponto origem, Ponto destino)
216         {
217             var cor = new SolidColorBrush();
218 
219             switch (_opcoes.CorLinha)
220             {
221                 case 0:
222                     cor.Color = Colors.Blue;
223                     break;
224                 case 1:
225                     cor.Color = Colors.Black;
226                     break;
227                 case 2:
228                     cor.Color = Colors.Green;
229                     break;
230                 case 3:
231                     cor.Color = Colors.Red;
232                     break;
233             }
234 
235             var linha = new Line
236             {
237                 X1 = origem.PosicaoX + _atividadeGlobal.PontoTamanho / 2.0,
238                 Y1 = origem.PosicaoY + _atividadeGlobal.PontoTamanho / 2.0,
239                 X2 = destino.PosicaoX + _atividadeGlobal.PontoTamanho / 2.0,
240                 Y2 = destino.PosicaoY + _atividadeGlobal.PontoTamanho / 2.0,
241                 StrokeThickness = _opcoes.EspessuraLinha,
242                 Stroke = cor
243             };
244 
245 
246             FolhaDesenho.Children.Add(linha);
247         }
248 
249         private void AddPontoFolhaDesenho(Ponto ponto)
250         {
251 
252             //Grid---------------------------------------
253             var pontoArea = new Grid
254             {
255                 Width = _atividadeGlobal.PontoTamanho,
256                 Height = _atividadeGlobal.PontoTamanho
257             };
258 
259 
260             var bola = new Ellipse{
261                 Name = string.Format("Ponto_{0}", ponto.Numero),
262                 Width = ponto.Largura,
263                 Height = ponto.Altura
264             };
265 
266             if (ponto.tipoPonto.Equals("objetivo"))
267             {
```

```
268                 bola.Fill = Brushes.Red;
269             }
270             else if (ponto.tipoPonto.Equals("atingido"))
271             {
272                 bola.Fill = Brushes.LawnGreen;
273             }
274             else
275             {
276                 bola.Fill = Brushes.DarkGray;
277             }
278 
279             Canvas.SetTop(bola, ponto.PosicaoY);
280             Canvas.SetLeft(bola, ponto.PosicaoX);
281             pontoArea.Children.Add(bola);
282 
283             //Texto-------------------------------
284             var idNumero = new TextBlock
285             {
286                 Text = "" + ponto.Numero,
287                 Foreground = Brushes.Black,
288                 FontSize = _atividadeGlobal.PontoTamanho / 2.0,
289                 FontWeight = FontWeights.Bold,
290                 HorizontalAlignment = HorizontalAlignment.Center,
291                 VerticalAlignment = VerticalAlignment.Center
292             };
293 
294             pontoArea.Children.Add(idNumero);
295 
296             //Centralizando o idNumero dentro do pontoArea
297             Canvas.SetTop(pontoArea, ponto.PosicaoY);// - (pontoArea.Height /
298 2));
299             Canvas.SetLeft(pontoArea, ponto.PosicaoX);// - (pontoArea.Width /
300 2));
301 
302             FolhaDesenho.Children.Add(pontoArea);
303 
304 
305         }
306 
307 
308 
309         private void CarregarComboBoxAtividades()
310         {
311             var idAtividade = new List<int>();
312 
313             foreach (var atividade in _listaAtividades)
314             {
315                 idAtividade.Add(((Atividade)atividade).Id);
316             }
317 
318             //Ordenando a lista de atividades
319             idAtividade.Sort();
320 
321             cboAtividades.ItemsSource = idAtividade;
322             cboAtividades.SelectedIndex = 1;
323         }
324 
325 
326         private Ponto PaintBola(Point currentPosition)
327         {
328             var bola = new Ellipse
329                             {
```

```
330                                 Name = "Finder",
331                                 Width = (int) _atividadeGlobal.PontoTamanho/2.0,
332                                 Height = (int) _atividadeGlobal.PontoTamanho/2.0,
333                                 Fill = _opcoes.ShowEsqueleto ? Brushes.Black :
334 Brushes.White
335                             };
336 
337             Canvas.SetTop(bola, currentPosition.Y);
338             Canvas.SetLeft(bola, currentPosition.X);
339 
340             int count = paintCanvas.Children.Count;
341             paintCanvas.Children.Add(bola);
342 
343             if (count > 1)
344             {
345                 paintCanvas.Children.RemoveAt(0);
346             }
347 
348             var ponto = new Ponto
349                         {
350                             PosicaoX = (int) currentPosition.X,
351                             PosicaoY = (int) currentPosition.Y,
352                             Largura = 10,
353                             Altura = 10
354                         };
355 
356             return ponto;
357 
358         }
359 
360         private bool HouveColisao(Ponto bolaPosicao)
361         {
362             var alvo = (Ponto)_listaPontos[_contPontos];
363             var alvoPosicaoX1 = 0;
364             var alvoPosicaoX2 = 0;
365             var alvoPosicaoY1 = 0;
366             var alvoPosicaoY2 = 0;
367 
368             //Console.WriteLine(); ;
369             if (alvo != null)
370             {
371                 alvoPosicaoX1 = (int)alvo.PosicaoX;
372                 alvoPosicaoX2 = (int)(alvoPosicaoX1 + alvo.Largura);
373 
374                 alvoPosicaoY1 = (int)alvo.PosicaoY;
375                 alvoPosicaoY2 = (int)(alvoPosicaoY1 + alvo.Altura);
376 
377 
378                 //Console.WriteLine("Alvo: {0} - {1} | {2} - {3}",
379                 //    alvoPosicaoX1, alvoPosicaoX2,
380                 //    alvoPosicaoY1, alvoPosicaoY2);
381             }
382 
383             var finderPosicaoX1 = bolaPosicao.PosicaoX;
384             var finderPosicaoX2 = finderPosicaoX1 + bolaPosicao.Largura/2.0;
385             var finderPosicaoY1 = bolaPosicao.PosicaoY; //bolaPosicao.Y;
386             var finderPosicaoY2 = finderPosicaoY1 + bolaPosicao.Altura/2.0;
387 
388 
389             //Console.WriteLine("Finder: {0} - {1} | {2} - {3}",
390             //        finderPosicaoX1, finderPosicaoX2,
391             //        finderPosicaoY1, finderPosicaoY2);
```

```
392
393
394             if ((finderPosicaoX1 >= alvoPosicaoX1 && finderPosicaoX2 <=
395 alvoPosicaoX2) &&
396                 (finderPosicaoY1 >= alvoPosicaoY1 && finderPosicaoY2 <=
397 alvoPosicaoY2)
398                 )
399             {
400                 if (_contPontos == 0)
401                 {
402                     _cronometroFase.Start();
403                     _iniciarCapturaPontosTrajetoria = true;
404                 }
405
406                 if (_opcoes.ShowPonto)
407                 {
408                     _listaPontos[_contPontos++].tipoPonto = "atingido";
409                 } else
410                 {
411                     _listaPontos[_contPontos++].tipoPonto = "atingido";
412                     if (_contPontos < _listaPontos.Count)
413                         _contPontos = _listaPontos.Count-1;
414                 }
415                 if (_contPontos < _listaPontos.Count)
416                     _listaPontos[_contPontos].tipoPonto = "objetivo";
417
418
419                 return true;
420             }
421             return false;
422         }
423
424         #region Kinect
425
426         private KinectSensor _myKinect;
427
428         private void IniciarKinect()
429         {
430             try
431             {
432                 _myKinect = KinectSensor.KinectSensors[0];
433                 //Para pessoa sentada
434                 _myKinect.SkeletonStream.TrackingMode =
435 SkeletonTrackingMode.Seated;
436                 _myKinect.ColorStream.Enable();
437                 _myKinect.SkeletonStream.Enable();
438                 _myKinect.ColorFrameReady += MyKinectColorFrameReady;
439                 _myKinect.SkeletonFrameReady += MyKinectSkeletonFrameReady;
440                 _myKinect.Start();
441                 _ocorreuErroKinect = false;
442             }
443             catch
444             {
445                 MessageBox.Show("Erro ao inicializar o Kinect");
446                 _ocorreuErroKinect = true;
447             }
448         }
449
450         void MyKinectColorFrameReady(object sender,
451 ColorImageFrameReadyEventArgs e)
452         {
453             using (var colorFrame = e.OpenColorImageFrame())
```

```
454             {
455                 if (colorFrame != null)
456                 {
457                     var colorData = new byte[colorFrame.PixelDataLength];
458                     colorFrame.CopyPixelDataTo(colorData);
459                     var bitmap = BitmapSource.Create(
460                         colorFrame.Width, colorFrame.Height, //Dimensões da
461 imagem
462                         96, 96, //Resolução
463                         PixelFormats.Bgr32, //Formato do vídeo
464                         null, //Paleta de cores
465                         colorData, //dados do video
466                         colorFrame.Width*colorFrame.BytesPerPixel);
467 
468                     if (_opcoes.ShowPaciente)
469                     {
470                         KinectVideo.Source = bitmap;
471                     }
472 
473                 }
474             }
475         }
476 
477         private void DesenharEsqueleto()
478         {
479             if (_skeletonPaciente == null) return;
480 
481             paintEsqueleto.Children.Clear();
482             var userBrush = Brushes.Black;
483             //Desenhando a cabeca e o torso
484             var joints = new[]
485                                 {
486                                     JointType.Head, JointType.ShoulderCenter
487 //,JointType.Spine
488                                 };
489             paintEsqueleto.Children.Add(GeraFigura(_skeletonPaciente, userBrush,
490 joints));
491 
492             ////Desenhando o braco esquerdo
493             joints = new JointType[]
494                                 {
495                                     JointType.ShoulderCenter,
496 JointType.ShoulderLeft,
497                                     JointType.ElbowLeft, JointType.WristLeft,
498 JointType.HandLeft
499                                 };
500             paintEsqueleto.Children.Add(GeraFigura(_skeletonPaciente, userBrush,
501 joints));
502 
503             //Desenhando o braco direito
504             joints = new JointType[]
505                             {
506                                 JointType.ShoulderCenter, JointType.ShoulderRight,
507                                 JointType.ElbowRight, JointType.WristRight,
508 JointType.HandRight
509                             };
510             paintEsqueleto.Children.Add(GeraFigura(_skeletonPaciente, userBrush,
511 joints));
512         }
513 
514         private Polyline GeraFigura(Skeleton skeletonPaciente, SolidColorBrush
515 userBrush, JointType[] joints)
```

```
516         {
517             var figure = new Polyline {StrokeThickness = 8, Stroke = userBrush};
518 
519             foreach (var t in joints)
520             {
521                 figure.Points.Add(GetJointPoint(skeletonPaciente.Joints[t]));
522             }
523             return figure;
524         }
525 
526         private Point GetJointPoint(Joint joint)
527         {
528 
529             var maoPosition = joint.Position;
530 
531             var posicaoPonto = _myKinect.MapSkeletonPointToDepth(
532                 joint.Position, DepthImageFormat.Resolution640x480Fps30);
533 
534             //Calculando a posição X na tela
535             var x = (int)
536 ((posicaoPonto.X*TelaJogo.ActualWidth/_myKinect.DepthStream.FrameWidth));
537             //Centralizando o objeto na mao.
538             x += (int) (_atividadeGlobal.PontoTamanho/2.0);
539             //Calculando a posição Y na tela
540             var y = (int)
541 ((posicaoPonto.Y*TelaJogo.ActualHeight/_myKinect.DepthStream.FrameHeight));
542             //Centralizando o objeto na mao.
543             y += (int) (_atividadeGlobal.PontoTamanho/2.0);
544             return new Point(x, y);
545 
546         }
547 
548         Skeleton[] skeletons = null;
549         private Skeleton _skeletonPaciente;
550         private void MyKinectSkeletonFrameReady(object sender,
551 SkeletonFrameReadyEventArgs e)
552         {
553             var retangulo = new Rectangle
554             {
555                 Name = "Retangulo",
556                 Stroke = Brushes.Red,
557                 Width = 110,
558                 Height = 50
559             };
560 
561             var posicao = new Point(0, 0);
562 
563             using (var frame = e.OpenSkeletonFrame())
564             {
565                 if (frame != null)
566                 {
567                     skeletons = new Skeleton[frame.SkeletonArrayLength];
568                     frame.CopySkeletonDataTo(skeletons);
569                 }
570                 if (skeletons == null) return;
571 
572                 _skeletonPaciente = null;
573                 foreach (var skeleton in skeletons)
574                 {
575                     if (skeleton.TrackingState == SkeletonTrackingState.Tracked)
576                     {
577                         _skeletonPaciente = skeleton;
```

```
578
579                         if (_opcoes.ShowEsqueleto && _atividadeGlobal != null)
580                         {
581                             DesenharEsqueleto();
582                         }
583
584                         if (_atividadeGlobal != null)
585                         {
586                             var maoJoint =
587 _atividadeGlobal.LadoTreino.Equals("Esquerdo")
588                                                     ?
589 skeleton.Joints[JointType.HandLeft]
590                                                     :
591 skeleton.Joints[JointType.HandRight];
592                             var maoPosition = maoJoint.Position;
593
594                             var posicaoMaoTela =
595 _myKinect.MapSkeletonPointToDepth(
596                                 maoJoint.Position,
597 DepthImageFormat.Resolution640x480Fps30);
598                             //Calculando a posição X na tela
599                             var x = (int)
600 ((posicaoMaoTela.X*TelaJogo.ActualWidth/_myKinect.DepthStream.FrameWidth));
601                             //Centralizando o objeto na mao.
602                             x += (int) (_atividadeGlobal.PontoTamanho/2.0);
603
604                             //Calculando a posição Y na tela
605                             var y = (int)
606 ((posicaoMaoTela.Y*TelaJogo.ActualHeight/_myKinect.DepthStream.FrameHeight));
607                             //Centralizando o objeto na mao.
608                             y += (int) (_atividadeGlobal.PontoTamanho/2.0);
609
610
611                             posicao = new Point(x, y);
612                             PaintBola(posicao);
613                             canvasMensagens.Children.Add(ShowMessages(5));
614
615
616                         }
617                     }
618                 }
619                 if (_skeletonPaciente == null)
620                 {
621                     paintEsqueleto.Children.Clear();
622                     canvasMensagens.Children.Add(ShowMessages(0));
623
624
625                     return;
626                 }
627                 canvasMensagens.Children.Add(ShowMessages(5));
628
629
630
631                 // Console.WriteLine(message);
632                 if (_listaPontos == null) return;
633
634                 //if (contPontos == listaPontos.Count)
635                 if (_contPontos == _listaPontos.Count)
636                 {
637                     _cronometroFase.Stop();
638                     var tempo = lblCronometro.Content.ToString().Replace(":",
639 ",");
```

```
640                     //Console.WriteLine(tempo);
641                     _tempoDesenvolvimentoAtividade = Convert.ToDouble(tempo);
642                     canvasMensagens.Children.Add(ShowMessages(3));
643                     paintEsqueleto.Children.Clear();
644                     _iniciarFase = false;
645                 }
646                 else
647                 {
648                     if (PacienteEmPosicao())
649                     {
650                         if (_iniciarFase)
651                         {
652                             if (_contPontos < _listaPontos.Count)
653                             {
654                                 HouveColisao(PaintBola(posicao));
655                                 GeraAtividade();
656                                 if (_iniciarCapturaPontosTrajetoria)
657                                 {
658                                     var pontoTrajetoria = new PontosTrajetoria{
659                                         X = (int) posicao.X,
660                                         Y = (int) posicao.Y,
661                                         IdPontoDestino = _contPontos
662                                     };
663                                     if (_contPontos < _listaPontos.Count)
664                                         _trajetoria.Add(pontoTrajetoria);
665                                 }
666                             }
667 
668                             lblCronometro.Content =
669 string.Format("{0:0}:{1:000}",
670 
671 _cronometroFase.Elapsed.TotalSeconds,
672 
673 _cronometroFase.Elapsed.Milliseconds);
674                         }
675                         if (canvasGeral.Children.Contains(retangulo))
676                         {
677                             canvasGeral.Children.Remove(retangulo);
678                         }
679                     }
680                     else
681                     {
682                         canvasMensagens.Children.Add(ShowMessages(2));
683 
684                         if (_atividadeGlobal != null)
685                         {
686                             Canvas.SetTop(retangulo,
687 _atividadeGlobal.DistanciaSensorY - 25);
688                             Canvas.SetLeft(retangulo,
689 _atividadeGlobal.DistanciaSensorX - 50);
690                             canvasGeral.Children.Add(retangulo);
691                         }
692                     }
693                 }
694             }
695         }
696 
697         Atividade atvTemp = new Atividade();
698 
699         private bool PacienteEmPosicao()
700         {
701             if (_skeletonPaciente != null && _atividadeGlobal != null)
```

```
            {
                //Limite para a frente da posição ideal Eixo Z
                float porcentagemLimiteZ = _atividadeGlobal.DistanciaSensorZ *
0.1f;
                float limiteMaximoAZ = _atividadeGlobal.DistanciaSensorZ +
porcentagemLimiteZ;
                //Limite para trás da posicação ideal.
                float limiteMaximoBZ = _atividadeGlobal.DistanciaSensorZ -
porcentagemLimiteZ;

                //Limite para a frente da posição ideal Eixo X
                float porcentagemLimiteX = _atividadeGlobal.DistanciaSensorX *
0.1f;
                float limiteMaximoAX = _atividadeGlobal.DistanciaSensorX +
porcentagemLimiteX;
                //Limite para trás da posicação ideal.
                float limiteMaximoBX = _atividadeGlobal.DistanciaSensorX -
porcentagemLimiteX;


                //Limite para a frente da posição ideal Eixo Y
                float porcentagemLimiteY = _atividadeGlobal.DistanciaSensorY *
0.1f;
                float limiteMaximoAY = _atividadeGlobal.DistanciaSensorY +
porcentagemLimiteY;
                //Limite para trás da posicação ideal.
                float limiteMaximoBY = _atividadeGlobal.DistanciaSensorY -
porcentagemLimiteY;

                DeterminandoDistanciaDoSensor();

                float distanciaPacienteSensorZ = atvTemp.DistanciaSensorZ;
                float distanciaPacienteSensorX = atvTemp.DistanciaSensorX;
                float distanciaPacienteSensorY = atvTemp.DistanciaSensorY;


                //Console.WriteLine("X {0}, Y{1}, Z{2:0.0}",
                //    distanciaPacienteSensorX,
                //    distanciaPacienteSensorY,
                //    distanciaPacienteSensorZ);
                if (
                    (distanciaPacienteSensorZ < limiteMaximoAZ &&
distanciaPacienteSensorZ > limiteMaximoBZ) &&
                    (distanciaPacienteSensorX < limiteMaximoAX &&
distanciaPacienteSensorX > limiteMaximoBX) &&
                    (distanciaPacienteSensorY < limiteMaximoAY &&
distanciaPacienteSensorY > limiteMaximoBY)
                    )
                {
                    return true;
                }
            }
            return false;
        }

        private void DeterminandoDistanciaDoSensor()
        {
            var ombroCentral =
_skeletonPaciente.Joints[JointType.ShoulderCenter];
            var point = _myKinect.MapSkeletonPointToDepth(
                    ombroCentral.Position,
                    DepthImageFormat.Resolution640x480Fps30
```

```
764                     );
765
766             point.X = (int)((point.X * TelaJogo.ActualWidth /
767 _myKinect.DepthStream.FrameWidth));
768             point.Y = (int)((point.Y * TelaJogo.ActualHeight /
769 _myKinect.DepthStream.FrameHeight));
770
771
772             var z = atvTemp.DistanciaSensorZ = ombroCentral.Position.Z;
773             var x = atvTemp.DistanciaSensorX = point.X;
774             var y = atvTemp.DistanciaSensorY = point.Y;
775
776             AuxilioPosicaoPaciente(x, y, z);
777         }
778
779         private void AuxilioPosicaoPaciente(float x, float y, float z)
780         {
781             //Calculando porcentagem fora da posição X
782             var terapeutaX = _atividadeGlobal.DistanciaSensorX;
783             var pacienteX = x;
784             var porX = (1 - pacienteX/terapeutaX) * 100;
785             lblPosicaoX.Content = string.Format("{0:0.00} % dentro do limite
786 X.",
787                                                 porX);
788
789             //Calculando porcentagem fora da posição Y
790             var terapeutaY = _atividadeGlobal.DistanciaSensorY;
791             var pacienteY = y;
792             var porY = (1 - pacienteY / terapeutaY) * 100;
793             lblPosicaoY.Content = string.Format("{0:0.00} % dentro do limite
794 Y.",
795                                                 porY);
796
797             //Calculando porcentagem fora da posição Z
798             var terapeutaZ = _atividadeGlobal.DistanciaSensorZ;
799             var pacienteZ = z;
800             var porZ = (1 - pacienteZ / terapeutaZ) * 100;
801             lblPosicaoZ.Content = string.Format("{0:0.00} % dentro do limite
802 Z.",
803                                                 porZ);
804
805             var bola = new Ellipse { Name = "Finder", Fill = Brushes.White,
806 Width = 10,
807                 Height = 10 };
808             Canvas.SetTop(bola, y);
809             Canvas.SetLeft(bola, x);
810             canvasAjuste.Children.Add(bola);
811             if (canvasAjuste.Children.Count > 1)
812             {
813                 canvasAjuste.Children.RemoveAt(0);
814             }
815
816         }
817
818         private void FinalizarKinect()
819         {
820             try
821             {
822                 if (_myKinect == null) return;
823                 _myKinect.Stop();
824                 _myKinect.Dispose();
825             }
```

```
826             catch
827             {
828                 MessageBox.Show("Erro ao finalizar o Kinect");
829             }
830         }
831 
832         #endregion
833 
834         private void WindowClosed(object sender, EventArgs e)
835         {
836             FinalizarKinect();
837         }
838 
839         private Label ShowMessages(int showMensagem)
840         {
841 
842             var lblStatus = new Label
843                                 {
844                                     BorderBrush = Brushes.Black,
845                                     Background = Brushes.White,
846                                 };
847 
848             switch (showMensagem)
849             {
850                 case 0:
851                     lblStatus.Content = "Paciente não encontrado!";
852                     lblStatus.FontSize = 36;
853                     lblStatus.FontWeight = FontWeights.Bold;
854                     lblStatus.Foreground = Brushes.Red;
855                     Canvas.SetTop(lblStatus, 100);
856                     Canvas.SetLeft(lblStatus, 286);
857                     break;
858                 case 1:
859                     lblStatus.Content = "Escolha a atividade";
860                     lblStatus.FontSize = 36;
861                     lblStatus.FontWeight = FontWeights.Bold;
862                     lblStatus.Foreground = Brushes.Red;
863                     Canvas.SetTop(lblStatus, 100);
864                     Canvas.SetLeft(lblStatus, 335);
865                     break;
866                 case 2:
867                     lblStatus.Content = "Coloque o paciente em posição";
868                     lblStatus.FontSize = 36;
869                     lblStatus.FontWeight = FontWeights.Bold;
870                     lblStatus.Foreground = Brushes.Red;
871                     Canvas.SetTop(lblStatus, 100);
872                     Canvas.SetLeft(lblStatus, 241);
873                     break;
874                 case 3:
875                     lblStatus.Content = "Atividade Concluída com Sucesso";
876                     lblStatus.FontFamily = new FontFamily("Comic Sans MS");
877                     lblStatus.FontSize = 36;
878                     lblStatus.FontWeight = FontWeights.Bold;
879                     lblStatus.Foreground = Brushes.GreenYellow;
880                     Canvas.SetTop(lblStatus, 100);
881                     Canvas.SetLeft(lblStatus, 214);
882                     break;
883                 default:
884                     lblStatus = new Label();
885                     break;
886             }
887             canvasMensagens.Children.Clear();
```

```
888
889             return lblStatus;
890         }
891
892     }
893 }
```

Listagem 7.31: Código fonte do arquivo Resultados.xaml

```
1  <Window x:Class="UIGraphic.Janela.Resultados"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="AVARFMS - Resultados"
5          Height="768" Width="1024" WindowStartupLocation="CenterScreen"
6  ResizeMode="NoResize">
7      <Grid Name="TelaJogo">
8          <Canvas x:Name="FolhaDesenho"/>
9      </Grid>
10 </Window>
```

Listagem 7.32: Código fonte do arquivo Resultados.cs

```
1  using System.Collections.Generic;
2  using System.Linq;
3  using System.Windows;
4  using System.Windows.Controls;
5  using System.Windows.Media;
6  using System.Windows.Shapes;
7  using Dominio;
8
9  namespace UIGraphic.Janela
10 {
11     /// <summary>
12     /// Interaction logic for Resultados.xaml
13     /// </summary>
14     public partial class Resultados
15     {
16         private readonly int _idPaciente;
17         private readonly HistoricoAtividadePaciente _historico;
18         private Atividade _atividade;
19         private List<Ponto> _pontosAtividade;
20         private List<PontosTrajetoria> _pontoTrajetoriaAtividade;
21
22         public Resultados()
23         {
24             InitializeComponent();
25         }
26
27         public Resultados(int idPaciente)
28         {
29             _idPaciente = idPaciente;
30             InitializeComponent();
31         }
32
33
34         public Resultados(HistoricoAtividadePaciente historicoAtividadePaciente)
35         {
36             _historico = historicoAtividadePaciente;
37             InitializeComponent();
38             RecuperarDados();
```

```
39              MontarTela();
40          }
41
42          private void RecuperarDados()
43          {
44              _atividade =
45  Principal.appAtividade.ConsultarAtividade(_historico.IdAtividade);
46              _pontosAtividade = Principal.appPonto.Lista().Where(p =>
47  p.IdAtividade == _historico.IdAtividade).ToList();
48              _pontoTrajetoriaAtividade =
49  Principal.appPontosTrajetoria.ConsultarPontoPorAtividade(_historico.Id).ToList();
50          }
51
52
53          private void MontarTela()
54          {
55              DesenharLinhas();
56              DesenharPontos();
57              DesenharTrajetoria();
58          }
59
60          public void DesenharTrajetoria()
61          {
62              for(var i = 0; i < _pontoTrajetoriaAtividade.Count - 1; i++)
63              {
64                  AddLinhaTrajetoria(_pontoTrajetoriaAtividade[i],
65  _pontoTrajetoriaAtividade[ i + 1]);
66              }
67          }
68
69          public void AddLinhaTrajetoria(PontosTrajetoria origem, PontosTrajetoria
70  destino)
71          {
72              var cor = new SolidColorBrush
73              {
74                  Color = Colors.Black
75              };
76
77              var linha = new Line
78              {
79                  X1 = origem.X+ _atividade.PontoTamanho / 2,
80                  Y1 = origem.Y + _atividade.PontoTamanho / 2,
81                  X2 = destino.X + _atividade.PontoTamanho / 2,
82                  Y2 = destino.Y + _atividade.PontoTamanho / 2,
83                  StrokeThickness = 1,
84                  Stroke = cor
85              };
86
87
88              FolhaDesenho.Children.Add(linha);
89          }
90
91          public void DesenharLinhas()
92          {
93              for (var i = 0; i < _pontosAtividade.Count - 1; i++)
94              {
95                  AddLinha(_pontosAtividade[i], _pontosAtividade[i + 1]);
96              }
97          }
98
99
100         public void DesenharPontos()
```

```
101         {
102             foreach (var ponto in _pontosAtividade)
103             {
104                 AddPontoFolhaDesenho(ponto);
105             }
106         }
107 
108         public void AddLinha(Ponto origem, Ponto destino)
109         {
110             var cor = new SolidColorBrush
111             {
112                 Color = Colors.Black
113             };
114 
115             var linha = new Line
116             {
117                 X1 = origem.PosicaoX + _atividade.PontoTamanho / 2,
118                 Y1 = origem.PosicaoY + _atividade.PontoTamanho / 2,
119                 X2 = destino.PosicaoX + _atividade.PontoTamanho / 2,
120                 Y2 = destino.PosicaoY + _atividade.PontoTamanho / 2,
121                 StrokeThickness = 3,
122                 Stroke = cor
123             };
124 
125 
126             FolhaDesenho.Children.Add(linha);
127         }
128 
129         private void AddPontoFolhaDesenho(Ponto ponto)
130         {
131 
132             //Grid---------------------------------------
133             var pontoArea = new Grid
134             {
135                 Width = _atividade.PontoTamanho,
136                 Height = _atividade.PontoTamanho
137             };
138 
139 
140             var bola = new Ellipse
141             {
142                 Name = string.Format("Ponto_{0}", ponto.Numero),
143                 Width = ponto.Largura,
144                 Height = ponto.Altura
145             };
146 
147             if (ponto.tipoPonto.Equals("objetivo"))
148             {
149                 bola.Fill = Brushes.Red;
150             }
151             else if (ponto.tipoPonto.Equals("atingido"))
152             {
153                 bola.Fill = Brushes.LawnGreen;
154             }
155             else
156             {
157                 bola.Fill = Brushes.DarkGray;
158             }
159 
160             Canvas.SetTop(bola, ponto.PosicaoY);
161             Canvas.SetLeft(bola, ponto.PosicaoX);
162             pontoArea.Children.Add(bola);
```

```
163
164             //Texto------------------------------
165             var idNumero = new TextBlock
166             {
167                 Text = "" + ponto.Numero,
168                 Foreground = Brushes.Black,
169                 FontSize = _atividade.PontoTamanho / 2.0,
170                 FontWeight = FontWeights.Bold,
171                 HorizontalAlignment = HorizontalAlignment.Center,
172                 VerticalAlignment = VerticalAlignment.Center
173             };
174
175             pontoArea.Children.Add(idNumero);
176
177             //Centralizando o idNumero dentro do pontoArea
178             Canvas.SetTop(pontoArea, ponto.PosicaoY);// - (pontoArea.Height /
179 2));
180             Canvas.SetLeft(pontoArea, ponto.PosicaoX);// - (pontoArea.Width /
181 2));
182
183             FolhaDesenho.Children.Add(pontoArea);
184
185         }
186     }
187 }
```