Daniela Cristina de Almeida Silvestre

Licenciada em Engenharia Eletrotécnica e Computadores

Desenvolvimento e Avaliação de um Sistema de Monitorização de Fisioterapia

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e Computadores

Orientador: Prof. Doutor José Manuel Fonseca, Professor Auxiliar, Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Júri:

Presidente: Prof. Doutor André Teixeira Bento Damas Mora

Arguente: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso

Vogal: Prof. Doutor José Manuel Matos Ribeiro da Fonseca

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Janeiro, 2017

Desenvolvimento e Avaliação de um Sistema de Monitorização de Fisioterapia

Agradecimentos

A escrita desta dissertação significa o término de uma fase bastante importante da minha vida. Chegar até aqui foi sem dúvida um percurso bastante difícil, mas muito gratificante que não teria sido possível sem o apoio das pessoas que me rodeiam.

Primeiramente, quero agradecer ao meu orientador, professor José Fonseca, por toda a ajuda disponibilizada, compreensão e paciência que teve comigo durante este trabalho. Quero agradecer também, à professora Helena Fino pela simpatia e por todas as ajudas e esclarecimentos académicos que me deu ao longo destes anos. Um especial agradecimento à Dª Rosário, que com a sua boa disposição sempre alegrou as minhas entradas no X.

Não podia deixar de agradecer à minha família, em especial aos meus pais e irmão.

Aos meus pais, agradeço todo o apoio e ajuda na concretização deste curso.

A ti, mano, agradeço o bom papel de irmão mais velho: os bons conselhos, a constante preocupação, carinho, amizade e a cumplicidade única que temos.

Obrigada a todos os colegas de curso com quem me cruzei durante estes anos, e que de alguma forma contribuíram para que o fecho desta etapa fosse possível.

Aos que para além de colegas se tornaram grandes amigos, o meu grande obrigado!

Não podia deixar de agradecer em especial ao Zeca, por todo o apoio, por todo o carinho, e por toda a paciência que teve comigo durante estes anos. Obrigada por acreditares e por me fazeres acreditar que era capaz.

Resumo

Quando um individuo apresenta uma limitação a nível cinético-funcional do seu corpo é provável que o seu médico de família o encaminhe para a fisioterapia. Estes problemas podem ser de origem genética ou resultado de traumas. Independentemente do tipo de reabilitação, existe um objetivo comum: o de tornar o tratamento o mais eficaz possível. A força de vontade e a motivação por parte do paciente são fatores importantes e influenciadores na recuperação. Um paciente motivado e empenhado na execução dos seus tratamentos tem maior probabilidade de recuperar mais rapidamente do que um paciente desanimado. Assim, surge a necessidade de tornar os tratamentos não só eficazes como também interessantes e apelativos ao paciente, motivando-o a seguir rigorosamente os exercícios que lhe são prescritos e deste modo, obter resultados positivos na sua recuperação.

O *biofeedback* pode ser utilizado como auxílio ao processo de reabilitação. Esta técnica pode ser aplicada a qualquer exercício e permite que o paciente tenha uma melhor perceção dos seus movimentos durante a execução das sessões, e, por conseguinte, corrigir possíveis falhas nos movimentos ou más posturas.

Posto isto, o sistema implementado consiste num software de exercícios de fisioterapia relativos a reabilitação ao nível dos ombros com uma componente de *biofeedback*, isto é, permitir ao paciente ter um *feedback* relativo à execução dos exercícios, em tempo real, e com isto compreender o impacto desta técnica no processo de recuperação.

Palavras-chave: Fisioterapia, Reabilitação, Kinect, *Biofeedback*

Abstract

When a person has a kinect-functional limitation of his body, it is possible that general practitioner doctor refers him to a treatment based on physical therapy. These problems can be genetic or trauma's result. Whatever type of rehabilitation, there is a common goal: make treatment as effective as possible. Willpower and motivation's patient are very important factor that influence a recovery.

A motivated patient and committed in his treatments execution has a higher probability a fast recuperation than a despondent patient. Thus occur the necessity to make treatments not only effective but also interesting and appealing to the patient, motivate him to follow strictly the prescribed treatments and notice positive results in recovery.

The *biofeedback* can be used to aid the rehabilitation process. This technique can be applied to any exercise and allows the patient to have a better perception of their movements during the sessions, and therefore, to correct flaws in movements or bad postures.

Finally, the implemented system is a physiotherapy exercises software, related to shoulder injuries. This software has a Biofeedback component that allows patient to have real-time feedback related to the exercises execution. Also allows understand the impact this technique has in rehabilitation.

Keywords: Physiotherapy, Rehabilitation, Kinect, Biofeedback

Conteúdo

1 INTRODUÇÃO 1
1.1 MOTIVAÇÃO 3
1.2 OBJETIVOS 3
1.3 ESTRUTURA DA DISSERTAÇÃO 4
2 ESTADO DE ARTE 7
2.1 VIRTUALREHAB 8
2.2 SEEME 11
2.3 NEUMIMIC 12
2.4 VERA ™ 13
2.5 JINTRONIX 13
2.6 MIRA 14
2.7 SÍNTESE 14
3 ENQUADRAMENTO TEÓRICO 17
3.1 REABILITAÇÃO TRADICIONAL 17
3.2 REABILITAÇÃO COM BIOFEEDBACK 19
3.2.1 Biofeedback 19
3.2.2 Microsoft Kinect 20
4 DESENVOLVIMENTO 27
4.1 INICIALIZAÇÃO DO KINECT E OBTENÇÃO DE FRAMES 28
4.2 DETEÇÃO DO CORPO 31
4.3 ESTRUTURA DO PROGRAMA 34
4.3.1 Interface gráfica 35
4.3.2 Base de dados 40
4.3.3 Abdução de ombro 42

4.3.4 Flexão de ombro *43*
4.3.5 Resistência de ombros *44*
4.3.6 Abdução de ombro com bola *45*
4.3.7 Flexão de ombros com halteres *46*
5 VALIDAÇÃO DE RESULTADOS **47**
5.1 DETEÇÃO DO CORPO 47
5.2 ABDUÇÃO DE OMBRO 48
5.2.1 Validação com biofeedback *48*
5.2.2 Validação sem biofeedback *50*
5.3 FLEXÃO DE OMBRO 51
5.3.1 Validação com biofeedback *51*
5.3.2 Validação sem biofeedback *53*
5.4 RESISTÊNCIA DE OMBROS 54
5.4.1 Validação com biofeedback *54*
5.4.2 Validação sem biofeedback *55*
5.5 ABDUÇÃO DE OMBRO COM BOLA 56
5.5.1 Validação com biofeedback *56*
5.5.2 Validação sem biofeedback *57*
5.6 FLEXÃO DE OMBROS COM HALTERES 58
5.6.1 Validação com biofeedback *59*
5.6.2 Validação sem biofeedback *61*
5.7 SÍNTESE DE RESULTADOS 62
6 CONCLUSÕES E TRABALHO FUTURO **65**
7 BIBLIOGRAFIA **67**
ANEXOS **71**

Lista de Figuras

FIGURA 2.1 - PACIENTE A USAR O SISTEMA *VIRTUALREHAB*. 8
FIGURA 2.2 – ESTRUTURA DAS VERSÕES DO *VIRTUALREHAB* 10
FIGURA 2.3 – PROTÓTIPO DE UMA SESSÃO DE EXERCÍCIOS EXECUTADA COM O *SEEME*. 12
FIGURA 3.1 – REABILITAÇÃO DE OMBRO. 18
FIGURA 3.2 – COMPONENTES DO SENSOR KINECT V2.0. 21
FIGURA 3.3 – DETEÇÃO DO CORPO INTEIRO E DETEÇÃO DO CORPO SENTADO 23
FIGURA 3.4 – ESPAÇO DE COORDENADAS DA CAMARA DO SISTEMA. 25
FIGURA 4.1 – ARQUITETURA DA IMPLEMENTAÇÃO DO SISTEMA. 27
FIGURA 4.2 – DECLARAÇÃO DAS CLASSES *KINECTSENSOR* E *MULTISOURCEFRAMEREADER* 28
FIGURA 4.3 - INICIALIZAÇÃO DO KINECT E LEITURA DE *FRAMES* 28
FIGURA 4.4 - MÉTODO PARA ACEDER ÀS *FRAMES* DE COR E DO CORPO. 29
FIGURA 4.5 – DECLARAÇÃO DE UM OBJETO DE IMAGEM NO XAML. 29
FIGURA 4.6 – MÉTODO DE CONVERSÃO DAS IMAGENS OBTIDAS DA *COLORFRAME* EM *BITMAPIMAGE*. 30
FIGURA 4.7 – CLASSE *CONSTANTS* 30
FIGURA 4.8 – VARIÁVEIS RELATIVAS À DETEÇÃO DO CORPO. 31
FIGURA 4.9 – MÉTODO PARA ACEDER ÀS *FRAMES* DE CORPO. 31
FIGURA 4.10 – CLASSE *BODYINFORMATION* 32
FIGURA 4.11 – MÉTODO *REFRESH*: INICIALIZAÇÃO DAS ARTICULAÇÕES. 33
FIGURA 4.12 – MÉTODO *CONVERT3DTO2D*. 33
FIGURA 4.13 – MÉTODO DE CÁLCULO DE ÂNGULOS ENTRE ARTICULAÇÕES. 34
FIGURA 4.14 - MÉTODO DE CÁLCULO DO COMPRIMENTO ENTRE MEMBROS 34
FIGURA 4.15 – MENU INICIAL DE IDENTIFICAÇÃO DO UTILIZADOR 35
FIGURA 4.16 – MENU DE OPÇÕES DO TERAPEUTA E DO PACIENTE. 35
FIGURA 4.17 - QUESTIONÁRIO DE INFORMAÇÕES DO PACIENTE. 36
FIGURA 4.18 – PAINEL DE EXERCÍCIOS. 37
FIGURA 4.19 – FORMULÁRIO DE IDENTIFICAÇÃO DO PACIENTE. 38
FIGURA 4.20 – FORMULÁRIO DE CONTACTO ENTRE OS PACIENTES E OS TERAPEUTAS 38
FIGURA 4.21 - PAINEL DE EXECUÇÃO DE EXERCÍCIOS 39

FIGURA 4.22 – MODELO RELACIONAL. 40
FIGURA 4.23 – COMANDO PARA A PESQUISA DO HISTÓRICO DO PACIENTE 41
FIGURA 4.24 – HISTÓRICO DE SESSÕES DA PACIENTE "MARIA JOAQUINA". 42
FIGURA 4.25 – ESQUEMA DO MOVIMENTO DE ABDUÇÃO PARA AMBOS OS BRAÇOS PARA UM DADO ÂNGULO. 42
FIGURA 4.26 – EXEMPLO DE ARTICULAÇÕES NECESSÁRIAS PARA O CÁLCULO DOS ÂNGULOS DOS MOVIMENTOS. 44
FIGURA 4.27 - ESQUEMA DO MOVIMENTO DE FLEXÃO. 44
FIGURA 4.28 – ESQUEMA DO MOVIMENTO DO EXERCÍCIO RESISTÊNCIA DE OMBROS 45
FIGURA 4.29 – POSIÇÃO INICIAL E FINAL DO MOVIMENTO DE ABDUÇÃO COM O BRAÇO ESQUERDO. 45
FIGURA 4.30 - POSIÇÃO INICIAL E FINAL DO MOVIMENTO DE ABDUÇÃO COM O BRAÇO ESQUERDO. 45
FIGURA 4.31 – ESQUEMA DO MOVIMENTO ALTERNADO DE FLEXÃO. 46
FIGURA 5.1 – DESENHO DAS ARTICULAÇÕES E DOS MEMBROS DETETADOS PELO KINECT SOBRE A IMAGEM DE COR. 47
FIGURA 5.2 – ABDUÇÃO COM O BRAÇO ESQUERDO (ÂNGULO ESTIMADO = 13º) 49
FIGURA 5.3 – ABDUÇÃO COM O BRAÇO ESQUERDO (ÂNGULO ESTIMADO = 36º) 49
FIGURA 5.4 – ABDUÇÃO COM O BRAÇO ESQUERDO (ÂNGULO ESTIMADO = 41º) 49
FIGURA 5.5 – ABDUÇÃO COM O BRAÇO ESQUERDO SEM *BIOFEEDBACK.* 50
FIGURA 5.6 – DETEÇÃO DO TRONCO E DO BRAÇO DIREITO 51
FIGURA 5.7 - FLEXÃO (ÂNGULO ESTIMADO = 13º) 52
FIGURA 5.8 - FLEXÃO (ÂNGULO ESTIMADO = 148º) 52
FIGURA 5.9 - FLEXÃO (ÂNGULO ESTIMADO = 173º) 52
FIGURA 5.10 – FLEXÃO SEM *BIOFEEDBACK.* 53
FIGURA 5.11 – RESISTÊNCIA DE OMBROS (ÂNGULO ESQUERDO = 7º; ÂNGULO DIREITO = 10º) 54
FIGURA 5.12 - RESISTÊNCIA DE OMBROS (ÂNGULO ESQUERDO = 84º; ÂNGULO DIREITO = 83º) 54
FIGURA 5.13 - RESISTÊNCIA DE OMBROS (ÂNGULO ESQUERDO = 90º; ÂNGULO DIREITO = 94º) 55
FIGURA 5.14 – RESISTÊNCIA DE OMBROS SEM *BIOFEEDBACK* 55
FIGURA 5.15 – ABDUÇÃO COM BOLA (POSIÇÃO INICIAL; ÂNGULO ESTIMADO = 179º) 57
FIGURA 5.16 – ABDUÇÃO COM BOLA (POSIÇÃO FINAL; ÂNGULO ESTIMADO = 111º) 57
FIGURA 5.17 – ABDUÇÃO COM BOLA (POSIÇÃO INICIAL; SEM *BIOFEEDBACK*) 57
FIGURA 5.18 - ABDUÇÃO COM BOLA (POSIÇÃO FINAL; SEM *BIOFEEDBACK*) 58
FIGURA 5.19 – FLEXÃO COM HALTERE (BRAÇO DIREITO; ÂNGULO = 115º) 59
FIGURA 5.20 – FLEXÃO COM HALTERE (BRAÇO DIREITO; ÂNGULO = 166º) 59
FIGURA 5.21 – FLEXÃO COM HALTERE (BRAÇO ESQUERDO; ÂNGULO = 68º) 60
FIGURA 5.22 – FLEXÃO COM HALTERE (BRAÇO ESQUERDO; ÂNGULO = 157º) 60
FIGURA 5.23 - FLEXÃO COM HALTERE (BRAÇO DIREITO; SEM *BIOFEEDBACK*) 61
FIGURA 5.24 - FLEXÃO COM HALTERE (BRAÇO ESQUERDO; SEM *BIOFEEDBACK*) 61

Lista de Tabelas

TABELA 3.1 – TABELA COMPARATIVA DE ESPECIFICAÇÕES DO KINECT V1.0 E KINECT V2.0 21
TABELA 3.2 – DESCRIÇÃO DOS DIFERENTES ESTADOS DETETADOS DAS MÃOS. 23
TABELA 3.3 – PROPRIEDADE DA CLASSE *FACE* 24
TABELA 4.1 – TABELA DE PACIENTES. 41
TABELA 4.2 – TABELA DE SESSÕES. 41
TABELA 4.3 – TABELA DE EXERCÍCIOS. 41
TABELA 5.1 – ÂNGULOS ESTIMADOS - EXERCÍCIO DE ABDUÇÃO. 50
TABELA 5.2 - ÂNGULOS ESTIMADOS - EXERCÍCIO DE FLEXÃO 53
TABELA 5.3 - ÂNGULOS ESTIMADOS - EXERCÍCIO DE ABDUÇÃO COM HALTERES. 56
TABELA 5.4 - ÂNGULOS ESTIMADOS - EXERCÍCIO DE ABDUÇÃO 58
TABELA 5.5 - ÂNGULOS ESTIMADOS - FLEXÃO DE OMBROS COM HALTERES. 62

Acrónimos

2D Espaço Bidimensional

3D Espaço Tridimensional

API Application Programming Interface

Bpp Bites per pixel

DPI Dots Per Inch

HD High Definition

IR Infrared

MFR Medicina Física e de Reabilitação

RGB Red Green Blue

ROM Range Of Motion

SaaS Software as a Service

SDK Software Developer Kit

SQL Structured Query Language

WPF Windows Presentation Form

XAML eXtensible Application Markup Language

1 Introdução

Um método muito conhecido de reabilitação física é a fisioterapia. A fisioterapia auxilia a recuperação de pacientes com problemas cinéticos funcionais que condicionam a sua qualidade de vida.

São muitas as pessoas que fazem fisioterapia, pelas mais diversas razões: acidentes rodoviários, acidentes de trabalho, problemas de saúde, etc.

Dependendo do problema do indivíduo o fisiatra que avalia a condição clínica do paciente, prescreve um conjunto de exercícios adequados à problemática. Para que o tratamento se torne eficaz é importante que o paciente o execute de forma correta. Deste modo, surge a necessidade de uma monitorização mais rigorosa da realização de cada exercício. O exercício irá exigir uma determinada amplitude (de braços por exemplo) ou até mesmo uma determinada postura. No caso de exercícios com pesos (halteres) o rigor torna-se maior, pois a pessoa ao executar o exercício de forma incorreta, corre o risco de piorar a lesão ou até mesmo criar uma nova.

Esta monitorização nem sempre pode ser feita pelo fisioterapeuta e muitas vezes, existem tratamentos que também são importantes realizar fora das clínicas de reabilitação. O paciente, em alguns casos, também pode realizar os exercícios em casa. É aqui que surge o problema. O terapeuta recomenda um determinado exercício para ser executado em casa e o paciente, por vezes não cumpre ou faz os exercícios de forma incorreta, podendo retardar a recuperação. Assim, surge uma desmotivação por parte do paciente, pois os exercícios que tem de executar são aborrecidos e como não obtém *feedback* sobre o que está a fazer perde a vontade de os realizar.

Uma boa maneira de motivar o paciente e ajudá-lo a recuperar de um problema físico é tornar os exercícios divertidos e ao mesmo tempo desafiantes.

Assim, surge a necessidade de tornar a fisioterapia não só num tratamento necessário, como também em algo prazeroso.

Com o *biofeedback* é possível tornar exercícios físicos normais em algo motivador para o paciente. O paciente ao ver resultados imediatos, e tendo uma melhor perceção dos movimentos que está a executar, sente uma motivação extra em corrigir possíveis falhas e assim melhorar o seu estado físico e consequentemente o problema que está a tratar. O *biofeedback* apresenta um dos mais eficazes mecanismos de reabilitação motora [1].

O desenvolvimento de sistemas com *biofeedback* permite diminuir o tempo de recuperação e apresentam um incremento a nível de qualidade de reabilitação [2]. Vários autores têm vindo a desenvolver sistemas deste tipo e diversos estudos mostram que estes sistemas contribuem de forma bastante positiva no processo de reabilitação física [3].

Neste projeto foi desenvolvido um sistema capaz de monitorizar os exercícios feitos pelo paciente através do uso do Kinect v2. O Kinect é um sensor de movimento desenvolvido pela Microsoft para a Xbox que permite que o utilizador interaja com jogos, através de voz e de gestos e movimentos do seu corpo sem que seja necessário usar um comando, como era comum nas consolas de videojogos tradicionais. O Software Development Kit (SDK) 2.0 disponibiliza as ferramentas e APIs necessárias que permitem criar aplicações interativas com gestos, voz e movimentos usando a tecnologia do sensor Kinect v2.0 [4].

Os exercícios aqui desenvolvidos são destinados a problemas relacionados com os ombros, sendo basicamente exercícios de abdução e flexão existindo algumas variantes. Em alguns exercícios é recomendado o uso de objetos, nomeadamente halteres e bola de pilates. A maior parte dos exercícios será executado em pé sendo que apenas um será executado sentado.

1.1 Motivação

Em qualquer programa de reabilitação física, o fisioterapeuta numa fase inicial tem que explicar e demonstrar os exercícios a serem executados pelo paciente. Nem sempre o terapeuta acompanha de perto a sessão completa de cada paciente. Deste modo, cabe ao paciente prestar atenção às explicações, anteriormente dadas e reproduzir de forma correta os exercícios. Por vezes, as pessoas não prestam atenção à explicação da execução do exercício ou têm dificuldade em memorizar os movimentos necessários.

Uma forma de controlar com mais rigor as sessões de cada paciente, é garantir a monitorização de cada sessão por parte do fisioterapeuta. Esta prática implicaria que cada fisioterapeuta dedicasse mais tempo a cada paciente, aumentando as filas de espera e por conseguinte o descontentamento relativo aos serviços clínicos. Para reduzir as filas de espera a clínica teria que aumentar o número de profissionais contratados e comprar mais equipamentos clínicos, o que levaria a um aumento significativo das despesas.

Para o paciente, a reabilitação física em clínica apresenta custos de transporte, tempo de espera, entre outros desconfortos que podem ser evitados se houver a possibilidade de executarem os exercícios no conforto de suas casas.

Com o constante desenvolvimento das tecnologias, têm vindo a surgir cada vez mais propostas de sistemas que permitem a reabilitação física a partir de casa, possibilitando o acompanhamento do profissional à distância.

1.2 Objetivos

Existe uma lacuna de métodos práticos e de baixo custo que possam ser utilizados no contexto clínico a partir de casa do paciente com a garantia de acompanhamento especializado à distância [5].

Deste modo, o objetivo deste projeto consiste em desenvolver um software simples que permite a medição da amplitude de movimentos. Estas medições, em clínica, são feitas manualmente com recurso a um instrumento de medição designado goniómetro. Este instrumento apresenta um erro estimado entre 5 a 10º [6].

No sistema desenvolvido, estas medições são calculadas através de fórmulas matemáticas com base nas coordenadas das articulações obtidas através da câmara de profundidade do Microsoft Kinect.

Os exercícios de reabilitação são relacionados com problemáticas a nível dos ombros e apresentam uma vertente de *biofeedback*. Este sistema pode ser usado por qualquer pessoa, a partir de casa, sendo necessário apenas um computador com portas USB 3.0 e um sensor Kinect v2.

O foco do trabalho concentra-se não só no desenvolvimento dos exercícios como também em torná-los mais interessantes e divertidos para os pacientes cuja fisioterapia é prescrita. Deste modo, cada exercício terá uma componente de *feedback* que permite ao paciente ter uma melhor perceção dos movimentos que está a executar permitindo-lhe corrigir posturas menos corretas e assim tornar a fisioterapia mais eficaz e consequentemente a sua recuperação.

1.3 Estrutura da dissertação

Após este capítulo introdutório onde foi feita uma abordagem ao tema, motivação e objetivos do projeto, seguem-se os seguintes capítulos:

- Estado de arte: Este capitulo surge na sequência da pesquisa feita acerca do que já há feito na área abordada neste projeto. São mencionados detalhadamente alguns softwares dedicados à reabilitação com *biofeedback*, desenvolvidos com a tecnologia do Microsoft Kinect.

- Enquadramento teórico: Nesta seção é feita uma abordagem teórica aos dois tipos de reabilitação: tradicional e com *biofeedback*. Nesta última, é feita uma descrição teórica sobre especificações e funcionamento a nível de hardware e software da mais recente versão do Microsoft Kinect, usada na implementação do projeto desta dissertação.

- Desenvolvimento: Neste capítulo são abordados os passos necessários para o desenvolvimento deste projeto. É explicada a forma como se inicializou o Kinect

e o processo de obtenção das diferentes *frames*. Por fim, é explicado como se desenvolveu cada exercício.

- Validação de resultados: Neste capítulo são apresentados alguns testes para cada exercício, efetuados com e sem *biofeedback* e por fim são tiradas algumas conclusões com base nos resultados obtidos.

- Conclusões e trabalho futuro: Consiste no capítulo final desta dissertação e apresenta as conclusões feitas acerca deste projeto, nomeadamente se os objetivos propostos foram atingidos. Neste capítulo há espaço também para sugestões relativas a melhorias ou novas ideias para trabalhos futuros.

2 Estado de arte

São diversos os motivos que levam inúmeras pessoas a recorrer à fisioterapia. Deste modo, a procura por clínicas de reabilitação aumenta e as filas de espera também. As clínicas estão cheias de pacientes e os terapeutas que os acompanham não conseguem monitorizar continuamente todos os exercícios executados por todos os pacientes. Isto significa, que durante o processo de reabilitação, o utente faz parte dos exercícios sem monitorização, e como não recebe feedback relativo à execução dos seus movimentos e postura, é gerado um desânimo e aborrecimento por parte do paciente. Este desânimo leva a que a pessoa salte repetições ou séries, sabotando o seu próprio tratamento.

Desde 1980 que têm surgido diversos estudos acerca de aplicações clínicas com base na amplitude dos movimentos das articulações de um indivíduo [7]. Para este efeito eram utilizados sensores magnéticos, óticos ou inerciais. Os sistemas que usam sensores magnéticos não são práticos de utilizar devido à quantidade de cabos necessários e apresentam problemas na precisão das medições devido à existência de distorções magnéticas causadas por objetos metálicos [8]. Os sensores óticos são difíceis de utilizar em aplicações práticas devido à sua complexidade e espaço requerido [8]. Por sua vez, os sensores inerciais são fáceis de utilizar e possuem uma sensibilidade alta, constituindo uma boa opção de deteção de movimentos 3D. No entanto, à semelhança dos sensores magnéticos, estes são suscetíveis a erros de medição quando próximos de metais [9].

Após o lançamento do Microsoft Kinect foram várias as pessoas e empresas que começaram a desenvolver software clínico com base neste sensor. O crescente interesse deveu-se sobretudo ao custo acessível do equipamento, à sua portabilidade e ao facto de não serem necessários sensores extra para detetar as articulações do corpo das pessoas.

Assim, surgiram diversos estudos que visam compreender o impacto do uso deste sensor no processo de reabilitação física. Devido ao Microsoft Kinect v2 ser relativamente recente, a maioria dos estudos existentes são feitos com base na versão antiga do sensor.

Tilak Dutta [9] liderou um estudo cujo objetivo era avaliar o desempenho do Kinect como ferramenta de medição cinemática. Este estudo visou comparar a precisão, o alcance e o campo de visão do sensor com as características do sistema VICON [10] que utiliza sensores óticos para medição cinemática. O autor concluiu que com algum trabalho de desenvolvimento extra, o Kinect pode ser utilizado como um bom sistema portátil de captura de movimentos 3D para avaliações ergonómicas.

Até então, havia uma necessidade de tornar o processo de reabilitação divertido e mais interativo para o paciente por forma a aumentar a motivação e deste modo, fazer com que os pacientes seguissem rigorosamente os seus tratamentos melhorando progressivamente os seus resultados.

Assim, surgiram diversas *startups* que criaram softwares complementares à área da fisioterapia, aliando os exercícios fisioterapêuticos ao *biofeedback*.

2.1 *VirtualRehab*

O *VirtualRehab* [11] consiste num sistema clínico que incide sobre a reabilitação física. Este sistema utiliza tecnologia *cloud*, o que significa que pode ser acedido em diversos lugares, independentemente da plataforma que é usada, através do recurso à internet. Os programas de reabilitação são baseados no uso de videojogos e o progresso dos pacientes pode ser acompanhado. Este sistema permite que os pacientes o utilizem em suas casas.

Figura 2.1 - Paciente a usar o sistema *VirtualRehab*. (*VirtualRehab*)

O *VirtualRehab* permite tratar diversas patologias sendo indicado no tratamento das seguintes problemáticas:

- Lesões cerebrais (AVC por exemplo);
- Doenças degenerativas (Esclerose múltipla, Parkinson, Alzheimer e Esclerose Lateral Amiotrófica);
- Distúrbios neuromusculares (Distrofias, Miopatias, Atrofias e Neuropatias);
- Mobilidade para o idoso, monitorizando os níveis de atividade do idoso.

Este programa encontra-se dividido em nove jogos diferentes. Estes jogos são introduzidos nos programas de reabilitação física, permitindo treinar diversas funções. Incidem essencialmente em problemáticas ao nível do equilíbrio (em pé e em movimento), postura, força (empurrar objetos e transferir pesos), problemas motores (dificuldade em executar tarefas básicas tais como sentar, manter-se em pé ou caminhar) e coordenação. Estes jogos têm o principal objetivo de tornar a reabilitação algo divertido, motivando os pacientes e permitindo-lhes melhorar a sua situação e, por conseguinte, permitir a melhoria na execução de atividades vitais diárias necessárias a qualquer ser humano.

Para usar este sistema o material necessário é um Kinect para Windows, um computador e um monitor.

O *VirtualRehab* usa a tecnologia do Microsoft® Kinect para Windows para detetar o utilizador, permitindo ao utilizador comandar o jogo apenas com os seus movimentos. Assim, não existe necessidade de dispositivos extra, tais como comandos ou outro tipo de aparelho, sendo o corpo do utilizador e os seus movimentos os comandos dos jogos. Como este sistema é baseado na tecnologia *cloud*, cada sessão é gravada e fica disponível para consulta do fisioterapeuta. Assim, é possível haver um acompanhamento constante de cada paciente, permitindo ao fisioterapeuta acompanhar o progresso dos seus pacientes.

Existem duas versões diferentes do programa:

- *SaaS (Software as a Service)* : nesta versão o paciente pode usar o *VirtualRehab* em casa desde que tenha acesso à internet. Toda a informação do utente encontra-se armazenada no servidor *cloud* (Microsoft® Azure).
- *On Premises*: consiste na versão exclusiva para utilização em clínica (não podendo ser utilizada em casa dos pacientes). Não é necessária ligação à internet, estando a informação dos pacientes armazenada no servidor da própria clínica.

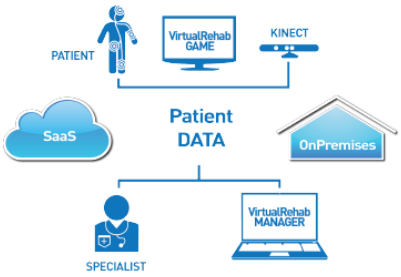

Figura 2.2 – Estrutura das versões do *VirtualRehab* (*VirtualRehab)*

As vantagens do uso do *VirtualRehab* encontram-se separadas em três setores:

1. Clínicas de reabilitação:

   - Melhoram os serviços oferecidos aos utentes;

   - Permite-lhes a prescrição de tratamentos específicos a cada paciente;

   - Diminuem as filas de espera;

   - Existe um maior número de adesões por parte dos pacientes com base neste tipo de reabilitação;

   - Diminuem as despesas de transporte devido à possibilidade de tele reabilitação.

2. Terapeutas:

   - Controlam remotamente os programas de tratamento;

   - Recebem alertas;

   - Programa simples e intuitivo, facilitando o utilizador comum;

   - Permite-lhes avaliar e monitorizar o progresso dos seus pacientes;

3. Pacientes:

   - Aderem aos tratamentos com mais facilidade;

   - O programa é simples de utilizar;

   - É-lhes apresentado não só um fator de reabilitação como também de diversão, com base no recurso aos jogos;

   - É desenvolvido para diversas problemáticas e inclusive para pacientes que necessitam de cadeiras de rodas;

   - Sentem-se mais motivados, aumentando a frequência com que executam o tratamento.

2.2 SeeMe

SeeMe [12] consiste num sistema clínico de diagnóstico e de exercícios controlados, que à semelhança do anterior usa a tecnologia do Microsoft Kinect.

Este sistema foi desenvolvido para ajudar os pacientes no processo de reabilitação, apostando na realidade virtual juntamente com a componente de *biofeedback.* Deste modo, o *SeeMe* é uma ferramenta importante para a recuperação de pessoas cujas funções corporais ou cognitivas se encontram comprometidas. As aplicações desenvolvidas visam melhorar a coordenação motora, o equilíbrio, aumentar a força muscular, melhorar problemas de memória e tempos de reação. As atividades são apresentadas em forma de jogos, o que motiva o paciente no processo de reabilitação. O programa é individualizado para cada paciente e o terapeuta pode acompanhar o progresso dos seus pacientes em tempo real e ajustar os parâmetros dos jogos de acordo com as necessidades do paciente.

O *SeeMe* apresenta uma componente de *biofeedback,* apresentando um relatório de progresso dos tratamentos de cada paciente. O sistema deteta os movimentos, os ângulos entre vários membros do corpo, monitoriza tempos de execução dos exercícios, entre outros. Os relatórios das sessões de cada utente possibilitam o terapeuta perceber o progresso do paciente, sendo uma mais valia em tomadas de decisão clínicas. Este programa pode ser utilizado como complemento no tratamento das seguintes ocorrências:

- Acidentes Vasculares Cerebrais (AVC);
- Esclerose Múltipla;
- Paralisia Cerebral;
- Doenças degenerativas (Alzheimer, Parkinson, etc.);
- Recuperação pós-operatório de cirurgias cerebrais;
- Problemas de coluna;
- Distúrbios de equilíbrio;
- Atrofias musculares;
- Entre outros.

Este sistema possui uma biblioteca de atividades que podem ser executadas tanto por crianças como adultos com diferentes problemáticas. De maneira a ser possível o uso do *SeeMe*, são necessários os seguintes requisitos:

- Kinect for Windows;
- Monitor ou TV;

- Sistema operativo Windows 7 ou Windows 8;
- Processador Intel® i3 ou superior;
- Memória RAM de 2GB;
- Placa gráfica NVIDIA® GeForce® ou ATI™Radeon™ ou Intel HD Graphics 4000 (ou melhor);
- 2 GB de espaço disponível de disco rígido.

O *SeeMe* foi desenvolvido com o objetivo de transformar a reabilitação num processo divertido e interativo para o utilizador. Este sistema é certificado (ISO 13485) [2] como dispositivo médico, sendo usado em diversas clínicas e centros de investigação do mundo.

Figura 2.3 – Protótipo de uma sessão de exercícios executada com o *SeeMe.* (*SeeMe*)

2.3 Neumimic

Neumimic é um software de reabilitação de baixo custo que surgiu duma colaboração entre a Academia da Força Aérea dos Estados Unidos e os serviços de saúde da *Penrose-St.Francis.* Este sistema pretende ajudar os pacientes a recuperar os seus movimentos afetados após AVC ou algum outro trauma [14]. Os cadetes desenharam um suporte para o braço do paciente de maneira a que este apenas mexa o braço de acordo com as indicações do terapeuta. Ao mesmo tempo, o progresso dos pacientes é gravado através da conexão do Microsoft Kinect, pelo que os exercícios podem ser executados mesmo que o terapeuta não esteja presente. Qualquer pessoa pode ter acesso a este software desde que pague uma subscrição mensal que permite aceder à *cloud* do software.

2.4 VERA ™

VERA [15] é uma plataforma de tele reabilitação desenvolvida pela empresa *Reflexion Health.* Este software usa a deteção do movimento (proveniente do Microsoft Kinect) para monitorizar movimentos e remotamente compreender a eficácia dos exercícios prescritos.

O progresso dos pacientes é guardado e é apresentado *feedback* em tempo real. A tele reabilitação permite que o paciente se motive a recuperar através de exercícios executados em casa.

Um grande foco deste programa é ajudar na prevenção de quedas que é muito comum ocorrerem em pessoas mais idosas. Através desta plataforma são apresentados programas de prevenção de risco de queda clinicamente comprovados.

2.5 Jintronix

Jintronix [16] é uma ferramenta que pretende combinar jogos virtuais com tratamentos especializados em reabilitação. Foi desenvolvido com a apoio de fisioterapeutas e terapeutas ocupacionais, o que garante que cada exercício e atividades desenvolvidos são desenhados a pensar numa recuperação segura e eficaz.

Este sistema encontra-se implementado em três modelos diferentes:

- Modelo integrado: é combinado com a terapia tradicional e inclui acompanhamento profissional, permitindo que o terapeuta avalie de forma rápida o progresso do paciente.
- Modelo livre: é usado em clínica sem necessidade de supervisão.
- Modelo remoto: é o modelo a ser usado pelo paciente em sua casa. Deste modo, o paciente pode continuar a efetuar os exercícios mesmo depois de ter alta médica e assim potenciar a sua recuperação.

2.6 MIRA

MIRA [17] à semelhança dos softwares anteriores foi desenvolvido para tornar a fisioterapia divertida e eficaz na recuperação de traumas ou pós-cirurgias.

Este sistema transformou exercícios de reabilitação já existentes relativos aos membros superiores, em vídeo jogos e usa o Microsoft Kinect em conjunto com sensores externos para monitorizar o progresso das atividades desenvolvidas pelos pacientes. Os jogos desenvolvidos para esta aplicação foram criados com uma probabilidade de 80% de hipóteses de vitória, como forma de motivação extra para o paciente se manter motivado durante o processo de reabilitação. Os estudos efetuados sobre a eficácia da plataforma concluem que os pacientes permaneceram motivados durante o processo de reabilitação e que a aplicação permitiu diminuir o tempo necessário de monitorização da atividade do paciente por parte do fisioterapeuta [18].

2.7 Síntese

O número de softwares desenvolvido com base em exercícios de reabilitação com *biofeedback* recorrendo à tecnologia do Microsoft Kinect tem vindo a aumentar.

São bastante semelhantes, pois são desenvolvidos com o mesmo objetivo: monitorizar exercícios de reabilitação e torná-los divertidos para o utilizador.

Todos os softwares mencionados anteriormente começaram por ser desenvolvidos com base no Microsoft Kinect v1. Com o lançamento da nova versão foram feitas atualizações de maneira a permitir que o seu sistema funcione com o novo aparelho.

Vários estudos [19] [20] comprovam a eficácia do uso do Microsoft Kinect em aplicações de reabilitação. Foi feito um estudo piloto, sobre a eficácia e a segurança do uso de um jogo computacional que usa o Microsoft Kinect como auxílio no processo de reabilitação postural em doentes de Parkinson [21]. O jogo desenvolvido treina a postura do individuo a partir de tarefas de alcance de objetos e de caminhada em múltiplas direções. Este estudo concluiu que os jogos de reabilitação que usam o Microsoft Kinect são seguros e viáveis para os doentes de Parkinson.

O Microsoft Kinect também pode ser utilizado como ferramenta de auxílio à aprendizagem para crianças [22]. Um estudo preliminar em contexto escolar, sobre a eficácia do Kinect como modalidade terapêutica para crianças com paralisia cerebral, mostrou que esta ferramenta auxilia na melhoria do equilíbrio destas crianças.

São muitos os estudos que relacionam o desenvolvimento de sistemas que usam o Microsoft Kinect com a reabilitação e tem vindo a ser comprovada cientificamente a eficácia do uso deste aparelho no contexto clinico. É uma área que tem vindo a ser bastante explorada, e há inúmeras possibilidades de aplicação deste aparelho.

Assim, neste projeto pretendeu-se desenvolver um sistema que permite que as pessoas executem exercícios de reabilitação, sem acompanhamento direto de um profissional, dando-lhes uma perceção visual dos seus movimentos e assim compreender a eficácia e o impacto na recuperação do individuo.

3 Enquadramento Teórico

3.1 Reabilitação tradicional

Designa-se por Medicina Física e de Reabilitação (MFR) a especialidade responsável por tratar da recuperação de funções comprometidas por lesões de ossos, músculos, tendões, articulações e nervos [3].

O objetivo da reabilitação física é possibilitar que indivíduos cujas capacidades se encontram afetadas, devido a lesões ou doenças, sejam autónomos e independentes.

A reabilitação física potencia um aumento de qualidade de vida e um papel mais ativo na vida social do individuo.

Nesta área trabalham terapeutas especialistas (fisiatras e fisioterapeutas), sendo os fisiatras responsáveis pela parte do diagnóstico e de elaboração do programa de reabilitação e o fisioterapeuta o responsável por acompanhar e monitorizar o paciente durante a sessão de reabilitação. Cada programa é ajustado, ao longo do tempo, de acordo com as necessidades e ritmos de progressão do paciente. Deste modo, a equipa de profissionais garante um acompanhamento personalizado a cada um dos seus pacientes.

O programa de treino prescrito pelo fisiatra numa clínica de fisioterapia normalmente incide nas seguintes atividades:

- Aplicações de calor e frio;
- Manipulação terapêutica das articulações e cinesioterapia;
- Eletroterapia, ultrassons, ondas curtas, laser, ondas de choque, magnetoterapia;
- Hidroterapia e balneoterapia;

- Massagens e drenagem linfática manual.

Para além destas atividades, são ainda prescritos exercícios físicos, com ou sem objetos.

Figura 3.1 – Reabilitação de ombro. (All Sports Physical Therapy)

As lesões a ser tratadas podem ser provenientes de acidentes rodoviários, de desporto ou de trabalho, má postura, movimentos repetitivos, doenças degenerativas, pré e pós-operatório de cirurgias, etc.

Assim, a MFR incide nas seguintes áreas:

- Perda de capacidade física e psicológica;
- Dores músculo-esqueléticas;
- Doenças do sistema nervoso;
- Perda da capacidade de andar e dependência de terceiros;
- Espasticidade;
- Disfunções a nível da bexiga e intestinais;
- Lesões neurológicas (AVC, traumatismos, etc).
- Disfagia (dificuldade em engolir);
- Problemas de comunicação;
- Disfunções sexuais;
- Problemas comportamentais, etc.

3.2 Reabilitação com biofeedback

Os exercícios prescritos num programa de reabilitação física podem ser bastante variados, pois vão depender de pessoa para pessoa e do tipo de problemática de cada um. O *biofeedback* pode ser aplicado a qualquer tipo de exercício, e a sua eficácia no processo de reabilitação tem vindo a ser estudada.

Quando uma pessoa executa um exercício de equilíbrio sem *biofeedback,* o seu corpo oscila mais, exibindo uma postura mais instável e incorreta. Por sua vez, se executar o mesmo exercício, mas com a componente de *biofeedback* a oscilação do corpo é menor. Em problemas de equilíbrio, o *biofeedback* contribui para melhorias significativas em termos de correção de postura [5].

Uma faixa etária muito propensa a problemas de saúde são os idosos. Estes são afetados essencialmente por dores músculo-esqueléticas e apresentam dificuldades nos movimentos. Na maioria dos casos são prescritos programas de reabilitação para melhoria destas condições. Os programas de reabilitação com *biofeedback* podem ser implementados na casa destas pessoas, sem grandes custos. O terapeuta escolhe o programa de reabilitação mais adequado à pessoa, e à distância consegue ter acesso aos resultados dos pacientes e adaptar sempre que necessário os exercícios consoante o progresso do tratamento. Assim, para pessoas mais idosas, que não têm maneira de se deslocar constantemente aos centros de fisioterapia, esta é uma boa alternativa pois conseguem executar os exercícios a partir de casa, evitar o stress passado em filas de espera e deslocações e ainda assim ter acompanhamento especializado [25].

Os exercícios executados com *feedback* visual são executados de forma mais correta do que quando executados sem *feedback* [26].

3.2.1 Biofeedback

Biofeedback consiste numa técnica de treino que permite que uma pessoa melhore a sua saúde através do controlo de determinados parâmetros do seu corpo: pressão arterial, tensão muscular e temperatura [27]. Estes parâmetros são medidos através de sensores colocados no corpo.

O *biofeedback* pode ser de diferentes tipos:

- Electromiografia (EMG) – Mede a tensão muscular.
- *Biofeedback* térmico – Mede a temperatura da pele.
- *Neufeedback* ou Eletroencefalografia – Mede a atividade sensorial.

O Microsoft Kinect consiste num sensor de movimentos de baixo custo que permite aceder a 25 articulações do corpo de uma pessoa. Estes pontos podem ser usados para conectar os ossos de uma pessoa e assim controlar a sua postura através dos seus movimentos [28]. Assim, o Microsoft Kinect pode ser usado como um sensor de *biofeedback*.

3.2.2 Microsoft Kinect

Com o avanço da deteção do movimento usando o Microsoft Kinect, surgiram diversas ideias de desenvolvimento de aplicações baseadas na captura de movimentos e na realidade virtual [29].

3.2.2.1 Hardware

O Kinect foi desenvolvido com o objetivo de ser utilizado para controlo de vídeo jogos da consola Xbox através do movimento do corpo e sem necessidade de um comando. A sua tecnologia foi desenvolvida pela PrimeSense [30] e encontra-se descrita em mais detalhe nas suas patentes [31], [32], [33] .

O Kinect v2.0 consiste na versão mais recente do sensor e foi a versão utilizada no desenvolvimento desta dissertação. Possui uma camara RGB que permite obter *streams* de 1080p (*Full HD Color*), que vistas na resolução do ecrã, apresentam uma ótima qualidade de imagem. É constituído também por uma camara de profundidade, que apresenta melhor fidelidade que a do Kinect v1. Deste modo, dispõe de uma deteção do corpo mais estável e de uma deteção de objetos mais pequenos três vezes melhor. Possui também um emissor de infravermelhos, que permite visão noturna, isto é, permite aceder a campos de visão cujas condições de luminosidade são mais fracas. Este emissor emite com uma frequência de

30 fps. Por fim, possui um conjunto de quatro microfones, que permitem não só capturar som e gravar áudio como também perceber a origem e a direção da onda de áudio.

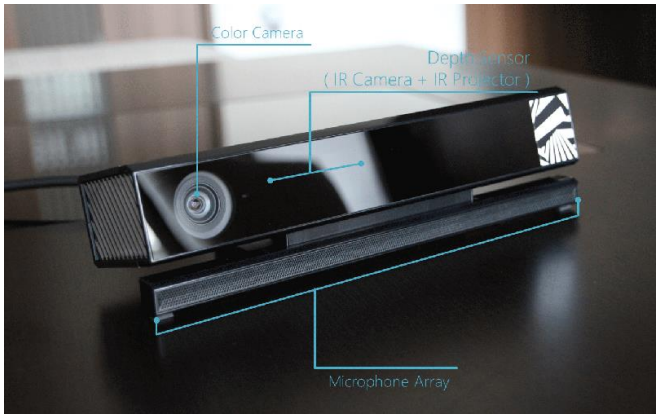

Figura 3.2 – Componentes do sensor Kinect v2.0. (Microsoft)

Este sensor veio revolucionar o mundo dos videojogos que até então necessitavam de comandos ou joysticks para controlar as funcionalidades dos jogos. O Kinect permite que o jogador interaja de forma dinâmica com o jogo, através dos movimentos do seu corpo sem necessitar do tradicional comando.

Rapidamente começaram a surgir interfaces criadas com base no uso deste sensor, passando não só a ser utilizado em jogos como também em trabalhos de desenvolvimento e de investigação.

O Kinect v2, comparativamente com o Kinect v1 (primeira versão lançada pela Microsoft), sofreu bastantes melhorias. A informação é processada mais rapidamente e com melhor precisão [34]. Na tabela seguinte, é possível comparar as duas versões e perceber as melhorias.

Tabela 3.1 – Tabela comparativa de especificações do Kinect v1.0 e Kinect v2.0

	Kinect v1.0	Kinect v2.0
Camara cor	640 x 480	1920 x 1080
Camara profundidade	320 x 240	512 x 424
Infravermelhos	No	512 x 424
Ângulo de visão	43º vertical 57º horizontal	60º vertical 70º horizontal

Limites físicos (da deteção do corpo)	[0.4 – 4.5] m	[0.5 – 4.5] m
Número máximo de corpos detetados	2	6
Número de articulações	20	25
USB Standard	2.0	3.0

3.2.2.2 Software

Embora a API criada para o novo sensor seja semelhante à do anterior, não é possível testar com o novo sensor, projetos criados com o Kinect SDK v1, pois foram mudados alguns nomes e metodologias das classes.

O Software Development Kit (SDK) 2.0 disponibiliza as ferramentas e APIs necessárias para desenvolver aplicações para Microsoft Windows que usem a tecnologia do sensor Kinect, possibilitando a criação de aplicações que usem gestos e/ou reconhecimento de voz.

3.2.2.2.1 Body Tracking

A deteção do corpo foi melhorada, as posições do corpo estão mais precisas e estáveis [35] o que possibilita criar aplicações mais realistas.Com este novo sensor é possível detetar até seis pessoas (esqueletos completos) enquanto que com o anterior apenas era possível detetar duas. É possível detetar o esqueleto de um utilizador quer ele esteja em pé ou sentado. São detetadas 25 articulações do corpo, enquanto que a versão anterior apenas detetava 20. No caso de a pessoa estar sentada, o Kinect consegue detetar 10 articulações.

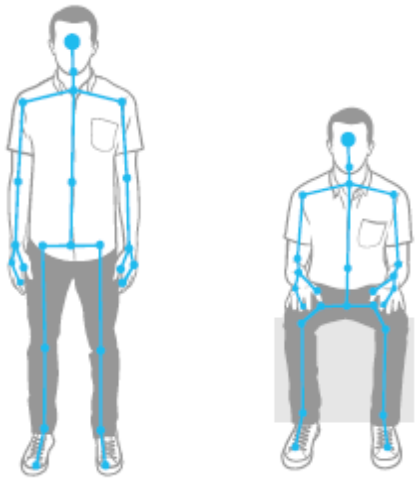

Figura 3.3 – Deteção do corpo inteiro e deteção do corpo sentado. (Human Interface Guidelines 2.0)

Uma nova funcionalidade implementada foi a deteção do estado das mãos do utilizador que se encontram descriminados na tabela 3.2.

Tabela 3.2 – Descrição dos diferentes estados detetados das mãos.

Membro	Descrição
Closed	A mão está fechada.
Lasso	A mão está “lassa” (nem aberta nem fechada).
NotTracked	Mão não encontrada.
Open	A mão está aberta.
Unknown	Não percebe o gesto que está a ser executado.

3.2.2.2.2 Face Tracking

A classe *FaceFrame* contém informação relativa à cara que está a ser detetada. A partir desta classe é possível compreender onde se encontra a cara, para onde está a olhar,

deteta algumas expressões faciais e permite também saber se o utilizador está a usar óculos ou não. Toda esta informação é obtida através do sensor de infravermelhos, o que significa que é possível obter informação relativa à cara de uma pessoa independentemente das condições luminosas em que se encontre e até mesmo em cenários com pouca iluminação. No entanto, a má iluminação pode ter um efeito negativo na deteção do olho esquerdo fechado (*LeftEyeClosed*) ou no olho direito fechado (*RightEyeClosed*) pois a deteção destes dois estados depende da componente de cor.

O Kinect v1 tem a capacidade de detetar 40 pontos da cara do utilizador. Esta nova versão, apresenta um rastreio do rosto bastante melhorado, apresentando uma resolução 20 vezes superior ao anterior o que permite criar uma rede de mais de 1000 pontos, possibilitando uma representação mais precisa do rosto da pessoa. Estes pontos podem ser obtidos no espaço de cor ou no infravermelho.

Para além de detetar estes pontos consegue reconhecer algumas expressões faciais (neutra ou de felicidade), atividades (olhos fechados, olhos abertos, boca fechada e boca aberta) e acessórios (óculos) descritas na tabela 3.3.

Tabela 3.3 – Propriedade da classe *Face* (MSDN)

Propriedade	Descrição
Happy	Quanto o utilizador sorri ou solta uma gargalhada é atribuída a expressão de felicidade. Não é considera uma emoção exata.
Engaged	Combina os resultados das propriedades *LookingAway* e *EyeClosed* para perceber o interesse do utilizador no conteúdo.
WearingGlass	Verifica se o utilizador usa óculos.
LeftEyeClosed	Verifica se o olho esquerdo está fechado.
RightEyeClosed	Verifica se o olho direito está fechado.
MouthOpen	Verifica se a boca está aberta.
MouthMoved	Deteta o movimento da cabeça do utilizador.
LookingAway	Deteta se o utilizador está a olhar para fora da zona do conteúdo.

3.2.2.2.3 Coordinate mapping

O Kinect integra sensores com diferentes resoluções: a camara de cor RGB apresenta uma resolução de 1920x1080 e os sensores de profundidade e de infravermelhos 512x424. Isto mostra que a camara RGB consegue ver uma maior área do que a camara de profundidade e infravermelhos. Deste modo, a informação visível na camara de profundidade, por exemplo, não pode ser projetada diretamente na *stream* de cor, pois existe necessidade de alinhar a informação.

No caso da deteção do corpo do utilizador, a recolha de informação das posições das articulações é feita através do sensor de profundidade. Estas posições são recolhidas no espaço 3D (X, Y, Z) pelo que estes pontos estão alinhados apenas com a *frame* de profundidade.

Por forma a contornar esta questão, o SDK possui uma classe chamada *CoordinateMapper* que serve para identificar se os pontos obtidos no espaço 3D correspondem aos pontos em 2D, da representação de cor ou de profundidade. Também permite a verificação inversa.

No desenvolvimento de aplicações, a deteção do corpo é crucial. Para permitir que o utilizador consiga compreender o que se está a passar com base nos movimentos do seu corpo, existe a necessidade de converter as coordenadas obtidas no espaço 3D em coordenadas 2D visíveis no ecrã (pixéis).

Antes de se usar esta classe é necessário compreender os diferentes espaços de coordenadas: camara (3D) e profundidade e cor (2D).

O sistema de coordenadas usado pelo Kinect é designado *Camera Space* e corresponde ao espaço 3D visto pelo sensor.

Figura 3.4 – Espaço de coordenadas da camara *(Camera Space*) do sistema. (MSDN Microsoft)

O *Depth Space* corresponde à representação 2D ocupada pela imagem em profundidade. Caso seja necessário obter o valor de infravermelhos correspondente a cada pixel da imagem em profundidade é bastante simples pois tanto a profundidade como os infravermelhos provêm do mesmo sensor do Kinect. Por sua vez, o *Color Space* corresponde à representação 2D ocupada pela imagem de cor.

4 Desenvolvimento

Neste capitulo, explicam-se detalhadamente os passos seguidos para o desenvolvimento do software implementado (figura 4.1). Começa-se por explicar o método de inicialização do Microsoft Kinect e de seguida a forma de aceder às *frames* vistas pelo dispositivo. De seguida, será feita uma abordagem de como é feita a deteção do corpo e quais os métodos de conversão necessários para desenhá-lo no ecrã. Por fim, é abordada toda a estrutura do programa e a forma como foram desenvolvidos cada um dos exercícios.

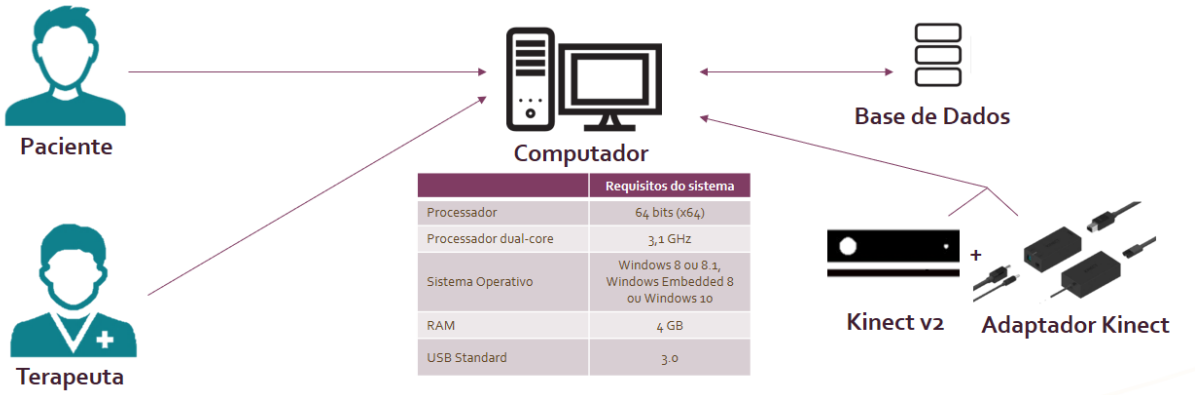

Figura 4.1 – Arquitetura da implementação do sistema.

4.1 *Inicialização do Kinect e obtenção de frames*

O primeiro passo consiste na inicialização do sensor, pelo que se cria um objeto do tipo classe *KinectSensor* responsável pela sua representação.

A conexão do hardware é estabelecida e seguidamente é necessário obter as *streams* da camara. Para isso, instancia-se um objeto do tipo classe *MultiSourceFrameReader* responsável pela leitura das múltiplas *frames*.

```
public KinectSensor kinectSensor { get; set; }
MultiSourceFrameReader msfReader = null;
```

Figura 4.2 – Declaração das classes *KinectSensor* e *MultiSourceFrameReader*

O Kinect v2.0 possui cinco tipos de *streams*: *Color, Depth, Infrared, Body, Audio.*

A maneira de inicializar o sensor e de obter as diferentes *frames* encontra-se exemplificada na figura 4.3.

```
public void InitializeKinect(object sender, EventArgs e)
   {
       // Initializing the sensor
       (App.Current as App).kinectSensor = KinectSensor.GetDefault();

       if ((App.Current as App).kinectSensor != null)
       {
          (App.Current as App).kinectSensor.Open();

          // Reading the streams
          msfReader = (App.Current as

          App).kinectSensor.OpenMultiSourceFrameReader(FrameSourceTypes.Color
          | FrameSourceTypes.Depth | FrameSourceTypes.Infrared |
          FrameSourceTypes.Body | FrameSourceTypes.Audio);

          msfReader.MultiSourceFrameArrived +=Reader_MultiSourceFrameArrived;
       }
       else
       {
          Console.WriteLine("Error! Kinnect not connected");
       }
   }
```

Figura 4.3 - Inicialização do Kinect e leitura de *frames*

Quando uma *frame* é detetada o evento *MultiSourceFrameArrived* dispara. Deste modo, foi criado o método *Reader_MultiSourceFrameArrived*, que permite aceder às *frames* detetadas. No método da figura 4.4, apenas constam as *frames* "*Color*" e "*Body*" porque foram as únicas usadas na implementação deste projeto.

```
        void Reader_MultiSourceFrameArrived(object sender,
MultiSourceFrameArrivedEventArgs e)
        {
            // Get a reference to the multiframe
            var reference = e.FrameReference.AcquireFrame();
            (App.Current as App).bodyFound = false;

            #region OpenColorFrame
            using (var frame = reference.ColorFrameReference.AcquireFrame())
            {
                if (frame != null)
                {
                    if (camMode == CameraSensor.Color)
                        camera.Source = frame.ToBitmap();
                }
            }
            #endregion

            #region OpenBodyFrame
              (…)
            #endregion
        }
```

Figura 4.4 - Método para aceder às *frames* de cor e do corpo.

Paralelamente, no ficheiro XAML que será abordado no subcapítulo 4.3 (estrutura do programa) foi declarado um elemento para mostrar as *frames* obtidas pelo Kinect. Este elemento suporta imagens do tipo *BitmapImage*.

```
<Image Name="camera" />
```

Figura 4.5 – Declaração de um objeto de imagem no XAML.

A resolução da camara de cor é de 1920x1080 e o que Kinect devolve quando obtém a *frame* de cor é um vetor de pixéis com valores RGB. Deste modo, para representar a imagem obtida pelo Kinect no ecrã do utilizador é necessário converte-la em *BitmapImage*.

Assim, criou-se um método, designado por *ToBitmap* (figura 4.6), que recebe uma imagem 3D (*ColorFrame*) e converte-a para o espaço 2D.

```
public static ImageSource ToBitmap(this ColorFrame frame)
{
    width = frame.FrameDescription.Width;
    height = frame.FrameDescription.Height;
    pixels = new byte[width * height * Constants.bytesPerPixel];

    if (frame.RawColorImageFormat == ColorImageFormat.Bgra)
    {
        frame.CopyRawFrameDataToArray(pixels);
    }
    else
    {
        frame.CopyConvertedFrameDataToArray(pixels,
         ColorImageFormat.Bgra);
    }

    stride = width * (Constants.format).BitsPerPixel / 8;

    return BitmapSource.Create(width, height, Constants.DPI,
      Constants.DPI, Constants.format, null, pixels, stride);
}
```

Figura 4.6 – Método de conversão das imagens obtidas da *ColorFrame* em *BitmapImage*.

Na implementação deste método, foram necessários os valores declarados num ficheiro de constantes, da figura 4.7. De seguida, segue-se uma explicação detalhada sobre cada variável.

```
public static class Constants
{
    // Default format
    public static readonly PixelFormat format = PixelFormats.Bgr32;
    // Bytes per pixel
    public static readonly int bytesPerPixel = (format.BitsPerPixel) / 8;
    // Kinect DPI
    public static readonly double DPI = 96.0;
}
```

Figura 4.7 – Classe *Constants*

- **format**: Define o formato de pixéis para as imagens. Neste caso, considerou-se o formato Bgr32 que é um sRGB (*standard* RGB) com 32 bits por pixel. Cada canal de cor (*blue*, *green* and *red*) aloca 8 bits por pixel.
- **bytesPerPixel:** Define o número de bytes que existe por cada pixel de cor.
- **DPI:** *Dots per inch* ou pontos por polegada. O Kinect v2 tem 96 DPI.

4.2 Deteção do corpo

Dentro do método Reader_MultiSourceFrameArrived consta ainda a forma de aceder à *Body Frame*.

```
IList<Body> bodies = null;

Body body = null;

bodyInformation newBody = new bodyInformation();
```

Figura 4.8 – Variáveis relativas à deteção do corpo.

```
using (var frame = reference.BodyFrameReference.AcquireFrame())
{
       if (frame != null)
       {
              bodies = new Body[frame.BodyFrameSource.BodyCount];
              frame.GetAndRefreshBodyData(bodies);

              foreach (var body in bodies)
              {
                     if (body != null)
                     {
                            if (body.IsTracked)
                                   (…) usar as coordenadas obtidas do corpo
                     }
              }
      }
}
```

Figura 4.9 – Método para aceder às *frames* de corpo.

Cada vez que o sensor deteta um novo corpo, é adicionado à lista dos corpos, sendo que no máximo o Kinect consegue detetar seis pessoas. Desta forma, o Kinect consegue

reconhecer as pessoas cada vez que aparecem no ecrã. As informações relativas às posições do corpo da pessoa são atualizadas a cada *frame*.

Por forma a aceder a estes valores foi criada a classe *bodyInformation,* a partir da qual se pode aceder a toda a informação necessária relativa ao corpo da pessoa e aos métodos necessários para manipulação e uso destes valores.

```
public class bodyInformation
{
        public Point head; (…)

        public void Refresh(Body newBody, Canvas canvasExercise)[…]

        public void DrawBody(Canvas canvas)[…]

        public void DrawLine(Canvas canvas, Point point1, Point point2)[…]

        public void DrawPoint(Canvas canvas, Point point)[…]

        public Point Convert3Dto2D(CameraSpacePoint cameraPoint, Canvas canvas)[…]

        public double calculateAngle(Point shoulder, Point elbow)[…]

        public double calculateLenght(Point p1, Point p2)[…]

}
```

Figura 4.10 – Classe *bodyInformation*

A classe *bodyInformation* é constituída pelos seguintes métodos:

1. **Refresh:** É o método responsável por inicializar as 25 articulações do corpo. As posições das articulações do corpo são obtidas no espaço 3D (*Camera Space*), pois são detetadas pela câmara de profundidade. Como nos interessa desenhar estas coordenadas por cima da imagem de cor, para dar uma melhor perceção aos utilizadores da posição correta do seu corpo no ecrã e dos seus movimentos, existe a necessidade de converter os pontos para a representação bidimensional (*Color Space*).

```
public void Refresh(Body newBody, Canvas canvasExercise)
{
        head = Convert3Dto2D(newBody.Joints[JointType.Head].Position,
canvasExercise);
        (…)
}
```

Figura 4.11 – Método *Refresh*: Inicialização das articulações.

2. **DrawBody:** Junta o desenho dos pontos ao desenho das linhas e assim desenha o esqueleto do corpo detetado.
3. **DrawLine:** Desenha uma linha entre dois pontos. Útil para representar os membros do esqueleto.
4. **DrawPoint:** Desenha um ponto. É usado para desenhar os pontos relativos às articulações.
5. **Convert3Dto2D:** É o método responsável por converter os pontos em 3D recebidos pela camera (*Camera Space*) em pontos 2D (*Color Space*).

```
public Point Convert3Dto2D(CameraSpacePoint cameraPoint, Canvas canvas)
{
        CameraSpacePoint joint3D = cameraPoint; // 3D point
        Point joint2D = new Point()
        ColorSpacePoint colorPosition = (App.Current as
           App).kinectSensor.CoordinateMapper.MapCameraPointToColorSpace(joint3D);

            // Ponto ajustado ao tamanho do canvas
            joint2D.X = float.IsInfinity(colorPosition.X) ? 0 : ((colorPosition.X *
               canvas.ActualWidth) / 1920);
            joint2D.Y = float.IsInfinity(colorPosition.Y) ? 0 : ((colorPosition.Y *
               canvas.ActualHeight) / 1080);

            return joint2D;
}
```

Figura 4.12 – Método *Convert3Dto2D.*

6. **calculateAngle:** Calcula o ângulo entre duas articulações, cujos valores já se encontram convertidos para a representação 2D.

```
public double calculateAngle(Point p1, Point p2)
{
	double catOp, catAdj, radians, angle;

	catOp = Math.Abs(p1.X - p2.X);
	catAdj = Math.Abs(p1.Y - p2.Y);
	radians = Math.Atan(catOp / catAdj);
	angle = (radians * 180) / Math.PI;

	if (p1.Y > p2.Y)
		angle = 180 - angle;

	if (angle < 0.5)
		angle = Math.Floor(angle);
	else
		angle = Math.Ceiling(angle);

	return angle;
}
```

Figura 4.13 – Método de cálculo de ângulos entre articulações.

7. **calculateLenght:** Calcula o comprimento entre duas articulações, cujos valores já se encontram convertidas para o espaço 2D.

```
public double calculateLenght(Point p1, Point p2)
{
	double lenght = Math.Abs(Math.Sqrt(Math.Pow((p1.X - p2.X), 2) +
	   Math.Pow((p1.Y - p2.Y), 2)));

	return lenght;
}
```

Figura 4.14 - Método de cálculo do comprimento entre membros

4.3 Estrutura do programa

Nesta seção começar-se-á pela apresentação da interface do programa e da forma como a aplicação está organizada para o utilizador, quer seja terapeuta ou paciente. De seguida, abordar-se-á a execução de cada exercício, de forma detalhada, nomeadamente como foram feitos os cálculos e os desenhos que permitem que o utilizador acompanhe o seu desempenho.

4.3.1 Interface gráfica

Quando se executa o programa é mostrado um menu inicial composto por dois ícones que pretendem identificar o tipo de utilizador: paciente ou terapeuta (figura 4.15).

Figura 4.15 – Menu inicial de identificação do utilizador.

O utilizador clica no ícone que o identifica e consoante ser paciente ou terapeuta aparecerá o seu menu correspondente tal como é demonstrado na figura 4.16.

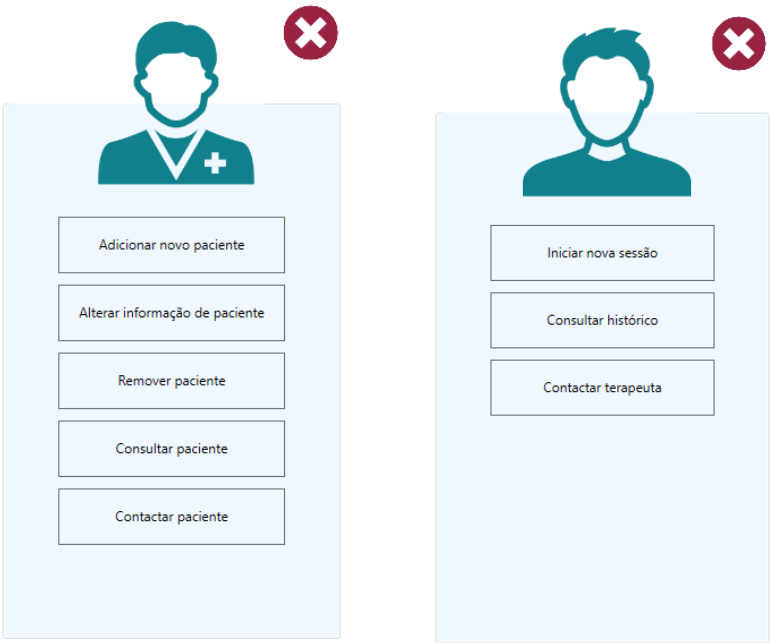

Figura 4.16 – Menu de opções do terapeuta e do paciente.

A gestão dos pacientes é feita através dos terapeutas, pelo que cada terapeuta poderá adicionar novos pacientes, alterar informação relativa a um determinado paciente, remover a ficha de inscrição do paciente, consultar informações acerca dos seus pacientes nomeadamente dados pessoais e os respetivos históricos das sessões de reabilitação física executada. Por fim, é disponibilizado um campo de contacto com o paciente que é feito através do e-mail.

No caso dos pacientes, é possível iniciar uma nova sessão de treino, consultar histórico das suas sessões efetuadas e à semelhança do terapeuta, possui um campo de contacto com o seu terapeuta via e-mail.

4.3.1.1 Interface do terapeuta

Quando o terapeuta seleciona a opção “Adicionar novo paciente” é aberto um questionário de dados relativos a informações pessoais do paciente (figura 4.16).

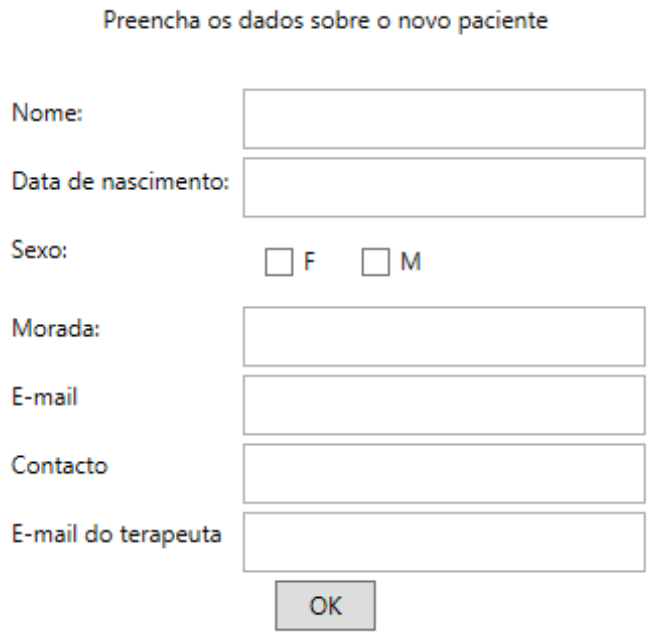

Figura 4.17 - Questionário de informações do paciente.

Após conclusão do preenchimento do formulário anterior, o terapeuta terá que selecionar qual ou quais os exercícios que o paciente terá que executar durante as suas sessões de

reabilitação. Estes exercícios são todos relacionados com problemáticas a nível dos ombros. São eles:

1. Abdução de ombro
2. Flexão de ombro
3. Abdução de ombros com halteres
4. Abdução de ombro com bola
5. Flexão com halteres

O terapeuta terá ainda que definir o número de séries por sessão, o número de repetições em cada série, o número total de sessões e no caso dos exercícios cujo parâmetro seja necessário o ângulo objetivo. O terapeuta ao selecionar um exercício é aberto um painel de informações relativas ao exercício selecionado tal como mostra a figura 4.18.

Figura 4.18 – Painel de exercícios.

Posto isto, o registo do paciente está completo e o paciente pode iniciar sessão através do identificador que lhe foi atribuído. O terapeuta através do identificador de registo do paci-

ente (figura 4.19) pode ainda alterar dados de determinado paciente, apagar um dado paciente da base de dados, consultar o seu histórico de sessões e contactá-lo via e-mail através do formulário da figura 4.20.

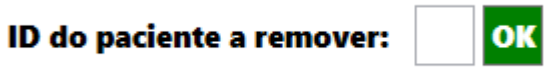

Figura 4.19 – Formulário de identificação do paciente.

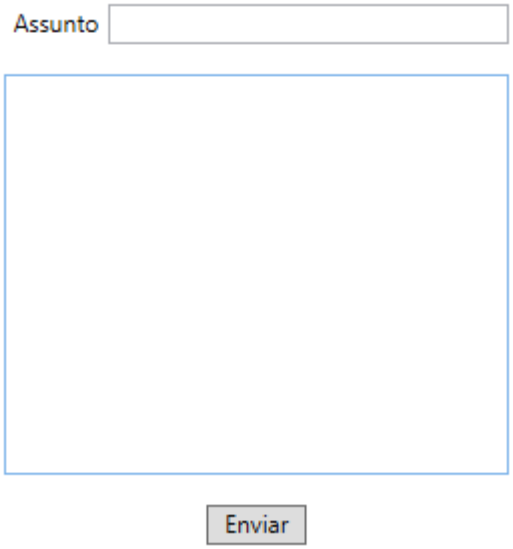

Figura 4.20 – Formulário de contacto entre os pacientes e os terapeutas.

De maneira a ser possível armazenar informação acerca dos pacientes, nomeadamente dados pessoais, que possibilitam o contacto entre paciente-terapeuta e a visualização do histórico de sessões dos pacientes, foi criada uma base de dados em SQL que será abordada em mais detalhe na seção 4.3.2.

4.3.1.2 Interface do paciente

Na interface do paciente a primeira opção consiste em “Iniciar nova sessão”, e é a partir desta opção que o paciente executa as sessões de reabilitação física prescritas pelo terapeuta. Na base de dados implementada, é guardada a informação do paciente e as sessões que o terapeuta prescreveu. Deste modo, o paciente quando inicia sessão, apenas tem que introduzir o seu identificador de utilizador e assim é iniciada uma nova janela de execução de exercícios já com todas as informações para execução preenchidas (figura 4.21).

Nesta janela é possível ver a *frame* de cor, o desenho do esqueleto (caso esteja selecionada a opção “mostrar articulações”), o desenho dos ângulos entre articulações, o valor dos ângulos entre articulações em tempo real, bem como outros detalhes relativos à execução dos exercícios.

Figura 4.21 - Painel de execução de exercícios.

Após conclusão do exercício, os dados relativos à sessão são guardados na base de dados implementada. De seguida, abordar-se-á detalhadamente a base de dados implementada para armazenamento e visualização de dados.

4.3.2 Base de dados

Por forma a ser possível armazenar um conjunto de dados relativos aos pacientes e às suas sessões de treino, foi implementada uma base de dados num servidor local.

Esta base de dados é constituída por três tabelas:

1. Pacientes: Nesta tabela constam os dados pessoais de todos os pacientes adicionados à base de dados e o e-mail do respetivo terapeuta. Estes dados são introduzidos pelo terapeuta através da sua interface gráfica.
2. Sessoes: Nesta tabela constam os dados relativos às sessões atribuídas pelo terapeuta a cada paciente e os dados relativos ao histórico das sessões realizadas.
3. Exercicios: O objetivo desta tabela é ter o registo de todos os exercícios que o programa oferece.

Por sua vez, estas tabelas encontram-se relacionadas como é possível visualizar no modelo relacional da figura 4.22.

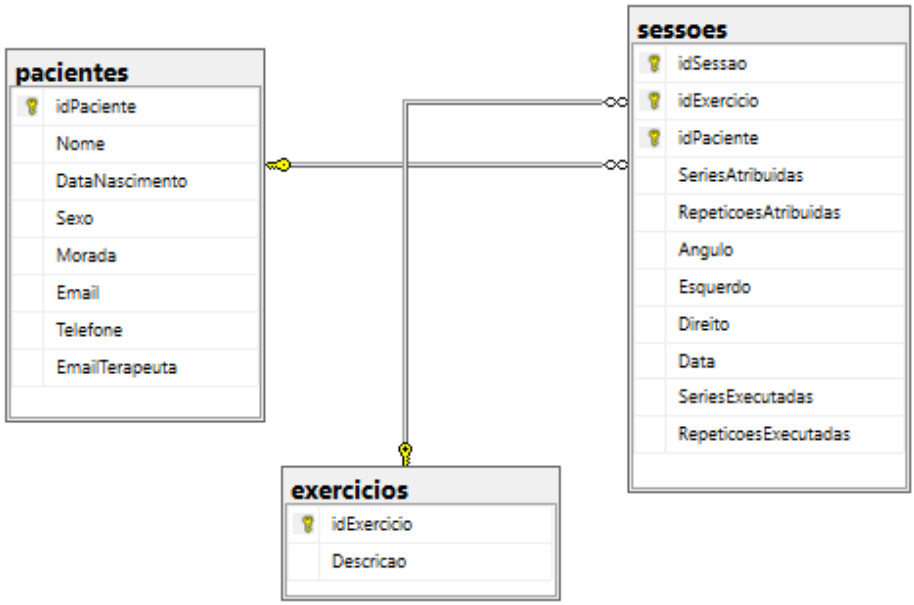

Figura 4.22 – Modelo relacional.

Através desta base de dados é possível consultar toda a informação que o utilizador quiser relativa a cada uma das três tabelas. De seguida, mostrar-se-á um exemplo das três-tabelas preenchidas com dados.

Tabela 4.1 – Tabela de pacientes.

idPaciente	Nome	DataNascimento	Sexo	Morada	Email	Telefone	EmailTerapeuta
1	Maria Joaquina	1965-10-12	F	Rua das malas	mariajoaquina@gmail.com	152369856	manulnacio@gmail.com
2	José Jesus	1945-03-01	M	Avenida 1º Maio	josejesus@gmail.com	152364568	manulnacio@gmail.com
3	Ana Maria	1990-02-16	F	Rua das carmélias	anamaria@gmail.com	154254789	manulnacio@gmail.com
4	Rosa Andrade	1978-11-26	F	Avenida 5 de Outubro	rosandrade@gmail.com	154236589	manulnacio@gmail.com
5	Luís Santos	1960-05-19	M	Rua das barcas	luisantos@gmail.com	152365896	manulnacio@gmail.com

Tabela 4.2 – Tabela de sessões.

	idSessao	idExercicio	idPaciente	SeriesAtribuidas	RepeticoesAtribuidas	Angulo	Esquerdo	Direito	Data	SeriesExecutadas	RepeticoesExecutadas
1	10	1	1	2	3	45	1	0	2016-12-15	2	0
2	11	1	1	2	3	45	1	0	NULL	NULL	NULL
3	12	1	1	2	3	45	1	0	NULL	NULL	NULL
4	13	1	1	2	3	45	1	0	NULL	NULL	NULL
5	14	1	1	2	3	45	1	0	NULL	NULL	NULL

Tabela 4.3 – Tabela de exercícios.

idExercicio	Descricao
1	Abdução
2	Flexão
3	Abdução com halteres
4	Abdução com bola
5	Flexão com halteres

Qualquer utilizador pode consultar o histórico dos pacientes, através do identificador do paciente. Deste modo, o terapeuta pode consultar o histórico das sessões realizadas de um dado paciente e acompanhar o seu progresso. Supondo que o terapeuta quer visualizar o desempenho da paciente “Maria Joaquina”, cujo id é 1. A pesquisa na base de dados é feita através do id da paciente.

```
SELECT p.Nome, s.*
  FROM pacientes AS p, sessoes AS s
  WHERE p.idPaciente = 1 AND s.idPaciente = 1 AND s.Data IS NOT NULL
```

Figura 4.23 – Comando para a pesquisa do histórico do paciente.

A partir deste comando é apresentado o histórico das sessões do paciente.

	Nome	idSessao	idExercicio	idPaciente	SeriesAtribuidas	RepeticoesAtribuidas	Angulo	Esquerdo	Direito	Data	Duracao	SeriesExecutadas	RepeticoesExecutadas
1	Maria Joaquina	15	1	67	2	3	45	1	0	2016-12-15	NULL	2	0

Figura 4.24 – Histórico de sessões da paciente “Maria Joaquina”.

De seguida, abordar-se-á a forma como foram desenvolvidos os exercícios.

4.3.3 Abdução de ombro

O exercício de “Abdução de ombro” pode ser executado apenas com um dos braços ou com os dois em simultâneo. É selecionado o ângulo (entre 0º e 180º de 10º em 10º) que é suposto o braço fazer em relação ao corpo. Assim, o utilizador irá executar o exercício, movimentando o braço até ao ângulo selecionado, e de seguida tem que voltar à posição inicial. Este exercício é suposto repetir o número de vezes e séries estabelecido.

São desenhadas no ecrã pequenas elipses, de 10º em 10º até ao ângulo escolhido.

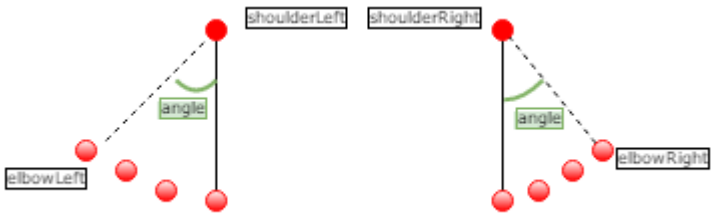

Figura 4.25 – Esquema do movimento de abdução para ambos os braços para um dado ângulo.

A posição de cada elipse, corresponde à posição onde é suposto o cotovelo estar (de 10º em 10º) até à posição final correspondente ao ângulo pretendido. Cada posição relativa ao braço esquerdo é calculada segundo o seguinte conjunto de equações que se segue.

$$\begin{cases} aux = 180 - i \ (em\ que\ 0 < i \leq angle\ e\ é\ incrementado\ de\ 10º) \\ elbow.X = shoulderLeft.X - [\cos(aux) * \tan(aux) * shoulderElbowLenght] \\ elbow.Y = shoulderLeft.Y - \dfrac{sen(aux) * shoulderElbowLenght}{\tan(aux)} \end{cases}$$

No caso do braço direito, o processo é semelhante.

$$\begin{cases} aux = 180 - i \ (em\ que\ 0 < i \leq angle\ e\ é\ incrementado\ de\ 10º) \\ elbow.X = shoulderLeft.X + [\cos(aux) * \tan(aux) * shoulderElbowLenght] \\ elbow.Y = shoulderLeft.Y - \dfrac{sen(aux) * shoulderElbowLenght}{\tan(aux)} \end{cases}$$

Considerando o braço direito, o ângulo desenhado entre o braço e o corpo é calculado a partir da seguinte forma:

$$\tan angle = \frac{cateto\ oposto}{cateto\ adjacente}$$

Sendo que:

$$\begin{cases} cateto\ oposto\ = |shoulder.X - elbow.X| \\ cateto\ adjacente\ = |shoulder.Y - elbow.Y| \end{cases}$$

Logo,

$$angle = \tan^{-1}\left[\left|\frac{shoulder.X - elbow.X}{shoulder.Y - elbow.Y}\right|\right] * \frac{180}{\pi}$$

4.3.4 Flexão de ombro

No exercício “Flexão de ombro” o objetivo é que o utilizador execute, com ambos os braços em simultâneo, um movimento de “semi-arco” ou seja de 0º a 180º e desfazê-lo. De maneira a ser possível desenhar o movimento dos braços e o ângulo feito entre o braço e o corpo em 2D pede-se inicialmente que o utilizador se coloque de lado para o sensor. Definiu-

se o lado direito da pessoa como lado principal. É feita uma verificação se a pessoa se colocou corretamente tal como foi pedido e em caso de sucesso pode começar a dar início ao movimento.

A maneira de calcular os ângulos é sempre a mesma, através do método *calculateAngle* da classe *bodyInformation*, apenas os parâmetros mudam. De acordo com os movimentos escolhem-se as articulações mais adequadas aos cálculos.

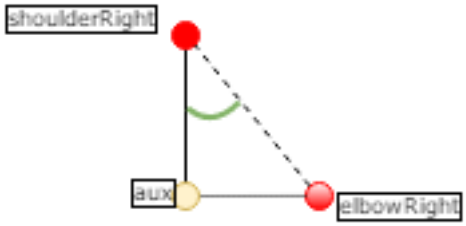

Figura 4.26 – Exemplo de articulações necessárias para o cálculo dos ângulos dos movimentos.

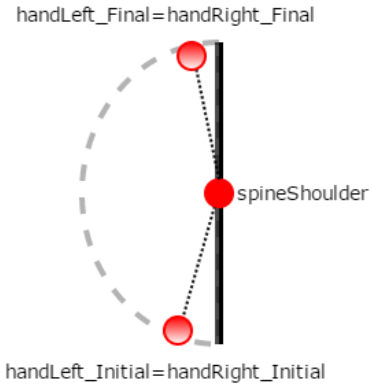

Figura 4.27 - Esquema do movimento de flexão.

4.3.5 Resistência de ombros

O exercício “Resistência de ombros” é um derivado do exercício 1 “Abdução” com alguns parâmetros pré-estabelecidos. Aqui é suposto o utilizador usar dois halteres (um em cada mão), embora não tenha sido desenvolvida uma forma de controlar se a pessoa realmente tem o objeto nas mãos. Neste exercício é suposto usar os dois braços em simultâneo

e executar o movimento de abdução de 0º até aos 90º (ângulo máximo), fazendo um ângulo reto entre os braços e o corpo na posição final do exercício.

A forma de calcular os ângulos entre os braços e o corpo é exatamente a mesma do exercício 1.

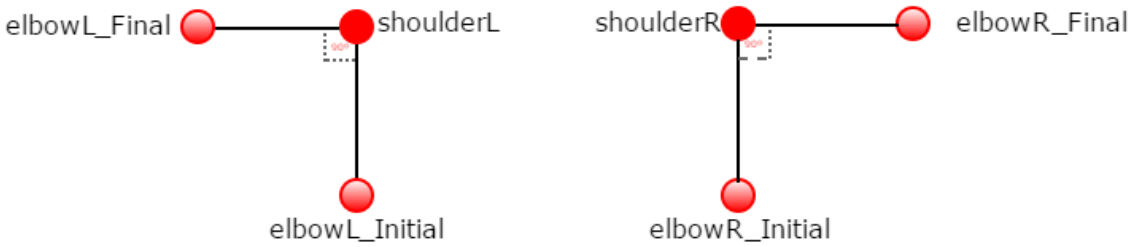

Figura 4.28 – Esquema do movimento do exercício resistência de ombros.

4.3.6 Abdução de ombro com bola

No exercício "Abdução de ombro com bola" o utilizador tem que fazer o movimento de abdução com uma bola de pilates para auxiliar o movimento. É pedido para selecionar o braço com que pretende executar o exercício e qual o ângulo a ser executado.

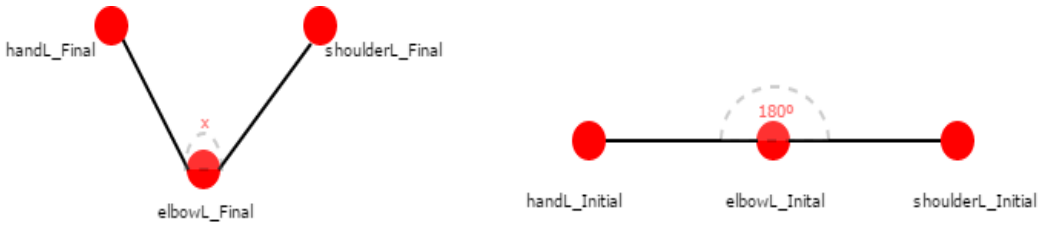

Figura 4.29 – Posição inicial e final do movimento de abdução com o braço esquerdo.

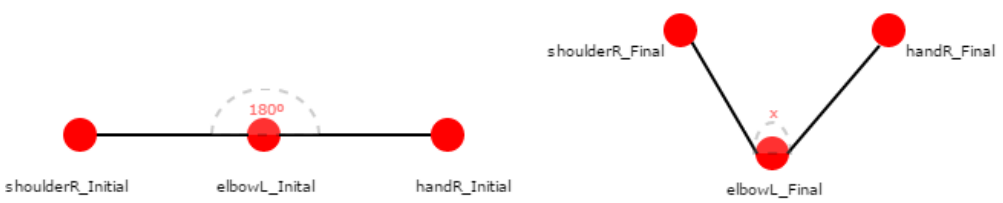

Figura 4.30 - Posição inicial e final do movimento de abdução com o braço esquerdo.

4.3.7 Flexão de ombros com halteres

O exercício “Flexão com halteres” é bastante semelhante ao exercício 2 “Flexão”. A única diferença é que os braços invés de executarem o movimento em simultâneo, será executado o mesmo movimento, mas com os braços alternados. É recomendado que o utilizador use um haltere em cada mão, embora este objeto não seja reconhecido nem controlado pelo sistema desenvolvido. O movimento será feito de lado para o sensor e assume-se que a pessoa se coloca com o lado direito do corpo voltado para o sensor.

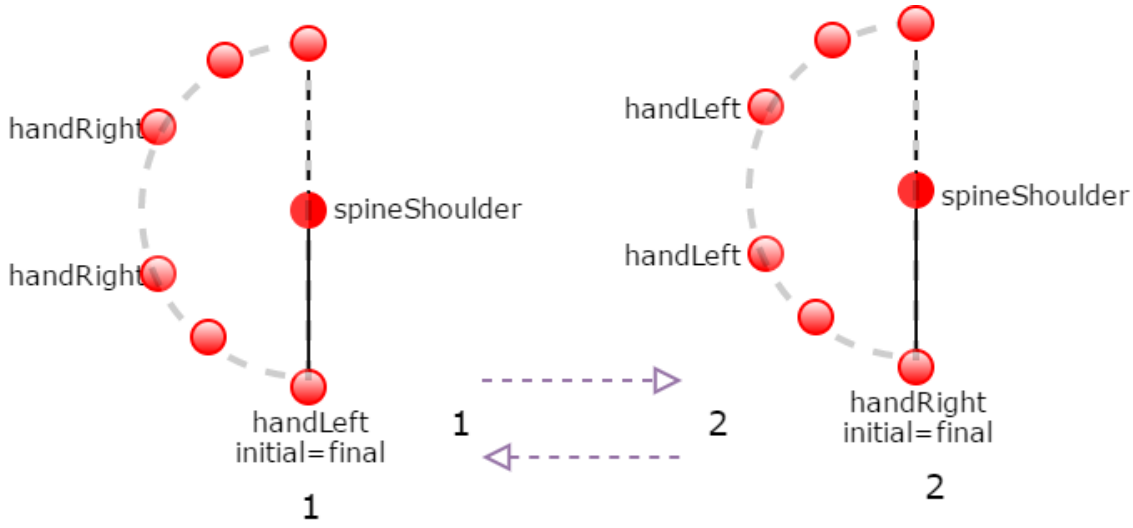

Figura 4.31 – Esquema do movimento alternado de flexão.

5 Validação de resultados

5.1 Deteção do corpo

Primeiramente, testou-se a deteção do corpo, pois a base da execução deste programa está na deteção correta do corpo do utilizador. Todos os testes foram feitos sobre a imagem de cor obtida pelo sensor. A deteção do corpo efetuada pelo Kinect é obtida através do sensor de profundidade pelo que as articulações detetadas pelo aparelho vão estar alinhadas apenas com a *frame* de profundidade. Assim, criou-se um método de conversão das coordenadas *cameraSpace* (X,Y,Z) em coordenadas *colorSpace* (X,Y) permitindo desenhos e cálculos na representação bi-dimensional. Na figura 5.1 é mostrado o desenho completo do esqueleto (25 articulações) sobreposto à imagem a cores obtida pelo sensor.

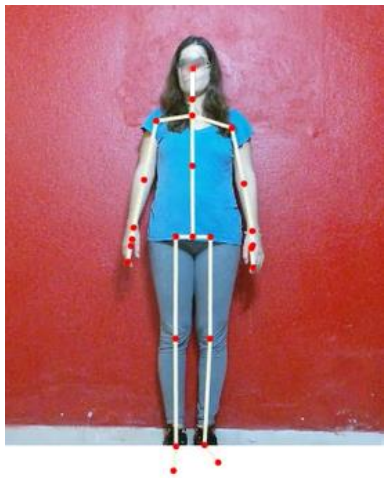

Figura 5.1 – Desenho das articulações e dos membros detetados pelo Kinect sobre a imagem de cor.

Através desta imagem, é possível verificar que o método de conversão de coordenadas 3D em 2D funcionou corretamente, pois as coordenadas das articulações encontram-se alinhadas com a imagem de cor.

De seguida, serão mostrados os testes efetuados para a resolução de cada exercício com e sem *biofeedback.* O objetivo é que cada exercício apresente uma componente de *biofeedback* para que o paciente compreenda visualmente o movimento que está a executar e perceba se está perto do objetivo proposto. Deste modo, todos os exercícios apresentados têm esta vertente. No entanto, para compreender a eficácia do *biofeedback* na execução dos exercícios, efetuaram-se testes sem *biofeedback*, ou seja, sem a ajuda dos desenhos dos movimentos no ecrã e sem os valores das amplitudes visíveis ao utilizador.

5.2 Abdução de ombro

Para realizar o exercício de abdução de ombro o paciente posiciona-se de frente para o sensor. Este exercício pode ser recomendado apenas para um dos braços ou para os dois em simultâneo.

Para efeitos de teste do movimento de abdução, selecionou-se o braço esquerdo e um ângulo de 40º.

5.2.1 Validação com *biofeedback*

Durante o movimento do braço é feito o desenho do ângulo entre o braço e o corpo do utilizador no ecrã. Este desenho é feito em três cores diferentes consoante a pessoa se esteja a aproximar (verde) ou afastar do ângulo pretendido (vermelho), sendo o amarelo o intermédio. Na tabela ao lado do ecrã é mostrado o valor estimado do ângulo. Quando a pessoa atinge o ângulo selecionado, desfaz o movimento e é incrementado o número de repetições. Assim que atinja o número de repetições estabelecido incrementa o número de séries. Atingido o número máximo de séries o objetivo proposto para este exercício está concluído.

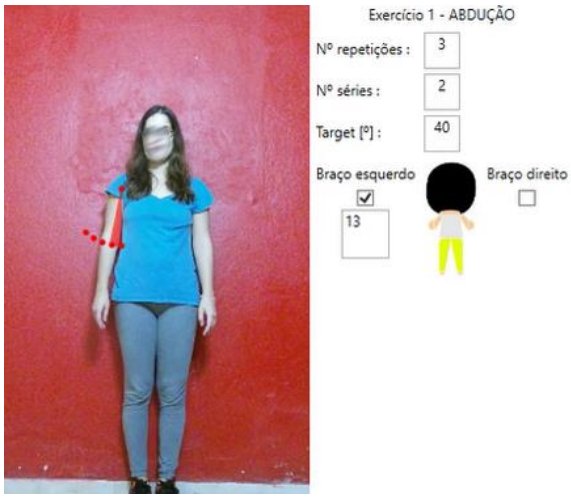

Figura 5.2 – Abdução com o braço esquerdo (Ângulo estimado = 13º)

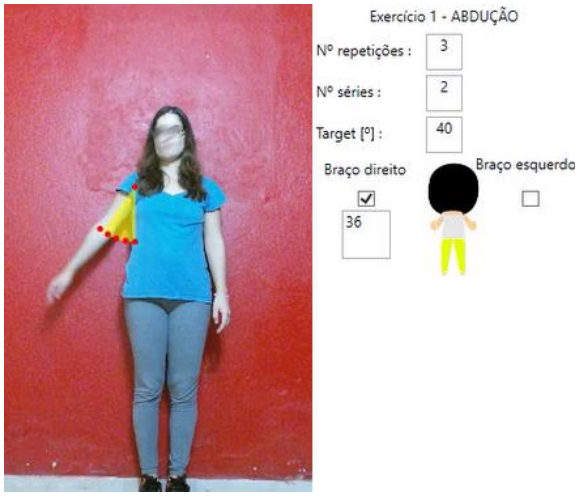

Figura 5.3 – Abdução com o braço esquerdo (Ângulo estimado = 36º)

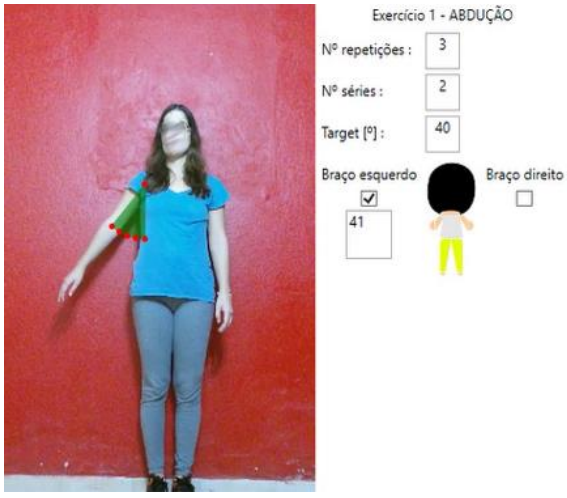

Figura 5.4 – Abdução com o braço esquerdo (Ângulo estimado = 41º)

5.2.2 Validação sem *biofeedback*

Neste teste, escolheram-se exatamente os mesmos requisitos que no teste anterior. Foram selecionadas três repetições, duas séries e pediu-se ao utilizador que fizesse o movimento de abdução até um ângulo de 40º. O utilizador executou o exercício até à amplitude que achava corresponder ao ângulo pretendido.

Figura 5.5 – Abdução com o braço esquerdo sem *biofeedback.*

Embora, não fosse visível ao utilizador, as amplitudes do movimento continuaram a ser calculadas. Na tabela 5.1, são mostradas as amplitudes atingidas em cada repetição. É importante referir que estes valores não foram mostrados ao utilizador durante a execução dos exercícios, para não haver influencia na execução do exercício.

Tabela 5.1 – Ângulos estimados em cada repetição durante a 1ª série do exercício de abdução.

	Ângulo estimado
1ª repetição	43º
2ª repetição	45º
3ª repetição	43º

Com base na análise da tabela 5.1, é de notar que as amplitudes ficaram bastante próximas do objetivo.

5.3 Flexão de ombro

Para testar este exercício não foi necessário estabelecer um ângulo nem definir um braço pois trata-se do movimento de flexão completo (de 0º a 180º) com ambos os braços. De maneira a facilitar o desenho do movimento no ecrã, pediu-se inicialmente que a pessoa se colocasse de lado para o sensor, estabelecendo o lado direito como o lado a ser colocado de frente para o sensor (figura 5.6). O exercício começa assim que é detetado o alinhamento do cotovelo direito com o tronco, o que significa que a pessoa se encontra de lado tal como fora pedido.

Figura 5.6 – Deteção do tronco e do braço direito

Esta primeira fase é comum em ambos os testes com e sem *biofeedback*.

5.3.1 Validação com *biofeedback*

Assim que a pessoa coloca o lado direito do seu corpo virado para o sensor e o seu cotovelo se encontra alinhado com o tronco, inicia-se o desenho do movimento dos braços no ecrã.

À medida que a pessoa movimenta os braços, o desenho do ângulo entre o tronco e os braços vai aparecendo no ecrã (com a respetiva cor relativa à proximidade do objetivo) e o valor do ângulo é mostrado no painel à direita. Assim que perfaz um ângulo aproximado de

180º (para algumas pessoas não é possível atingirem precisamente este ângulo), é desfeito o movimento e contabilizadas as repetições. Este processo é repetido o número de repetições e séries estabelecido.

Figura 5.7 - Flexão (Ângulo estimado = 13º)

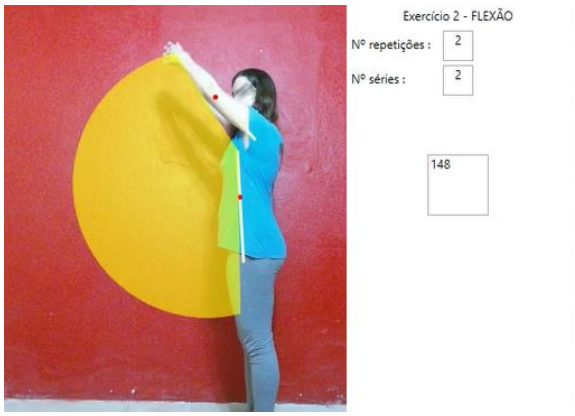

Figura 5.8 - Flexão (Ângulo estimado = 148º)

Figura 5.9 - Flexão (Ângulo estimado = 173º)

5.3.2 Validação sem *biofeedback*

Tal como na vertente com *biofeedback,* a execução do exercício só é iniciada quando a pessoa coloca o lado direito do seu corpo virado para o sensor e o seu cotovelo se encontra alinhado com o tronco. O objetivo deste teste é que o utilizador execute o movimento de flexão completo (de 0º a 180º) sem ter *feedback* visual da execução dos movimentos.

Figura 5.10 – Flexão sem *biofeedback.*

O utilizador executou o movimento de flexão sem o auxilio do desenho e do valor da amplitude descrita.

Sem que o utilizador visse no ecrã, as amplitudes do movimento foram calculadas para de seguida ser possível tirar conclusões. Na tabela 5.2, são mostradas as amplitudes atingidas em cada repetição. É importante referir que estes valores não foram mostrados ao utilizador durante a execução dos exercícios.

Tabela 5.2 - Ângulos estimados em cada repetição durante a 1ª série do exercício de flexão.

	Ângulo estimado
1ª repetição	177º
2ª repetição	179º

Analisando a tabela 5.2, observa-se que o utilizador executou o exercício dentro de amplitudes perto do que era desejado.

5.4 Resistência de ombros

Este exercício é semelhante ao exercício da abdução. A diferença é que não são selecionados os braços nem é estabelecido o ângulo pretendido pois o objetivo é usar os dois braços e executar o movimento de abdução até aos 90º. Este exercício é para ser efetuado com halteres, e embora os testes tenham sido feitos com eles não foi estabelecido um algoritmo de reconhecimento de objetos e por conseguinte não é feito o controlo do uso do objeto.

5.4.1 Validação com *biofeedback*

A validação com *biofeedback* foi feita através do desenho dos ângulos entre os braços e o corpo, por forma a que o utilizador conseguisse ter a perceção do ângulo que está a efetuar e à direita do ecrã aparece em tempo real o respetivo valor do ângulo.

Figura 5.11 – Resistência de ombros (Ângulo esquerdo = 7º; Ângulo direito = 10º)

Figura 5.12 - Resistência de ombros (Ângulo esquerdo = 84º; Ângulo direito = 83º)

Figura 5.13 - Resistência de ombros (Ângulo esquerdo = 90º; Ângulo direito = 94º)

Quando o utilizador atinge a amplitude pretendida (próxima de 90º) em ambos os braços, a repetição é contabilizada e pode desfazer o movimento, repetindo as vezes correspondentes às repetições e séries inicialmente definidas.

5.4.2 Validação sem *biofeedback*

Neste teste, o utilizador executou o movimento de abdução, com os dois braços em simultâneo, até onde na perceção dele corresponde a uma amplitude perto de 90º.

Figura 5.14 – Resistência de ombros sem *biofeedback*

Na tabela 5.3, são mostradas as amplitudes atingidas em cada repetição, que não estiveram visíveis ao utilizador durante a execução do exercício.

Tabela 5.3 - Ângulos estimados em cada repetição durante a 1ª série do exercício de abdução com halteres.

	Ângulo esquerdo estimado	Ângulo direito estimado
1ª repetição	99º	84º
2ª repetição	92º	80º

Observando a tabela, é de notar o desfasamento significativo entre os dois braços, no momento em que o utilizador achou que estava a executar uma amplitude de 90º. Isto mostra que, o utilizador estava a executar o exercício com má postura pois os braços estavam bastante desnivelados. Como não teve o *feedback* visual do movimento em tempo real, não corrigiu o movimento.

5.5 Abdução de ombro com bola

A abdução de ombro é feita sentada com o auxílio de uma bola de pilates. À semelhança do que foi referido para os halteres, como não foi desenvolvido um algoritmo de deteção de objetos, não existe maneira de controlar se o utilizador usa efetivamente a bola ou não. No entanto, os testes deste exercício foram executados com bola. Neste teste selecionou-se o braço esquerdo para efetuar o movimento. A posição inicial consiste no braço completamente esticado (180º) com a mão na bola. A bola serve como auxilio ao movimento pois o objetivo é que a pessoa aproxime o ombro do objeto. O ângulo não foi pré-estabelecido pois depende da capacidade de execução de pessoa para pessoa e é influenciado pelo uso da bola.

Novamente, foram realizados testes do exercício primeiramente com *biofeedback* e de seguida sem esta componente.

5.5.1 Validação com biofeedback

Neste teste obteve-se um ângulo de 111º. Na figura 5.16 é notado um desfasamento dos pontos das articulações o que pode estar relacionado com a bola.

Figura 5.15 – Abdução com bola (Posição inicial; Ângulo estimado = 179º)

Figura 5.16 – Abdução com bola (Posição final; Ângulo estimado = 111º)

5.5.2 Validação sem *biofeedback*

O exercício de abdução com bola foi repetido, mas desta vez sem o desenho dos pontos de referência ao movimento.

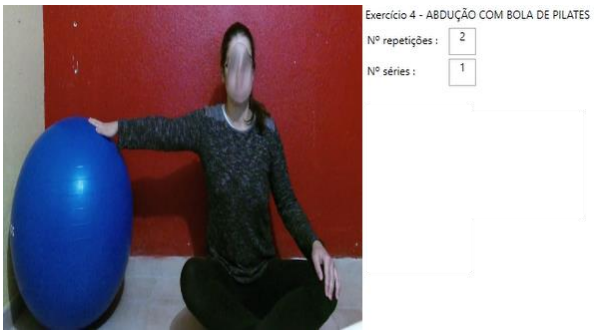

Figura 5.17 – Abdução com bola (Posição inicial; Sem *biofeedback*)

Figura 5.18 - Abdução com bola (Posição final; Sem *biofeedback*)

Na tabela 5.4 são mostradas as amplitudes iniciais e finais em cada repetição. Estes valores não estiveram visíveis ao utilizador.

Tabela 5.4 - Ângulos estimados em cada repetição durante a 1ª série do exercício de abdução

	Ângulo inicial estimado	Ângulo final estimado
1ª repetição	178	111
2ª repetição	177	109

A partir desta tabela não se consegue tirar conclusões concretas, pois a bola influencia a amplitude do movimento, e como não é definido inicialmente uma amplitude objetivo, o utilizador executa o movimento até à amplitude que lhe é confortável com o auxilio da bola.

5.6 Flexão de ombros com halteres

Neste exercício o objetivo é fazer o movimento completo de flexão (0º a 180º) com os braços alternados. É recomendado o uso de halteres, pelo que o teste foi feito com dois halteres. À semelhança do exercício de resistência de ombros, este exercício foi executado com o lado direito voltado para o sensor, de maneira a facilitar o desenho do movimento dos braços e assim dar uma melhor perceção dos movimentos ao utilizador. Como o exercício é feito alternadamente com os dois braços, iniciou-se com o braço direito.

5.6.1 Validação com *biofeedback*

Após o utilizador se colocar de lado para o sensor, é iniciada a execução do exercício. Quando o braço direito atinge próximo de 180º fica pronto a desfazer o movimento e recomeçar com o braço esquerdo o mesmo processo.

Figura 5.19 – Flexão com haltere (Braço direito; Ângulo = 115º)

Figura 5.20 – Flexão com haltere (Braço direito; Ângulo = 166º)

Figura 5.21 – Flexão com haltere (Braço esquerdo; Ângulo = 68º)

Figura 5.22 – Flexão com haltere (Braço esquerdo; Ângulo = 157º)

Como este exercício é executado de lado para o sensor e usa os dois braços alternadamente possui algumas limitações. O braço esquerdo, que não se encontra totalmente visível ao sensor, nem sempre é detetado, influenciando deste modo os cálculos da amplitude e por conseguinte os desenhos. Uma forma de contornar isto é a pessoa colocar-se ligeiramente na diagonal para o sensor, tal como é mostrado na figura 5.21.

5.6.2 Validação sem *biofeedback*

Após o utilizador colocar-se de lado para o sensor, o exercício foi iniciado. Iniciou-se pelo braço direito (figura 5.23) e assim que o utilizador achou que tinha chegado perto de 180º começou a desfazer o movimento e a iniciar o movimento com o braço oposto (figura 5.24).

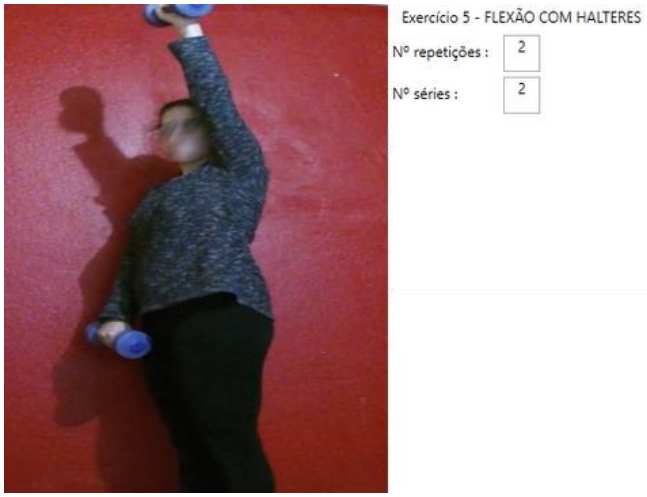

Figura 5.23 Flexão com haltere (Braço direito; Sem *biofeedback*)

Figura 5.24 - Flexão com haltere (Braço esquerdo; Sem *biofeedback*)

Na tabela 5.5, são mostradas as amplitudes atingidas em cada repetição durante a execução dos exercícios. Estes valores não estiveram acessíveis ao utilizador.

Tabela 5.5 - Ângulos estimados em cada repetição durante a 1ª série do exercício de flexão de ombros com halteres.

	Ângulo direito estimado	Ângulo esquerdo estimado
1ª repetição	173º	-
2ª repetição	175º	-

Tal como fora referenciado no teste deste exercício com *biofeedback,* o desenvolvimento deste exercício apresenta limitações. Quando o braço esquerdo está a executar o movimento e o utilizador não está colocado ligeiramente na diagonal para o sensor, a amplitude do movimento não é contabilizada. Esta dedução é observável através da análise da tabela 5.5.

5.7 Síntese de resultados

A análise dos resultados baseia-se nos resultados obtidos para cada exercício quando testados com e sem *biofeedback*. Embora este conjunto de dados resulte de uma amostra reduzida de testes, foi possível tirar algumas deduções. Tendo conhecimento dos valores correspondentes aos ângulos, em exercícios de abdução com apenas um braço e para um ângulo pré-estabelecido, o utilizador consegue facilmente executar o movimento pretendido até uma amplitude próxima da desejada. O mesmo se passou para o exercício de flexão.

No exercício de resistência de ombros, verificou-se que o utilizador consegue executar com mais rigor o movimento no teste com *biofeedback.* Os braços apareceram bastante desnivelados quando o teste foi executado sem *biofeedback*, o que mostra que o exercício estava a ser executado com uma postura incorreta, sem que o utilizador tivesse notado.

No caso do exercício de abdução de ombro com a bola não se conseguiram tirar conclusões concretas pois a execução do exercício depende do auxilio da bola e não é requerida nenhuma amplitude especifica.

Por fim, terminou-se este capitulo de validação de resultados com o teste do ultimo exercício desenvolvido, o da flexão de ombros com halteres. Este exercício apresenta limitações, pois exige que a pessoa se coloque numa posição especifica direcionada para o sensor, caso contrário o braço que está mais atrás não é corretamente detetado e os cálculos

das amplitudes não são efetuados. No teste com *biofeedback* o utilizador consegue ter essa perceção e deste modo corrigir a forma como está posicionado para o sensor. Em contra-partida, no teste sem *biofeedback* não existe esta perceção e o utilizador continua a realizar o exercício sem se aperceber que os cálculos não estão a ser efetuados corretamente.

6 Conclusões e trabalho futuro

Quando um individuo tem necessidade de fazer um programa de reabilitação física, pode acontecer ter que executar exercícios sozinho, em casa por exemplo, sem a monitorização de um terapeuta. Deste modo, surgem algumas dificuldades e desânimo por parte dos pacientes, pois sozinhos não conseguem controlar a execução dos exercícios e ter a perceção se os estão a executar de forma correta. É difícil uma pessoa sozinha compreender se a sua postura é a mais correta e se o movimento que está a executar corresponde exatamente ao que foi recomendado para recuperação.

Com o desenvolvimento de exercícios de reabilitação, com base na tecnologia do Microsoft Kinect e com uma vertente de *biofeedback,* o paciente consegue visualizar em tempo real os seus movimentos e deste modo perceber se está perto dos objetivos pretendidos. Esta importante componente, possibilita que o utilizador sozinho consiga corrigir más posturas, movimentos incorretos e melhorar a execução dos seus exercícios. Esta componente de *feedback* visual aumenta o entusiasmo do paciente e por conseguinte a eficácia do tratamento.

A implementação de bases de dados neste programa, possibilita o acesso ao histórico do paciente por parte do terapeuta e consequentemente aumenta o sentido de responsabilidade e compromisso do paciente ao saber que as suas sessões são monitorizadas.

Para uma compreensão mais exata acerca dos resultados obtidos no capitulo da validação, seria interessante arranjar uma amostra significativa de pessoas de ambos os sexos e de faixas etárias diferentes, com as mesmas problemáticas para realizar testes dos exercícios com e sem *biofeedback* em contexto clínico.

Por forma a incrementar ainda mais o entusiasmo da pessoa é possível introduzir inúmeras formas de divertimento enquanto a pessoa executa o seu plano de reabilitação. Uma possibilidade, já implementada em diversos softwares desta área, é o desenvolvimento de jogos (*serious games*) que visam exercitar os membros afetados do corpo, tendo sempre em conta o movimento mais adequado e seguro à reabilitação. Este tipo de jogos pode ser jogado individualmente ou em grupo, recorrendo à competição entre pacientes e a possíveis recompensas pelos programas de reabilitação serem concluídos com sucesso.

No caso dos exercícios com objetos, seria interessante aprofundar o seu reconhecimento através do Microsoft Kinect. Deste modo, o Kinect conseguiria reconhecer e identificar um conjunto de objetos necessários à execução dos exercícios e controlar o seu uso, tornando assim a execução do exercício ainda mais eficaz.

Seria também interessante, estudar o desempenho de cada paciente ao longo das sessões efetuadas. Este desempenho podia ser guardado com base no movimento efetuado ao longo das repetições. Para isto, o programa necessitaria de compreender melhor a execução do exercício e ter a capacidade de perceber se foi ou não executado da forma mais correta e assim, disponibilizar esta informação no histórico dos pacientes.

Por fim, seria interessante criar um editor de exercícios, isto é, permitir que o terapeuta introduzisse novos exercícios no sistema. O terapeuta ensinava ao sistema os movimentos de cada exercício, e deste modo seria possível escolher diferentes tipos de reabilitação para outros membros do corpo que não os ombros. Deste modo, o programa tornar-se-ia mais abrangente, não estando apenas limitado a problemáticas de ombros e aos cinco exercícios desenvolvidos neste projeto.

7 Bibliografia

[1] L. Lunenburger, G. Colombo, and R. Riener, "Biofeedback for robotic gait rehabilitation," *J. Neuroeng. Rehabil.*, vol. 4, no. 1, pp. 1–11, 2007.

[2] J. Nirme, A. Duff, and P. Verschure, "Adaptive rehabilitation gaming system: On-line individualization of stroke rehabilitation," in *33rd Annual International Conference of the IEEE EMBS Boston, Massachusetts USA,* 2011, pp. 6749–6752.

[3] B. Kim, M. Chun, L. Kim, and J. Park, "Effect of Virtual Reality on Cognition in Stroke Patients," *arm - Ann. Rehabil. Med.*, vol. 35, no. 4, pp. 450–459, 2011.

[4] Microsoft, "What's New in the October 2014 Kinect for Windows version 2.0 SDK." [Online]. Available: https://msdn.microsoft.com/en-us/library/dn782041.aspx. [Accessed: 19-Jul-2016].

[5] G. Kurillo, J. Han, S. Obdrzalek, P. Yan, R. Abresch, A. Nicorici, and R. Bajcsy, "Upper Extremity Reachable Workspace Evaluation With Kinect," *Stud. Health Technol. Inform.*, vol. 184, no. February, pp. 247–53, 2013.

[6] N. Kitsunezaki, E. Adachi, T. Masuda, and J. I. Mizusawa, "KINECT applications for the physical rehabilitation," *MeMeA 2013 - IEEE Int. Symp. Med. Meas. Appl. Proc.*, pp. 294–299, 2013.

[7] H. Zhou and H. Hu, "Human motion tracking for rehabilitation-A survey," *Biomedical Signal Processing and Control*, vol. 3, no. 1. pp. 1–18, 2008.

[8] R. Begg and M. Palaniswami, *Computational Intelligence for Movement Sciences: Neural Networks and Other Emerging Techniques*. United States of America and United Kingdom: Idea Group Publishing, 2006.

[9] T. Dutta, "Evaluation of the Kinect Ô sensor for 3-D kinematic measurement in the workplace," *Appl. Ergon.*, vol. 43, no. 4, pp. 645–649, 2012.

[10] "Motion Capture Systems | VICON." [Online]. Available: https://www.vicon.com/. Accessed: 08/09/2016

[11] "Virtualrehab | Virtual Rehabilitation System." [Online]. Available: http://www.virtualrehab.info/. Accessed: 10/09/2016.

[12] "KINECT REHABILITATION with Biofeedback | News, Healthcare, Research." [Online]. Available: http://www.virtual-reality-rehabilitation.com/ Accessed acesso: 10/09/2016.

[13] Brontes Processing, "Features of SeeMe," *SeeMe Rehabil. By Fun*, pp. 1–2, 2015.

[14] "Cadet-designed system aids stroke victim recovery > U.S. Air Force > Article Display." [Online]. Available: http://www.af.mil/News/ArticleDisplay/tabid/223/Article/467823/cadet-designed-system-aids-stroke-victim-recovery.aspx. Accessed: 24/08/2016

[15] "Reflexion Health." [Online]. Available: http://reflexionhealth.com/ Accessed: 24/08/2016.

[16] "(EN) Kinect rehabilitation software - Jintronix." [Online]. Available: http://www.jintronix.com/pt/. Accessed: 23/08/2016.

[17] "MIRA - Product." [Online]. Available: http://www.mirarehab.com/product. Accessed: 23/08/2016.

[18] A. Alin, A. Cantea, A. Alu, C. Mihaiu, and D. Suciu, "Mira –upper limb rehabilitation system using microsoft kinect" vol. LVI, no. 4, pp. 63–74, 2011.

[19] Y. Chang, S. Chen, and J. Huang, "Kinerehab: A Kinect-based System for Physical Rehabilitation: A Pilot Study for Young Adults with Motor Disabilities," *Res. Dev. Disabil.*, vol. 32, no. 6, pp. 319–320, 2011.

[20] H. Hondori and M. Khademi, "A Review on Technical and Clinical Impact of Microsoft Kinect on Physical Therapy and Rehabilitation," *J. Med. Eng.*, vol. 2014, pp. 1–16, 2014.

[21] B. Galna et al, "Retraining function in people with Parkinson's disease using the Microsoft kinect: game design and pilot testing.," *J Neuroeng Rehabil*, p. 60, 2014.

[22] B. Homer, C. Kinzer, J. Plass, S. Letourneau, D. Hoffman, M. Bromley, E. Hayward, S. Turkay, and Y. Kornak, "Moved to learn: The effects of interactivity in a Kinect-based literacy game for beginning readers," *Comput. Educ.*, vol. 74, pp. 37–49, 2014.

[23] "Medicina Física e Reabilitação - Apresentação | CUF." [Online]. Available: https://www.saudecuf.pt/areas-clinicas/medicina-fisica-e-reabilitacao. Accessed: 15/08/2016.

[24] M. Afzal, M. Oh, and J. Yoon, "Development of a multimodal biofeedback system for balance training," *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM*, vol. 2015–Augus, pp. 658–663, 2015.

[25] S. Bragaglia, S. Di Monte, and P. Mello, "A distributed system using MS kinect and event calculus for adaptive physiotherapist rehabilitation," *Proc. - 2014 8th Int. Conf. Complex, Intell. Softw. Intensive Syst. CISIS 2014*, pp. 531–538, 2014.

[26] M. Barandas, H. Gamboa, and J. M. Fonseca, "A Real Time Biofeedback System Using Visual User Interface for Physical Rehabilitation," *Procedia Manuf.*, vol. 3, no. Ahfe, pp. 823–828, 2015.

[27] "Biofeedback | University of Maryland Medical Center." [Online]. Available: http://umm.edu/health/medical/altmed/treatment/biofeedback. Accessed: 24/08/2016.

[28] N. Borghese, R. Mainetti, M. Pirovano, and P. Lanzi, "An intelligent game engine for the at-home rehabilitation of stroke patients," *SeGAH 2013 - IEEE 2nd Int. Conf. Serious Games Appl. Heal. B. Proc.*, 2013.

[29] H. Shum, E. Ho, Y. Jiang, and S. Takagi, "Real-time posture reconstruction for Microsoft Kinect," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1357–1369, 2013.

[30] "PrimeSense-MIT Technology Review." [Online]. Available: http://www2.technologyreview.com/tr50/primesense/. Accessed: 10/09/2016.

[31] E. Spektor et Al., "Integrated Processor for 3D Mapping. Prime Sense, Tel Aviv," vol. 1, no. 19, 2010.

[32] B. Freedman et Al., "Depth Mapping Using Projected Patterns, Prime Sense Ltd., Tel Aviv.," vol. 2, no. 12, 2012.

[33] Shpunt, "Depth Mapping Using Multi-Beam Illumination. Prime Sense LTD, Tel Aviv.," vol. 1, no. 19, pp. 0–2, 2010.

[34] "Kinect for Windows Programming Guide." [Online]. Available: https://msdn.microsoft.com/en-us/library/dn782037.aspx. Accessed: 03/09/2016.

[35] Vangos Pterneas, "Kinect for Windows version 2: overview | Vangos Pterneas," 2014. [Online]. Available: http://pterneas.com/2014/02/08/kinect-for-windows-version-2-overview/. Accessed: 03/09/2016.

Anexos

ExercisePage: É a classe principal do programa. É através dela que se obtêm as *frames* do Microsoft Kinect e onde são feitos todos os cálculos, desenhos e conteúdos relativos à execução de cada exercício.

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using PhysioExercises.Úteis;
using System.Windows.Threading;

namespace PhysioExercises
{
    public partial class ExercisesPage : Page
    {
        #region Variables
        MultiSourceFrameReader msfReader = null;  // Camera streams
        IList<Body> bodies = null; // List of bodies
        DispatcherTimer timer; //  Timer
        bodyInformation newBody = new bodyInformation();
        CameraSensor camMode = CameraSensor.Color;
        int time = 5;
        bool startExercise = false;
        int s = 0, r = 0;
        bool success = false, successL = false, successR = false, repeat = false, re-
set = false, begin = false, finish = false;
        int repetitions,series, target, aux;
        double finalLeftY, finalRightY;
        double shoulderElbowLenght_L, shoulderElbowLenght_R, spineShoulder-
HandLenght_L, spineShoulderHandLenght_R, shoulderHandLenght_R, shoulderWristLenght_L,
shoulderWristLenght_R;
        double angleL, angleR;
        double angle;
        string path;
        Point finalElbow = new Point();
        double degToRad = Math.PI / 180;
        int exerciseNumber;
        bool left = false, right = false;

        Path pathAngleLeft = new Path
        {
            Stroke = new SolidColorBrush(Colors.Red),
            Opacity = 0.60,
            Fill = new SolidColorBrush(Colors.Red)
        };
```

```
        Path pathAngleRight = new Path
        {
            Stroke = new SolidColorBrush(Colors.Red),
            Opacity = 0.60,
            Fill = new SolidColorBrush(Colors.Red)
        };

        Path pathFlexion = new Path
        {
            Stroke = new SolidColorBrush(Colors.Red),
            Opacity = 0.60,
            Fill = new SolidColorBrush(Colors.Red)
        };

        public class trigClass
        {
            public double sen, cos, tan;
        };

        Dictionary<double, trigClass> trig = new Dictionary<double, trigClass>();

        #endregion

        #region InitializePage
        public ExercisesPage()
        {
            InitializeComponent();

            exerciseNumber = (App.Current as App).exNumber;

            timer = new DispatcherTimer();
            timer.Interval = new TimeSpan(0, 0, 1); // 5 seconds
            timer.Tick += new EventHandler(timerTick);

            for (int i = 0; i <= 180; i += 10)
            {
                trig[i] = new trigClass() { sen = Math.Sin(i * degToRad), cos =
Math.Cos(i * degToRad), tan = Math.Tan(i * degToRad) };
            }

            switch (exerciseNumber)
            {
                case 1:
                    nameExercise.Content = "ABDUÇÃO";
                    target = (App.Current as App).targetAngle;
                    objetiveAngle.Text = target.ToString() + "º";
                    angleLabel.Visibility = Visibility.Hidden;
                    angleValue.Visibility = Visibility.Hidden;
                    leftAngleLabel.Visibility = Visibility.Visible;
                    leftAngleValue.Visibility = Visibility.Visible;
                    rightAngleLabel.Visibility = Visibility.Visible;
                    rightAngleValue.Visibility = Visibility.Visible;
                    break;
                case 2:
                    nameExercise.Content = "FLEXÃO";
                    target = 180;
                    objetiveAngle.Text = "180º";
                    break;
                case 3:
                    nameExercise.Content = "RESISTÊNCIA";
                    target = 90;
```

```
                    objetiveAngle.Text = "90º";
                    left = true;
                    right = true;
                    angleLabel.Visibility = Visibility.Hidden;
                    angleValue.Visibility = Visibility.Hidden;
                    leftAngleLabel.Visibility = Visibility.Visible;
                    leftAngleValue.Visibility = Visibility.Visible;
                    rightAngleLabel.Visibility = Visibility.Visible;
                    rightAngleValue.Visibility = Visibility.Visible;
                    break;
                case 4:
                    nameExercise.Content = "ABDUÇÃO COM BOLA";
                    break;
                case 5:
                    nameExercise.Content = "FLEXÃO COM HALTERES";
                    target = 180;
                    objetiveAngle.Text = "180º";
                    angleLabel.Visibility = Visibility.Hidden;
                    angleValue.Visibility = Visibility.Hidden;
                    leftAngleLabel.Visibility = Visibility.Visible;
                    leftAngleValue.Visibility = Visibility.Visible;
                    rightAngleLabel.Visibility = Visibility.Visible;
                    rightAngleValue.Visibility = Visibility.Visible;
                    break;
            }

            if ((exerciseNumber == 1) || (exerciseNumber == 4))
            {
                left = (App.Current as App).left;
                right = (App.Current as App).right;
            }

            repetitions = (App.Current as App).repetitions;
            series = (App.Current as App).series;
            repMax.Content = "/" + ((App.Current as App).repetitions).ToString();
            serMax.Content = "/" + ((App.Current as App).series).ToString();
            userName.Content = ((App.Current as App).name).ToString();
        }
        #endregion

        #region InitializeKinect
        public void InitializeKinect(object sender, EventArgs e)
        {
            // Initializing the sensor
            (App.Current as App).kinectSensor = KinectSensor.GetDefault();

            if ((App.Current as App).kinectSensor != null)
            {
                (App.Current as App).kinectSensor.Open();

                // Reading the streams
                msfReader = (App.Current as App).kinectSensor.OpenMulti-
SourceFrameReader(FrameSourceTypes.Color | FrameSourceTypes.Depth | FrameSource-
Types.Infrared | FrameSourceTypes.Body);
                msfReader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;
            }
            else
            {
                Console.WriteLine("Error! Kinnect not connected");
            }
        }
```

```
        #endregion

        #region Window_Closed
        private void Window_Closed(object sender, EventArgs e)
        {
            if (msfReader != null)
            {
                msfReader.Dispose();
            }

            if ((App.Current as App).kinectSensor != null)
            {
                (App.Current as App).kinectSensor.Close();
            }
        }
        #endregion

        #region ReaderFrame
        void Reader_MultiSourceFrameArrived(object sender, MultiSourceFrameArrive-
dEventArgs e)
        {
            // Get a reference to the multiframe
            var reference = e.FrameReference.AcquireFrame();

            #region OpenColorFrame
            using (var frame = reference.ColorFrameReference.AcquireFrame())
            {
                if (frame != null)
                {
                    if (camMode == CameraSensor.Color)
                        camera.Source = frame.ToBitmap();
                }
            }
            #endregion

            #region OpenBodyFrame
            // Each body consists of 25 joints
            // The sensor provides us with the position X,Y,Z and the rotation for
each body
            using (var frame = reference.BodyFrameReference.AcquireFrame())
            {
                if (frame != null)
                {
                    bodies = new Body[frame.BodyFrameSource.BodyCount];

                    frame.GetAndRefreshBodyData(bodies);

                    foreach (var body in bodies)
                    {
                        if (body != null)
                        {
                            if (body.IsTracked)
                            {
                                newBody.Refresh(body, canvasExercise);

                                exercises(newBody);
                            }
                        }
                    }
                }
            }
```

```
            #endregion
        }
        #endregion

        #region Exercises
        /******************** EXERCISES ********************/
        void exercises(bodyInformation newBody)
        {
            dockElements.Visibility = Visibility.Visible;

            if (finish == false)
            {
                if ((App.Current as App).feedback == true)
                {
                    canvasElements.Visibility = Visibility.Visible;
                }
                else
                {
                    canvasExercise.Visibility = Visibility.Visible;
                }
            }

            if (startExercise == false)
            {
                countdownClock.Visibility = Visibility.Visible;
                timer.Start();
            }

            if (startExercise == true)
            {
                if (drawBody.IsChecked == true)
                {
                    canvasExercise.Children.Clear(); // Clear canvas
                    newBody.DrawBody(canvasExercise);
                }
                Console.WriteLine("start true");

                #region exerciseNumber
                switch (exerciseNumber)
                {
                    #region EX.1-ABDUCTION
                    case 1:
                        string arm;

                        canvasExercise.Children.Clear();

                        if (finish == false)
                        {
                            instructions.Text = string.Format("Execute o movimento de
abdução pausadamente, repetindo {0} vezes num conjunto de {1} séries", repetitions,
series);
                        }

                        #region LeftArm
                        if ((left == true))
                        {
                            arm = "l";
                            drawPathLeft();
                            newBody.DrawPoint(canvasExercise, newBody.shoulderLeft);
```

```
                            drawAnglePoints(newBody.shoulderLeft, shoulderEl-
bowLenght_L, target, arm);
                        }
                        #endregion

                        #region RightArm
                        if ((right == true))
                        {
                            arm = "r";
                            drawPathRight();
                            newBody.DrawPoint(canvasExercise, newBody.shoulderRight);
                            drawAnglePoints(newBody.shoulderRight, shoulderEl-
bowLenght_R, target, arm);
                        }
                        #endregion

                        count(); // contagem das repetições e séries

                        break;
                    #endregion

                    #region EX.2-FLEXION
                    case 2:
                        canvasExercise.Children.Clear();

                        newBody.DrawLine(canvasExercise, newBody.spineShoulder, new-
Body.spineBase);
                        newBody.DrawLine(canvasExercise, newBody.shoulderRight, new-
Body.handRight);
                        newBody.DrawPoint(canvasExercise, newBody.spineMid);
                        newBody.DrawPoint(canvasExercise, newBody.elbowRight);

                        if (finish == false)
                        {
                            instructions.Text = string.Format("Coloque-se de lado para
a camera (ombro direito à frente), e alinhe o seu ombro direito com a linha do tronco
que aparece no ecra");

                            if ((newBody.elbowRight.X < newBody.spineMid.X + 10) &&
(newBody.elbowRight.X > newBody.spineMid.X - 10)) // cotovelo direito praticamente
alinhado com spineMid
                            {
                                Console.WriteLine("Cotovelo na posiçao correta!");
                                begin = true; // o exercicio pode começar
                            }

                            if (begin == true)
                            {
                                instructions.Text = string.Format("Execute o movimento
de flexão pausadamente, repetindo {0} vezes num conjunto de {1} séries", repetitions,
series);
                                drawPathFlexion();
                                count();
                            }
                        }
                        break;
                    #endregion

                    #region EX.3-SHOULDER RESISTANCE (with haltere)
                    case 3:
                        canvasExercise.Children.Clear();
```

```
                        if (finish == false)
                        {
                            instructions.Text = string.Format("Execute o movimento de
abdução até 90º pausadamente, repetindo {0} vezes num conjunto de {1} séries", repe-
titions, series);
                        }

                        /* Left arm */
                        drawPathLeft();
                        newBody.DrawPoint(canvasExercise, newBody.shoulderLeft);
                        drawAnglePoints(newBody.shoulderLeft, shoulderElbowLenght_L,
target, "l");

                        /* Right arm */
                        drawPathRight();
                        newBody.DrawPoint(canvasExercise, newBody.shoulderRight);
                        drawAnglePoints(newBody.shoulderRight, shoulderElbowLenght_R,
target, "r");

                        count();

                        break;
                    #endregion

                    #region EX.4 - SHOULDER ABDUCTION WITH BALL
                    case 4:
                        if (left == true)
                        {
                            shoulderWristLenght_L = newBody.calculateLenght(new-
Body.shoulderLeft, newBody.wristLeft);
                            angle = newBody.calculateAngle(newBody.shoulderLeft, new-
Body.elbowLeft);
                            angle += newBody.calculateAngle(newBody.wristLeft, new-
Body.elbowLeft);

                            if (angle > 100 && angle < 130)
                            {
                                success = true;
                            }
                            angleValue.Text = angle.ToString();
                            canvasExercise.Children.Clear();
                            newBody.DrawPoint(canvasExercise, newBody.handLeft);
                            newBody.DrawPoint(canvasExercise, newBody.elbowLeft);
                            newBody.DrawPoint(canvasExercise, newBody.shoulderLeft);

                            count();
                        }
                        if (right == true)
                        {
                            shoulderWristLenght_R = newBody.calculateLenght(new-
Body.shoulderRight, newBody.wristRight);
                            angle = newBody.calculateAngle(newBody.shoulderRight, new-
Body.elbowRight);
                            angle += newBody.calculateAngle(newBody.wristRight, new-
Body.elbowRight);
                            angleValue.Text = angle.ToString();
                            canvasExercise.Children.Clear();
                            newBody.DrawPoint(canvasExercise, newBody.handRight);
                            newBody.DrawPoint(canvasExercise, newBody.elbowRight);
                            newBody.DrawPoint(canvasExercise, newBody.shoulderRight);
```

```
                        count();
                    }
                    break;
                #endregion

                #region  EX.5 - SHOULDER FLEXION WITH HALTERE
                case 5:
                    shoulderElbowLenght_L = newBody.calculateLenght(newBody.shoul-
derLeft, newBody.elbowLeft);
                    finalLeftY = Math.Abs(newBody.shoulderLeft.Y + shoulderEl-
bowLenght_L);
                    shoulderElbowLenght_R = newBody.calculateLenght(newBody.shoul-
derRight, newBody.elbowRight);
                    finalRightY = Math.Abs(newBody.shoulderRight.Y + shoulderEl-
bowLenght_R);

                    canvasExercise.Children.Clear();

                    instructions.Text = string.Format("Execute o movimento de fle-
xao pausadamente, repetindo {0} vezes num conjunto de {1} séries", repetitions, se-
ries);

                    angleR = newBody.calculateAngle(newBody.shoulderRight, new-
Body.elbowRight);
                    rightAngleValue.Text = angleR.ToString();

                    path = string.Format("M{0},{1} L{2},{3} A{4},{5} 0 0 1 {6},{7}
z", newBody.shoulderRight.X, newBody.shoulderRight.Y, newBody.shoulderRight.X, final-
RightY, shoulderElbowLenght_R, shoulderElbowLenght_R, newBody.elbowRight.X, new-
Body.elbowRight.Y);
                    pathAngleRight.Data = System.Windows.Media.Geome-
try.Parse(path);

                    if ((angleR < target + 30) && (angleR > target - 30))
                    {
                        pathAngleRight.Stroke = new SolidColorBrush(Colors.Yel-
low);
                        pathAngleRight.Fill = new SolidColorBrush(Colors.Yellow);
                    }
                    if ((angleR < target + 15) && (angleR > target - 15))
                    {
                        pathAngleRight.Stroke = new SolidColorBrush(Colors.Green);
                        pathAngleRight.Fill = new SolidColorBrush(Colors.Green);
                        successR = true;

                    }
                    canvasExercise.Children.Add(pathAngleRight);
                    newBody.DrawPoint(canvasExercise, newBody.shoulderRight);
                    newBody.DrawPoint(canvasExercise, newBody.elbowRight);

                    if ((successR == true))
                    {
                        if ((angleR < 10) && (angleR > 0))
                        {
                            canvasExercise.Children.Clear();
                            newBody.DrawPoint(canvasExercise, newBody.shoulder-
Right);
                            newBody.DrawPoint(canvasExercise, newBody.elbowLeft);
```

```
                                finalRightY = Math.Abs(newBody.shoulderRight.Y +
shoulderElbowLenght_R);
                                angleL = newBody.calculateAngle(newBody.shoulderRight,
newBody.elbowLeft);
                                leftAngleValue.Text = angleL.ToString();

                                path = string.Format("M{0},{1} L{2},{3} A{4},{5} 0 0 1
{6},{7} z", newBody.shoulderRight.X, newBody.shoulderRight.Y, newBody.shoulderRight.X,
finalRightY, shoulderElbowLenght_R, shoulderElbowLenght_R, newBody.elbowLeft.X, new-
Body.elbowLeft.Y);
                                pathAngleLeft.Data = System.Windows.Media.Geome-
try.Parse(path);

                                if ((angleL < target + 40) && (angleL > target - 40))
                                {
                                    pathAngleLeft.Stroke = new SolidColorBrush(Col-
ors.Yellow);
                                    pathAngleLeft.Fill = new SolidColorBrush(Col-
ors.Yellow);
                                }
                                if ((angleL < target + 25) && (angleL > target - 25))
                                {
                                    pathAngleLeft.Stroke = new SolidColorBrush(Col-
ors.Green);
                                    pathAngleLeft.Fill = new SolidColorBrush(Col-
ors.Green);
                                    successL = true;
                                }
                                canvasExercise.Children.Add(pathAngleLeft);

                                if ((successL == true) && (successR == true))
                                {
                                    repeat = true; // repetição completa
                                    r++;
                                    atualRep.Content = r.ToString();
                                    repeat = false;
                                    success = false;
                                }
                                if (r == repetitions)
                                {
                                    s++;
                                    reset = true;
                                    r = 0;
                                    atualRep.Content = r.ToString();
                                }
                                if (s == series)
                                {
                                    instructions.Text = string.Format("Parabéns! Com-
pletou as {0} séries.", s);
                                    canvasExercise.Visibility = System.Windows.Visi-
bility.Hidden;
                                }
                            }
                        }
                    break;
                    #endregion
                }
                #endregion
            }
        }
        #endregion
```

```
    #region Buttons & Events

    #region Timer_Tick
    public void timerTick(object o, EventArgs sender)
    {
        countdownClock.Text = (time--).ToString();

        if (time < 0)
        {
            timer.Stop();
            countdownClock.Visibility = System.Windows.Visibility.Hidden;
            objetiveAngle.Visibility = System.Windows.Visibility.Visible;
            startExercise = true;
        }
    }
    #endregion

    #region Back_Page
    private void backButton_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
    {
        (App.Current as App).feedback = false;
        (App.Current as App).left = false;
        (App.Current as App).right = false;
        (App.Current as App).repetitions = 0;
        (App.Current as App).series = 0;
        (App.Current as App).targetAngle = 0;

        NavigationService.Navigate(new Uri("/MainMenu.xaml", UriKind.Relative));
    }
    #endregion

    #region Close_Page
    private void closeButton_MouseLeftButtonDown(object sender, MouseButtonEventArgs
e)
    {
        Environment.Exit(-1);
    }
    #endregion

    #region Angles_Points
    void drawAnglePoints(Point shoulder, double shoulderElbowLenght, int target,
string arm)
    {
        int aux, i;

        for (i = 0; i <= target; i += 10)
        {
            aux = (180 - i);
            if (arm == "l")
            {
                finalElbow.X = shoulder.X - (trig[aux].cos * trig[aux].tan * shoulder-
ElbowLenght);
            }
            else
            {
                finalElbow.X = shoulder.X + (trig[aux].cos * trig[aux].tan * shoulder-
ElbowLenght);
            }
```

```
            finalElbow.Y = shoulder.Y - ((trig[aux].sen * shoulderElbowLenght) /
trig[aux].tan);

            drawPoint(canvasExercise, finalElbow);
        }
    }

    #endregion

    #region Path_Left
    void drawPathLeft()
    {
        shoulderElbowLenght_L = newBody.calculateLenght(newBody.shoulderLeft, new-
Body.elbowLeft);
        finalLeftY = Math.Abs(newBody.shoulderLeft.Y + shoulderElbowLenght_L);

        angle = newBody.calculateAngle(newBody.shoulderLeft, newBody.elbowLeft);
        leftAngleValue.Text = angle.ToString();

        path = string.Format("M{0},{1} L{2},{3} A{4},{5} 0 0 1 {6},{7} z", new-
Body.shoulderLeft.X, newBody.shoulderLeft.Y, newBody.shoulderLeft.X, finalLeftY,
shoulderElbowLenght_L, shoulderElbowLenght_L, newBody.elbowLeft.X, newBody.el-
bowLeft.Y);
        pathAngleLeft.Data = System.Windows.Media.Geometry.Parse(path);

        if ((angle < target + 10) && (angle > target - 10))
        {
            pathAngleLeft.Stroke = new SolidColorBrush(Colors.Yellow);
            pathAngleLeft.Fill = new SolidColorBrush(Colors.Yellow);
        }

        if ((angle < target + 2) && (angle > target - 2))
        {
            pathAngleLeft.Stroke = new SolidColorBrush(Colors.Green);
            pathAngleLeft.Fill = new SolidColorBrush(Colors.Green);
            successL = true;
        }

        if (angle > target + 5)
        {
            pathAngleLeft.Stroke = new SolidColorBrush(Colors.Transparent);
            pathAngleLeft.Fill = new SolidColorBrush(Colors.Transparent);
        }
        canvasExercise.Children.Add(pathAngleLeft);
    }
    #endregion

    #region Path_Right

    void drawPathRight()
    {
        shoulderElbowLenght_R = newBody.calculateLenght(newBody.shoulderRight, new-
Body.elbowRight);
        finalRightY = Math.Abs(newBody.shoulderRight.Y + shoulderElbowLenght_R);
        angle = newBody.calculateAngle(newBody.shoulderRight, newBody.elbowRight);
        rightAngleValue.Text = angle.ToString();

        path = string.Format("M{0},{1} L{2},{3} A{4},{5} 0 0 0 {6},{7} z", new-
Body.shoulderRight.X, newBody.shoulderRight.Y, newBody.shoulderRight.X, finalRightY,
shoulderElbowLenght_R, shoulderElbowLenght_R, newBody.elbowRight.X, newBody.elbow-
Right.Y);
```

```
        pathAngleRight.Data = System.Windows.Media.Geometry.Parse(path);

        if ((angle < target + 10) && (angle > target - 10))
        {
            pathAngleRight.Stroke = new SolidColorBrush(Colors.Yellow);
            pathAngleRight.Fill = new SolidColorBrush(Colors.Yellow);
        }

        if ((angle < target + 2) && (angle > target - 2))
        {
            pathAngleRight.Stroke = new SolidColorBrush(Colors.Green);
            pathAngleRight.Fill = new SolidColorBrush(Colors.Green);
            successR = true;
        }

        if (angle > target + 5)
        {
            pathAngleRight.Stroke = new SolidColorBrush(Colors.Transparent);
            pathAngleRight.Fill = new SolidColorBrush(Colors.Transparent);
        }
        canvasExercise.Children.Add(pathAngleRight);
    }
    #endregion

    #region Path_Flexion
    void drawPathFlexion()
    {
        spineShoulderHandLenght_R = newBody.calculateLenght(newBody.spineShoulder,
newBody.handRight);
        finalRightY = Math.Abs(newBody.spineMid.Y + spineShoulderHandLenght_R);

        angle = newBody.calculateAngle(newBody.spineShoulder, newBody.handRight);
        angleValue.Text = angle.ToString();

        path = string.Format("M{0},{1} L{2},{3} A{4},{5} 0 0 1 {6},{7} z", new-
Body.spineShoulder.X, newBody.spineShoulder.Y, newBody.spineMid.X, finalRightY,
spineShoulderHandLenght_R, spineShoulderHandLenght_R, newBody.handRight.X, new-
Body.handRight.Y);
        pathFlexion.Data = System.Windows.Media.Geometry.Parse(path);

        if ((angle < target + 40) && (angle > target - 40))
        {
            pathFlexion.Stroke = new SolidColorBrush(Colors.Yellow);
            pathFlexion.Fill = new SolidColorBrush(Colors.Yellow);
        }

        if ((angle < target + 20) && (angle > target - 20))
        {
            pathFlexion.Stroke = new SolidColorBrush(Colors.Green);
            pathFlexion.Fill = new SolidColorBrush(Colors.Green);
            success = true;

        }
        canvasExercise.Children.Add(pathFlexion);
    }
    #endregion

    #region drawPoint
    public void drawPoint(Canvas canvas, Point point)
    {
        // Create an ellipse
```

```
        Ellipse ellipse = new Ellipse
        {
            Width = 6,
            Height = 6,
            Fill = new SolidColorBrush(Colors.Blue)
        };

        // Position the ellipse according to the joint's coordinates
        Canvas.SetLeft(ellipse, point.X - ellipse.Width / 2);
        Canvas.SetTop(ellipse, point.Y - ellipse.Height / 2);

        // Add the ellipse to the canvas
        canvas.Children.Add(ellipse);
    }
    #endregion

    #region Repetitions&Series
    void count()
    {
        if ((left == true) && (right == false))
        {
            if ((successL == true) && (Int32.Parse(leftAngleValue.Text) > 0) &&
(Int32.Parse(leftAngleValue.Text) < 15))
            {
                successL = false; // repetição concluída
                r++;
                atualRep.Content = r.ToString();
            }
        }

        if ((left == false) && (right == true))
        {
            if ((successR == true) && (Int32.Parse(rightAngleValue.Text) > 0) &&
(Int32.Parse(rightAngleValue.Text) < 15))
            {
                successR = false; // repetição concluída
                r++;
                atualRep.Content = r.ToString();
            }
        }

        if ((left == true) && (right == true))
        {
            if ((successL == true) && (Int32.Parse(leftAngleValue.Text) > 0) &&
(Int32.Parse(leftAngleValue.Text) < 15)
                && (successR == true) && (Int32.Parse(leftAngleValue.Text) > 0) &&
(Int32.Parse(leftAngleValue.Text) < 15))
            {
                successL = false; // repetição concluída
                successR = false;
                r++;
                atualRep.Content = r.ToString();
            }
        }

        if (exerciseNumber == 2)
        {
            if ((success == true) && (Int32.Parse(angleValue.Text) > 0) &&
(Int32.Parse(angleValue.Text) < 15))
            {
                success = false; // repetição concluída
```

```
                r++;
                atualRep.Content = r.ToString();
            }
        }

        if (exerciseNumber == 4)
        {
            if ((success == true) && (Int32.Parse(angleValue.Text) > 150) &&
(Int32.Parse(angleValue.Text) < 180))
            {
                success = false; // repetição concluída
                r++;
                atualRep.Content = r.ToString();
            }
        }


        if (r == repetitions) // atingiu o nº de repetições pretendidas
        {
            r = 0;
            s++;
            atualSer.Content = s.ToString();
        }

        if (s == series) // atingiu o nº de series pretendidas
        {
            r = 0;
            s = 0;
            instructions.Text = string.Format("Parabéns! Completou as {0} séries.",
series);
            finish = true; // fim do exercício
            canvasExercise.Visibility = System.Windows.Visibility.Hidden;
            canvasElements.Visibility = System.Windows.Visibility.Hidden;
        }
    }

    #endregion
    #endregion

}

enum CameraSensor
{
    Color,
    Depth,
    Infrared
}
}
```