ALEXANDRE YIP GONÇALVES DIAS
LUCCAS DE NIGRIS GARRITANO

MOVIMENTO DE UM AVATAR PARA REABILITAÇÃO SUBMETIDO A UM CAMPO DE FORÇA EM UM AMBIENTE DE REALIDADE VIRTUAL

Monografia apresentada no Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo para obtenção do titulo de Engenheiro. Area de Concentração: Engenharia Mecatrônica.

São Paulo
2015

ALEXANDRE YIP GONÇALVES DIAS
LUCCAS DE NIGRIS GARRITANO

MOVIMENTO DE UM AVATAR PARA REABILITAÇÃO SUBMETIDO A UM CAMPO DE FORÇA EM UM AMBIENTE DE REALIDADE VIRTUAL

Monografia apresentada no Departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro. Área de Concentração: Engenharia Mecatrônica

Curso de Graduação:
Engenharia Mecatrônica

Orientador:
Prof. Dr. Fabrício Junqueira

São Paulo
2015

Este relatório é apresentado como requisito parcial para obtenção do título de Engenheiro na Escola Politécnica da Universidade de São Paulo. É o produto do nosso próprio trabalho, exceto onde indicado no texto. O relatório pode ser livremente copiado e distribuído desde que a fonte seja citada.

AGRADECIMENTOS

Às nossas famílias, em especial aos nossos pais, que nos acompanharam em todos os momentos desses desafios e que, sem o suporte dos quais, não teríamos chegado até onde estamos.

Ao nosso orientador, Dr. Fabrício Junqueira que nos ajudou e nos guiou nesse último ano de trabalho. Esperamos que o convívio que tivemos tenha tido tanto impacto para ele quanto nos fez crescer como pessoas e profissionais.

À todos os amigos que fizemos, que nos acompanharam e ajudaram nesses anos de faculdade - memórias que levaremos para o resto da vida.

RESUMO

Um grande problema do processo de reabilitação é a falta de motivação do paciente. A partir disso, esse trabalho propõe, projeta e implementa um ambiente de realidade virtual voltado para esse setor utilizando o Microsoft Kinect e o conceito de campos de força. O *software* projetado deverá receber as ações do usuário através do Kinect, mostrá-los no ambiente criado por intermédio de um avatar, e exibir valores de torque com base no campo de força na qual seu avatar está imerso. A saída, em torque, permitirá o acoplamento de um exoesqueleto ao sistema em trabalhos subsequentes.

ABSTRACT

One of the major problems in the rehabilitation process is the patient's lack of motivation throughout the physical therapy. To solve that matter, this work design and implement a virtual reality environment focusing on the Health Care Sector and utilizing the Microsoft Kinect. The input of the system is the motion capture of the user by the Kinect and is displayed in a virtual reality environment in the form of an avatar. The output of the system is the torque created in the avatar's arm when it's in contact with a force field. The force field is an area in the virtual world that aids the user's movement or opposes it. The output will allow an exoskeleton to be incorporated to the system in forthcoming works.

LISTA DE SIGLAS

NUI - Natural User Interface

RV - Realidade Virtual

UML - *Unified Modeling Language*

SDK - *Software Development Kit*

IDE - *Integrated Development Environment*

TABELA DE FIGURAS

Figura 2.1 - Resultado do Teste de CHANG, CHEN e HUANG (2011) 5
Figura 2.2 - Resultado obtido por QUADRADO, NORIEGA e FORNER-CORDERO (2014), comparando a taxa de acertos dos dois grupos. B1, B2, B3, B4 e T refere-se à etapa de aquisição de dados. 5
Figura 2.3 – Arquitetura interna do sensor do Kinect (CATUHE, 2012) 9
Figura 2.4 - Área de melhor precisão do Kinect para modo padrão (CATUHE, 2012) 10
Figura 2.5 - Arquitetura do SDK do Kinect (CATUHE, 2012) 10
Figura 2.6 - Juntas encontradas pelo Kinect (CATUHE, 2012) 11
Figura 2.7 - Arquitetura do XNA (WEI, DONGSHENG e CHUN, 2013), modificada . 12
Figura 3.1 – Diagrama de Caso de Uso 14
Figura 3.2 – Diagrama de Atividade - Calibrar Sistema 15
Figura 3.3 - Diagrama de Atividade - Alterar Exercício Escolhido 15
Figura 3.4 - Diagrama de Atividade - Alterar Intensidade Máxima do Exercício 16
Figura 3.5 – Diagrama de Atividade - Alterar Multiplicador do Exercício 16
Figura 3.6 - Diagrama de Atividade - Alterar Proporção Esforço 17
Figura 3.7 - Diagrama de Atividade - Iniciar Simulação 17
Figura 3.8 - Diagrama de Atividade - Encerrar Simulação 17
Figura 3.9 - Diagrama de Atividade - Encerrar Programa 18
Figura 3.10 – Diagrama de Atividade - Movimentar/ Detectar Movimento 18
Figura 3.11 – Fragmento de código – Inicialização de componentes e serviços 19
Figura 3.12 – Fragmento de código – Retorno de um serviço do tipo camera 20
Figura 3.13 – Diagrama de Componentes 20
Figura 3.14 – Diagrama de Classe – Game 22
Figura 3.15 - Diagrama de Classe - TelaInicial 23
Figura 3.16 - Diagrama de Classe – KinectControlador 24
Figura 3.17 - Diagrama de Classe - Camera 24
Figura 3.18 - Diagrama de Classe – Chao 25
Figura 3.19 - Diagrama de Classe – Avatar 26
Figura 3.20 - Diagrama de Classe – DepthVideo 27

Figura 3.21 - Diagrama de Classe – Forca 28
Figura 3.22 – Diagrama de Sequência - Calibrar Sistema (visão do usuário).....29
Figura 3.23 - Diagrama de Sequência - Calibrar Sistema (visão do jogo).....30
Figura 3.24 – Diagrama de Sequência - Escolher Exercício (visão jogo).....31
Figura 3.25 – Diagrama de Sequência - Escolher Exercício (visão usuário).....31
Figura 3.26 – Diagrama de Sequência - Iniciar Simulação (visão do usuário) 32
Figura 3.27 – Diagrama de Sequência - Iniciar Simulação (visão do jogo) 32
Figura 3.28 - Diagrama de Sequência - Encerrar Simulação (visão do usuário).....33
Figura 3.29 - Diagrama de Sequência - Encerrar Simulação (visão do jogo).....33
Figura 3.30 - Diagrama de Sequência - Encerrar Programa 34
Figura 3.31 - Diagrama de Sequência - Movimentar/Detectar Movimento 34
Figura 4.1 - Modelo 3D do Homer, respectivamente na forma de arestas, superfície e com textura.....36
Figura 4.2 - Esqueleto confeccionado para o modelo e sua superposição à superfície 37
Figura 4.3 - Paint Weight do centro do quadril. 38
Figura 4.4 - Exemplo de movimento.....38
Figura 4.5 - Modelo do disco 38
Figura 4.6 - Comparação entre o sistema de coordenadas do Kinect (esquerda) e do modelo (direita) 40
Figura 5.1 – Tela Incial do Jogo 44
Figura 5.2 – Tela de Configuração do Exercício 45
Figura 5.3 – Simulação sem o usuário 46
Figura 5.4 – Simulação com o usuário 47

SUMÁRIO

1. Introdução 1
1.1. Objetivos do trabalho 2
1.2. Metodologia utilizada 2
1.3. Organização do Texto 3
2. Revisão Bibliográfica 4
2.1. Uso de tecnologia em reabilitação 4
2.2. Exoesqueleto 6
2.3. Realidade virtual 7
2.4. Microsoft Kinect 8
2.5. XNA 11
3. Projeto 13
3.1. Caso de Uso 13
3.2. Diagrama de Atividade 15
3.3. Diagrama de Componentes 19
3.4. Diagrama de Classe 21
3.4.1. Game 21
3.4.2. TelaInicial 22
3.4.3. KinectControlador 23
3.4.4. Camera 24
3.4.5. Chao 25
3.4.6. Avatar 25
3.4.7. DepthVideo 26
3.4.8. Forca 27
3.5. Diagrama de Sequência 29
3.5.1. Calibrar 29
3.5.2. Escolher Exercício 30

3.5.1. Iniciar Simulação 32
3.5.2. Encerrar Simulação 33
3.5.3. Encerrar Programa 33
3.5.4. Movimentar/Detectar Movimento 34
4. Implementação 35
4.1. Microsoft Kinect 35
4.2. Modelo 3D 35
4.3. Avatar 39
4.4. Campo de Força 41
5. Demonstração do Protótipo 44
6. Conclusão 48
7. Referência 49

1. Introdução

Segundo a COFFITO, a Fisioterapia “é uma ciência da saúde que estuda, previne e trata os distúrbios cinéticos funcionais intercorrentes em órgãos e sistemas do corpo humano, gerados por alterações genéticas, por traumas e por doenças adquiridas, na atenção básica, média complexidade e alta complexidade. Fundamenta suas ações em mecanismos terapêuticos próprios, sistematizados pelos estudos da biologia, das ciências morfológicas, das ciências fisiológicas, das patologias, da bioquímica, da biofísica, da biomecânica, da cinesia, da sinergia funcional, e da cinesia patológica de órgãos e sistemas do corpo humano e as disciplinas comportamentais e sociais”.

A COFFITO ainda divide a fisioterapia em 4 áreas de atuação: Fisioterapia Clínica, Saúde Coletiva, Educação e Outros. A Fisioterapia clínica deve “elaborar o Diagnóstico Cinesiológico Funcional, prescrever, planejar, ordenar, analisar, supervisionar e avaliar os projetos fisioterapêuticos, a sua eficácia, a sua resolutividade e as condições de alta do cliente submetido a estas práticas de saúde”, assim, abrangem os ambulatórios, as clínicas e hospitais, os consultórios e os centros de reabilitação”.

Quanto ao processo de reabilitação, Lozano-Quilis *et al.* (2013) apresentam dois problemas fundamentais. Primeiramente, o paciente precisa se manter motivado ao longo de todo o tratamento, o que pode ser difícil, dada a natureza insistente e pouco variante dos exercícios motores, diminuindo o interesse do praticante conforme o tratamento avança. Em segundo, por se tratar de um processo de recuperação, existe a necessita de um acompanhamento de uma pessoa qualificada e monitoração constante. Ainda em Lozano-Quilis *et al.* (2013), observa-se que o uso de novas tecnologias, como o uso de Realidade Virtual (RV) e *Natural User Interface* (NUI), melhora o aproveitamento do usuário.

O aumento da qualidade do tratamento e da difusibilidade com o uso de novas tecnologias, vêm de encontro com a tendência de aumento da população de idosos no mundo e poderão facilitar o acesso ao tratamento. Em 2045, é esperado que a população mundial de idosos superará a de crianças. Em 2050, a população de idosos será de aproximadamente 22% (DEPARTMENT OF ECONOMIC AND

SOCIAL AFFAIRS - POPULATION DIVISION, 2009). Essa faixa etária será muito beneficiada com o avanço de tecnologias na área da reabilitação e na medicina em geral.

Seguindo a ideia do uso de novas tecnologias durante tratamentos fisioterápicos, este trabalho consistirá no projeto e implementação de um ambiente de realidade virtual voltado à reabilitação e assistida pelo Kinect.

1.1. Objetivos do trabalho

O objetivo é modelar e simular um ambiente virtual que possui como entrada sinais físicos (movimento de uma pessoa) e como saída sinais de intensidade de torque. Em um próximo trabalho, essas saídas poderão ser interpretados por um exoesqueleto como um esforço que deverá ser aplicada ao usuário em resposta aos movimentos no mundo simulado.

Como objetivos secundários, será necessário a modelagem de um avatar e de um campo de força. A RV representará o corpo humano através de um avatar, e fará com que este sofra influência de campos de força gerados pelo software no ambiente criado. Assim, esses campos de força terão a finalidade de facilitar ou dificultar o deslocamento do avatar (ou de partes deste, como braços ou pernas) pelo ambiente de realidade virtual.

Por fim, como será focado na reabilitação, será criado um exercício focado nas partes do ombro e cotovelo.

Como não será desenvolvido o exoesqueleto nesse trabalho, o sinal de entrada será unicamente proveniente de uma fonte externa, e a resposta será demonstrada através de indicação de intensidade de torque, referente ao campo de força que envolve o avatar.

1.2. Metodologia utilizada

Primeiro temos a escolha da aplicação - a reabilitação. Partindo disso, foi pensado na ferramenta mais adequada para o projeto, chegando à conclusão que o Microsoft Kinect, em conjunto com sua SDK, seria a ferramenta mais completa e barata.

Avançando para um nível mais interno do projeto, foi necessário fixar outros pontos importantes. Escolheu-se o uso do *framework* XNA para criar o ambiente de realidade virtual, uma vez que foi desenvolvido pela própria Microsoft e é compatível com o Kinect, diminuindo qualquer possibilidade de conflito de programação com o

SDK do *hardware*. Por fim, para a implementação do XNA será utilizado o IDE *Visual Studio* da Microsoft. A implementação será a etapa de codificação do *software*.

Por fim, serão realizados testes de funcionalidade do software, entre os desenvolvedores e convidados, para a validação de tudo que foi proposto.

1.3. Organização do Texto

Inicialmente procura-se fazer uma breve análise do estado da arte no assunto trabalhado. Essa análise é encontrada na seção 2 - Revisão bibliográfica. Dentro dessa, será desenvolvido desde o conteúdo mais geral, até o mais específico, cobrindo todos os tópicos que cercam o assunto desse trabalho.

Em seguida, na seção 3 - Projeto será desenvolvido o projeto *per se*. A concepção do programa foi feita usando a UML, através dos Diagramas de Caso de Uso, Atividade, Componentes, Classe e Sequência, respectivamente nas seções 3.1, 3.2, 3.4, 3.5 e 3.5.

Na seção 4 - Implementação encontra-se as etapas e considerações para a implementação do *software* pensado na seção 3.

Na seção 5 - Demonstração do Protótipo será mostrado o funcionamento do programa através de telas e elementos visuais que ele utiliza.

A última seção do trabalho tratará das conclusões do mesmo

2. Revisão Bibliográfica

A revisão bibliográfica fará um estudo dos assuntos envolvidos na monografia. Esse estudo busca contextualizar o projeto e sua contribuição. Essa revisão oferece também um conhecimento que facilitará atingir os objetivos. Foram focados cinco tópicos: o uso de tecnologias e jogos em sessões fisioterápicas, exoesqueleto, realidade virtual, tecnologia Microsoft Kinect, XNA e campos de força. Primeiramente, será feito uma breve análise na interatividade entre a tecnologia e a fisioterapia, passando então para o uso de exoesqueletos. Após, será abordado uma análise do *hardware* de interesse, o Microsoft Kinect. E por último, será feita uma breve analise do *framework* da Microsoft para desenvolvimento de jogos voltados ao PC e ao Xbox, o XNA, que será usado para criar o ambiente virtual.

2.1. Uso de tecnologia em reabilitação

A reabilitação assistida por jogos vem se desenvolvendo muito na tentativa de melhorar o desempenho e o interesse do paciente, buscando diminuir a evasão do tratamento. O estudo de Lange *et al.* (2011) conduzido com 20 pacientes que possuíam várias formas de inabilidades motoras e diferentes graus de experiência com o uso de vídeo games comprovou que na grande maioria dos usuários, a reabilitação por Kinect foi considerada desafiadora e divertida, comprovando a grande capacidade dessa forma de tratamento. Indiretamente, a grande quantidade de sugestões mostra que ainda existe espaço para crescimento desse método.

Em consonância com as conclusões citadas anteriormente, vale ressaltar também o estudo feito por Chang, Chen e Huang (2011). Esse estudo buscou objetivar o aumento da qualidade do tratamento utilizando técnicas auxiliadas por um dispositivo tecnológico, nesse caso o Kinect. Para efeito de comparação foram feitos em duas fases. Na primeira, chamada de “*Baseline*”, o usuário (no caso Peter) foi ensinado como realizar o exercício da maneira tradicional. A partir do ponto onde seu entendimento foi considerado satisfatório, os movimentos, de forma não assistida, foram realizados e o clínico contou o número de ações corretas. Na segunda fase, chamada de “*Intervation*” foi utilizado o Kinerehab (habilitação assistida pelo Kinect). O jogo contou os movimentos certos enquanto mostrava de forma interativa se o movimento estava correto ou não. A Figura 2.1 mostra os resultados.

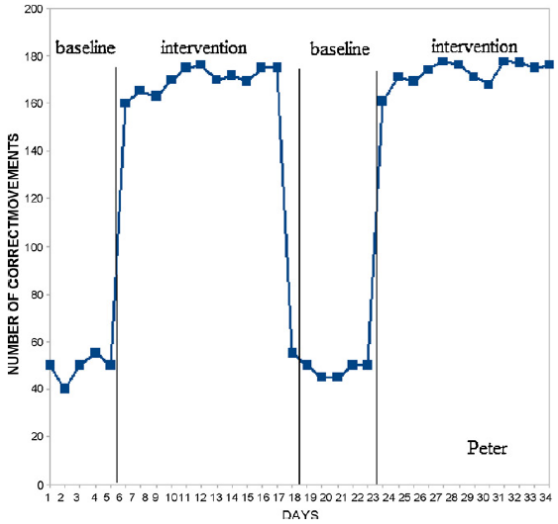
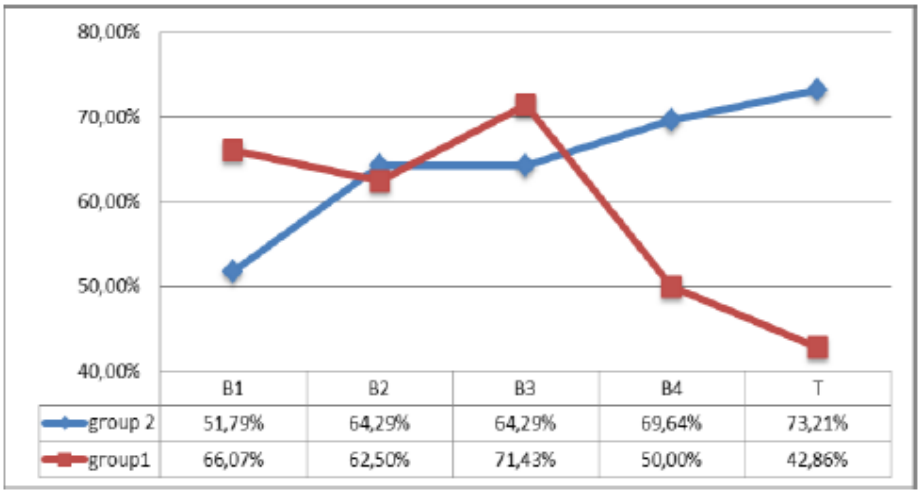


Figura 2.1 - Resultado do Teste de CHANG, CHEN e HUANG (2011)

Ainda um terceiro estudo, feito por Quadrado, Noriega e Forner-Cordero (2014) procurou correlacionar a existência de uma perturbação mecânica ao aprendizado de uma pessoa. Eles concluem que a existência de forças conhecidas durante a movimentação de uma pessoa aumenta consideravelmente o tempo de aprendizagem. Para isso, eles desenvolveram uma tarefa que deveria ser executada por 2 grupos de teste, um sob nenhuma influencia externa (grupo 1), e o outro sob a influência de uma força elástica acoplada à mão (grupo 2). Feita a tarefa, o número de acertos e erros foram comparados. Na última bateria de teste (T), um parâmetro foi alterado, no caso, a velocidade com que a tarefa deveria ser executada. Por fim, concluiu-se melhor taxa de aprendizagem no grupo submetido à força elástica.



	B1	B2	B3	B4	T
group 2	51,79%	64,29%	64,29%	69,64%	73,21%
group1	66,07%	62,50%	71,43%	50,00%	42,86%

Figura 2.2 - Resultado obtido por QUADRADO, NORIEGA e FORNER-CORDERO (2014), comparando a taxa de acertos dos dois grupos. B1, B2, B3, B4 e T refere-se à etapa de aquisição de dados.

Com esses três estudos, consegue-se demonstrar a eficácia da reabilitação assistida utilizando-se de um dispositivo tecnológico. Percebe-se que tal sucesso deve-se ao uso de sensores e a correção constante do movimento.

A assistência dos sensores é essencial, uma vez que não só a repetição é importante para a melhor reabilitação como também é imperativo que o exercício seja feito de forma correta e que não varie dentro de uma mesma série. O estado da arte empregado por pacientes e clínicos é o uso de instruções escritas, *log* de acompanhamento e simples dispositivos de contagem (ZHAO, *et al.*, 2014).

Já empregando as conclusões descritas nos dois parágrafos anteriores, pode-se citar o trabalho de Zhao *et al. (2014)*. Nesse trabalho, uma interface é criada e com o Kinect faz-se o acompanhamento em tempo real do exercício com *feedback* intuitivo e de rápida correção do movimento. Na interface desse projeto o paciente é representado por um avatar e divide a tela com outro avatar que demonstra a correta forma de comportamento.

Expandindo ainda mais o uso de tecnologia na área da reabilitação, unindo um sistema computacional porém com o uso de uma entrada diferente do Kinect, pode-se citar Andrade *et al. (2010).* Nesse projeto, com o auxílio de um dispositivo mecânico, que tem função de trabalhar como um *joystick* e um coletor de dados, e um jogo (no caso o pong) busca-se a reabilitação da fratura do rádio (osso localizado no antebraço). O dispositivo serve de auxílio fazendo com que o terapeuta consiga corrigir o tratamento enquanto esse está em andamento, e assim conseguir um melhor aproveitamento. Esse método assemelha-se ao método utilizando exoesqueletos.

Os exoesqueletos (WEI, *et al.*, 2013a) e os sistemas baseados em realidade virtual (LOZANO-QUILIS, *et al.*, 2013) são dois outros tipos de sistemas utilizados em tratamentos de reabilitação.

2.2. Exoesqueleto

O exoesqueleto é um mecanismo desenvolvido para mover-se em conjunto com um membro humano, enquanto distribui forças para ele. Seu modelo, geralmente, é feito a partir do corpo humano, mais especificamente dos ossos, unidos através das juntas (cotovelos, joelhos, entre outros). Eles procuram reproduzir o movimento dos membros. Os modelos variam no número de graus de liberdades (GL) que o

dispositivo possui. O aumento do número de GL possibilita maior capacidade de adaptação (JARASSÉ, CROCHER e MORE, 2012).

Segundo Wei *et al* (2013) é uma tecnologia que combina sentido, controle e informação. Através do *feedback* de sensores de posição e força o mecanismo proporciona uma reação que permite a reabilitação. Eles ainda citam que os vários graus de liberdade, o alcance de movimento e o controle preciso são vantagens desse tipo de mecanismo.

Wang *et al* (2014) procuram desenvolver um exoesqueleto para a perna. Partindo da ideia de que o dispositivo será acoplado ao corpo, ele deve proporcionar o maior conforto possível. A partir da analise cinemática da perna, eles desmembraram em vários conjuntos de juntas diferentes que simulassem esse movimento. E a partir desses esboços procuraram o conjunto que melhor reproduziria o movimento do membro de interesse e, assim, proporcionaria melhor conforto.

2.3. Realidade virtual

A Realidade Virtual vem sendo empregada de muitas maneiras, por exemplo, como uma alternativa a treinamentos para situações de difícil ou custosa reprodução física ou alta periculosidade, como visto em (OLIVEIRA e OLIVEIRA, 2014), onde utilizando um projeto 3D de uma plataforma de petróleo e um Kinect, criou-se um sistema onde no futuro poderá fazer com que funcionário sejam treinados em diversas atividades sem que haja a necessidade de levá-los a um ambiente tão perigoso quanto uma plataforma. Soma-se à isso o fato de que um sistema desse é bem mais barato do que os usados no mercado, como citado anteriormente. O treinamento de pilotos também pode ser feito dessa mesma forma, como mostrado em (WEI, DONGSHENG e CHUN, 2013b), aumentando a disponibilidade do sistema e a habilidade dos futuros comandantes de aeronaves visto que é um treinamento que deve ser repetido extensivamente para que não reste nenhum cenário que o piloto não tenha enfrentado, criando uma grande experiência. Há também o uso da realidade virtual em um ambiente educacional. Em (OUCH e ROUSE, 2011), o desenvolvimento de uma plataforma simulada da condução de um automóvel ensina os futuros motoristas a sinalização e significado das placas, leis de trânsito, segurança, responsabilidade, entre outros.

Algumas técnicas de manifestar fisicamente os sistemas de realidade virtual podem ser observadas em (OUARTI, LÉCUYER e BERTHOZ, 2014) e (KAUFMAN, 2007). Em (OUARTI, LÉCUYER e BERTHOZ, 2014) é citado 4 tipos de tecnologia. São elas o hexápode, o dispositivo baseado em trilhos, a estimulação galvânica dos nervos vestibulares e o dispositivo baseado em toque. O hexápode é um sistema de 6 graus de liberdade que simula movimento e transmite a sensação através de acelerações. O dispositivo baseado em trilhos trata-se de um hexápode montado sobre trilhos para aumentar o seu curso de movimento. Ambos são usados em simuladores, encontrados em parques de diversão. A estimulação galvânica procura estimular com pequenos choques as terminações nervosas localizada na região posterior à orelha, que controlam a sensação de equilíbrio, e a sensação de horizontal e vertical; mexendo nessas sensações torna-se possível transmitir ideia de aceleração para o usuário, mesmo que fisicamente ela não esteja ocorrendo; exposições a esse método por tempo prolongado pode causar vertigem. E por último, o dispositivo baseado no toque que procura transmitir forças para a região das mãos, mais especificamente, transmitir sensações para a ponta dos dedos. (OUARTI, LÉCUYER e BERTHOZ, 2014)

Já em (KAUFMAN, 2007) é apresentado um dispositivo de uso individual, com certa semelhança a um exoesqueleto, que procura identificar alguns movimentos básicos como agachar, deitar, ficar em pé, entre outros. Esse tipo de aparato é de interesse desse projeto, uma vez que, como foi citado anteriormente, o projeto poderá evoluir para um modelo mais desenvolvido num futuro próximo.

O Microsoft Kinect pode ser usado para interagir com o ambiente de RV, que pode ser criado pela framework XNA

2.4. Microsoft Kinect

O Microsoft Kinect é um produto que mudou a forma de interação entre o homem e o computador. Por se tratar de um equipamento de baixo custo, possibilitou trazer uma interatividade antes somente vista em laboratórios de alta tecnologia, com equipamentos caríssimos da ordem de $240.000, para dentro da casa das pessoas por cerca de $150 (TONG, *et al.*, 2012). Com o advento de tal tecnologia, primariamente empregada na indústria de video games, e a liberação de bibliotecas de programação próprias para esse equipamento, veio à procura de novas formas de trazer essa experiência para outras áreas. Existem estudos sobre seu uso em

áreas médicas (LOZANO-QUILIS, *et al.*, 2013), como reabilitação e fisioterapia, escaneamento 3D de objetos ou pessoas (TONG, *et al.*, 2012) e treinamento para situações extremas ou incomuns (OLIVEIRA e OLIVEIRA, 2014).

A forma de interação entre o Kinect e o usuário se dá por um programa gráfico, como um jogo. Esses métodos, em geral possibilitam a preparação psicológica do usuário a enfrentar uma situação, já que há pouca resposta física do computador no processo de comunicação entre a máquina e a pessoa.

O Kinect trata-se de um equipamento que combina uma câmara RGB, microfones e um emissor e receptor de infravermelho. A câmara é responsável pela capitação da imagem enquanto o infravermelho é responsável pela capitação de profundidade. Ao combinar esses 2 sinais, é possível obter uma imagem tridimensional em cores. Essas 5 informações (as 3 cores, a profundidade e o som) são pré-processadas e então enviadas para o PC ou XBox via USB, como ilustrado na Figura 2.3.

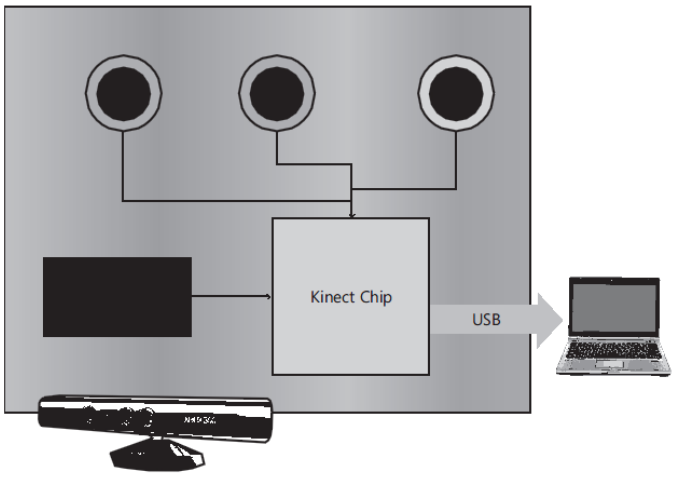


Figura 2.3 – Arquitetura interna do sensor do Kinect (CATUHE, 2012)

Segundo Catuhe (2012), a distância de melhor capitação em modo padrão do Kinect é entre 1,2m e 3,5m com margem lateral de mais ou menos 1,5m para a direita e esquerda. Para melhor esclarecimento, essa região é demonstrada na Figura 2.4. Mesmo fora dessa região, o Kinect consegue capitar pessoas com boa acurácia, sendo capaz de capitar até 6 pessoas simultaneamente.

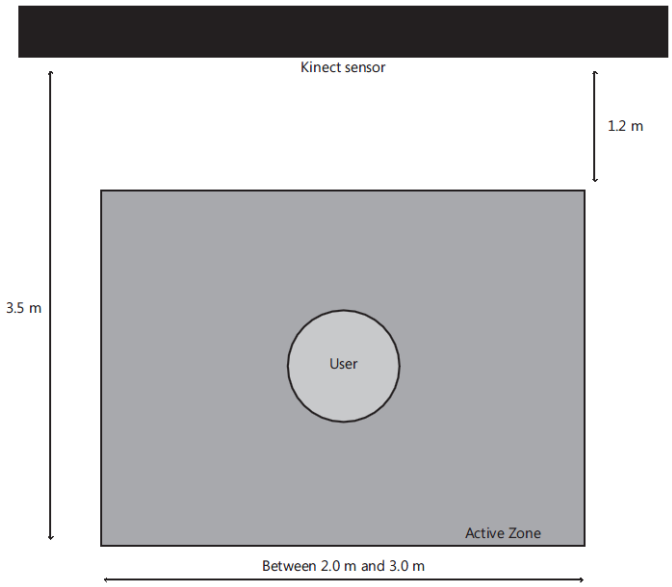


Figura 2.4 - Área de melhor precisão do Kinect para modo padrão (CATUHE, 2012)

Após a capitação e envio dos dados ao computador, a SDK do Kinect no Microsoft Windows deverá processar esses dados conforme o requisitado pela aplicação desenvolvida. A arquitetura desse SDK é apresentada na Figura 2.5.

Como pode ser observado na Figura 2.6, o SDK possui diversas funções, entre elas está configuração da captura de áudio e vídeo no Kinect e um processamento dos dados recebidos. É importante dar ênfase ao bloco identificado por NUI API. Essa é a funcionalidade do SDK que rastreia um usuário e modela seu esqueleto através do posicionamento de 20 juntas diferentes no corpo humano (10 se a pessoa estiver sentada).

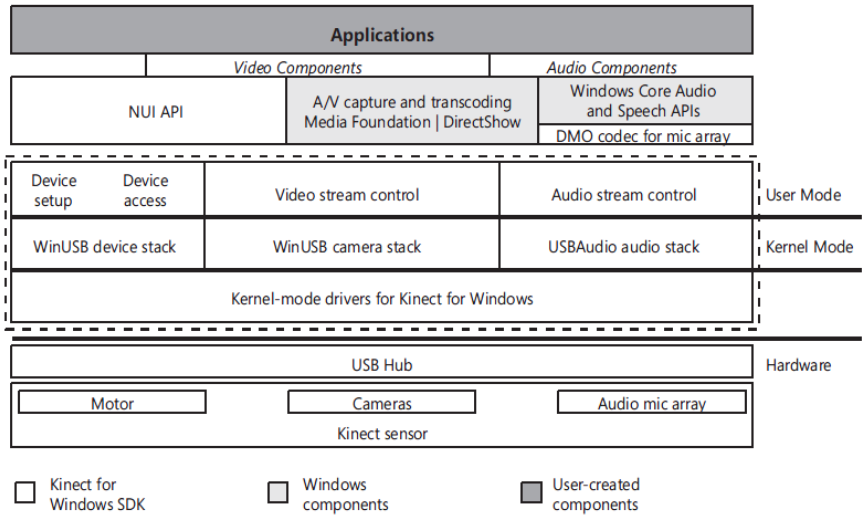


Figura 2.5 - Arquitetura do SDK do Kinect (CATUHE, 2012)

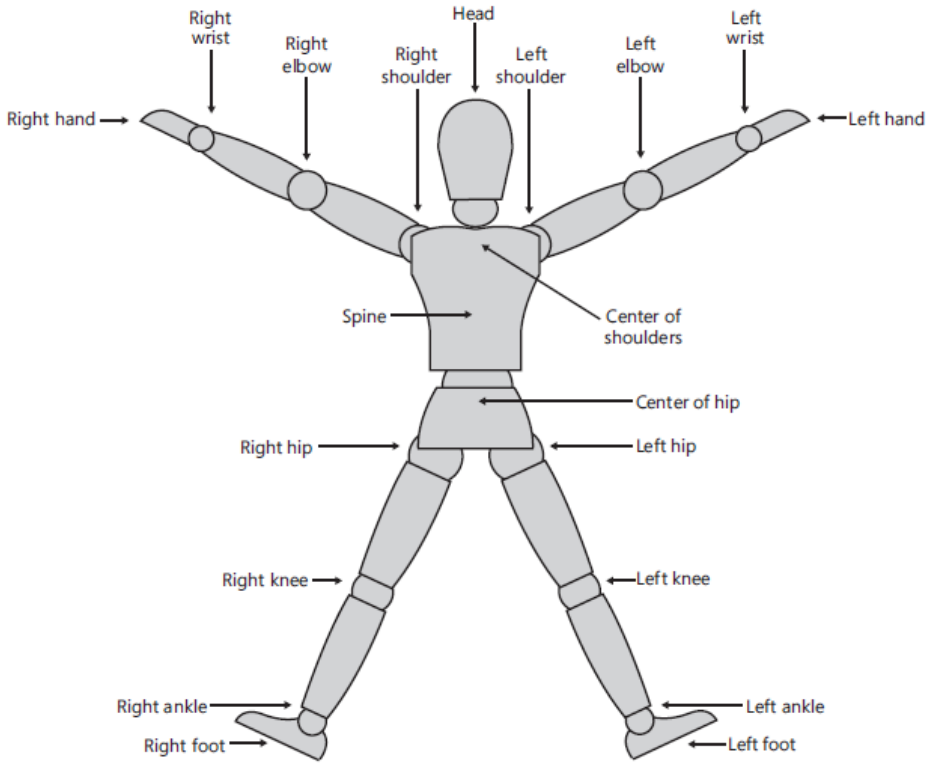


Figura 2.6 - Juntas encontradas pelo Kinect (CATUHE, 2012)

2.5. XNA

O uso de uma ferramenta de realidade virtual e de um avatar para representar uma pessoa se faz necessário, uma vez que pela Política de Privacidade para testes em humanos, não se deve mostrar a imagem do demonstrador nem do paciente além de que esse método aumenta o conforto do usuário (ZHAO, *et al.*, 2014).

O XNA é ferramenta de desenvolvimento distribuída pela Microsoft que auxilia a produção e codificação de jogos de video game. O X mostra a possibilidade da integração multiplataforma (*cross-platform* em inglês, de onde surge o uso da letra X) entre Xbox e o Windows. O N explicita que esse *software* é voltado para as novas gerações (“*new generation”*) de consoles e o A vem da palavra “*architecture”*. O *framework* consiste de uma extensa biblioteca de classes de códigos voltados à programação de jogos para PC (Windows) e Xbox. A linguagem utilizada é o C# (C Sharp) (WEI, DONGSHENG e CHUN, 2013b).

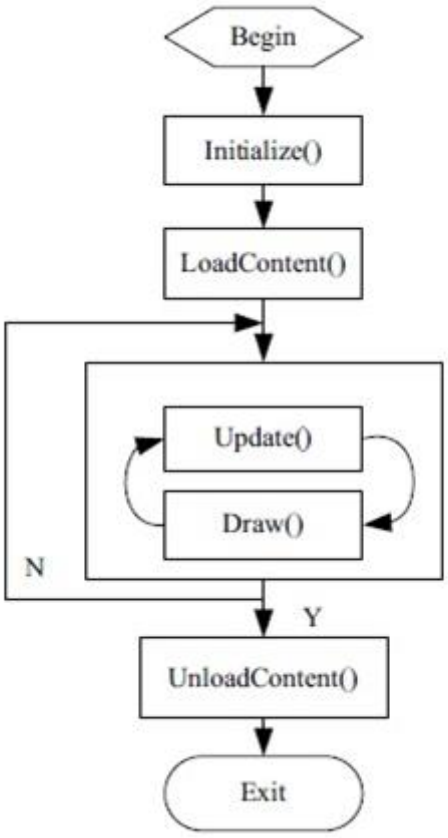


Figura 2.7 - Arquitetura do XNA (WEI, DONGSHENG e CHUN, 2013), modificada

A Figura 2.7 mostra a sequência de processos do *software* quando o jogo é executado. O Initialize() faz a inicialização dos recursos (áudios, vértices, etc.) e aplicações do programa como propriedades e carregamentos. O LoadContent() é usado uma vez por jogo e faz o carregamento do conteúdo necessário para rodar o jogo. O Update() é onde a lógica por trás do jogo funciona. O Draw() é chamado toda vez que precisa haver um redesenho do “mundo”. O UnloadContent() fecha o programa e libera os recursos que estavam sendo empregados (WEI, DONGSHENG e CHUN, 2013b).

3. Projeto

O *software* foi projetado e documentado através da Linguagem de Modelagem Unificada (UML). Seu projeto foi elaborado pensando nos objetivos propostos e procurando atacar os problemas da reabilitação anteriormente citados.

O projeto será apresentado por meio de 5 tipos de diagramas: diagrama de caso de uso, diagrama de atividade, diagrama de componente, diagrama de classe e diagrama de sequência.

3.1. Caso de Uso

O primeiro diagrama construido é o de caso de uso. O diagrama de caso de uso define as interações dos “atores” com o sistema.

Na Figura 3.1, existem três atores: o Paciente, o Especialista e o Kinect.

A única ação que o paciente pode executar é movimentar-se. Esse movimento é detectado pelo Kinect e seus dados passados ao programa.

O Especialista é o ator generalizado do paciente, ou seja, ele pode realizar o mesmo que o paciente, além de funções adicionais exclusivas a ele que garante o funcionamento ideal da sessão de exercício. Ele é o responsável por ações de controle do programa, que são: “Calibrar Sistema”, “Alterar Exercício Escolhido” (Anterior ou Posterior), “Alterar Intensidade Máxima” do Exercício (Aumentar ou Diminuir), “Alterar Multiplicador” do Exercício (Aumentar ou Diminuir), “Iniciar Simulação”, “Encerrar Simulação” e “Encerrar Programa”. A ação de “Calibrar Sistema” permitirá adequar o *software* ao tamanho do corpo do paciente, eliminando discrepâncias sentidas pelo paciente devido a seu tamanho corporal.

“Alterar Exercício Escolhido” permitirá a escolha de uma entre várias configurações de exercícios previamente definidos. “Alterar Intensidade Máxima”, “Alterar Multiplicador” e “Alterar Proporção Esforço” do Exercício mudará, respectivamente, a intensidade máxima que o esforço na junta poderá atingir, o quão rápido ele crescerá à medida que se afasta do objetivo e a proporção de distribuição do esforço entre as juntas trabalhadas. Uma vez que deve englobar o maior número possível de deficiências, desde as mais graves até as mais simples, é importante existir esses parâmetros para adequar o exercício ao paciente específico.

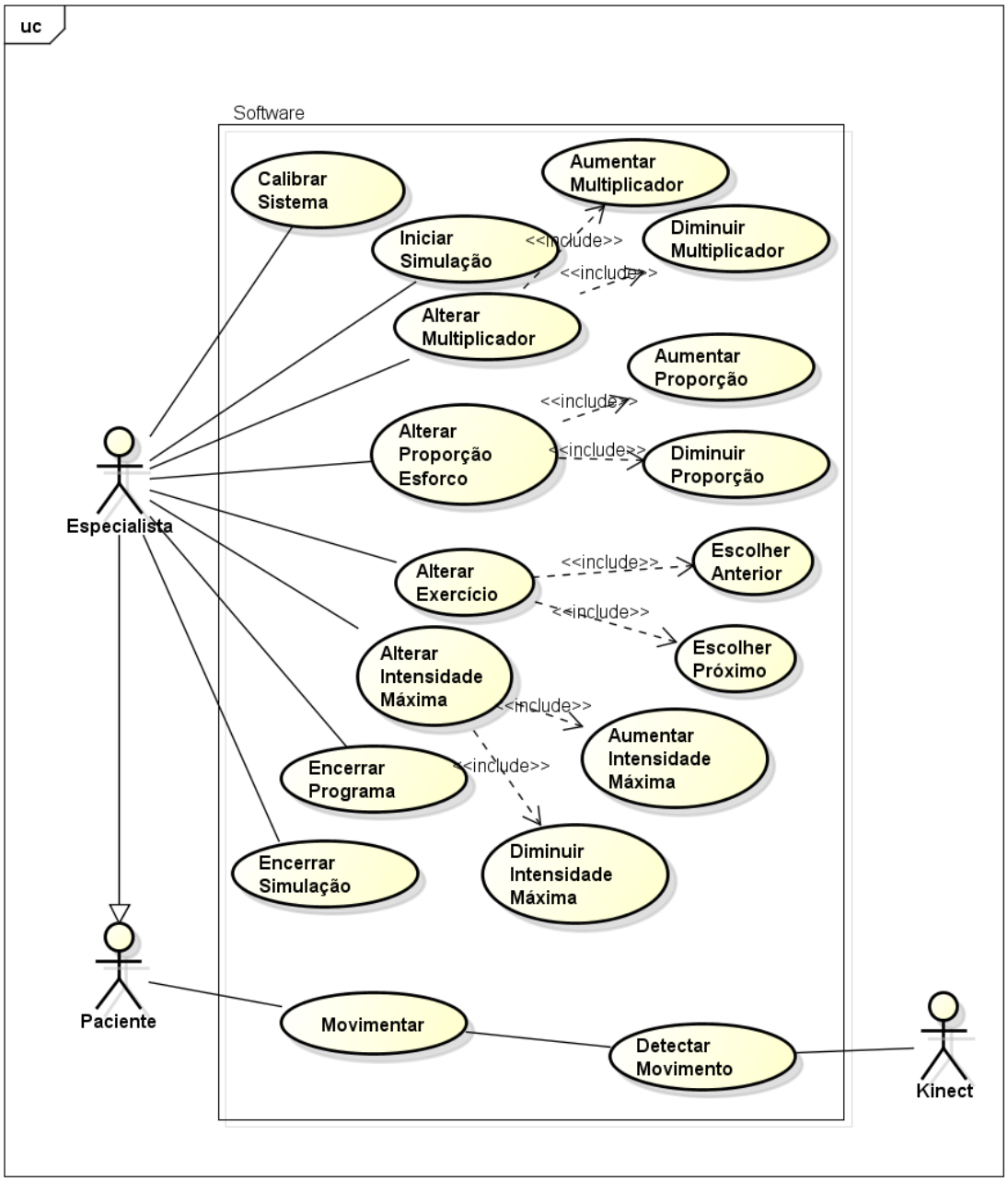


Figura 3.1 – Diagrama de Caso de Uso

Por fim, “Iniciar Simulação” e “Encerrar Simulação” deverá iniciar ou encerrar o ambiente de realidade virtual, dando início ao tratamento. E “Encerrar Programa” simplesmente desativa todas as funcionalidades e fecha o programa.

O principal motivo da separação entre o paciente e o especialista foi impedir que o paciente tivesse autonomia para fazer o exercício de forma independente, o que poderia agravar seu problema ou até causar novas lesões.

3.2. Diagrama de Atividade

O diagrama de atividade expressa o fluxo básico de ações tomadas pelo programa em resposta a uma ação de um ator prevista no caso de uso. Assim, deve haver um diagrama para cada uso expresso na Figura 3.1.

A calibragem do sistema (Figura 3.2) é um dos pontos sugeridos pelo teste de Lange *et al* (2011). O usuário se posicionará em frente ao Kinect de forma que as juntas sejam bem visíveis. Uma vez que o sistema consegue identificar esses pontos, ele faz as medidas do braço direito e os armazena nas configurações.

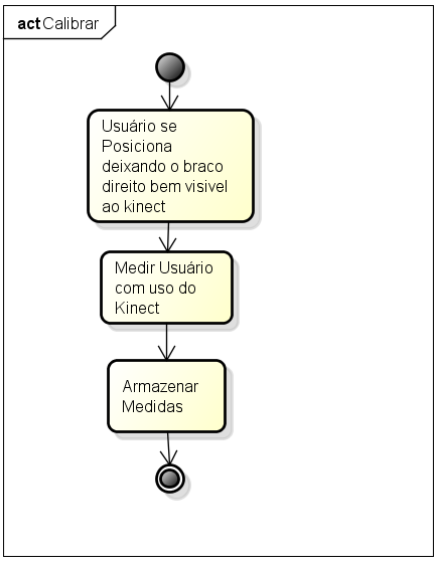


Figura 3.2 – Diagrama de Atividade - Calibrar Sistema

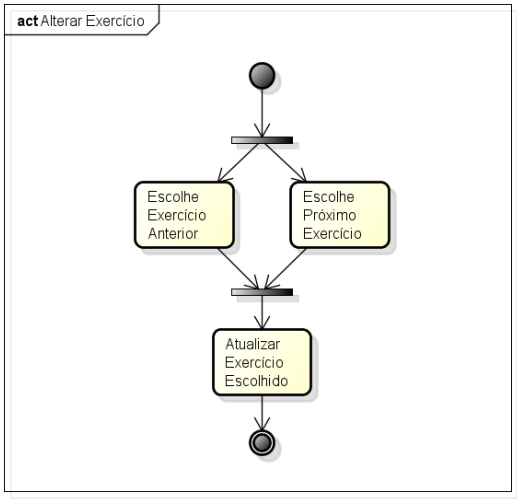


Figura 3.3 - Diagrama de Atividade - Alterar Exercício Escolhido

A configuração do exercício a ser executado será feita por meio de 4 casos de uso: um para a escolha do exercício (Figura 3.3), outro para selecionar a intensidade

máxima do esforço que o programa poderá externar (Figura 3.4), um para escolher o quão rápido esse esforço cresce com o distanciamento do objetivo. (Figura 3.5) e, por último um para indicar a distribuição do esforço entre as juntas trabalhadas (Figura 3.6).

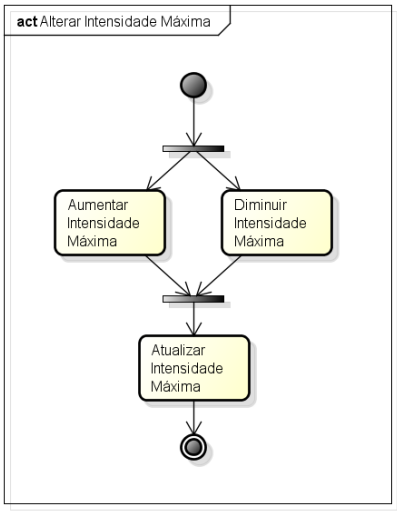


Figura 3.4 - Diagrama de Atividade - Alterar Intensidade Máxima do Exercício

Quanto à funcionalidade de iniciar simulação (Figura 3.7), a função deverá carregar os campos de forças e gerar o avatar com base na posição do paciente. Isso será feito utilizando as funções do Kinect e do Microsoft XNA.

Usando-o, o sistema entra em um *loop*. O *loop* deve comportar as funções que atualizam a posição do paciente, a pose do avatar, a força calculada e os gráficos do programa.

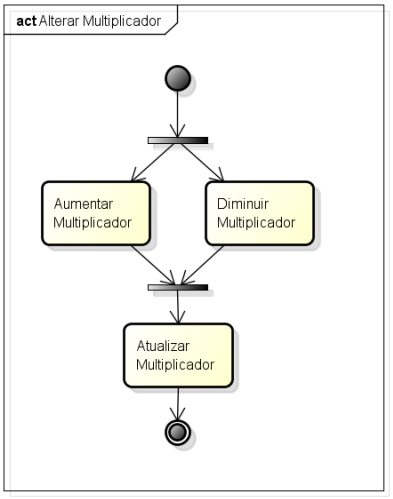


Figura 3.5 – Diagrama de Atividade - Alterar Multiplicador do Exercício

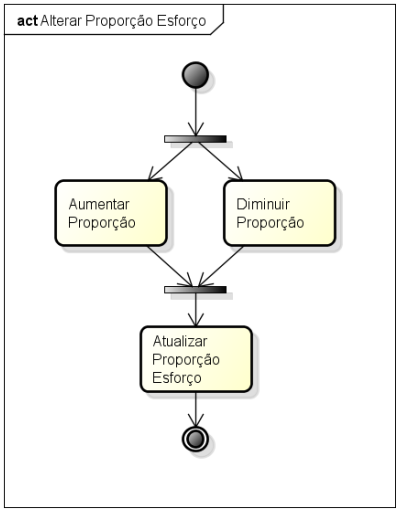


Figura 3.6 - Diagrama de Atividade - Alterar Proporção Esforço

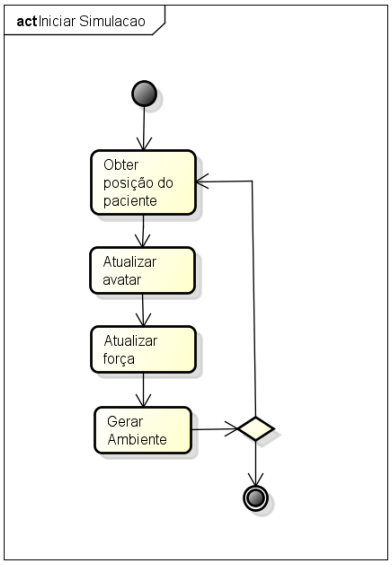


Figura 3.7 - Diagrama de Atividade - Iniciar Simulação

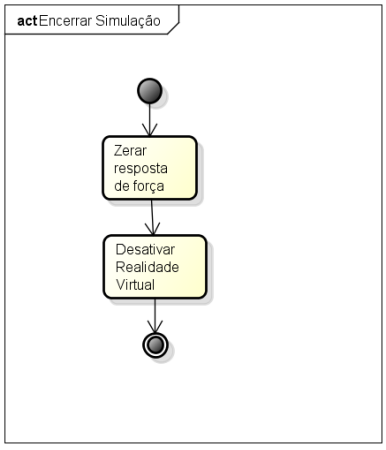


Figura 3.8 - Diagrama de Atividade - Encerrar Simulação

Quando se encerra a simulação (Figura 3.8), a resposta de força é zerada para impedir qualquer variação na saída que machuque o usuário. No final, a realidade virtual é desativada e todas as ferramentas são desativadas.

No encerramento do programa (Figura 3.9), o *software* desliga o Kinect e fecha a janela do programa.

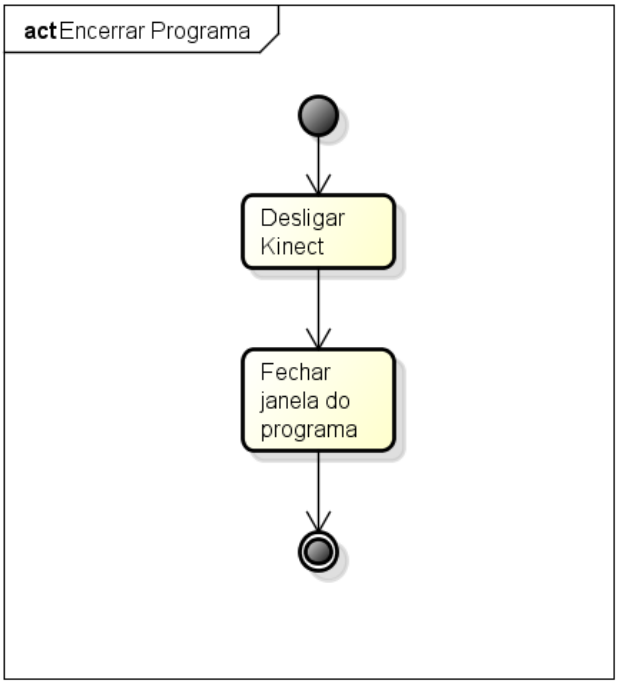


Figura 3.9 - Diagrama de Atividade - Encerrar Programa

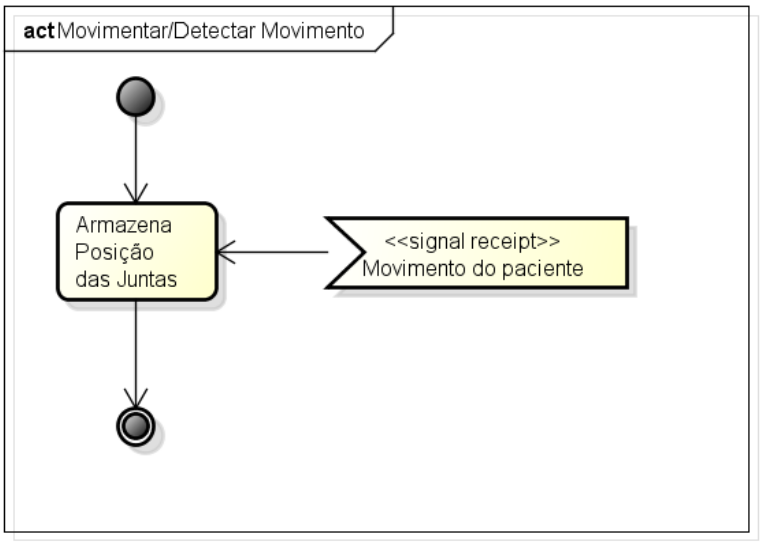


Figura 3.10 – Diagrama de Atividade - Movimentar/ Detectar Movimento

Para a movimentação do usuário, o sistema tem que receber os dados das juntas do usuário e armazená-las. O reconhecimento das juntas é feita pelo Kinect e enviada ao programa, onde é armazenado automaticamente.

3.3. Diagrama de Componentes

O Diagrama de componentes foi usado para representar o programa devido a características advindas do uso do XNA. Ao elaborar um projeto baseado em XNA é possível criar várias classes e identificar cada uma delas como componente e/ou serviço.

Essa declaração é feita da seguinte forma:

```
this.kinect = new KinectControlador(this,
ColorImageFormat.RgbResolution640x480Fps30,
DepthImageFormat.Resolution640x480Fps30);
this.Services.AddService(typeof(KinectControlador), this.kinect);
this.Components.Add(this.kinect);

this.camera = new Camera(this, new Vector3(0.0f, 0.0f, 20.0f),new Vector3(0.0f,
0.0f, 0.0f), new Vector3( 0.0f, 1.0f, 0.0f));
this.Services.AddService(typeof(Camera), this.camera);
this.Components.Add(this.camera);

this.avatar = new Avatar(this);
this.Components.Add(this.avatar);
this.Services.AddService(typeof(Avatar), this.avatar);

this.telaInicial = new TelaInicial(this);
this.Components.Add(this.telaInicial);
this.Services.AddService(typeof(TelaInicial), this.telaInicial);

this.chao = new Chao(this);
this.Components.Add(this.chao);

this.depthVideo = new DepthVideo(this);
this.Components.Add(this.depthVideo);

this.forca = new Forca(this);
this.Components.Add(this.forca);
```

Figura 3.11 – Fragmento de código – Inicialização de componentes e serviços

O XNA possui alguns métodos inerentes a seu *framework* que devem ser exploradas para seu máximo aproveitamento. Essas funções são mostradas na Figura 2.7. Referenciar um objeto como componente significa que a *framework* cuidará da execução dessas funções quando lhe for conveniente. Por exemplo o método Initialize() e LoadContent() de todos os objetos referenciados como

componente serão utilizados durante a inicialização do programa; e o método Draw() e Update() serão executados ciclicamente, assincronamente e automaticamente.

No XNA referenciar um componente como serviço ajuda a sincronizar os componentes do programa. Ao ser adicionado como serviço, torna-se muito simples acessar os atributos desse objeto a partir de qualquer outro objeto do programa. Para isso usa-se a seguinte declaração de objeto, variando a classe dele conforme desejado:

```
public Camera camera
{
    get
    {
        return (Camera)Game.Services.GetService(typeof(Camera));
    }
}
```

Figura 3.12 – Fragmento de código – Retorno de um serviço do tipo camera

A Figura 3.13 apresenta o diagrama de componentes. Na figura, componentes utilizados como serviço possuem a identificação $\ll Service \gg$. A função de cada componente será mostrada na seção 3.4.

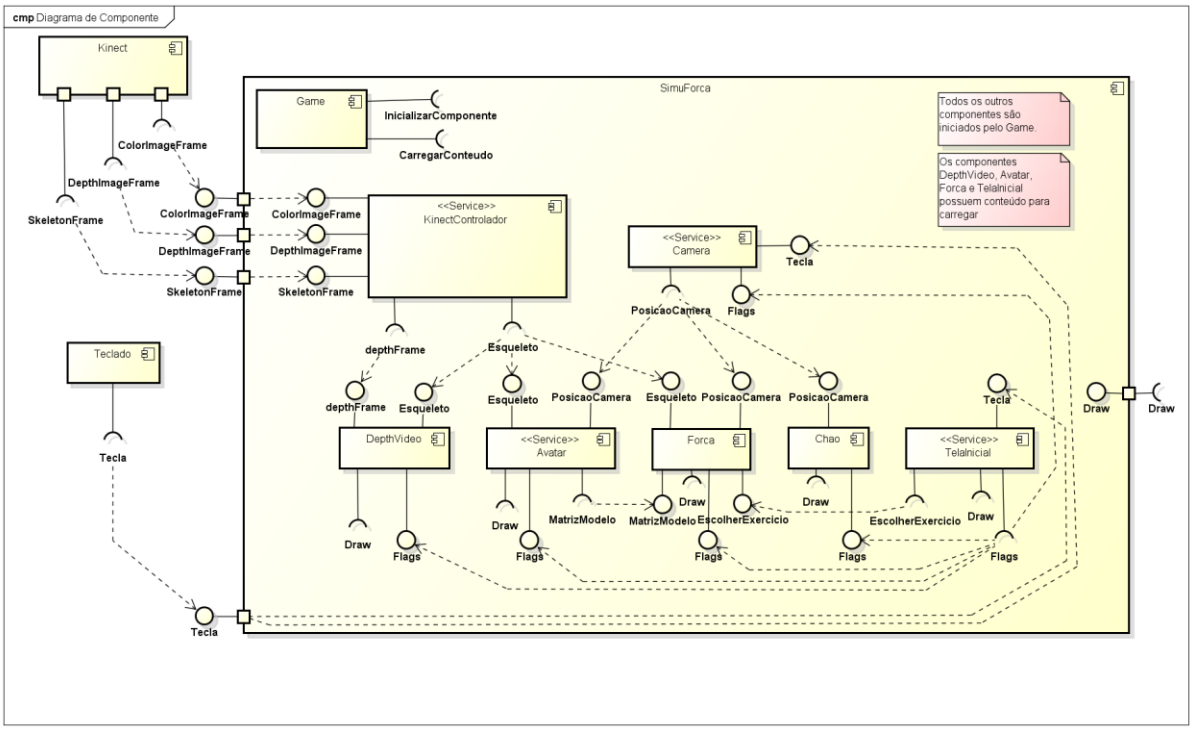


Figura 3.13 – Diagrama de Componentes

No programa existem 3 interfaces que se comunicam com o usuário, são eles o Teclado, o Kinect e uma tela (Monitor ou Televisão) onde o jogo será desenhado pelos métodos Draw(). O componente que interpreta os dados do Kinect é o KinectControlador, ele recebe os dados de profundidade, esqueleto e imagem RGB (este não usado) do Kinect. A TelaInicial faz uso do teclado para o controle do menu, e a câmera para o controle da posição de visão da RV.

A representação da RV do jogo é feita pela interação dos componentes DepthVideo, Avatar, Forca, Chao e TelaInicial.

Como o desenvolvimento do exoesqueleto não entra no escopo do projeto, nenhum componente envolvendo a comunicação com ele foi desenvolvida, uma vez que para o desenvolvimento dessa interface há a necessidade da especificação ser compatível com os dois lados da comunicação.

3.4. Diagrama de Classe

A seguir serão apresentados os diagramas de classe para cada componente apresentado na Figura 3.13. Como os componentes correspondem a uma classe do programa, a maioria dos diagramas de classe corresponderá às suas próprias classes, apresentando seus atributos e métodos.

Devido ao uso do XNA, grande parte da codificação será desenvolvida no interior dos métodos pertencentes a essa framework, assim pouco será possível interpretar pela leitura dos métodos do diagrama de classe, uma vez que a maior parcela dos enunciados serão métodos de auxilio ao XNA. Uma ideia da codificação dos métodos dos componentes mais relevantes será apresentada nas seções 4 e 5.

Nos diagramas a seguir, foi usado a codificação a seguir:

- + : public;
- - : private;
- # : private (service).

3.4.1. Game

O diagrama de classe do componente ***Game*** é apresentado na Figura 3.14. Essa classe é responsável pela inicialização de todos os objetos como componente, e alguns como serviço. Por isso, ele possui um objeto de cada classe que corresponderá a um componente, são eles: **spriteBatch**, **camera**, **kinect**, **avatar**, **chao**, **depthVideo**, **forca** e **telaInicial**. As classes SpriteBatch e

GraphicsDeviceManager pertencem ao XNA e são responsáveis pelo processamento gráfico do programa.

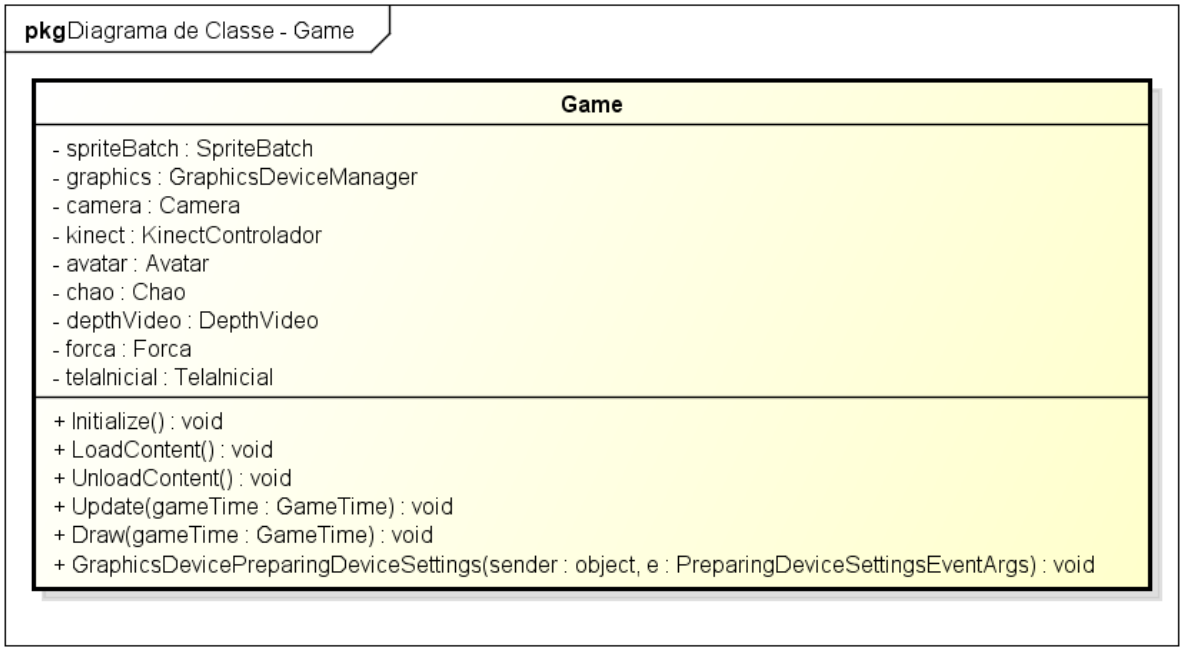


Figura 3.14 – Diagrama de Classe – Game

3.4.2. TelaInicial

O diagrama de classe do componente ***TelaInicial*** é apresentado na Figura 3.15. Esse componente é usado para gerar as telas de menu e controlar o programa através de *flags*. O objeto **fonte** carrega o tipo de fonte usada para a escrita do menu. A **flagTela** controla se deverá ser exibido o menu ou o ambiente de RV e a **flagEscolhendoExercicio** se deve ser exibido o menu principal ou o menu de configuração do exercício. Os atributos **flagCalibragem** e **flagExercicioEscolha** indica se o programa deverá, respectivamente, realizar a rotina de calibragem ou atualizar o exercício que foi escolhido. As informações do exercício são armazenadas nas variáveis **escolhaExercicio**, **intensidadeExercicio**, **maximoExercicio** e **propJuntas**. A variável **click** controla a opção que será selecionada.

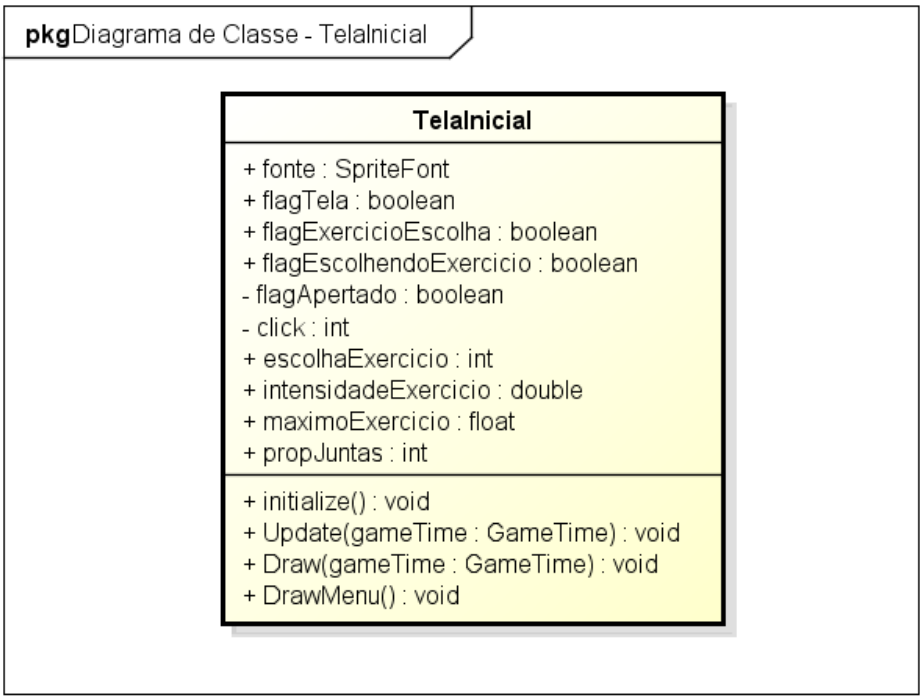


Figura 3.15 - Diagrama de Classe - TelaInicial

3.4.3. KinectControlador

O diagrama de classe do componente ***KinectControlador*** é apresentado na Figura 3.16. Esse componente é usado para armazenar os dados advindos do Kinect. Seus principais atributos são o **sensor**, que se refere e controla o funcionamento do Kinect; e **depthImagem**, **colorImagem** e **esqueleto**, que armazenam respectivamente os dados de profundidade, cor e juntas do usuário. Os atributos booleanos **novaImagemEsqueleto**, **novaImagemDepth** e **novaImagemColor** são usados para indicar a recepção de novos dados.

Junto com esses atributos, os principais métodos são **kinectSensor_ColorFrameReady**, **kinectSensor_DepthFrameReady** e **kinectSensor_SkeletonFrameReady**, onde é realizado a armazenagem de seus respectivos dados.

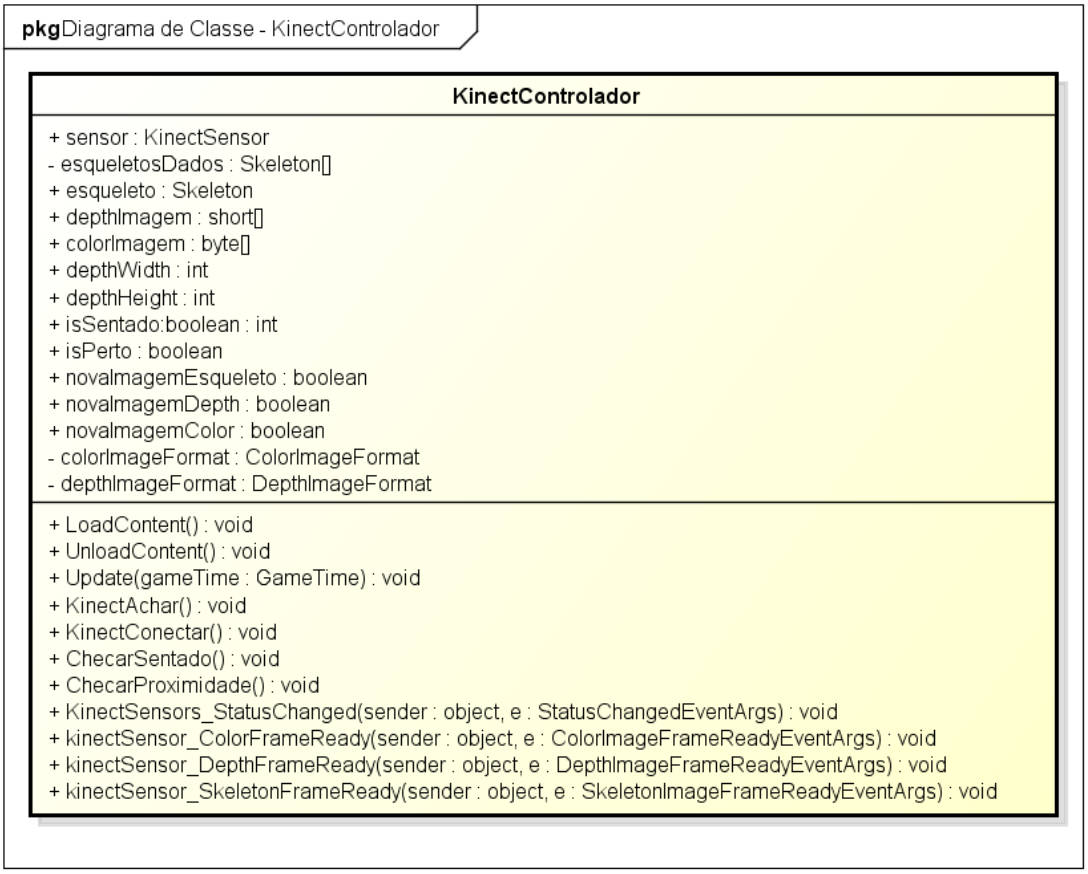


Figura 3.16 - Diagrama de Classe – KinectControlador

3.4.4. Camera

O diagrama de classe do componente ***Camera*** é apresentado na Figura 3.17. Esse componente é responsável por posicionar o ponto de vista que a tela do programa deverá utilizar. Essa informação é armazenada no atributo **view**. Essa matriz pode ser alterada através do teclado (implementado no método **Update()**). O serviço **telaInicial** é usado para indicar se o ponto de vista pode ser alterado pelo teclado.

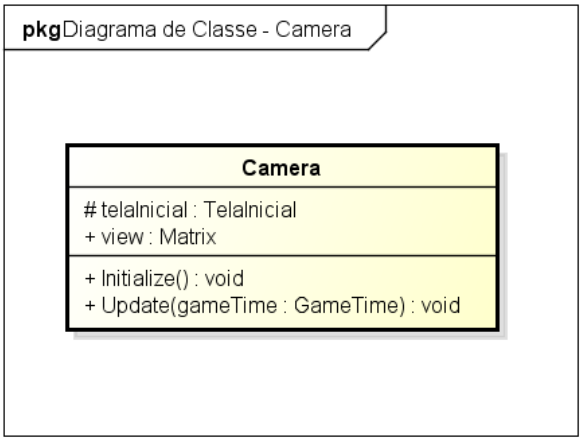


Figura 3.17 - Diagrama de Classe - Camera

3.4.5. Chao

O componente ***Chao*** é usado para o desenho do chão na RV, seu diagrama de classe é apresentado na Figura 3.18. Os serviços **telaInicial** e **camera** são usados para decidir se o chão deve ser desenhado e posicionar a visão do chão, respectivamente. A variável **effect** permite o desenho das linhas que compõem o chão. A variável **verticesChao** armazena uma posição e uma cor, podendo assim armazenar cada linha do chão e sua respectiva cor.

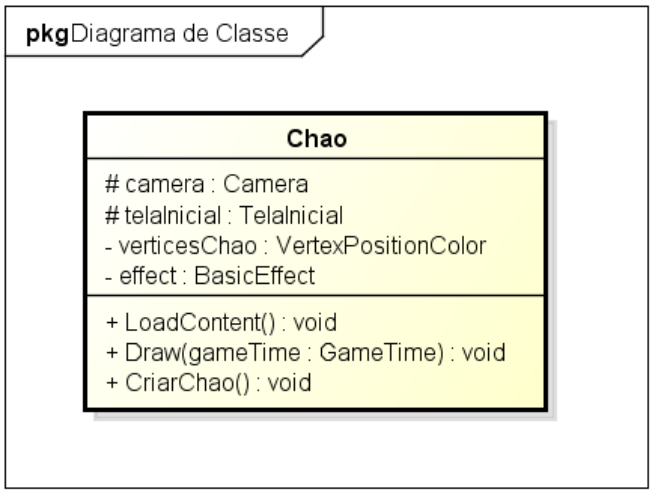


Figura 3.18 - Diagrama de Classe – Chao

3.4.6. Avatar

O componente ***Avatar*** será responsável pelo processamento e desenho do avatar na RV, seu diagrama de componente é apresentado na Figura 3.19. O serviço **camera** caracteriza o ponto de vista gerado. O serviço ***kinect***, em conjunto com o método **CopiarEsqueleto()**, é usada para pegar os dados de posição e orientação do usuário e armazená-los no **esqueleto**. O serviço **telaInicial** indica se a RV deve ser desenhada. O **dicionarioJuntaOsso** cria uma lista de relação entre as juntas do Kinect e as juntas do modelo 3D, armazenado no objeto **model**. As três transformadas (Pele, Global e Osso) armazenam informações de posição e orientação do modelo (mais será discutido na seção 4.3) e, junto com o **model**, são os resultados do processamento da variável **avatar** ao ser carregada com um modelo.

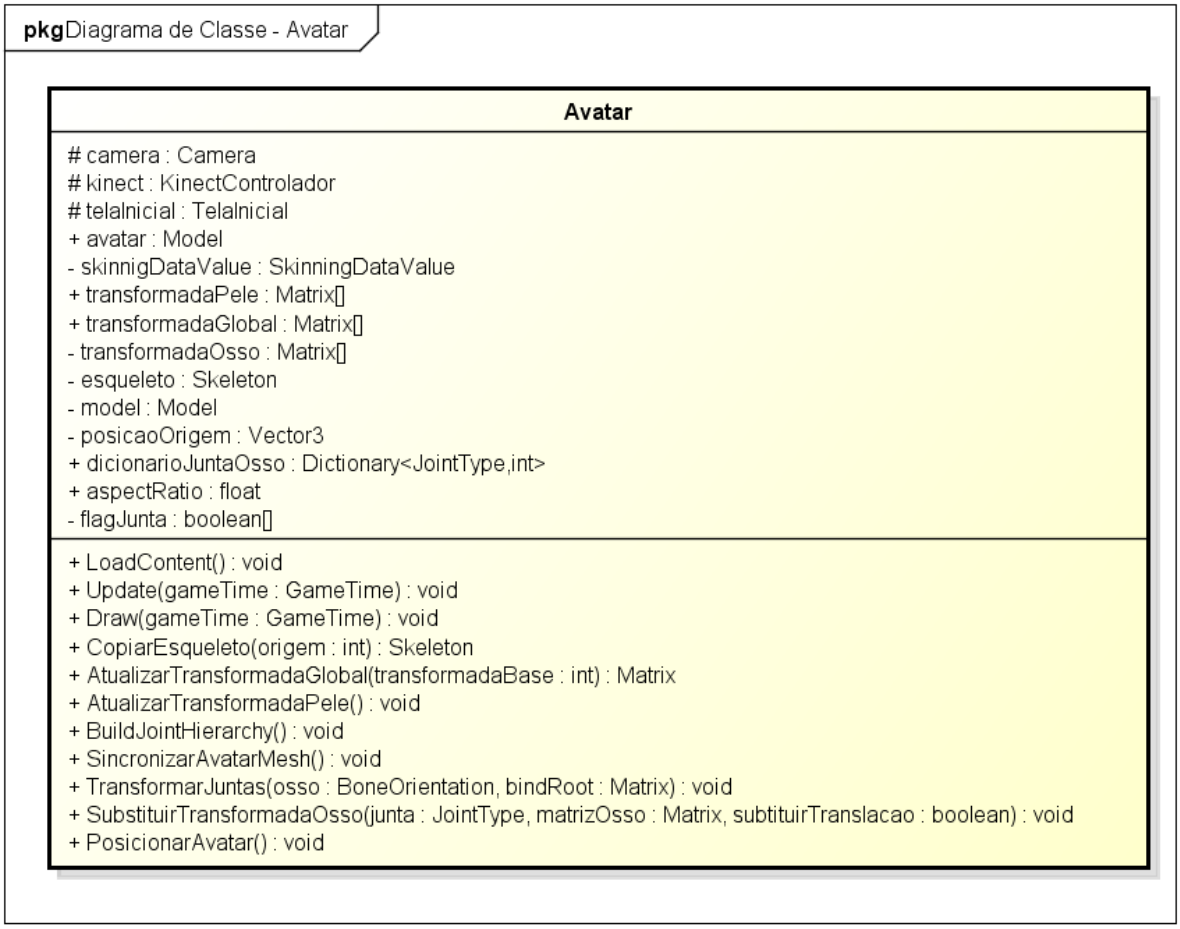


Figura 3.19 - Diagrama de Classe – Avatar

3.4.7. DepthVideo

Esse componente é usado para mostrar a filmagem da câmera de profundidade em tempo real na tela. A essa filmagem é adicionado um filtro que pinta as pessoas reconhecidas uma de cada cor, e desenha as juntas e ossos do primeiro usuário encontrado. Os dados do primeiro usuário reconhecido é armazenado no atributo **esqueleto** e os dados da câmera de profundidade no **depthImagem**, ambos dados são copiados a partir do serviço **kinect**. O **kinectDepthVisualizer** implementa o filtro aplicado a imagem para pintar as pessoas e as Texture2D armazenam o desenho que será usado para representar as juntas e ossos.

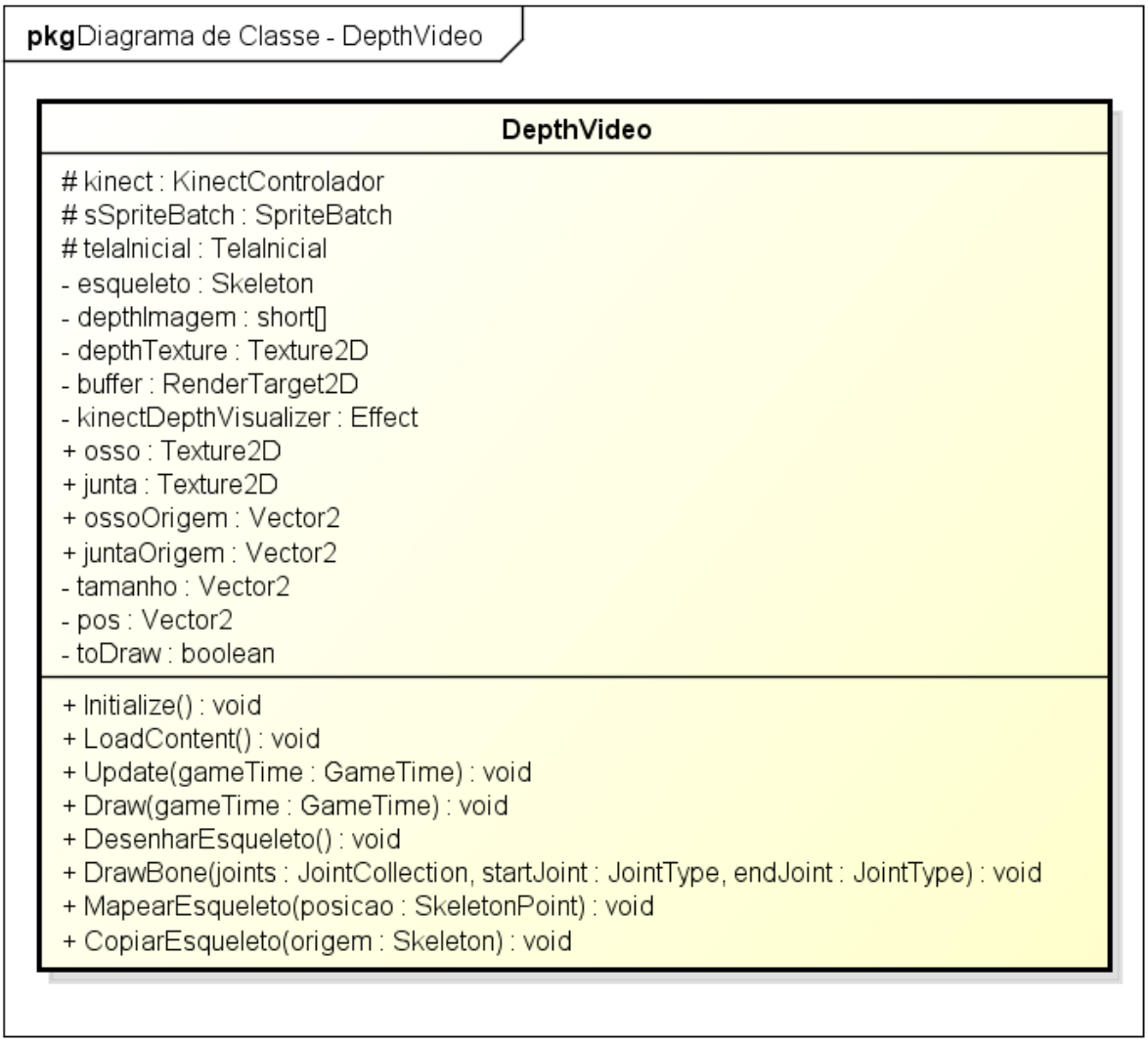


Figura 3.20 - Diagrama de Classe – DepthVideo

3.4.8. Forca

O componente ***Forca*** modela o campo de força, calcula a força aplicada a uma junta especificada e calcula e representa o torque existente na(s) junta(s) de interesse. A Figura 3.21 apresenta seu diagrama de classe.

A classe Forca utiliza 5 serviços: **camera**, para saber o ponto de vista; **kinect**, para obter os dados do usuário; **avatar**, para obter as matrizes global do modelo 3D na posição base e na posição atual; **telaInicial**, para saber se deve processar os dados e gerar o desenho; e **spriteBatch** para gerar as imagens 2D na tela (**barra** , **seta** e **seta2**).

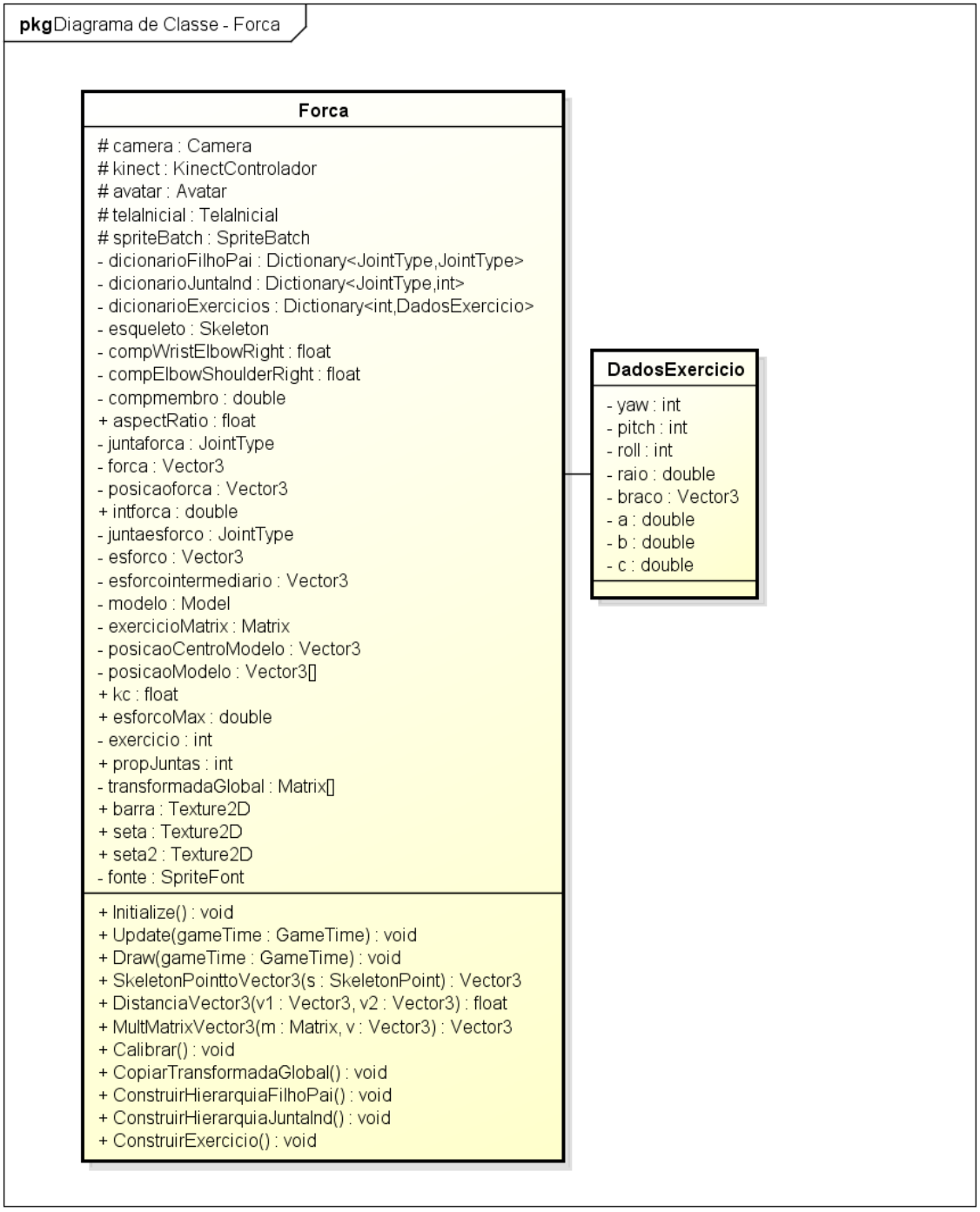


Figura 3.21 - Diagrama de Classe – Forca

A classe ainda utiliza um modelo 3D, sendo este representado várias vezes nas posições indicadas no *array* de vetor **posicaoModelo**, centrado na **posicaoCentroModelo**.

Os atributos **compWristElbowRight**, **compElbowShoulderRight** e **compMembro** são utilizados durante o método **Calibrar()** para a calibração do sistema. Os atributos **kc**, **forcaMax, exercício** e **propJuntas** são utilizado para

selecionar as características do exercício desejado. Esse atributo **exercício** caracteriza o exercício por meio da variável **dicionarioExercicios** que, com base no seu valor, o relaciona a um objeto da classe ***DadosExercicio***. Essa classe possui 8 atributos usados no calculo do campo de força.

Os cálculos envolvidos nessa classe serão mostrados na seção 4.4.

3.5. Diagrama de Sequência

O diagrama de sequência procura mostrar o encadeamento de métodos executados por uma ação do ator e prevista pelo diagrama de caso de uso.

Como parte do programa acontece de maneira autônoma, cada caso de uso será representado por 2 diagramas de sequência, sendo um do ponto de vista do usuário e o outro do ponto de vista do programa.

Além disso, como a maior parte do acesso e alteração das variáveis de uma classe é feita de forma direta, os métodos demonstrados não serão necessariamente os existentes nos diagramas de classe, e sim serão expostos de maneira mais literal, por exemplo, **Enter.Key()** se referirá a apertar a tecla *enter* do teclado e seta **flagCalibragem** se refere a mudar o valor dessa *flag* para *true*.

3.5.1. Calibrar

As Figura 3.22 e Figura 3.23 apresentam o diagrama de sequência para a calibração do sistema.

Quando o especialista pressiona o botão para começar a calibragem, ele muda a variável **flagCalibragem**.

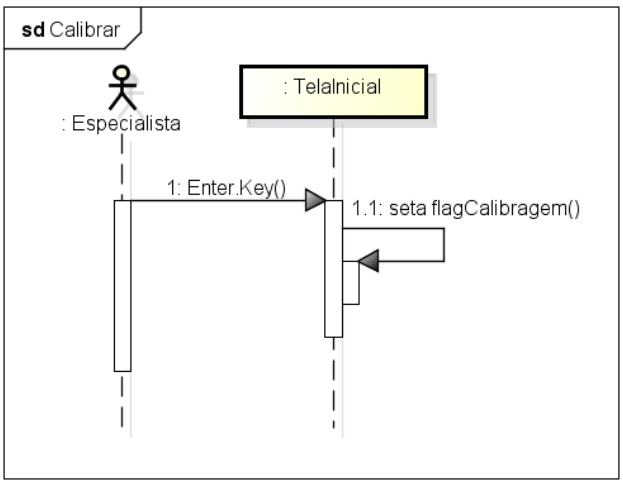


Figura 3.22 – Diagrama de Sequência - Calibrar Sistema (visão do usuário)

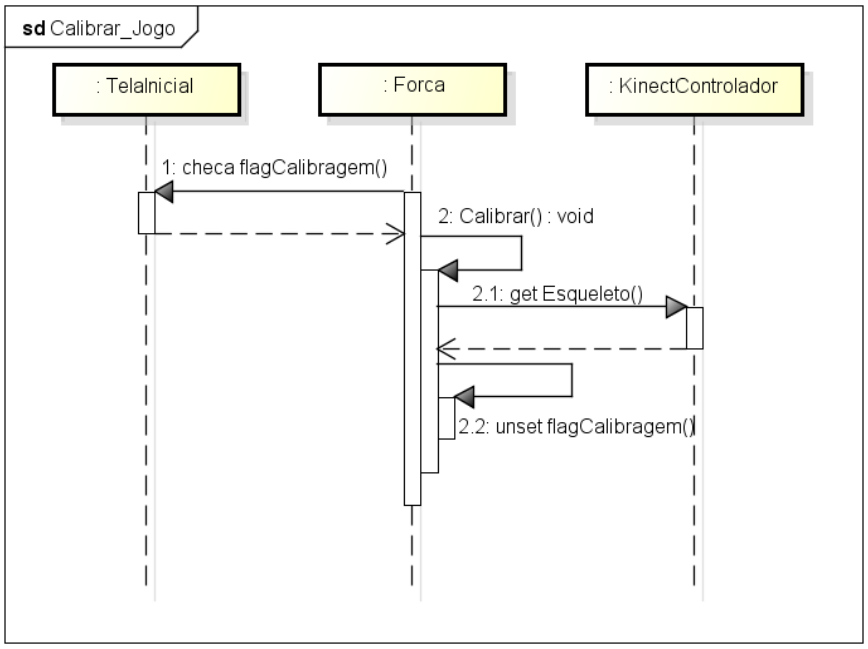


Figura 3.23 - Diagrama de Sequência - Calibrar Sistema (visão do jogo)

O sistema checa que essa **flagCalibragem** foi setada. Uma vez que ele reconhece o *set*, ele usa a função **Calibrar()**. A função **Calibrar()** pega as juntas do usuário, faz as medidas de comprimentos e muda a **flagCalibragem** para *false*, encerrando o processo.

3.5.2. Escolher Exercício

Esses diagramas abrangem os casos de uso de “Alterar Exercício”, “Alterar Intensidade Máxima”, “Alterar Multiplicador” e “Alterar Proporção Esforço”, e são apresentados na Figura 3.24. e na Figura 3.25.

Quando o usuário aperta *enter* na opção específica do menu inicial, ele seta a **flagEscolhendoExercício** e entra no menu de opções do exercício. Nesse menu ele tem a opção de apertar *baixo*, incrementando o valor de **click** e selecionando a linha de baixo; apertar *cima*, decrementando **click** e selecionando a linha de cima; apertar *direita*, aumentando o valor selecionado; *esquerda*, diminuindo o valor selecionado; ou apertar *enter* para mudar a **flagExercicioEscolha** para *true* e sair do menu de opções do exercício, ao mudar a **flagEscolhendoExercicio** para *false*.

A classe ***Forca*** checa a **flagExercicioEscolha** e, se for *true*, copia os valores das variáveis **exercicioEscolha**, **intensidadeExercicio**, **maximoExercicio** e **proJuntas** da classe ***TelaInicial*** para **exercicio**, **kc**, **esforcomax** e **propJuntas**, respectivamente, na classe Forca.

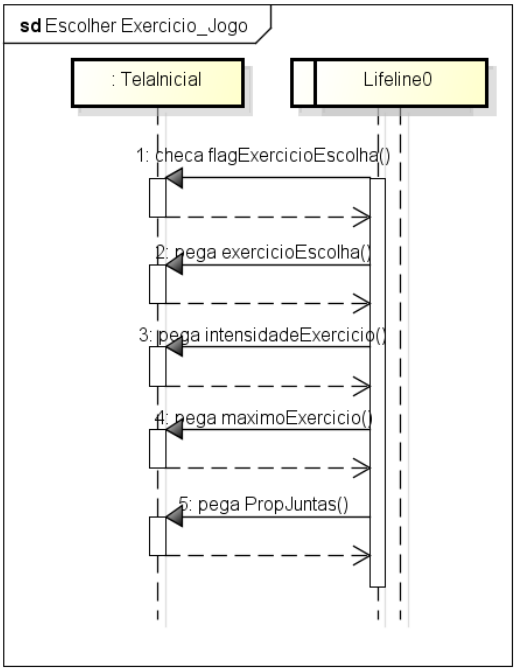


Figura 3.24 – Diagrama de Sequência - Escolher Exercício (visão jogo)

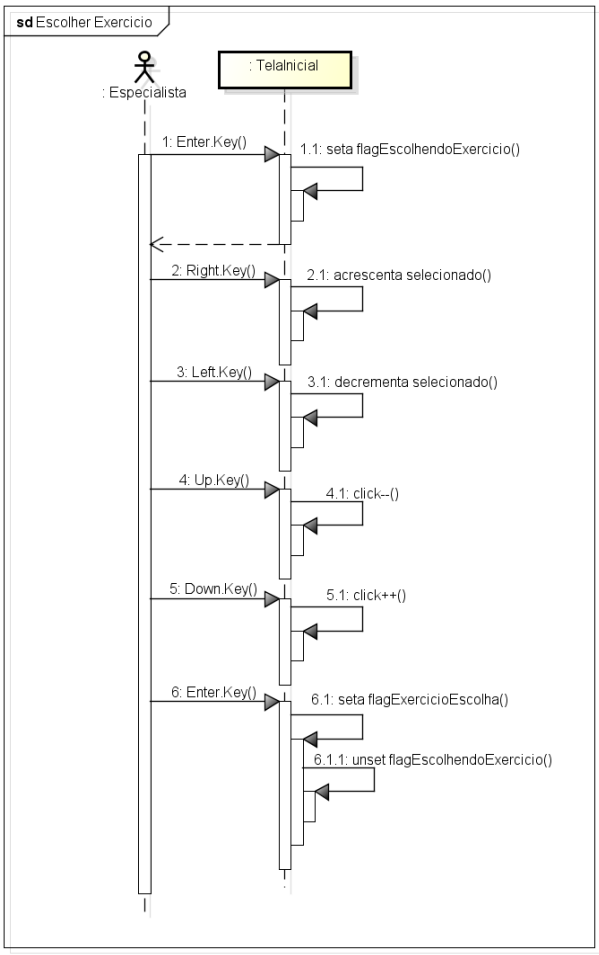


Figura 3.25 – Diagrama de Sequência - Escolher Exercício (visão usuário)

3.5.1. Iniciar Simulação

Iniciar a Simulação ocorre quando o usuário aperta *enter* na opção especificada do menu. Ao fazer isso, a flagTela() muda para *false*.

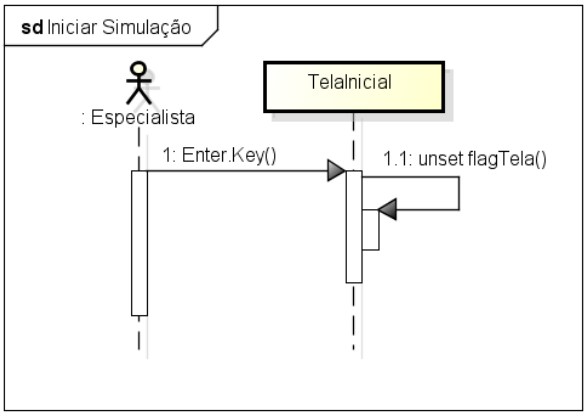


Figura 3.26 – Diagrama de Sequência - Iniciar Simulação (visão do usuário)

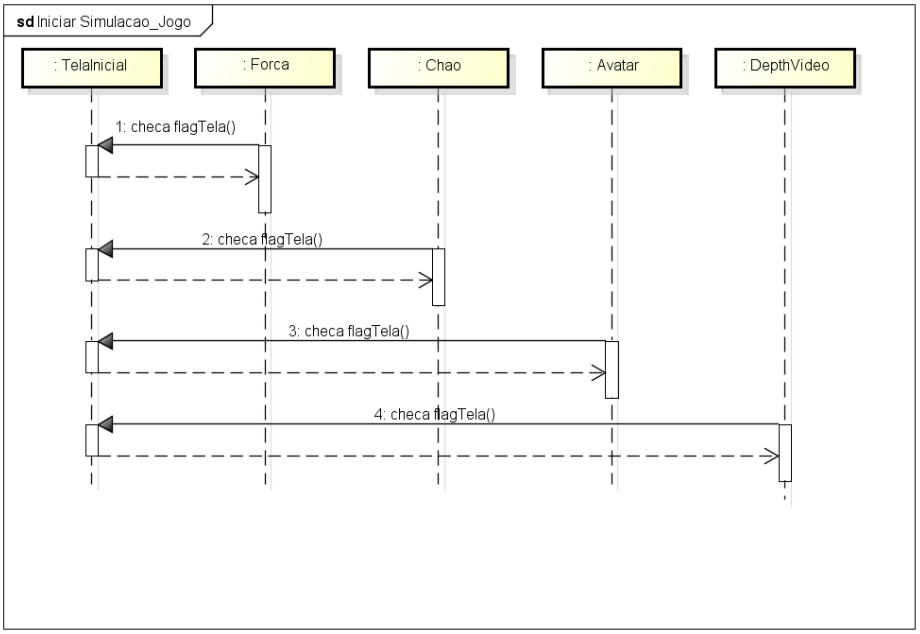


Figura 3.27 – Diagrama de Sequência - Iniciar Simulação (visão do jogo)

Assim, todas as classes que desenham algo leem a **flagTela** para, em caso de *false*, desenhar na tela o ambiente de RV.

O diagrama de sequência para esse caso de uso é apresentado na Figura 3.26 e na Figura 3.27.

3.5.2. Encerrar Simulação

O diagrama de sequência de Encerrar Simulação (Figura 3.28 e Figura 3.29) ocorre de modo semelhante ao de iniciar simulação. Durante a tela de realidade virtual, o usuário aperta *esc* e a **flagTela** muda para *true*.

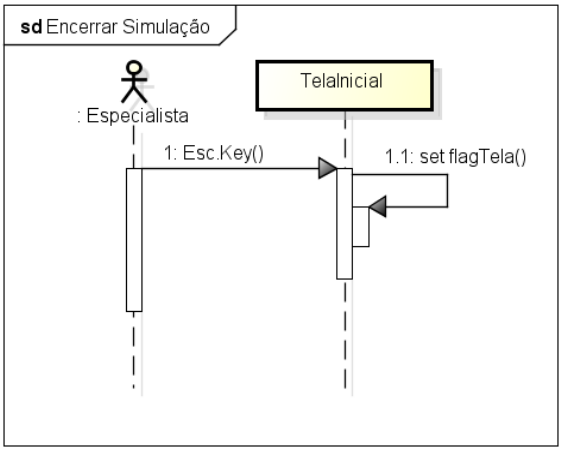


Figura 3.28 - Diagrama de Sequência - Encerrar Simulação (visão do usuário)

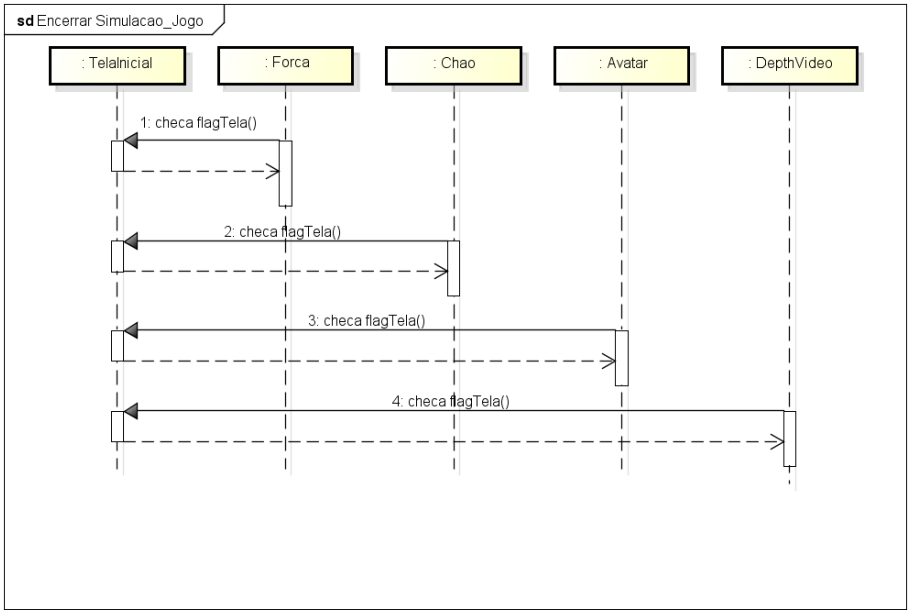


Figura 3.29 - Diagrama de Sequência - Encerrar Simulação (visão do jogo)

As classes que desenham a RV checam essa *flag* e param de fazer suas representações na tela. Em contrapartida, a *TelaInicial* volta a desenhar o menu.

3.5.3. Encerrar Programa

O encerramento do programa ocorre quando o usuário aperta *enter* em determinada função do menu. O sistema é encerrado e a janela é fechada. O XNA

cuida do descarregamento do jogo da memória e da desativação de cada componente por meio do método UnloadContent(). Seu diagrama de sequência é apresentado na Figura 3.30.

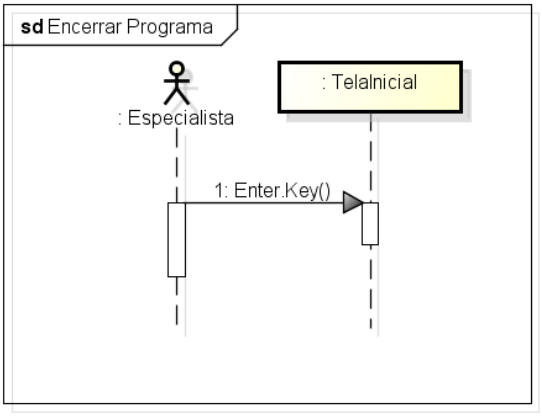


Figura 3.30 - Diagrama de Sequência - Encerrar Programa

3.5.4. Movimentar/Detectar Movimento

Esse diagrama ocorre a todo momento em que o Kinect está ligado. O Kinect captura a imagem que ele enxerga (com o paciente), processa e envia para o KinectControlador armazenar a informação por meio de um dos 3 métodos mostrados na Figura 3.31. Essa informação inclui o movimento do usuário.

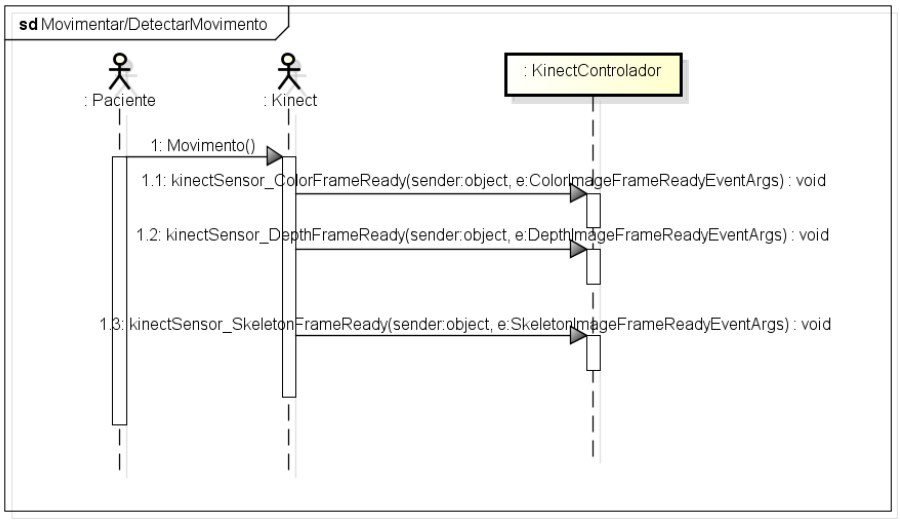


Figura 3.31 - Diagrama de Sequência - Movimentar/Detectar Movimento

4. Implementação

A implementação da solução pode ser pensada como a união de várias camadas.

A camada mais externa, que faz a coleta de dado do usuário, é o Microsoft Kinect. Seus dados são recebidos e manipulados pelo framework XNA, em conjunto com o Kinect SDK, dentro do Visual Studio. Também implementado pela combinação XNA e Visual Studio, um modelo 3D foi criado como avatar para representar gráficamente as mudanças de posição do usuário. Por fim, na parte mais interna do programa, é feito o cálculo do esforço que deve ser empregada a uma junta especificada, que será externado para o usuário na tela.

4.1. Microsoft Kinect

A codificação envolvendo o Kinect foi focada na tradução dos dados enviados por ele para o computador, de forma que ficasse mais fácil entender e manipular seus dados. Foi codificado a obtenção dos dados das câmaras RGB e infravermelho e o rastreamento de pessoas. A captação de dados foi elaborada de forma a ser feita por eventos, sendo acionado o respectivo método pelo envio de dados do Kinect, e não por uma chamada em alguma etapa do código.

Somente será usada no projeto a câmera infravermelha, para captar a profundidade, e o rastreamento de pessoas. O próprio Kinect é capaz de procurar e identificar os usuários e captar a posição das juntas indicadas na Figura 2.6 - Juntas encontradas pelo Kinect . Essa informação é armazenada numa estrutura chamada *Skeleton*, pertencente a SDK do Kinect.

4.2. Modelo 3D

Um modelo 3D manipulável por *frameworks* de jogos e animações (como o XNA) deve ser composto por no mínimo 2 elementos: a *waveform* e o esqueleto. Além disso, esses 2 elementos devem ser unidos por um processo que denominaram “pintura de peso” (*weight paint*). O programa usado para a manipulação do modelo 3D foi o Autodesk Maya. Esse programa foi escolhido por causa do seu sistema de coordenadas, compatível com as do XNA. Esse software é bastante usado em toda a indústria de entretenimento, desde séries de desenhos como o South Park, ou em jogos como o Resident Evil, até no cinema como no filme Avatar e em animações como em Shrek.

A *waveform* trata-se da parte externa do modelo, que o confere formato. Trata-se apenas de uma superfície formada por arestas modelada de forma cuidadosa a fim de parecer com o objeto, personagem, animal ou pessoa desejada, em um processo muito parecido com a escultura tradicional. Nessa “casca” pode ser adicionado um *template* com textura, conferindo cor ao objeto, enriquecendo-o de detalhes e permitindo identificar partes separadas do mesmo.

Como a escultura de avatares não é o escopo do trabalho, escolheu-se por descarregá-la da internet. Não é difícil encontrar sites de compartilhamento com modelos de personagens de filmes, desenhos ou jogos. A escolha do personagem foi feita de forma a ser atrativo para uma grande gama de público e idade, assim foi escolhido o personagem Homer Simpson do programa da Fox “Os Simpsons”, mostrado na Figura 4.1 - Modelo 3D do Homer, respectivamente na forma de arestas, superfície e com textura..[1]

Figura 4.1 - Modelo 3D do Homer, respectivamente na forma de arestas, superfície e com textura.

A Figura 4.1 mostra três imagens. A esquerda é a *waveform* do modelo. A figura central é o modelo conferido de uma textura padrão. E por fim, a imagem da direita é o modelo conferido de sua textura.

A partir desse conjunto *waveform*/textura, deve-se construir o esqueleto do objeto. O esqueleto fornece a estrutura básica de manipulação do modelo. A unidade fundamental do esqueleto é o osso, uma estrutura “direcional” composta por duas juntas e um segmento. O osso pode ser considerado direcional, pois é criada uma hierarquização durante o seu processo de confecção, assim ele possui uma orientação partindo do final do osso anterior ou um ponto de origem e indo para o posterior ou para uma junta na extremidade. Na Figura 4.2 - Esqueleto

[1] http://tf3dm.com/3d-model/simpsons-hit-and-run-homer-71869.html

confeccionado para o modelo e sua superposição à superfície pode ser observada essa hierarquização através da representação por cores. A junta base usada como origem foi a do centro do quadril.

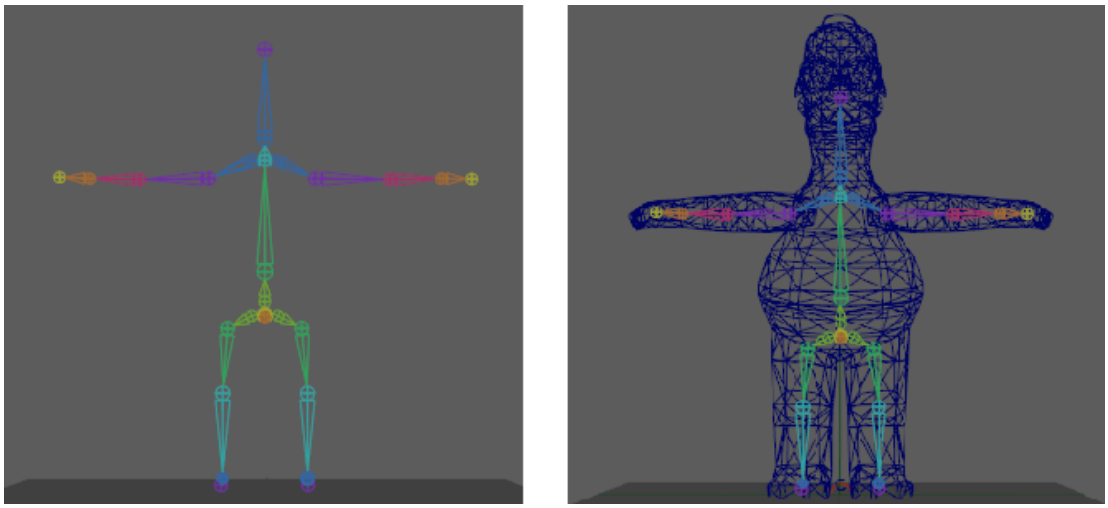


Figura 4.2 - Esqueleto confeccionado para o modelo e sua superposição à superfície

A confecção do esqueleto deve ser feito de forma tão detalhada quanto maior a complexidade do movimento a ser representado. Neste trabalho, o esqueleto foi feito de forma a ser compatível com as juntas do Kinect. O maior detalhamento do esqueleto não acrescentaria maior complexidade ao movimento, pois essas juntas não seriam controladas, mas poderia melhorar o seu detalhamento.

Ao fim do processo de construção do esqueleto, tem-se que transformar o objeto em uma estrutura coesa. O *Paint Weight* é a ferramenta usada para ligar a *waveform* ao esqueleto. Essa ferramenta deverá ser aplicada a cada um dos ossos, acoplando-os a uma parte da *waveform*. O nome *paint* se dá pela representação do processo, no qual se baseia na pintura da *waveform*; e o *weight* à capacidade de atribuir pesos à deformação sofrida pela *waveform* durante o movimento, essa representação é feita por meio de um degradê. Um exemplo de *paint weight* pode ser visto na Figura 4.3 - Paint Weight do centro do quadril.

Após o *Paint Weight*, o deslocamento de um osso acarretara na mudança da *waveform*, executando um movimento como visto na Figura 4.4 - Exemplo de movimento.

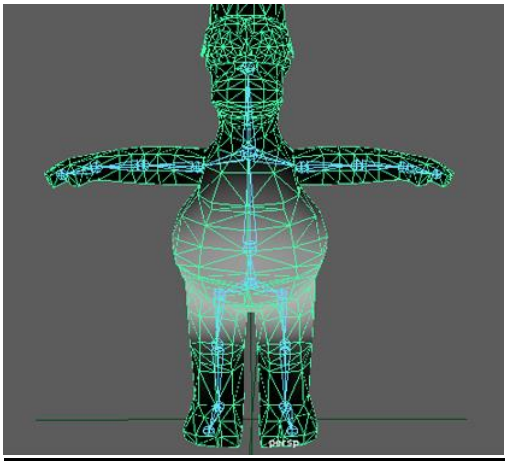


Figura 4.3 - Paint Weight do centro do quadril.

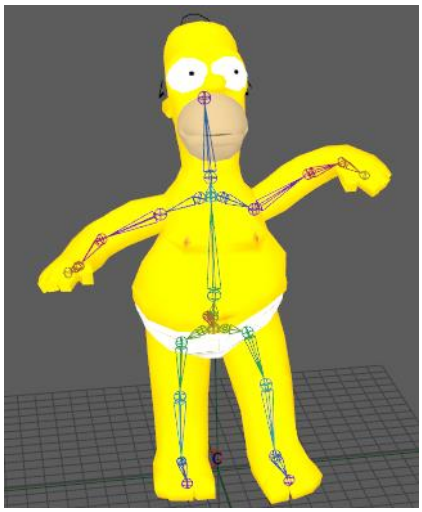


Figura 4.4 - Exemplo de movimento

Além do modelo do Homer, foi usado um modelo estático, composto somente pela *waveform*, no formato de um disco (Figura 4.5), representando as roscas de padaria, famosas pelo desenho em questão.

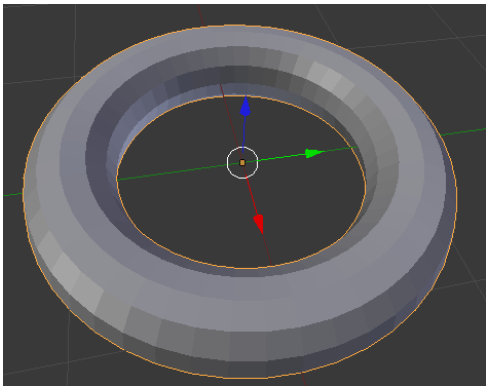


Figura 4.5 - Modelo do disco

4.3. Avatar

O processamento do modelo 3D do avatar gera 3 conjunto de matrizes, identificadas como **transformadaGlobal**, **transformadaOsso** e **transformadaPele** no diagrama de classes na seção 3.4.6. A **transformadaGlobal** possui a orientação e posição de uma junta em relação à origem; a **transformadaOsso** possui a orientação da junta em relação à junta anterior (seu "osso pai"); e a **transformadaPele** modifica a posição da *waveform* do Homer, ou seja, a modificação desta acarreta na modificação visual do modelo.

Para o sincronismo do modelo ao usuário é necessário utilizar as informações advindas do Kinect para modificar essas matrizes. Para manter as proporções do corpo do modelo (tamanho entre as juntas) foi utilizadas matrizes de orientação e rotação para a manipulação das matrizes do modelo, sem levar em conta a posição em si das juntas externadas pelo Kinect. As orientações locais das juntas do usuário são externadas pela SDK do Kinect.

Para a manipulação do Homer é necessário que cada junta do Kinect seja relacionada a uma junta do esqueleto. Essa relação deve ser feito de forma que a junta do Kinect seja referenciada à junta inicial do osso cuja junta final é a pretendida no do esqueleto. Por exemplo, no caso do braço direito e considerando um esqueleto no modelo com juntas no peito, ombro, cotovelo, pulso e mão ligadas por estruturas de osso, as juntas do Kinect referentes ao ombro direito, cotovelo direito, pulso direito e mão direita seriam ligados respectivamente às juntas peito, ombro, cotovelo e pulso do esqueleto.

Para o equacionamento a seguir, o termo "junta" se refere a uma junta do Kinect e o termo "osso" a um osso existente no modelo. A numeração desses dois elementos não são necessariamente os mesmos.

Assim, o primeiro passo para a manipulação do avatar será calcular as novas transformadas locais dos ossos. A manipulação genérica será mostrada nas expressões a seguir; a junta inicial possui tratamento diferente, pois sua posição estabelece o local onde o modelo será representado, então essa informação deve ser adicionada.

$$[transformadaOsso]_{junta(osso)} = [ajuste]_{junta} * [TrocaDeEixos]_{junta} * [Kinect]_{junta}{}^{2}$$

A matriz “Kinect” possui a orientação da respectiva junta observada por esse *hardware*. A matriz de “TrocaDeEixos” deve rotacionar os eixos do Kinect de forma a coincidir com o do modelo. Essa troca é realizada através de quaternions, devido à facilidade da operação; essa diferença entre os eixos do modelo e do Kinect pode ser observados na Figura 4.6. A matriz de “ajuste” se trata de uma matriz de rotação composta, e foi manipulada de forma a tornar a posição do modelo mais agradável em comparação com a do usuário (por exemplo, quando a cabeça do usuário estiver na vertical, a do modelo também deve estar). Durante a atribuição do resultado à **transformadaOsso** pode-se preservar seu valor de translação.

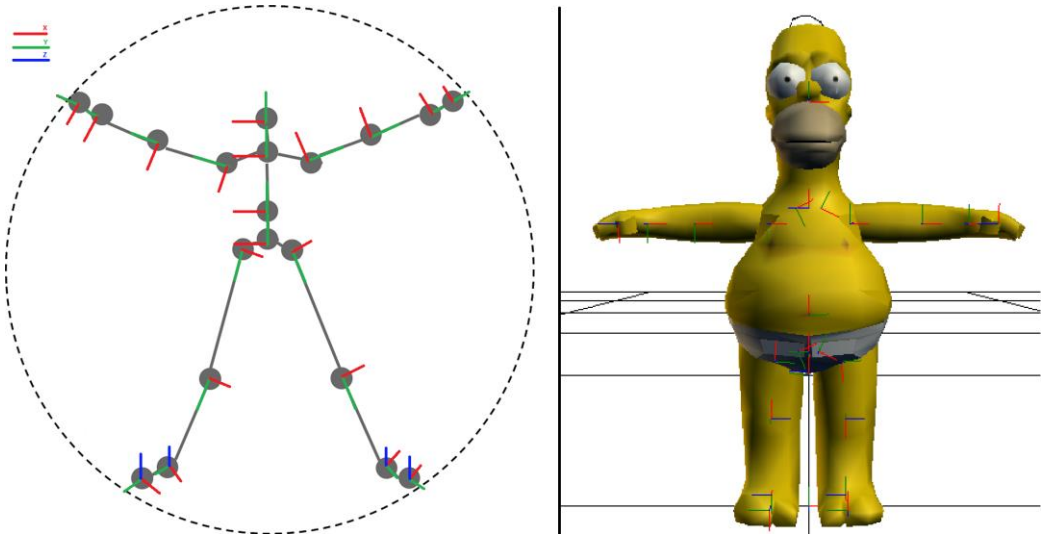


Figura 4.6 - Comparação entre o sistema de coordenadas do Kinect (esquerda) e do modelo (direita)

Sabendo-se as transformadas locais de cada osso, é possível calcular a sua **transformadaGlobal**.

$$[transformadaGlobal]_{osso} = [transformadaGlobal]_{ossoPai} * [transformadaOsso]_{osso}$$

Por fim calcula-se a **transformadaPele** que movimentará o modelo.

$$[transformadaPele]_{osso} = [transformadaGlobal]_{osso} * [transformadaAbsoluta]^{-1}_{osso}$$

[2] O equacionamento apresentado neste trabalho utiliza vetores em coluna e matrizes do tipo $\begin{bmatrix} R_{3x3} & T_{1x3} \\ 0_{3x1} & 1 \end{bmatrix}$. O *framework* XNA convenciona vetores em linha e matrizes $\begin{bmatrix} R^T_{3x3} & 0_{1x3} \\ T^T_{3x1} & 1 \end{bmatrix}$ em suas operações.

A matriz de “transformadaAbsoluta” pertence ao modelo.

4.4. Campo de Força

Para o campo de força, optou-se por modelá-lo como um campo elástico em torno de uma superfície elipsoidal, utilizando o modelo anelar (Figura 4.5) como representação dos pontos de referencia da localização desse elipsoide na RV.

A superfície elipsoidal possui uma modelagem simples e com uma grande variedade de formatos, e por isso foi escolhida.

Como o avatar deve ser capaz de alcançar pontos dessa superfície, seu posicionamento foi feito com base nos ângulos e tamanhos dos membros de interesse do avatar em sua posição inicial. O uso de ângulos ainda evita que deslocamentos do usuário influenciem no exercício, tornando o sistema mais consistente.

Como visto na seção 3.4.8, os dados característicos do exercício são armazenados na classe ***DadosExercicios*** e inicializados no **dicionarioExercicios**. Para finalizar a caracterização do campo de força é ainda necessário atribuir valores para **kc**, **compWristElbowRight**, **compElbowShoulderRight** e **compMembro**, isso é feito durante a calibração. No qual utiliza as posições do ombro direito, cotovelo direito e pulso direito do usuário para calcular a distancia entre eles, além de um valor base para **kc** que pode ser alterado nas configurações do exercício. Assim:

$$\begin{cases} compWristElbowRight = posWristRight - posElbowRight \\ compElbowShoulderRight = posElbowRight - posShoulderRight \\ comMembro = compWristElbowRight + compElbowShoulderRight \\ kc = \dfrac{kc_{base} * compMembro}{compBracoHomer} \end{cases}$$

Para posicionar o campo de força foi necessário ainda especificar duas juntas do modelo, uma para onde será calculada a força e uma para onde deve ser imposta a rigidez. Para obter a posição dessas juntas no modelo foi usada as matrizes de **transformadaGlobal** na posição inicial do avatar. Para o trabalho, as juntas utilizadas foram respectivamente o pulso direito e o ombro direito. Não foi utilizada a mão direita, pois a localização do pulso se mostrou mais eficiente.

Nos equacionamentos a seguir, o termo “rigidez” se referirá à junta de rigidez (ombro direito), e “força”, à junta de força (pulso direito). O calculo da posição do centro da elipsoide é apresentado a seguir:

$$[\overrightarrow{posicaoCentroModelo}] = [translacao]_{rigidez} * [exercicio] * [\overrightarrow{posicaoForca} - \overrightarrow{posicaoRigidez} - \overrightarrow{braco}]$$

$[\overrightarrow{posicaoForca} - \overrightarrow{posicaoRigidez} - \overrightarrow{braco}]$ trata-se de um vetor que indica a direção e distancia entre os dois pontos de interesse (junta da forca e de rigidez), decrescido de um vetor braço especificado nas configurações do exercício. A matriz de exercício aplica uma rotação a esse vetor com base no *roll-pitch-yaw* especificados na configuração do exercício; e a matriz de translação insere o centro de giro em cima da junta onde se calcula a rigidez alvo. Com essa multiplicação é possível calcular uma posição para centrar a elipsoide de forma que esse ponto use o comprimento do braço como referencia.

Assim, para um pré-calculo da força faz-se:

$$\overrightarrow{força} = kc * [\overrightarrow{posicaoCentroModelo} - \overrightarrow{posicaoForca}]$$

“Deforma-se” essa força usando-se os parâmetros **a**,**b** e **c** do exercício.

$$\begin{cases} X_{\overrightarrow{força}} = X_{\overrightarrow{força}} * a \\ Y_{\overrightarrow{força}} = Y_{\overrightarrow{força}} * b \\ Z_{\overrightarrow{força}} = Z_{\overrightarrow{força}} * c \end{cases}$$

Calcula-se o módulo dessa força “deformada” e aplica-se um offset (**raio**).

$$intforça = \left\|\overrightarrow{força}\right\| - kc * raio$$

E, por fim, para se calcular um vetor força que aplica um campo elástico em torno de uma superfície elipsoidal, normaliza-se o vetor força que se tem até agora e multiplica-o pela $intforca$ calculada:

$$\overrightarrow{força} = \frac{\overrightarrow{força}}{\left\|\overrightarrow{força}\right\|} * intforça$$

Para o calculo do esforço em torque, primeiro faz-se:

$$\overrightarrow{esforco} = (compElbowShoulder\left[\overrightarrow{posicaCotovelo} - \overrightarrow{posicaoOmbro}\right] + compWristElbow\left[\overrightarrow{posicaPulso} - \overrightarrow{posicaoCotovelo}\right]) \times \overrightarrow{forca}$$

Em seguida rotaciona-se esse vetor, de forma a tornar seus eixos base paralela aos eixos da junta de esforço (ombro), para isso utiliza-se a matriz inversa da transformada global do ombro.

$$\overrightarrow{esforco} = [rotacão]_{juntaesforço} * \overrightarrow{esforco}$$

Assim, a distribuição desse resultado entre o ombro e o cotovelo será mostrada a seguir. Essas duas juntas foram modeladas como tendo 2 graus de liberdade, para a determinação do nome, foi utilizado os eixo da base da própria junta baseada no Kinect (*X* e *Z*).

$$\begin{cases} \tau_{ombro,x} = \dfrac{propJuntas}{100} * X_{\overrightarrow{esforco}} \\ \tau_{ombro,z} = Z_{\overrightarrow{esforco}} \\ \tau_{cotovelo,x} = (1 - \dfrac{propJuntas}{100}) * X_{\overrightarrow{esforco}} \\ \tau_{cotovelo,x} = Y_{\overrightarrow{esforco}} \end{cases}$$

O valor de **propJuntas** indica a distribuição do esforço entre as duas juntas em questão e varia entre 0 e 100. Valores mais próximos de 100 aumenta o esforço no ombro e diminui no cotovelo. Essa variável pode ser ajustada durante a configuração do exercício.

O **esforço** é *cal*culado em torque, uma vez que as juntas do corpo humano executam movimentos rotativos e, ao contrário de uma força, é possível aplicar e controlar um torque de forma mais precisa, sem aplicar uma alavanca, usando o membro anterior e posterior de contrapoio.

5. Demonstração do Protótipo

Quando o jogo é aberto, ele tem o aspecto mostrado na Figura 5.1. Todo o menu está descrito na classe ***TelaInicial*** e documentada na Seção 3.4.2. O funcionamento de cada opção é mostrado durante a seção 3.5.

O “Iniciar o Jogo” começa a simulação e vai para a tela da Figura 5.3. O “Calibrar” executa a sequencia da Figura 3.22. O “Escolher Exercício” vai para a tela de configuração do programa (Figura 5.2). “Sair do Jogo” encerra o programa.

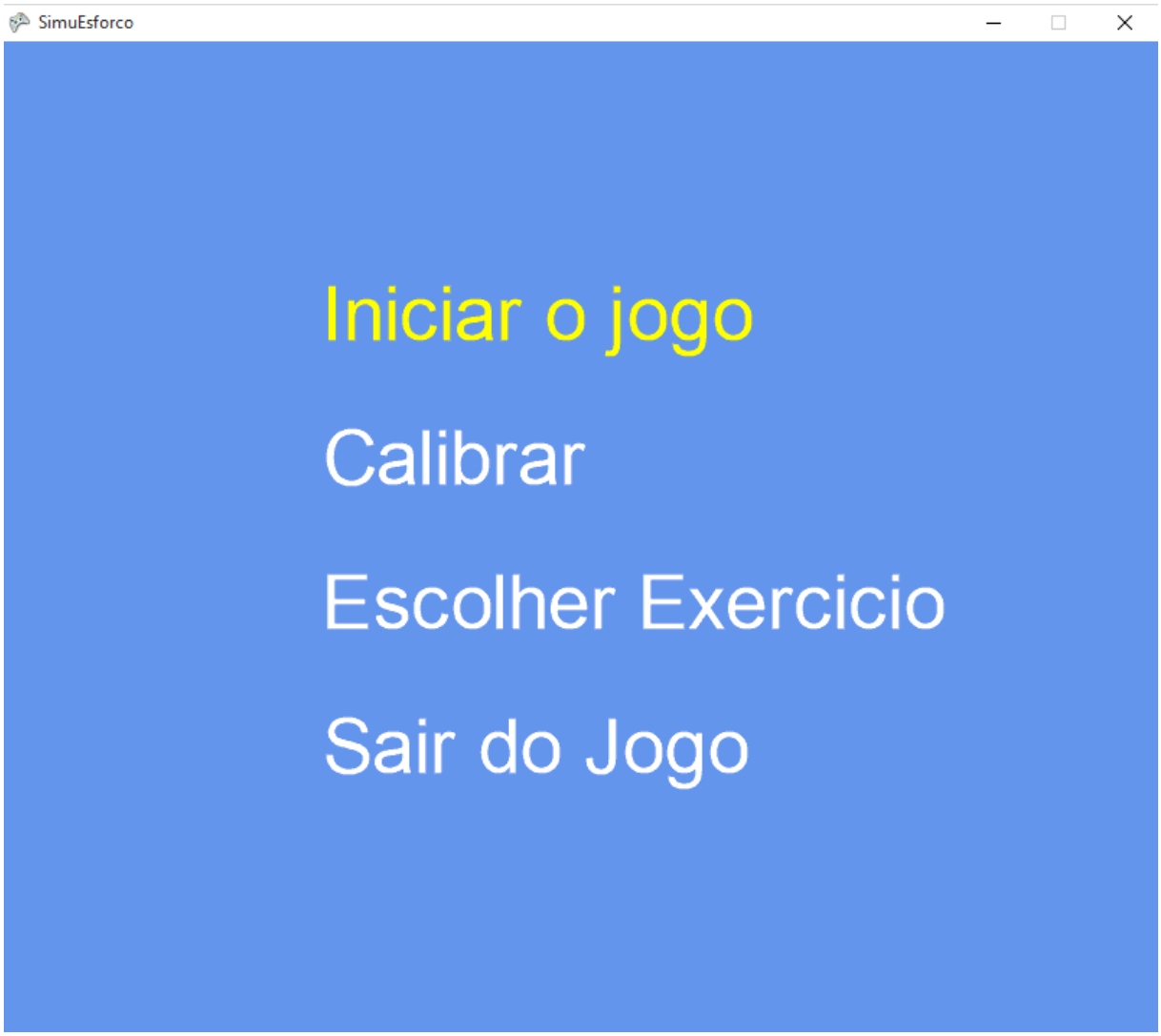


Figura 5.1 – Tela Incial do Jogo

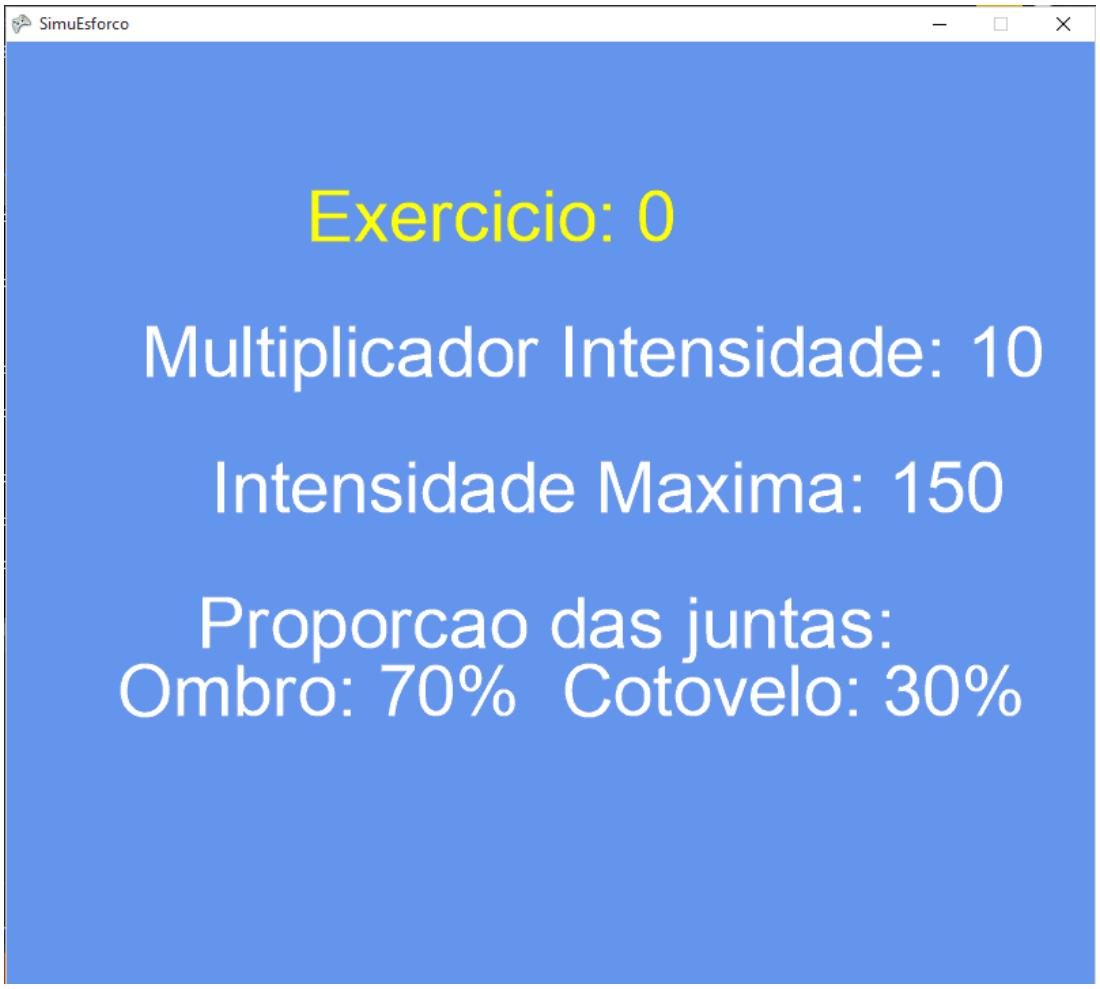


Figura 5.2 – Tela de Configuração do Exercício

A tela de configuração do exercício (Figura 5.2) permite que o usuário escolha uma das configurações de exercício, a intensidade máxima do exercício e o multiplicador da intensidade. O “Multiplicador de Intensidade” muda a constante **kc** da equação de força. A “Intensidade Máxima” configura o esforço máximo no qual o usuário poderá ser submetido dependendo de suas limitações ou da etapa do seu tratamento. A “Proporção das juntas” alterará o valor da **propJuntas** no equacionamento da distribuição de torques.

A primeira imagem após entrada a simulação é mostrada na Figura 5.3. Do lado esquerdo da tela, existem quatro barras mostrando cada uma o esforço nas juntas do Ombro e Cotovelo, nas direções X e Z. As “rosquinhas” do exercício já são posicionadas, mesmo sem o Kinect ter reconhecido o usuário. Existe também uma tela pequena no canto superior direito que reproduz para o usuário o que a câmera do Kinect está visualizando, para ajudá-lo a se posicionar de forma mais correta.

Apertar “Esc” nessa tela voltará ao menu principal.

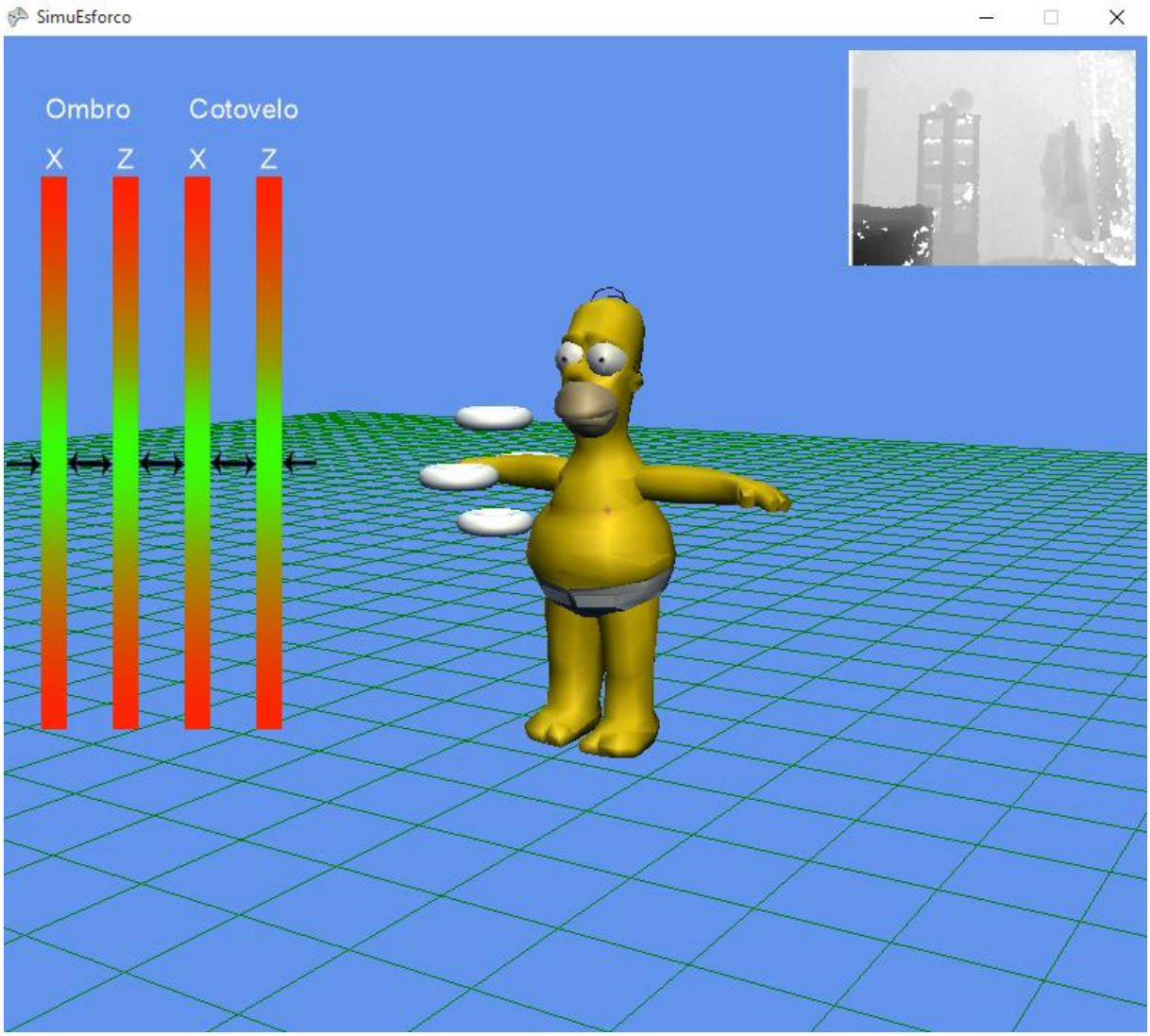


Figura 5.3 – Simulação sem o usuário

A partir do momento que o usuário entra no campo visual do Kinect e o sistema o identifica, o usuário é pintado na tela de profundidade e aparecem marcações em cada uma das juntas reconhecidas, como pode ser vista na Figura 5.4. O usuário tentará cumprir o exercício e as barras variarão de acordo com a proximidade da superfície elipsoidal e o funcionamento do exoesqueleto. Quanto mais próximo do verde, menor o momento calculado para determinada junta, e quanto mais próximo do vermelho, maior o valor. A existência de vermelho acima e abaixo do verde permite a visualização dos dois sentidos de torque. O exercício consiste em fazer, com o braço direito, um movimento circular. As rosquinhas foram usadas como marcação dos extremos da elipsoide.

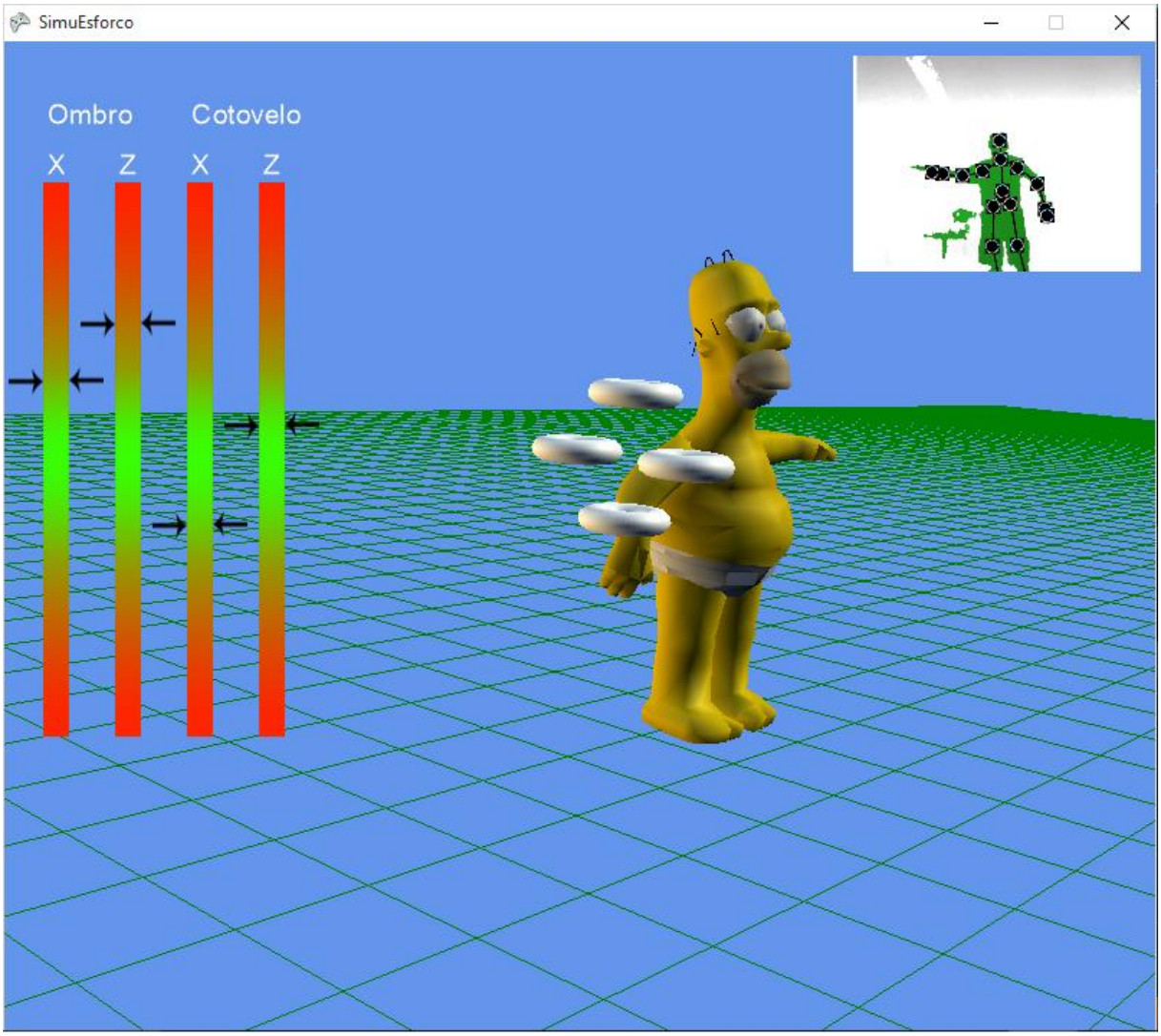


Figura 5.4 – Simulação com o usuário

6. Conclusão

Um software de captura e processamento de movimento foi projetado, documentado e implementado com todas as funcionalidades propostas.

A documentação do projeto foi baseada na UML, consistindo dos diagramas: Caso de Uso, Atividade, Componente, Classe e Sequência.

A codificação da interpretação dos dados vindos do Kinect foi feita de forma genérica, podendo ser reutilizado em qualquer programa que utilize esse Hardware.

O Kinect demonstrou, por vezes, dificuldade de reconhecer juntas sobrepostas e reconhecer com precisão rotações ao longo do próprio eixo do membro, possivelmente esses problemas poderiam ser resolvidos com o uso do Microsoft Kinect 2.0.

Para a implementação do avatar, seu esqueleto foi construído de forma a representar um movimento humano, como mostrado na seção 4.2. Posteriormente, o avatar foi inserido no programa.

Apesar de ele apresentar a movimentação necessária para o projeto, sua movimentação diverge da realidade; por exemplo, parte da *waveform* na região do abdomem se movimenta em conjunto com o braço. Refazer o esqueleto e a *paint weight* corrige esse problema.

Outro ponto de dificuldade foi o método de conectar o esqueleto do modelo e os dados advindos do Kinect para a sincronização do movimento da pessoa com o avatar.

Em seguida, os exercícios e todo o equacionamento dos esforços envolvidos foram desenvolvidos na seção 4.3 e aplicados ao programa.

Além disso, outras classes foram introduzidas para melhorar o entendimento da RV, dentre eles cita-se o elemento criador do chão e o manipulador da câmera.

Os testes feitos no *software* mostraram sua grande capacidade e como ele poderá, em trabalhos futuros, ser mais bem desenvolvido. Apesar de ele ter sido desenvolvido para o braço direito, ele foi codificado de forma a possibilitar sua adaptação para o braço esquerdo, ou até mesmo para os membros inferiores, desde que o Kinect tivesse uma boa rastreabilidade dele, sem grandes alterações.

7. Referência

ANDRADE, K. O.; ITO, G. G.; JOAQUIM, R. C.; JARDIM, B.; SIQUEIRA, A. A. G.; CAURIN, G. A. P.; BECKER, M. **A Robotic System for Rehabilitation of Distal Radius Fracture using Games**. Brazilian Symposium on Games and Digital Entertainment. [S.l.]: [s.n.]. 2010. p. 25-32.

CATUHE, D. **Programming with the Kinect for Windows Software Development Kit**. 1ª. ed. [S.l.]: Microsoft Press, 2012.

CHANG, Y.-J.; CHEN, S.-F.; HUANG, J.-D. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. **Research in Developmental Disabilities**, v. 32, p. 2566-2570, 2011.

COFFITO. Definição: Fisioterapia. **Site da COFFITO**. Disponivel em: <http://www.coffito.org.br/site/index.php/fisioterapia/definicao.html>. Acesso em: 16 Junho 2015.

DEPARTMENT OF ECONOMIC AND SOCIAL AFFAIRS - POPULATION DIVISION. **World Population Ageing 2009**. United Nation. New York. 2009.

JARASSÉ, N.; CROCHER, V.; MORE, G. **A Method for Measuring the Upper Limb Motion and Computing a Compatible Exoskeleton Trajectory**. International Conference on Intelligent Robots and Systems. Vilamoura: [s.n.]. 2012. p. 3461-3466.

KAUFMAN, R. E. **Harness Mechanisms for Full-Body Motions in Virtual Environments**. IEEE Virtual Reality Conference. Charlotte, North Carolina: [s.n.]. 2007. p. 279-280.

LANGE, B.; CHANG, C.-Y.; SUMA, E.; NEWMAN, B.; RIZZO, A. S.; BOLAS, M. **Development and Evaluation of Low Cost Game-Based Balance Rehabilitation Tool Using the Microsoft Kinect Sensor**. 33rd Annual International Conference of the IEEE EMBS. Boston, Massachusetts USA: [s.n.]. 2011.

LOZANO-QUILIS, J. A.; GIL-GOMÉZ, H.; GIL-GÓMEZ, J. A.; ALBIOL-PÉREZ, S.; PALACIOS, G.; FARDOUM, H. M.; MASHAT, A. S. Virtual Reality System for Multiple Sclerosis, Valencia, 2013. 4.

OLIVEIRA, R. E. M. D.; OLIVEIRA, J. C. D. **Rastreamento Utilizando o Dispositivo Kinect para Treinamento em Plataforma de Petróelo**. XVI Symposium on Virtual and Augmented Reality. [S.l.]: [s.n.]. 2014. p. 135-138.

OUARTI, N.; LÉCUYER, A.; BERTHOZ, A. **Haptic Motion:** Improving Sensation of Self-Motion in Virtual Worlds with Force Feedback. IEEE Haptics Symposium. Houston, Texas: [s.n.]. 2014. p. 167-174.

OUCH, R.; ROUSE, B. **Developing a Driving Training Game on Windows Mobile Phone Using C# and XNA**. The 16th Internationa Conference on Computer Games. [S.l.]: [s.n.]. 2011. p. 254-256.

QUADRADO, V. H.; NORIEGA, C.; FORNER-CORDERO, A. **Experimental assessment of a coincident timing motor task of the arm under a passive mechanical perturbation**. Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS & EMBS International Conference on. São Paulo: IEEE. 2014. p. 616-620.

TONG, J.; ZHOU, J.; LIU, L.; PAN, Z.; YAN, H. Scanning 3D Full Human Bodies Using Kinects. **IEEE Transaction on Visualization and Computer Graphics**, v. 18, p. 643-650, Abril 2012.

WANG, D.; LEE, K.-M.; GUO, J.; YANG, C.-J. Adaptive Knee Joint Exoskeleton Based on Biological Geometries. **IEEE/ASME TRANSACTIONS ON MECHATRONICS**, v. 19, n. 4, p. 1268-1278, Agosto 2014.

WEI, W.; DONGSHENG, L.; CHUN, L. **Fixed-wing Aircraft Interactive Flight Simulation and Training System Based on XNA**. Internartional Conference on Virtual Reality and Visualization. [S.l.]: [s.n.]. 2013b. p. 191-198.

WEI, W.; GUO, S.; ZHANG, F.; GUO, J.; JI, Y.; WANG, Y. **A Novel Upper Limb Rehabilitation System with Hand Exoskeleton Mechanism**. IEEE International Conference on Mechatronics and Automation. Takamatsu: [s.n.]. 2013a. p. 285-290.

ZHAO, W.; ESPY, D. D.; REINTHAL, M. A.; FENG, H. **A Feasibility Study of Using a Single Kinect Sensor for Rehabilitation Exercises Monitoring:** A Rule Based Approach. IEEE Symposium on Computational Intelligence in Healthcare and e-health. Orlando, FL: [s.n.]. 2014. p. 1-8.