



UNIVERSIDADE FEDERAL DE SÃO PAULO
CAMPUS SÃO JOSÉ DOS CAMPOS

LUIZ GUSTAVO ALVES ASSIS DA SILVA

Maratona da Fisioterapia

Computação Gráfica

Profa. Regina Célia Coelho

SÃO JOSÉ DOS CAMPOS

04 DE JANEIRO DE 2023

Sumário

1. Introdução	2
2. Objetivo	3
3. Metodologia	4
3.1) Paleta de cores	4
3.2) Modelagem do corpo humano	5
3.3) Modelagem do cenário	7
3.4) Escolha dos exercícios	9
3.4.1) Alongamento do quadríceps	10
3.4.2) Alongamento do pescoço	11
3.4.3) Flexão lateral do tronco	12
3.4.4) Agachamento	13
3.4.5) Polichinelo	14
3.4.6) Rosca alternada	15
3.4.7) Rosca simultânea	16
3.4.8) Elevação frontal	17
3.4.9) Elevação lateral	18
3.4.10) Afundo com halteres	19
3.5) Animação	20
4. Resultados e discussões	23
4.1) Imagens e detalhes do cenário	23
4.2) Imagens da animação	28
4.3) Opções de menu	35
4.4) Opções de teclado	37
4.5) Discussões sobre o desenvolvimento do projeto	38
5. Conclusão	39

1. Introdução

O programa “Maratona da Fisioterapia”, em alusão a quantidade de exercícios a ser executados pelo personagem, tem como principal objetivo a criação de um ambiente que auxilie na reabilitação de pacientes que precisam de fisioterapia. Utilizando a API de Computação Gráfica OpenGL, é criado um ambiente de visualização tridimensional interativo com usuário através do teclado e menu de opções pré-estabelecidas.

A motivação da escolha dos exercícios tem como base em trabalhar a resistência, flexibilidade, fortalecimento da musculatura, alívio de dores e posicionamento da postura corporal. Para este projeto foi feita a escolha de 10 exercícios no total em que 5 destes exercícios são executados sem equipamentos e os outros 5 exercícios com equipamentos, sendo o equipamento utilizado pelo personagem é um par de halteres.

2. Objetivo

O trabalho traz como objetivo apresentar as principais técnicas e métodos de Computação Gráfica desenvolvidas no decorrer da disciplina, tais como: visualização tridimensional, animação, interação com usuário, iluminação e textura. Para cumprir este objetivo, o trabalho em questão demonstra a aplicação direta do aprendizado adquirido por meio da criação de um ambiente tridimensional interativo que simula sequências de exercícios de reabilitação para pacientes que precisam de fisioterapia.

A visualização tridimensional está presente durante todo desenvolvimento do projeto, seja pela criação do cenário, iluminação e textura, criação do personagem e animação que sucede as sequências dos exercícios.

A interação com usuário ocorre através do teclado e da criação de um menu de opções permitindo a manipulação dos movimentos individuais de uma dada estrutura do corpo do personagem tal como a execução dos exercícios atribuídos ao personagem.

3. Metodologia

Com a finalidade de fragmentar a complexidade da animação e na modelagem do cenário e personagem, foi necessário o desenvolvimento paralelo de arquivos de cabeçalhos (*header files*), sendo esses arquivos os seguintes:

- paleteCores.h: definição da paleta de cores utilizada no projeto.
- textura.h: definição da estrutura e método para implementação de texturas.
- fractal.h: construção do fractal Árvore de Pitágoras.
- corpoHumano.h: definição e modelagem individual das estruturas do corpo.
- cenario.h: construção de paredes, objetos e equipamentos do ambiente.
- animacao.h: implementação lógica da animação dos exercícios.

3.1) Paleta de cores

A definição da paleta de cores utilizada no projeto é efetuada através de uma tabela de cores RGB pertencente a este website externo: [Philip Rideout :: Colors](#). Cada cor do arquivo de cabeçalho *paleteCores.h* é implementada da seguinte forma:

```
typedef struct {

    float color[3];

} Color;

Color  aliceblue =      {0.941f, 0.973f, 1.000f},
      antiquewhite =   {0.980f, 0.922f, 0.843f},
      aqua =           {0.000f, 1.000f, 1.000f},
      aquamarine =     {0.498f, 1.000f, 0.831f},
      ...
      yellow =         {1.000f, 1.000f, 0.000f},
      yellowgreen =    {0.604f, 0.804f, 0.196f};
```

3.2) Modelagem do corpo humano

A modelagem do corpo humano (arquivo corpoHumano.h) é realizada por meio da combinação do uso de superfícies quádricas (a exemplo de esferas, cubos e cilindros) e aplicação das transformações geométricas hierárquicas. Ainda, é definido uma *struct* responsável por armazenar informações de altura e raio de uma dada estrutura do corpo e, com base nesses dados armazenados, é possível modelar cada estrutura do corpo humano de forma individualizada seguindo um modelo hierárquico.

```
typedef struct {

    float height, radius;

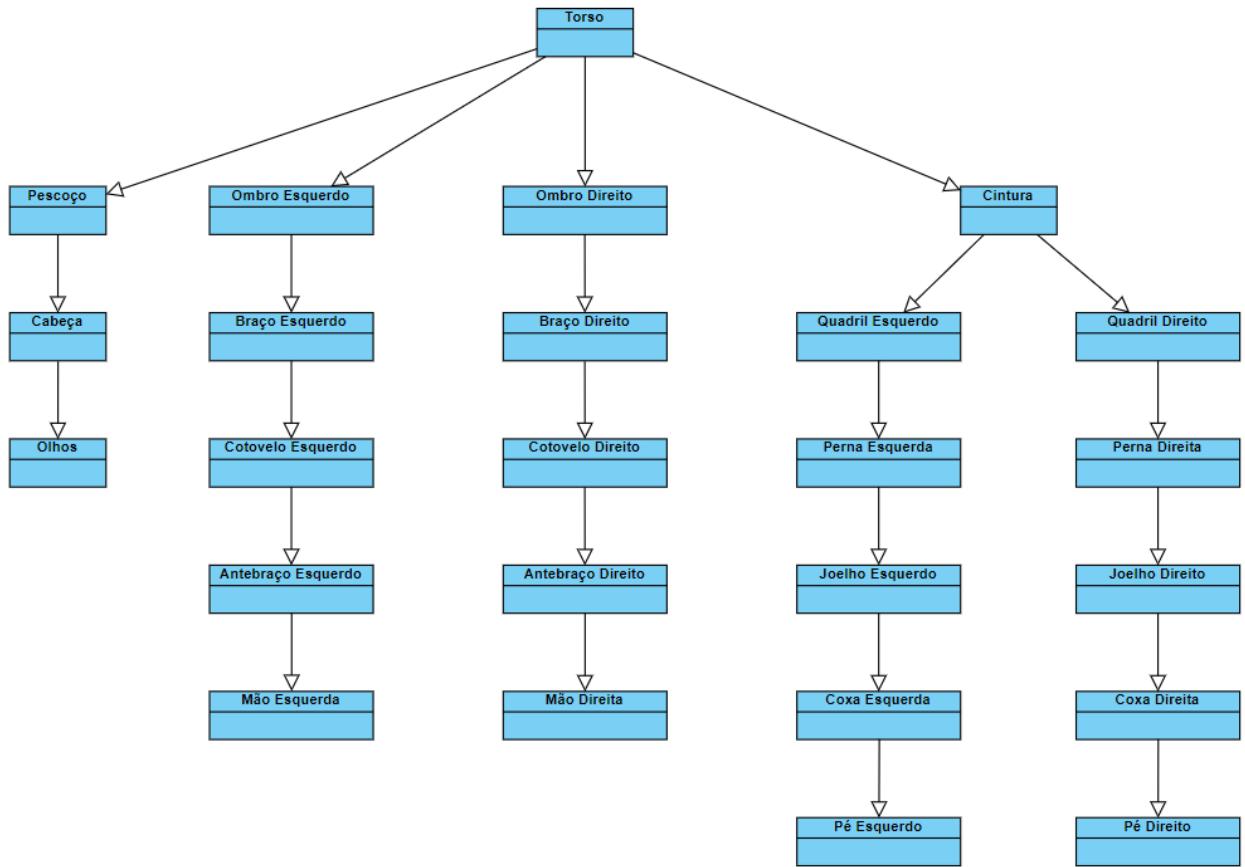
} Body_Structure;

/* Estruturas do corpo humano */
Body_Structure torso = {4.0, 2.0}, /* Torso */
    neck = {1.0, 0.4}, /* Pescoco */
    head = {1.5, 1.0}, /* Cabeça */
    eye = {0.0, 0.3}, /* Olho */
    shoulder = {0.0, 0.6}, /* Ombro */
    upperArm = {3.5, 0.5}, /* Braco */
    elbow = {0.0, 0.4}, /* Cotovelo */
    lowerArm = {3.0, 0.4}, /* Antebraço */
    hand = {0.0, 0.5}, /* Mao */
    waist = {3.5, 1.7}, /* Cintura */
    hip = {0.0, 0.6}, /* Quadril */
    upperLeg = {2.5, 0.6}, /* Coxa */
    knee = {0.0, 0.5}, /* Joelho */
    lowerLeg = {3.0, 0.5}, /* Canela */
    feet = {0.0, 0.8}; /* Pe */
```

A estrutura que sustenta os membros superiores, como: pescoço, cabeça, olhos, ombros, braços, cotovelos, antebraços e mãos (*neck*, *head*, *eye*, *shoulder*, *upperArm*, *elbow*, *lowerArm*, *hand*, respectivamente) é o torso.

A estrutura que sustenta os membros inferiores, como: quadris, pernas, joelhos, coxas e pés (*hip*, *upperLeg*, *knee*, *lowerLeg*, *feet*, respectivamente) é a cintura (*waist*).

Portanto, o desenvolvimento do personagem é feito seguindo o modelo hierárquico abaixo:



A cada ramificação do modelo acima é feita a chamada das funções *glPushMatrix* e *glPopMatrix* seguido de aplicações de transformações geométricas, como translação, escala e rotação, sendo a última transformação de extrema importância para a movimentação individual das articulações do corpo.

3.3) Modelagem do cenário

A modelagem do cenário é realizada por meio da combinação do uso de superfícies quádricas (a exemplo de esferas, cubos e cilindros), primitivas gráficas e transformações geométricas hierárquicas. Os principais componentes do cenário se dão pela construção das paredes e do chão da sala, funções *drawWalls* e *drawFloor* respectivamente, e para isso foi necessário utilizar variáveis globais que limitam o tamanho horizontal e vertical do cenário como um todo, sendo essas variáveis: *roomSize* e *humanWidth*, as demais variáveis globais são responsáveis por limitar os tamanhos individuais dos objetos da cena. Além disso, foi feita a definição das seguintes constantes para auxiliar no desenvolvimento do código:

```
#define SLICES 10
#define STACKS 10

#define CAMERA_RIGHT_DIRECTION (cameraAlpha > 0.50 && cameraAlpha < 2.80) || cameraTheta > 1.15
#define CAMERA_BACK_DIRECTION (cameraAlpha > 2.15 && cameraAlpha < 4.40) || cameraTheta > 1.15
#define CAMERA_LEFT_DIRECTION (cameraAlpha > 3.70 && cameraAlpha < 5.90) || cameraTheta > 1.15
#define CAMERA_FRONT_DIRECTION (cameraAlpha > 5.25 || cameraAlpha < 1.20) || cameraTheta > 1.15
```

A definição das duas primeiras constantes, *SLICES* e *STACKS*, diz respeito a quantidade de subdivisões em relação aos eixos X e Y das superfícies quádricas. As próximas constantes são relativas a posição atual da câmera permitindo controlar o que será desenhado na tela. Por exemplo, o código abaixo desenha a parede direita da sala se, e somente se, a posição atual da câmera não estiver do lado direito do cenário.

```
if (!CAMERA_RIGHT_DIRECTION) {

    /* Parede direita */
    glVertex3f(roomSize + wallSize - 1.0, -humanWidth,      roomSize);
    glVertex3f(roomSize + wallSize - 1.0, -humanWidth,      -roomSize);
    glVertex3f(roomSize + wallSize - 1.0, humanWidth * 3, -roomSize);
    glVertex3f(roomSize + wallSize - 1.0, humanWidth * 3,  roomSize);
}
```

Ainda, é aplicada uma textura no chão da sala por meio de uma definição da estrutura e método para implementação de texturas presente no arquivo de cabeçalho *textura.h*, Por fim, é utilizado o fractal Árvore de Pitágoras, presente no arquivo de cabeçalho *fractal.h* para compor a ambientação do cenário. As funções que desenham toda ambientação do projeto foram assim definidas:

```
/* Desenha todas as mobiliarias */
void drawFurniture() {

    drawDumbbellRack();
    drawSofa();
    drawTVRack();
    drawStool();
}

/* Desenha todos os objetos */
void drawObjects() {

    drawWindow();
    drawDoor();
    drawTV();
    drawRemoteController();
    drawMat();
    drawFlowerPot();
    drawSpeakers();
    drawTableLamp();
}

/* Desenha todo o cenario */
void drawScenario(float alpha, float theta) {

    cameraAlpha = alpha;
    cameraTheta = theta;

    drawWalls();
    drawFloor();
    drawFurniture();
    drawObjects();
}
```

3.4) Escolha dos exercícios

Para este projeto, foi feita a escolha de 10 exercícios no total, sendo 5 desses exercícios sem equipamento e outros 5 exercícios com equipamento. O equipamento a ser utilizado pelo personagem é um par de halteres.

Os exercícios sem equipamento a serem executados são os seguintes:

- Alongamento de quadríceps.
- Alongamento do pescoço.
- Flexão lateral do tronco.
- Agachamento.
- Polichinelo.

Os exercícios com equipamento a serem executados são os seguintes:

- Rosca alternada.
- Rosca simultânea.
- Elevação frontal.
- Elevação lateral.
- Afundo com halteres.

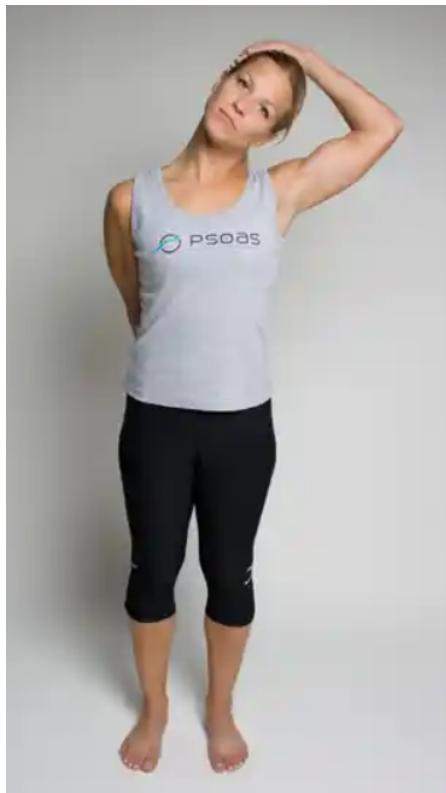
Cabe ressaltar que, para a reabilitação de pacientes, os exercícios com equipamento acima devem ser executados em menor amplitude ou de forma estática, no qual o músculo é contraído e fica parado em uma posição.

3.4.1) Alongamento do quadríceps



O alongamento do quadríceps consiste em flexionar uma das pernas, segurar o pé com a mão do mesmo lado, dobrar o joelho e puxar o calcanhar na direção do glúteo. A finalidade deste exercício corresponde em melhorar a flexibilidade, aliviar a rigidez muscular e reduzir as dores na costa e joelhos.

3.4.2) Alongamento do pescoço



O alongamento do pescoço consiste em inclinar a cabeça para o lado, colocar a mão que segura o pescoço atrás da cabeça e esticar o pescoço. A finalidade deste exercício corresponde em aliviar dores musculares e articulares como torcicolo.

3.4.3) Flexão lateral do tronco



A flexão lateral do tronco consiste em separar as pernas e esticar um dos braços para cima na direção do flexionamento do tronco. A finalidade deste exercício permite fortalecer os músculos anteriores e posteriores do tronco, além de trabalhar a mobilidade articular.

3.4.4) Agachamento



O agachamento consiste em ficar em pé e estender os membros superiores, afastar os pés à largura dos ombros e apoiá-los totalmente no chão, dobrar os joelhos jogando o quadril para baixo e esticar as pernas para voltar à posição original. A finalidade deste exercício corresponde em fortalecer a musculatura dos membros inferiores, melhorar a coordenação motora e o equilíbrio postural.

3.4.5) Polichinelo



O polichinelo consiste em iniciar mantendo-se numa posição ereta com as pernas juntas e mãos estendidas ao longo do corpo, saltar do mesmo lugar flexionando os joelhos e elevar os braços encostando as palmas das mãos acima da cabeça. A finalidade deste exercício corresponde em trabalhar a resistência muscular e cardiovascular, ideal para aquecimentos antes de iniciar um treinamento físico.

3.4.6) Rosca alternada



A rosca alternada consiste em segurar um halter em cada mão, flexionar um dos braços trazendo-o para cima até contrair o bíceps e alternar o movimento entre os braços. A finalidade deste exercício permite construir e fortalecer a musculatura dos flexores e bíceps, além de trabalhar a coordenação motora do praticante.

3.4.7) Rosca simultânea



A rosca simultânea, assim como a rosca alternada, consiste em segurar um halter em cada mão e flexionar os dois braços trazendo-os para cima até contrair o bíceps. A finalidade deste exercício permite construir e fortalecer a musculatura dos flexores e bíceps, além de trabalhar a coordenação motora do praticante. A principal diferença entre a rosca alternada e simultânea é que a rosca alternada vai ter menos intensidade comparado a rosca simultânea, uma vez que um dos braços estará em descanso.

3.4.8) Elevação frontal



A elevação frontal consiste em segurar um halter em cada mão e iniciar o flexionamento de ombros, elevando os braços com uma angulação paralela ao solo. A finalidade deste exercício permite reduzir a tensão e fortalecer a musculatura dos ombros e parte superior do tórax, posicionar a postura corporal e também aliviar dores como bursite no ombro.

3.4.9) Elevação lateral



A elevação lateral consiste em segurar um halter em cada mão e iniciar o flexionamento de ombros, elevando os braços lateralmente até que as mãos atingem a altura dos ombros. A finalidade deste exercício permite reduzir a tensão e fortalecer a musculatura dos ombros e parte superior do tórax, posicionar a postura corporal e também aliviar dores como bursite no ombro.

3.4.10) Afundo com halteres



O afundo com halteres consiste em segurar um halter em cada mão mantendo os braços estendidos e com os pés afastados (com pé esquerdo na frente do direito), dar um passo para frente e flexionar o joelho da perna avançada até que ela fique paralela ao chão, abaixar o quadril até a articulação da frente formar uma angulação de 90° graus e o joelho da perna que ficou atrás aproximar do chão, após isso empurrar o corpo para cima e retornar à posição original. A finalidade deste exercício permite reduzir a tensão e fortalecer as coxas, glúteos e panturrilha, além de melhorar a coordenação motora e postura corporal.

3.5) Animação

A implementação lógica da animação dos exercícios é feita no arquivo de cabeçalho *animacao.h*, nele é definido todas as *flags* de controle de fluxo da animação, tal como declaração de *structs* para manipulação dos eixos de rotação de cada estrutura do corpo humano e posição atual e inicial de um objeto e, por fim, a utilização de uma estrutura de dados do tipo lista encadeada para facilitar o armazenamento da escolha dos exercícios atribuídos pelo usuário através do menu (cada exercício possui um identificador próprio).

Segue abaixo as constantes e *flags* de controle de fluxo da animação:

```
#define AXIS 3
#define NUM_JOINTS 9
#define STEP 0.1

/* Flags de controle de fluxo da animacao */
float timer = 0.01;
int animationFinished = 0;
int newAnimation = 0;
int animationId = -1;
int allAnimationFlag = 0;
intwaitForAllAnimationFinish = 0;
int hasEquipment = 0;
int hasLeftEquipment = 0;
int hasRightEquipment = 0;
int walkingCycle = 1;
int stopWalkingAnimation = 1;
int waitReturnEquipment = 0;
```

As constantes *AXIS* e *NUM_JOINTS* definem a quantidade de eixos de rotação utilizados e a quantidade de articulações declaradas para as estruturas do corpo humano. A constante *STEP* representa o “passo” da animação, em outras palavras, controla a velocidade para manipulação dos eixos de rotação e posicionamento das articulações, personagem e os halteres. A variável global *timer* representa um contador que é incrementado a cada chamada de função para execução de um dado exercício.

As *flags* *animationFinished* e *newAnimation* são utilizadas para identificar se uma animação chegou ao fim e se há outra animação pendente na lista encadeada, caso não haja outra animação o personagem continuará executando a última animação do exercício o qual foi atribuído no menu do usuário.

As *flags* *allAnimationFlag* e *waitForAllAnimationFinish* são utilizadas quando o usuário escolhe executar todas as animações de exercícios sem ou com equipamento.

As *flags* *hasLeftEquipment*, *hasRightEquipment*, *hasEquipment* e *waitReturnEquipment* são utilizadas quando o usuário escolhe que o personagem utilize o equipamento (o personagem consegue realizar os exercícios com halteres apenas se estiver com os equipamentos em mãos). Após o personagem estiver em posse dos equipamentos não será possível selecionar outras opções de menu além de executar os exercícios com equipamentos e devolver equipamento. Apenas quando o personagem devolver os halteres que o usuário será capaz de utilizar as outras opções de menu. As *flags*: *walkingCycle* e *stopWalkingAnimation* são utilizadas para controlar a animação de andar do personagem.

Segue abaixo a definição e as declarações das articulações do corpo humano e a definição da rotação do corpo humano e dos halteres:

```

typedef struct {

    /* Coordenadas iniciais de rotacao 3D */
    float rotation[AXIS];

    /* Coordenadas limite para rotacao */
    float minRotation[AXIS];
    float maxRotation[AXIS];

} Animation;

/* Juntas (articulacoes) do corpo humano */
Animation neckJoint = {{0.0, 0.0, 0.0}, {20.0, 20.0, 20.0}, {20.0, 20.0, 20.0}},
    leftShoulderJoint = {{0.0, 0.0, 0.0}, {170.0, 0.0, 110.0}, {60.0, 0.0, 10.0}},
    rightShoulderJoint = {{0.0, 0.0, 0.0}, {170.0, 0.0, 10.0}, {60.0, 0.0, 110.0}},
    leftElbowJoint = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}, {0.0, 0.0, 110.0}},
    rightElbowJoint = {{0.0, 0.0, 0.0}, {0.0, 0.0, 110.0}, {0.0, 0.0, 0.0}},
    leftHipJoint = {{0.0, 0.0, 0.0}, {70.0, 0.0, 40.0}, {50.0, 0.0, 40.0}},
    rightHipJoint = {{0.0, 0.0, 0.0}, {70.0, 0.0, 40.0}, {50.0, 0.0, 40.0}},
    leftKneeJoint = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}, {140.0, 0.0, 0.0}},
    rightKneeJoint = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}, {140.0, 0.0, 0.0}},
    torsoJoint = {{0.0, 0.0, 0.0}},
    waistJoint = {{0.0, 0.0, 0.0}};

/* Definindo Rotacao do corpo humano e dos halteres que serao utilizados durante o exercicio */
Animation leftDumbbellRot = {{0.0, 0.0, 0.0}},
    rightDumbbellRot = {{0.0, 0.0, 0.0}},
    humanBodyRot = {{0.0, 0.0, 0.0}};

```

Segue abaixo a definição e as declarações das posições atuais e inicial do corpo humano, estruturas do corpo e dos halteres:

```
typedef struct {

    /* Coordenadas de posicao */
    float current[AXIS];
    float init[AXIS];

} ObjectPosition;

/* Posicao inicial do corpo humano e das estruturas do corpo */
ObjectPosition humanBodyPosition = {{0.0, -1.0, 0.0}, {0.0, -1.0, 0.0}},
                waistPosition      = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}},
                leftLegPosition    = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}},
                rightLegPosition   = {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}};

/* Posicao inicial e atual dos halteres */
ObjectPosition leftDumbbellPos = {{0.0, 0.0, 0.0}, {-40.75, -5.5, -30.0}},
                                rightDumbbellPos = {{0.0, 0.0, 0.0}, {-35.75, -1.5, -30.0}};
```

Segue abaixo a definição e declaração da lista encadeada para armazenamento dos exercícios a serem atribuídos e executados pelo personagem:

```
typedef struct AnimationLst {

    int animationId;
    struct AnimationLst *next;

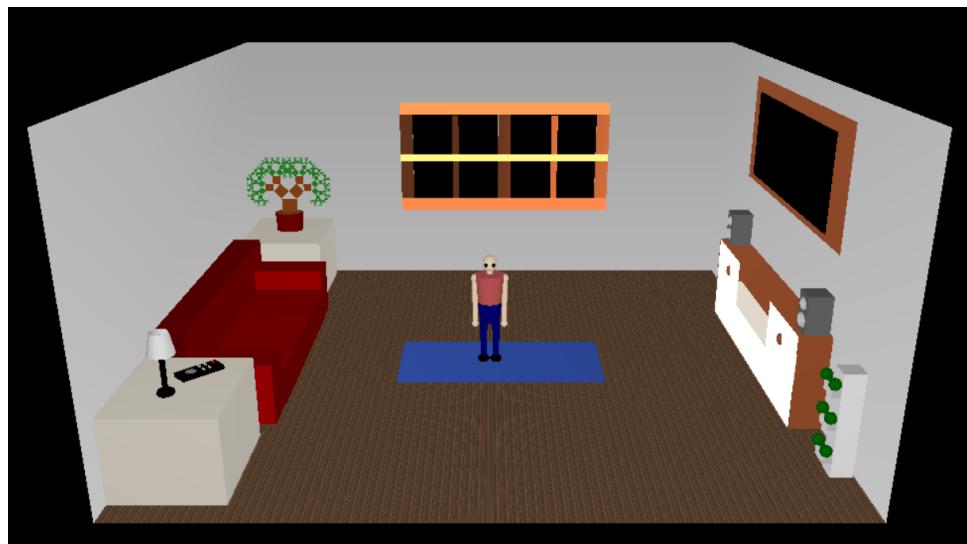
} AnimationLst;

/* Inicio da lista */
AnimationLst *headNode = NULL;
```

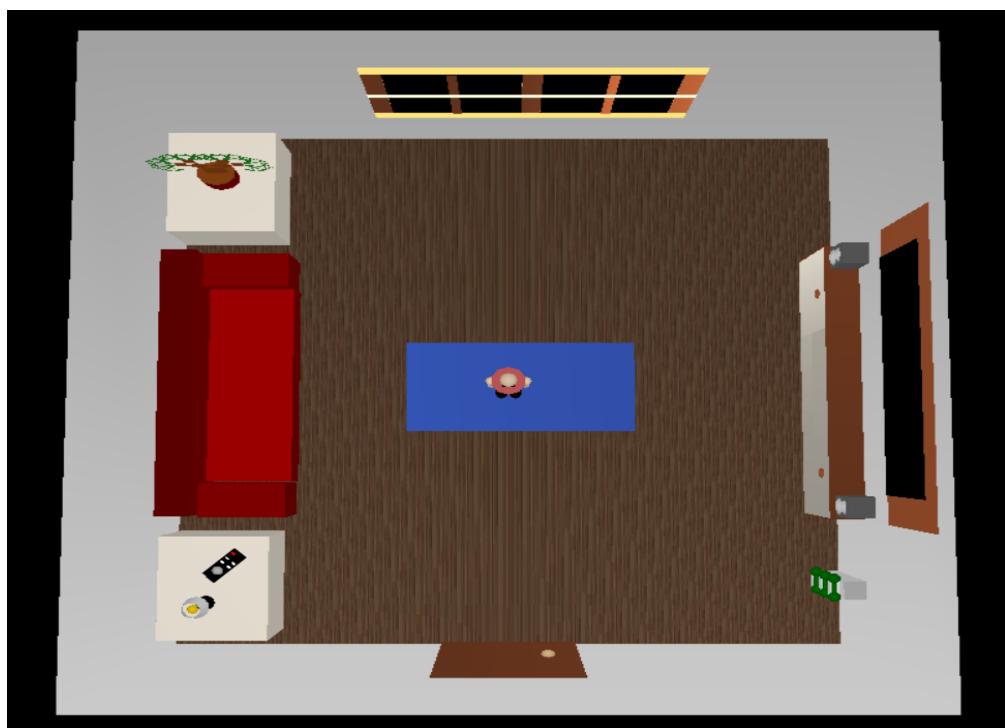
Quando o usuário escolhe executar um exercício, é feita uma chamada de função para atualizar a animação corrente adicionando um novo nó na lista encadeada contendo o identificador próprio da animação escolhida. Para cada passo da animação existem duas funções responsáveis por controlar o ritmo da animação corrente e inversa: *kinematics*, *inverseKinematics*, respectivamente.

4. Resultados e discussões

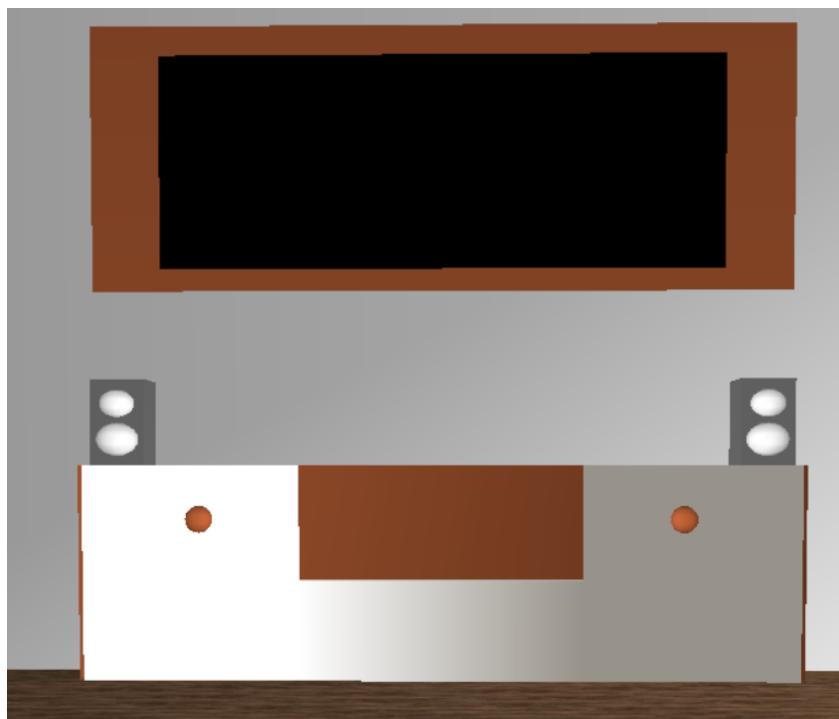
4.1) Imagens e detalhes do cenário



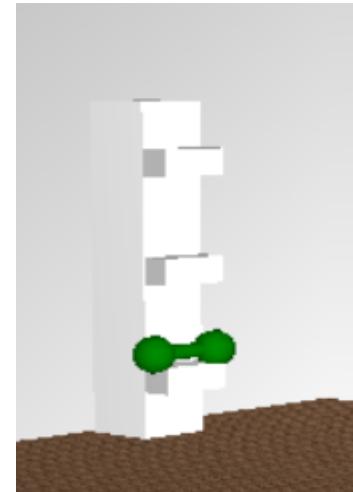
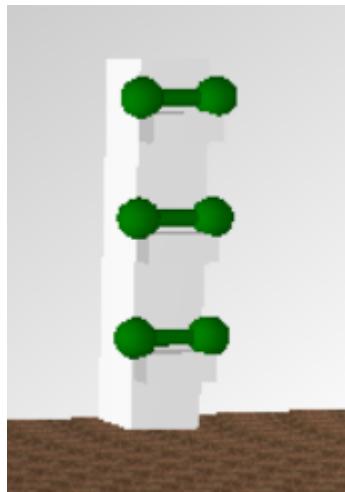
Visão horizontal do cenário.



Visão vertical do cenário.

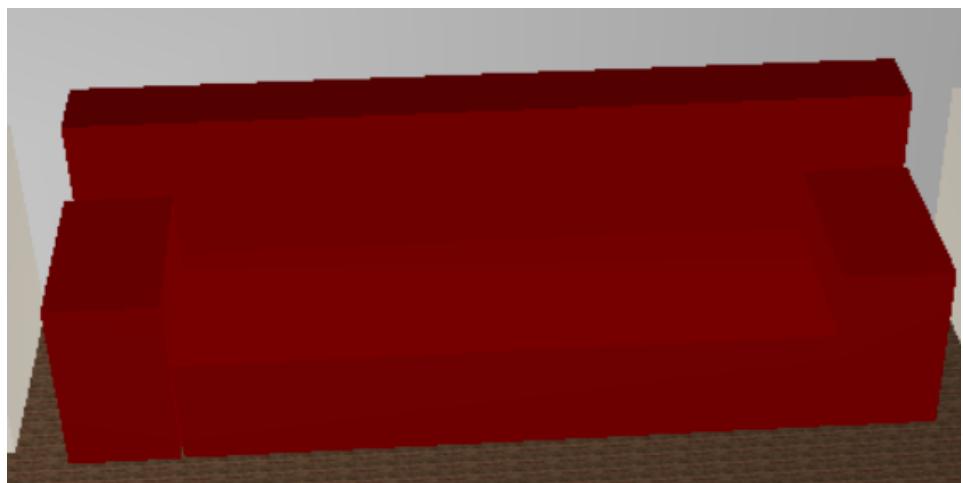


Rack com painel de TV, aparelhos de som para televisão.
Feito utilizando superfícies quádricas (cubos, esferas) e primitivas gráficas (quads)



Rack dos halteres.
Feito utilizando superfícies quádricas (cubos, esferas e cilindros).

A imagem à esquerda indica que os equipamentos não estão sendo utilizados.
A imagem à direita indica que os equipamentos estão sendo utilizados.



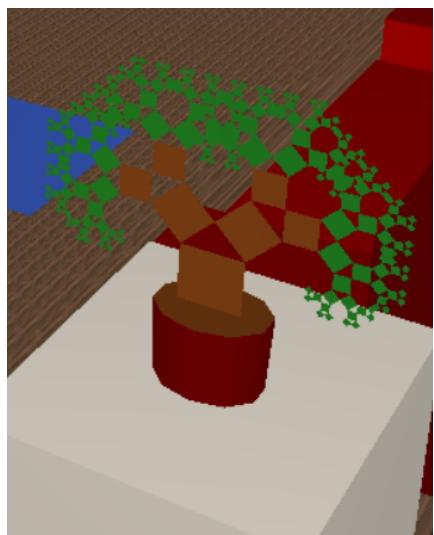
Sofá.

Feito utilizando superfícies quádricas (cubos).



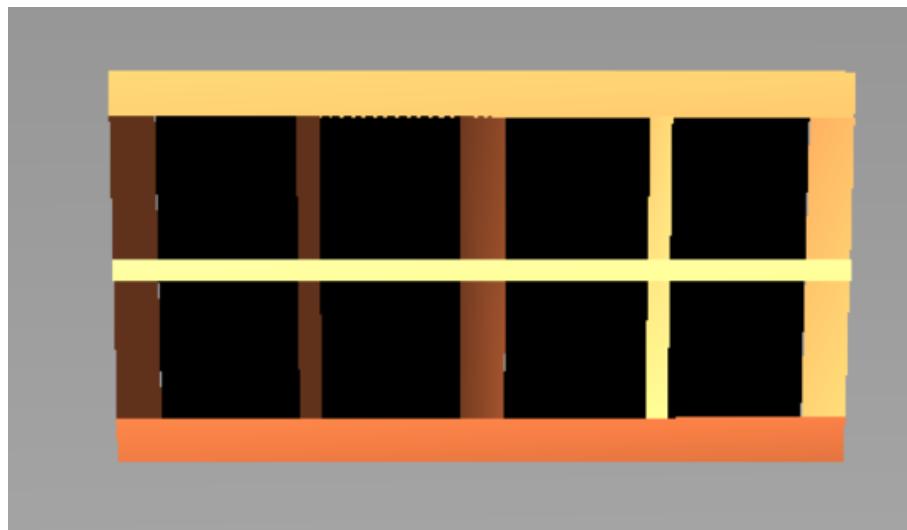
Móvel para sala, controle remoto de televisão, abajur de mesa.

Feito utilizando superfícies quádricas (cubos, esferas e cilindros).



Vaso de planta.

Feito utilizando superfícies quádricas (cilindro, disco) e fractal Árvore de Pitágoras.

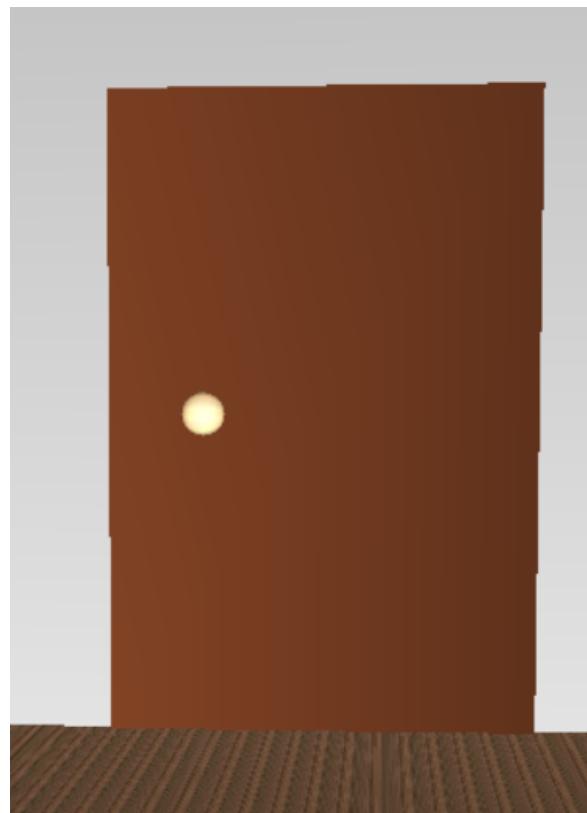


Janela.

Feito utilizando primitivas gráficas (quads).

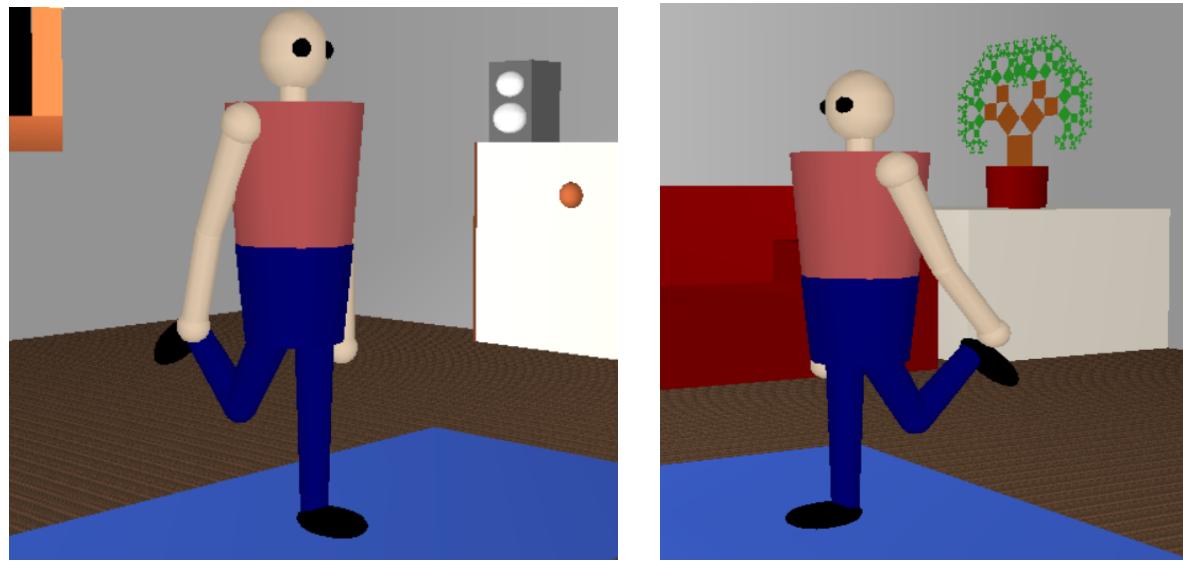


Tapete de treino.
Feito utilizando primitivas gráficas (quads).

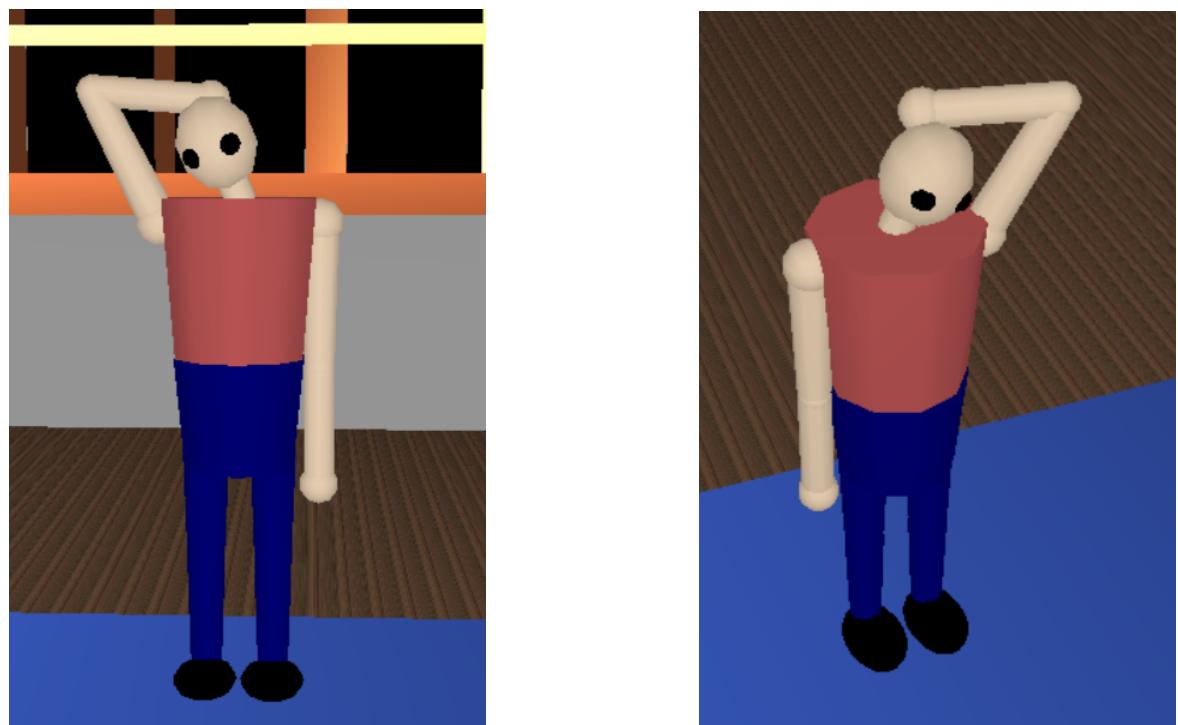


Porta.
Feita utilizando superfícies quádricas (esfera) e primitivas gráficas (quads).

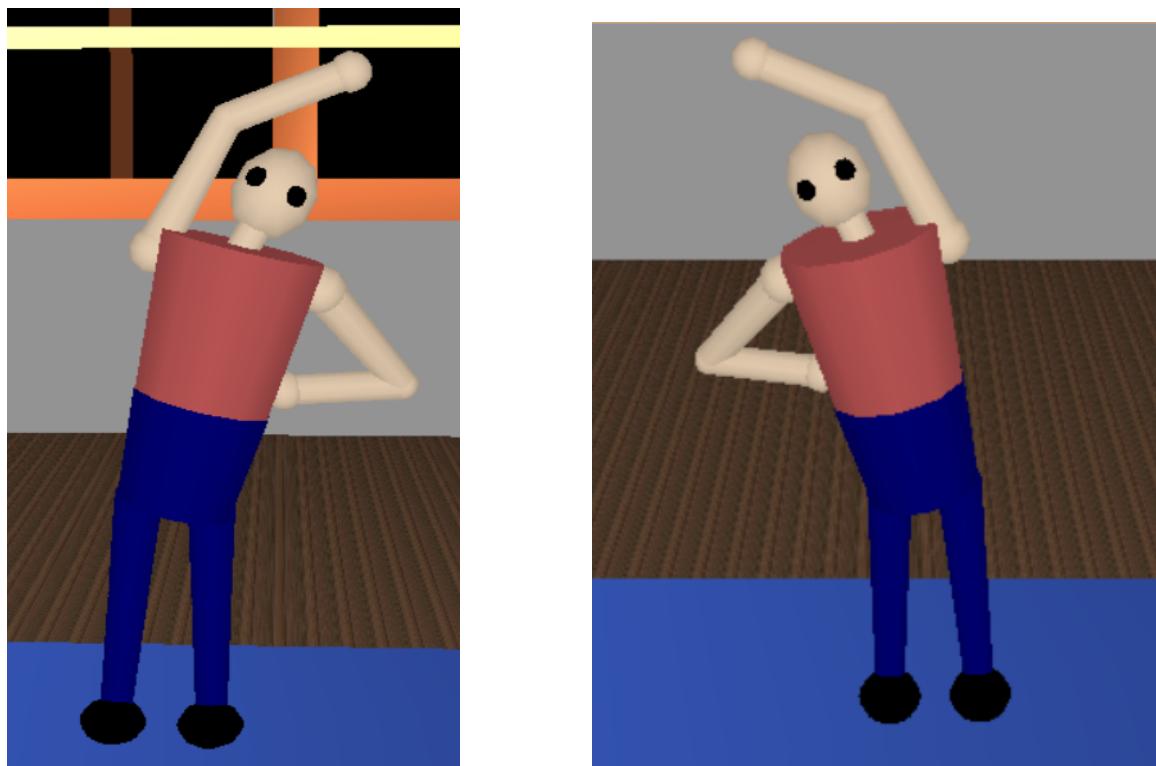
4.2) Imagens da animação



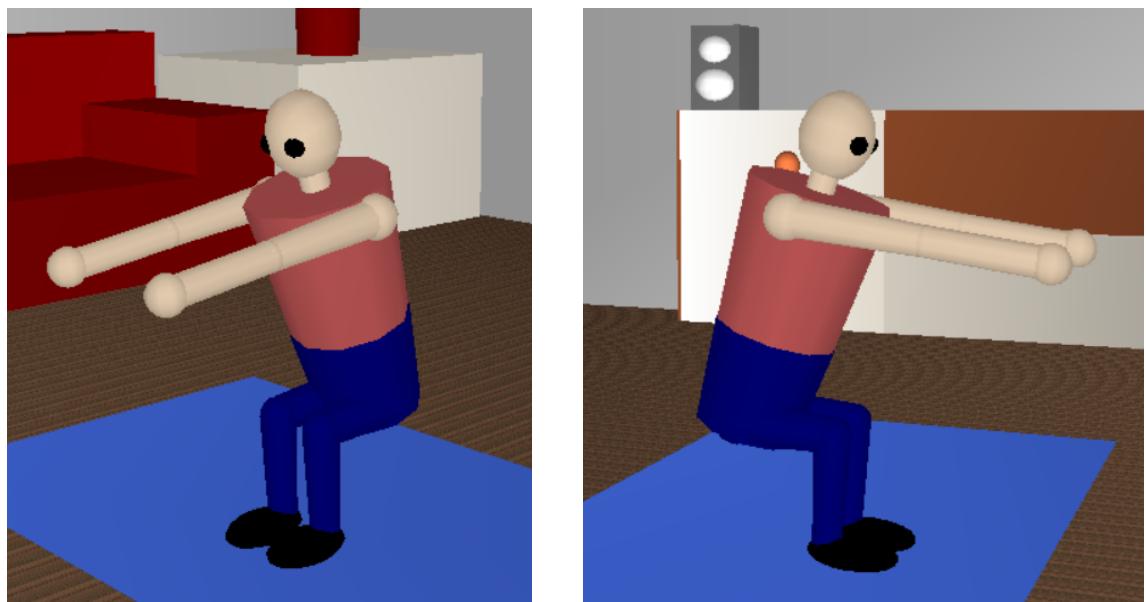
Exercícios de aquecimento - (Alongamento do quadríceps).



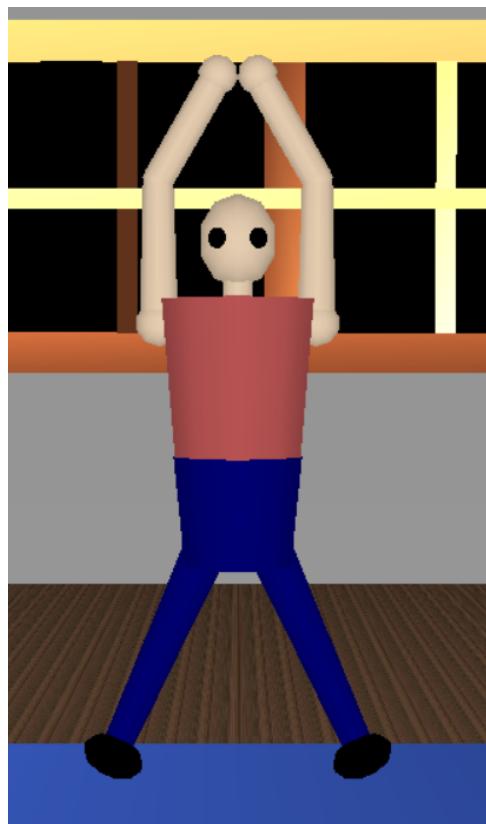
Exercícios de aquecimento - (Alongamento do pescoço).



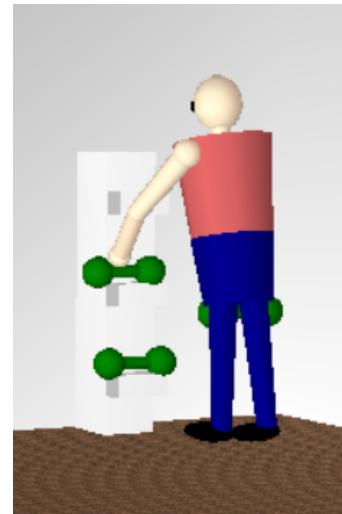
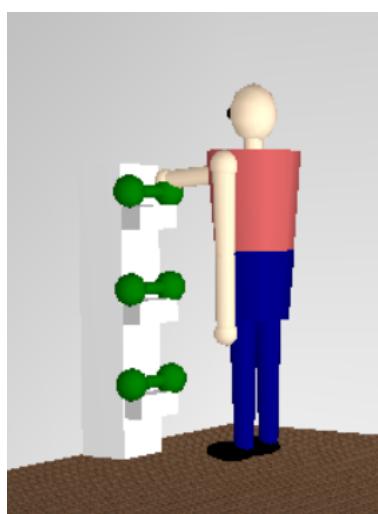
Exercícios de aquecimento - (Flexão lateral do tronco).



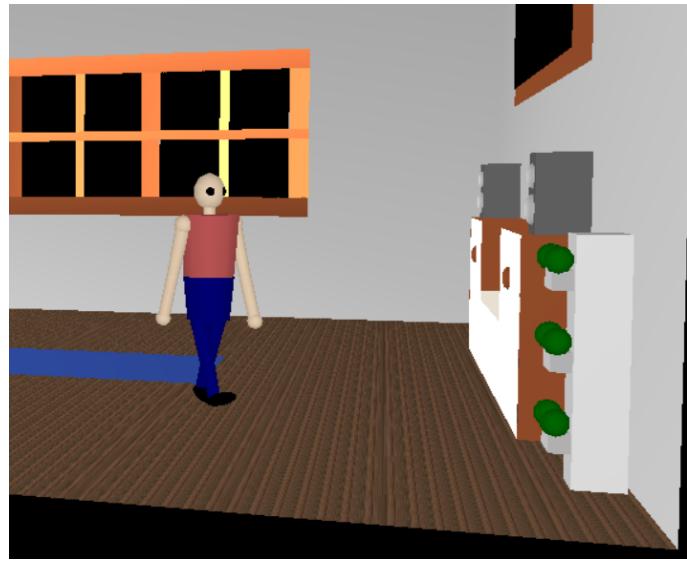
Exercícios de aquecimento - (Agachamento).



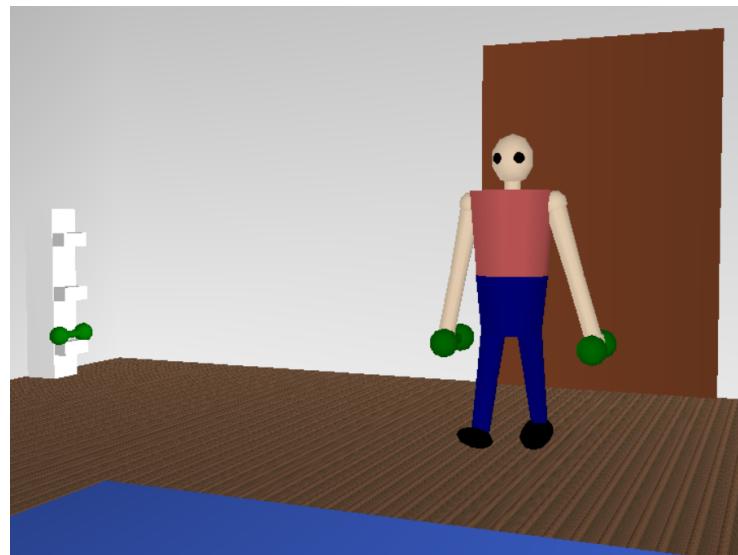
Exercícios de aquecimento - (Polichinelo).



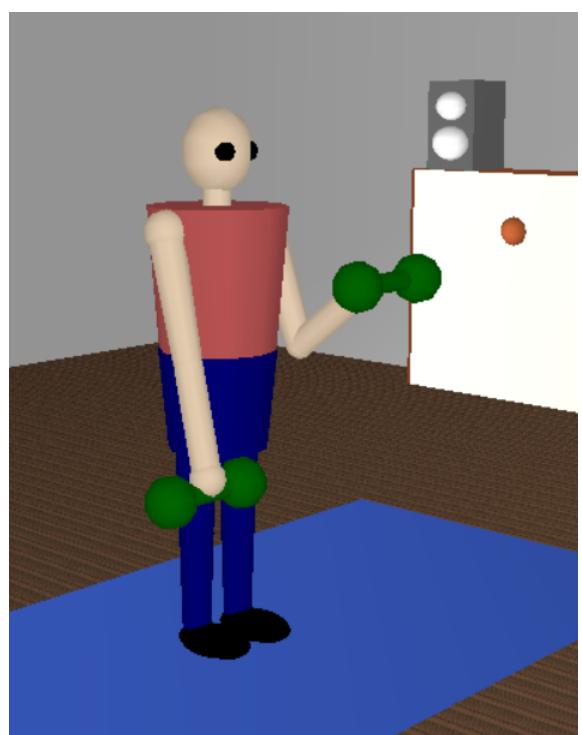
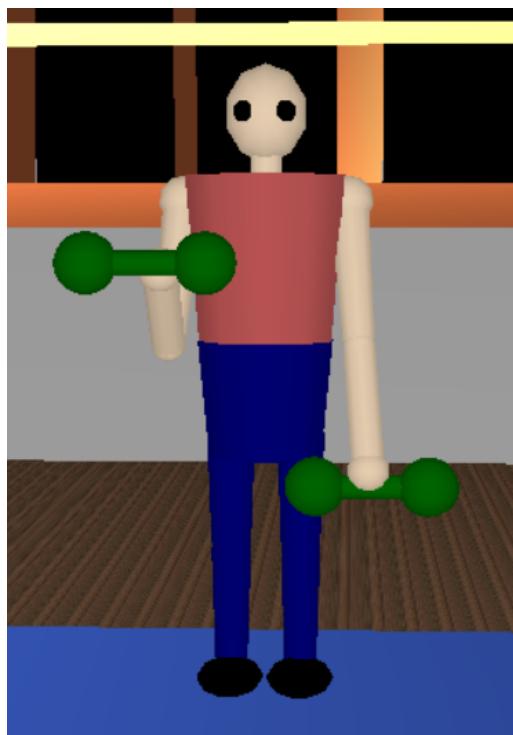
Animação para pegar e devolver equipamentos.



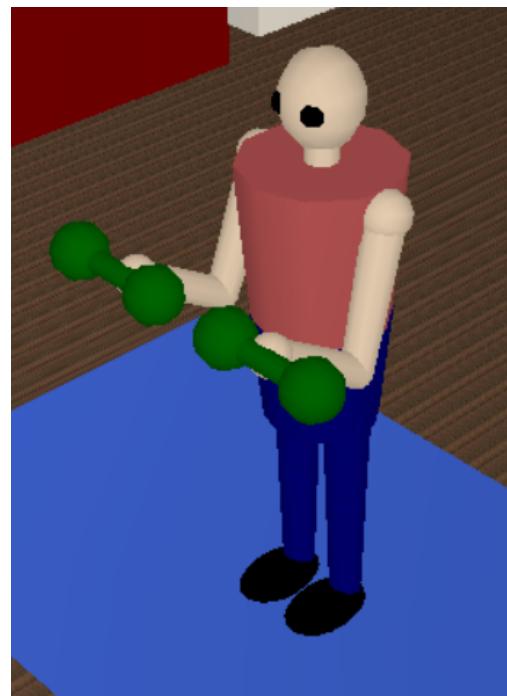
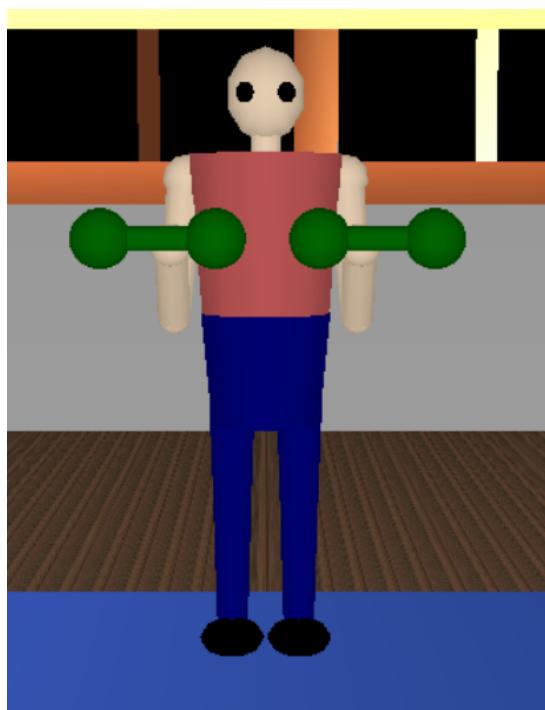
Animação para andar - (Sem equipamento).



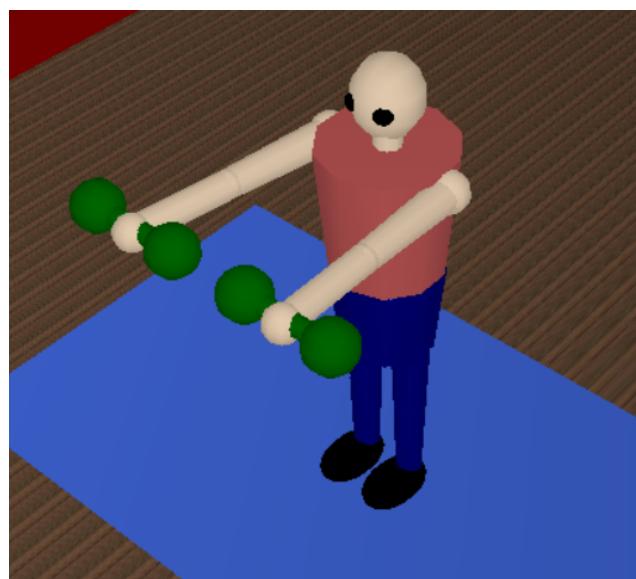
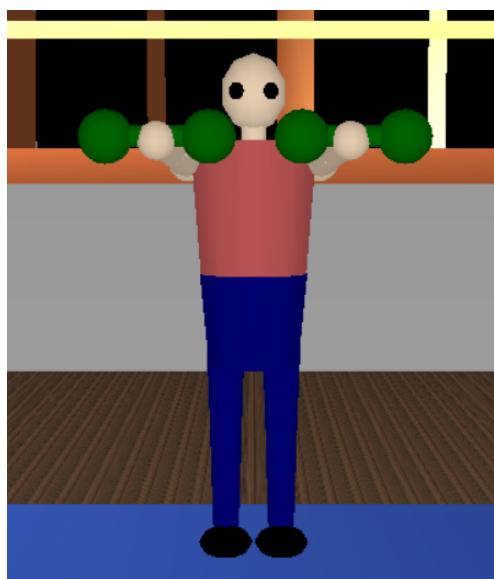
Animação para andar - (Com equipamento).



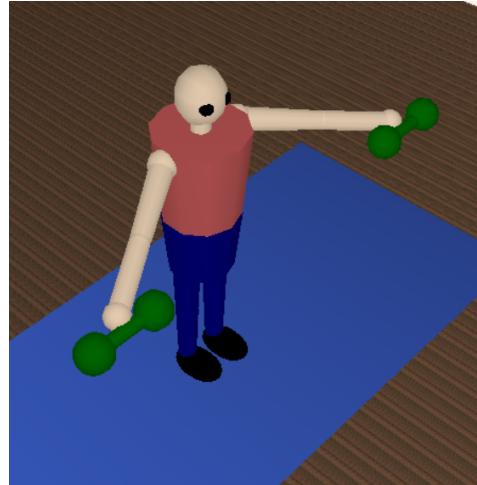
Exercícios com halteres - (Rosca alternada).



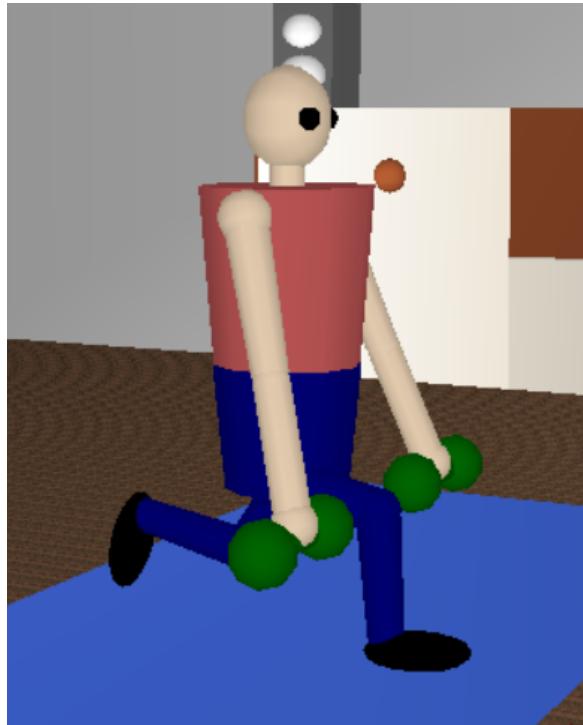
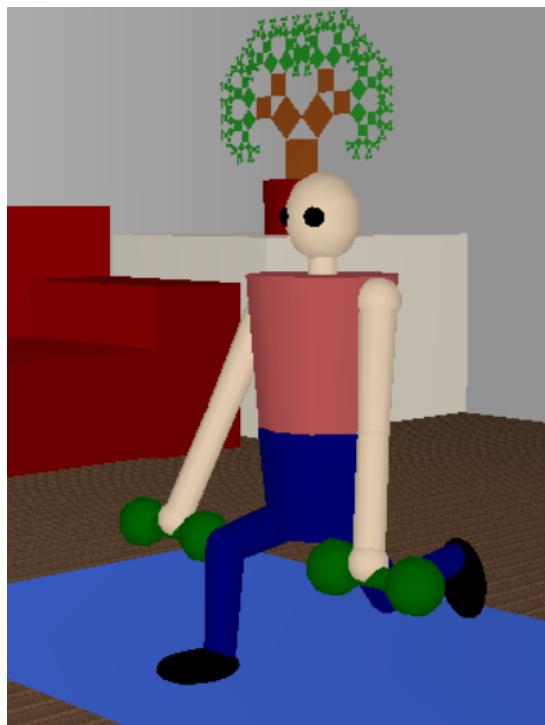
Exercícios com halteres - (Rosca simultânea).



Exercícios com halteres - (Elevação frontal).



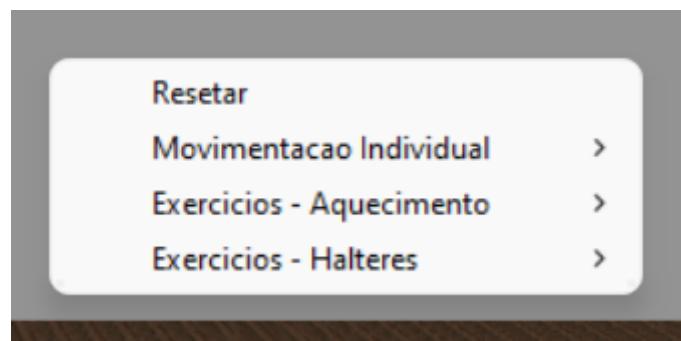
Exercícios com halteres - (Elevação lateral).



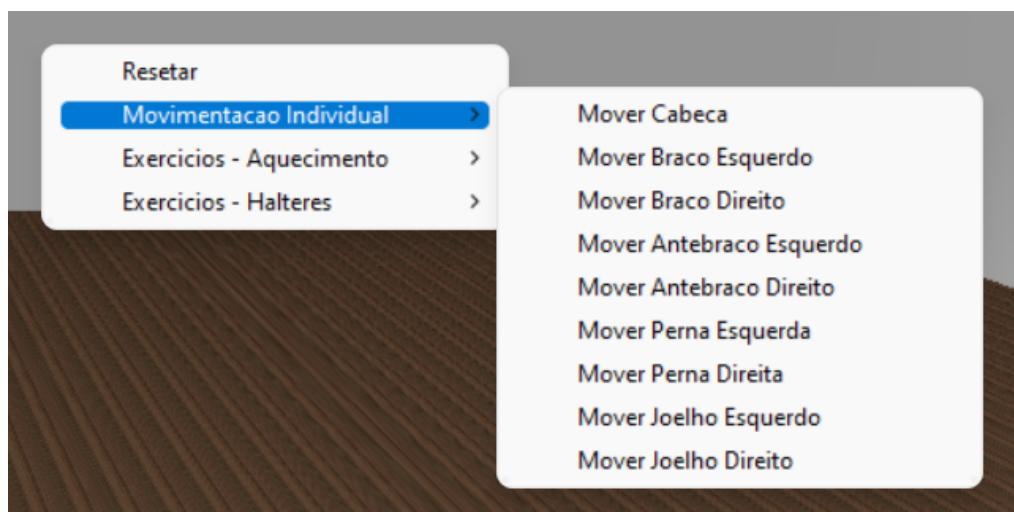
Exercícios com halteres - (Afundo).

4.3) Opções de menu

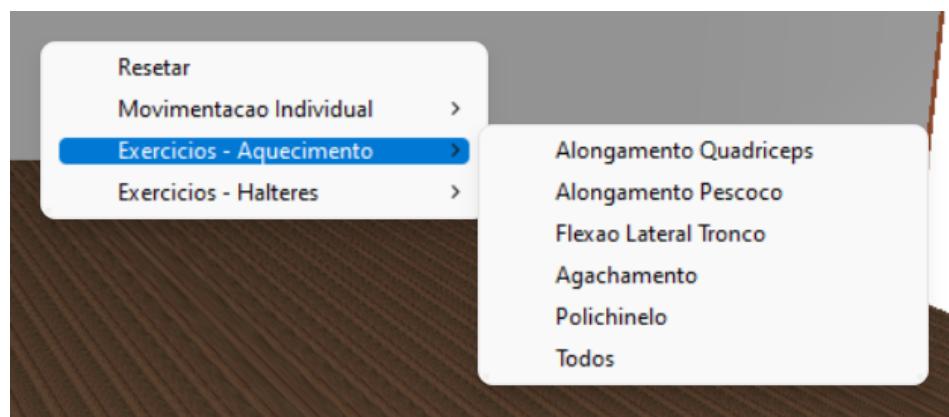
Para acessar o menu, basta clicar com o botão direito do mouse. O menu principal do projeto consiste em 4 opções disponíveis:



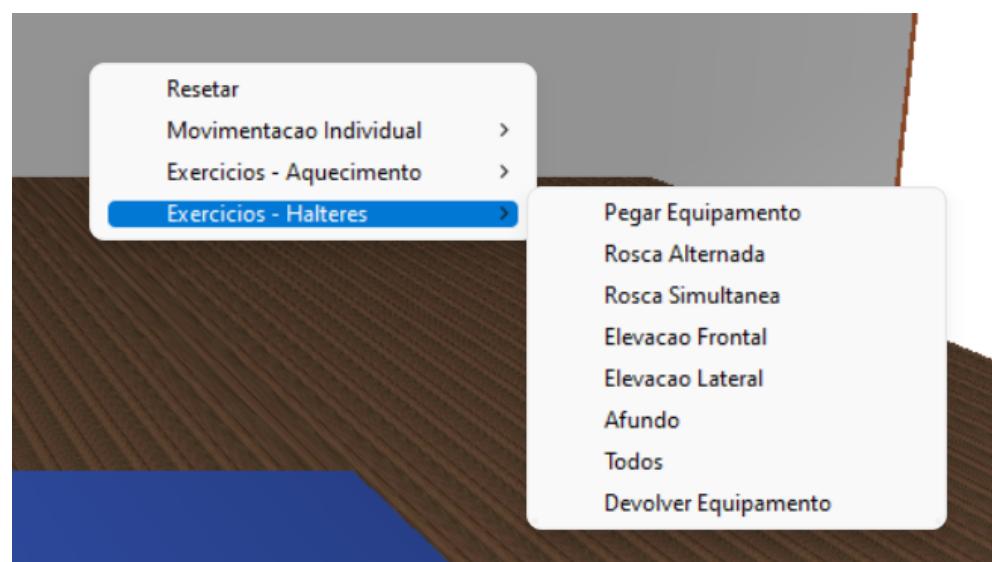
- Resetar: para de executar animação corrente (apenas para exercícios sem equipamento) e retorna todos os ângulos das articulações do corpo humano para posição inicial.
- Movimentação individual: permite mexer cada parte do corpo do personagem de modo separado (quando uma animação estiver sendo executada e o usuário selecionar esta opção, a animação será automaticamente resetada). Dentre essas partes do corpo, temos as seguintes opções de movimento:



- Exercícios - Aquecimento: executa as animações dos exercícios sem equipamentos. Ainda, possui a opção de executar todas as animações deste menu.



- Exercícios - Halteres: executa as animações dos exercícios com equipamentos. Para tanto, é necessário primeiramente que o usuário selecione a opção de “Pegar Equipamento” antes de realizar qualquer exercício. Quando o personagem estiver com equipamento não é possível selecionar as outras opções do menu principal além deste menu com exercícios com equipamentos, para poder selecionar outras opções é necessário que o usuário selecione a opção de “Devolver Equipamento”. Ainda, possui a opção de executar todas as animações deste menu.



4.4) Opções de teclado

As opções de teclado disponíveis no programa são as seguintes:

- ESC: sair do programa.
- x, y, z: aplica uma rotação positiva em um dos eixos x, y, ou z. Utilizado em conjunto quando o usuário escolher movimentar individualmente uma dada estrutura do corpo humano.
- X, Y, Z: aplica uma rotação negativa em um dos eixos x, y, ou z. Utilizado em conjunto quando o usuário escolher movimentar individualmente uma dada estrutura do corpo humano.
- +: aplica um *zoom in* da câmera.
- -: aplica um *zoom out* da câmera.
- Teclas direcionais (esquerda, direita): aplica uma rotação azimutal (rotação em y) para a posição da câmera, sendo a tecla direcional da esquerda uma rotação negativa e a tecla direcional da direita uma rotação positiva.
- Teclas direcionais (baixo, cima): aplica uma rotação elevação (rotação em x) para a posição da câmera, sendo a tecla direcional para baixo uma rotação negativa e a tecla direcional para cima uma rotação positiva.

4.5) Discussões sobre o desenvolvimento do projeto

Problemas ocorridos no decorrer do programa: reposicionamento de objetos no cenário, depuração para animação dos exercícios, aplicação de textura, *bugs* decorrentes do uso incorreto de *flags*, complexidade do código.

- Repositionamento de objetos no cenário: uma vez criado os objetos, fiquei interessado em aplicar transformações geométricas, como rotação e translação, em objetos individuais como um todo. A forma mais eficiente que encontrei para solucionar este problema foi utilizar um modelo hierárquico dentro das funções próprias dos objetos.
- Depuração para animação dos exercícios: a forma que encontrei para visualizar se a lógica da animação estava correta foi utilizar o menu de opções para movimentar individualmente cada estrutura do corpo humano e imprimindo as coordenadas dos eixos de rotação.
- Aplicação de textura: devido a complexidade para utilização de texturas, acabei me equivocando em detalhes para implementá-las da forma correta. Por exemplo, estava carregando a textura diretamente no arquivo cabeçalho do cenário ao invés do programa *main*, o que ocasionou em falhas no programa.
- *bugs* decorrentes ao uso incorreto de *flags*: durante o desenvolvimento do arquivo cabeçalho de animação, foi necessário a criação de diversas *flags* para controlar o fluxo de animação corrente. No entanto, como é de se esperar, quanto maior a quantidade de variáveis de controle lógico, mais sujeito o programa está para falhas e maior ainda é a dificuldade em encontrá-las. Foi necessário reservar umas boas horas da semana para corrigir essas falhas.
- Complexidade do código: somando todas as linhas de código do programa (1 arquivo *main* e 6 arquivos de cabeçalho), temos mais de 3.000 linhas de código no total, sendo que o arquivo de cabeçalho *animacao.h* contém mais de 1.500 linhas e o arquivo de cabeçalho *cenario.h* contém mais de 800. Para facilitar a leitura do código inclui uma quantidade considerável de comentários e técnicas de *clean code*, como refatoração do código e nome adequado para variáveis e funções.

5. Conclusão

Em resumo, o programa “Maratona da Fisioterapia” permitiu a realização de um ambiente que auxilie na reabilitação de pacientes que necessitam de fisioterapia através da criação de um cenário tridimensional, utilizando a API de Computação Gráfica OpenGL, em conjunto com a criação de um personagem, criando um corpo humano utilizando um modelo hierárquico, e animações dos exercícios a serem executados pelo personagem, sendo 10 exercícios no total no qual 5 destes exercícios são executados sem equipamentos e os outros 5 exercícios com equipamentos.

Por fim, em conjunto com as disciplinas de geometria analítica e álgebra linear, o desenvolvimento do projeto permitiu aprimorar meu aprendizado da disciplina de Computação Gráfica como um todo, solidificando os conhecimentos de visualização bi e tridimensional, projeção e perspectiva, transformações geométricas e modelos hierárquicos.